

# SCons

## API Documentation

November 9, 2015

## Contents

<b>Contents</b>	<b>1</b>
<b>1 Package SCons</b>	<b>2</b>
1.1 Modules . . . . .	2
1.2 Variables . . . . .	4
<b>2 Module SCons.Action</b>	<b>5</b>
2.1 Functions . . . . .	6
2.2 Variables . . . . .	6
2.3 Class ActionBase . . . . .	6
2.3.1 Methods . . . . .	7
2.3.2 Properties . . . . .	7
2.4 Class CommandAction . . . . .	8
2.4.1 Methods . . . . .	8
2.4.2 Properties . . . . .	9
2.5 Class CommandGeneratorAction . . . . .	9
2.5.1 Methods . . . . .	10
2.5.2 Properties . . . . .	11
2.6 Class LazyAction . . . . .	11
2.6.1 Methods . . . . .	11
2.6.2 Properties . . . . .	12
2.7 Class FunctionAction . . . . .	13
2.7.1 Methods . . . . .	13
2.7.2 Properties . . . . .	14
2.8 Class ListAction . . . . .	14
2.8.1 Methods . . . . .	14
2.8.2 Properties . . . . .	15
2.9 Class ActionCaller . . . . .	15
2.9.1 Methods . . . . .	16
2.9.2 Properties . . . . .	16
2.10 Class ActionFactory . . . . .	17
2.10.1 Methods . . . . .	17
2.10.2 Properties . . . . .	17
<b>3 Module SCons.Builder</b>	<b>18</b>
3.1 Functions . . . . .	19
3.2 Variables . . . . .	20

3.3	Class DictCmdGenerator . . . . .	20
3.3.1	Methods . . . . .	20
3.3.2	Class Variables . . . . .	21
3.4	Class CallableSelector . . . . .	21
3.4.1	Methods . . . . .	21
3.4.2	Class Variables . . . . .	22
3.5	Class DictEmitter . . . . .	22
3.5.1	Methods . . . . .	22
3.5.2	Class Variables . . . . .	23
3.6	Class ListEmitter . . . . .	23
3.6.1	Methods . . . . .	23
3.6.2	Properties . . . . .	24
3.6.3	Class Variables . . . . .	24
3.7	Class OverrideWarner . . . . .	24
3.7.1	Methods . . . . .	24
3.7.2	Class Variables . . . . .	25
3.8	Class EmitterProxy . . . . .	25
3.8.1	Methods . . . . .	25
3.8.2	Properties . . . . .	25
3.9	Class BuilderBase . . . . .	26
3.9.1	Methods . . . . .	26
3.9.2	Properties . . . . .	28
3.10	Class CompositeBuilder . . . . .	29
3.10.1	Methods . . . . .	29
3.10.2	Properties . . . . .	29
<b>4</b>	<b>Module SCons.CacheDir . . . . .</b>	<b>30</b>
4.1	Functions . . . . .	30
4.2	Variables . . . . .	30
4.3	Class CacheDir . . . . .	30
4.3.1	Methods . . . . .	31
4.3.2	Properties . . . . .	32
<b>5</b>	<b>Module SCons.Conftest . . . . .</b>	<b>33</b>
5.1	Functions . . . . .	33
5.2	Variables . . . . .	37
<b>6</b>	<b>Module SCons.Debug . . . . .</b>	<b>38</b>
6.1	Functions . . . . .	38
6.2	Variables . . . . .	38
<b>7</b>	<b>Module SCons.Defaults . . . . .</b>	<b>40</b>
7.1	Functions . . . . .	40
7.2	Variables . . . . .	41
7.3	Class NullCmdGenerator . . . . .	42
7.3.1	Methods . . . . .	43
7.3.2	Properties . . . . .	43
7.4	Class Variable_Method_Caller . . . . .	43
7.4.1	Methods . . . . .	44
7.4.2	Properties . . . . .	44
<b>8</b>	<b>Module SCons.Environment . . . . .</b>	<b>45</b>

8.1	Functions . . . . .	45
8.2	Variables . . . . .	45
8.3	Class MethodWrapper . . . . .	46
8.3.1	Methods . . . . .	46
8.3.2	Properties . . . . .	47
8.4	Class BuilderWrapper . . . . .	47
8.4.1	Methods . . . . .	47
8.4.2	Properties . . . . .	48
8.5	Class BuilderDict . . . . .	48
8.5.1	Methods . . . . .	48
8.5.2	Class Variables . . . . .	49
8.6	Class SubstitutionEnvironment . . . . .	49
8.6.1	Methods . . . . .	50
8.6.2	Properties . . . . .	53
8.7	Class Base . . . . .	53
8.7.1	Methods . . . . .	53
8.7.2	Properties . . . . .	61
8.8	Class OverrideEnvironment . . . . .	61
8.8.1	Methods . . . . .	62
8.8.2	Properties . . . . .	64
8.9	Class Base . . . . .	64
8.9.1	Methods . . . . .	64
8.9.2	Properties . . . . .	72
<b>9</b>	<b>Module SCons.Errors</b>	<b>74</b>
9.1	Functions . . . . .	74
9.2	Variables . . . . .	74
9.3	Class BuildError . . . . .	74
9.3.1	Methods . . . . .	76
9.3.2	Properties . . . . .	76
9.4	Class InternalError . . . . .	76
9.4.1	Methods . . . . .	77
9.4.2	Properties . . . . .	77
9.5	Class UserError . . . . .	77
9.5.1	Methods . . . . .	77
9.5.2	Properties . . . . .	78
9.6	Class StopError . . . . .	78
9.6.1	Methods . . . . .	78
9.6.2	Properties . . . . .	78
9.7	Class EnvironmentError . . . . .	79
9.7.1	Methods . . . . .	79
9.7.2	Properties . . . . .	79
9.8	Class MSVCErrors . . . . .	80
9.8.1	Methods . . . . .	80
9.8.2	Properties . . . . .	80
9.9	Class ExplicitExit . . . . .	81
9.9.1	Methods . . . . .	81
9.9.2	Properties . . . . .	81
<b>10</b>	<b>Module SCons.Executor</b>	<b>82</b>
10.1	Functions . . . . .	82
10.2	Variables . . . . .	82

10.3 Class Batch . . . . .	83
10.3.1 Methods . . . . .	83
10.3.2 Properties . . . . .	83
10.4 Class TSList . . . . .	84
10.4.1 Methods . . . . .	84
10.4.2 Properties . . . . .	85
10.4.3 Class Variables . . . . .	85
10.5 Class TSOBJECT . . . . .	86
10.5.1 Methods . . . . .	86
10.5.2 Properties . . . . .	86
10.6 Class Executor . . . . .	86
10.6.1 Methods . . . . .	87
10.6.2 Properties . . . . .	90
10.7 Class NullEnvironment . . . . .	91
10.7.1 Methods . . . . .	91
10.7.2 Properties . . . . .	91
10.8 Class Null . . . . .	91
10.8.1 Methods . . . . .	92
10.8.2 Properties . . . . .	93
<b>11 Module SCons.Job</b>	<b>94</b>
11.1 Variables . . . . .	94
11.2 Class InterruptState . . . . .	94
11.2.1 Methods . . . . .	94
11.2.2 Properties . . . . .	95
11.3 Class Jobs . . . . .	95
11.3.1 Methods . . . . .	95
11.3.2 Properties . . . . .	96
11.4 Class Serial . . . . .	96
11.4.1 Methods . . . . .	96
11.4.2 Properties . . . . .	97
11.5 Class Worker . . . . .	97
11.5.1 Methods . . . . .	98
11.5.2 Properties . . . . .	99
11.6 Class ThreadPool . . . . .	99
11.6.1 Methods . . . . .	99
11.6.2 Properties . . . . .	100
11.7 Class Parallel . . . . .	100
11.7.1 Methods . . . . .	101
11.7.2 Properties . . . . .	101
<b>12 Module SCons.Memoize</b>	<b>102</b>
12.1 Functions . . . . .	103
12.2 Variables . . . . .	104
12.3 Class Counter . . . . .	104
12.3.1 Methods . . . . .	105
12.3.2 Properties . . . . .	105
12.4 Class CountValue . . . . .	105
12.4.1 Methods . . . . .	106
12.4.2 Properties . . . . .	106
12.5 Class CountDict . . . . .	106
12.5.1 Methods . . . . .	107

12.5.2 Properties . . . . .	107
<b>13 Package SCons.Node</b>	<b>108</b>
13.1 Modules . . . . .	108
13.2 Functions . . . . .	108
13.3 Variables . . . . .	110
13.4 Class NodeInfoBase . . . . .	111
13.4.1 Methods . . . . .	112
13.4.2 Properties . . . . .	112
13.4.3 Class Variables . . . . .	112
13.5 Class BuildInfoBase . . . . .	113
13.5.1 Methods . . . . .	113
13.5.2 Properties . . . . .	114
13.5.3 Class Variables . . . . .	114
13.6 Class Node . . . . .	114
13.6.1 Methods . . . . .	115
13.6.2 Properties . . . . .	127
13.7 Class NodeList . . . . .	128
13.7.1 Methods . . . . .	128
13.7.2 Properties . . . . .	129
13.7.3 Class Variables . . . . .	129
13.8 Class Walker . . . . .	129
13.8.1 Methods . . . . .	130
13.8.2 Properties . . . . .	130
<b>14 Module SCons.Node.Alias</b>	<b>131</b>
14.1 Variables . . . . .	131
14.2 Class AliasNameSpace . . . . .	131
14.2.1 Methods . . . . .	131
14.2.2 Class Variables . . . . .	131
14.3 Class AliasNodeInfo . . . . .	132
14.3.1 Methods . . . . .	132
14.3.2 Properties . . . . .	132
14.3.3 Class Variables . . . . .	133
14.4 Class AliasBuildInfo . . . . .	133
14.4.1 Methods . . . . .	133
14.4.2 Properties . . . . .	133
14.4.3 Class Variables . . . . .	134
14.5 Class Alias . . . . .	134
14.5.1 Methods . . . . .	134
14.5.2 Properties . . . . .	136
<b>15 Module SCons.Node.FS</b>	<b>137</b>
15.1 Functions . . . . .	137
15.2 Variables . . . . .	139
15.3 Class EntryProxyAttributeError . . . . .	141
15.3.1 Methods . . . . .	141
15.3.2 Properties . . . . .	141
15.4 Class DiskChecker . . . . .	142
15.4.1 Methods . . . . .	142
15.4.2 Properties . . . . .	142
15.5 Class EntryProxy . . . . .	143

15.5.1	Methods . . . . .	143
15.5.2	Properties . . . . .	143
15.5.3	Class Variables . . . . .	143
15.6	Class Base . . . . .	144
15.6.1	Methods . . . . .	144
15.6.2	Properties . . . . .	149
15.6.3	Instance Variables . . . . .	149
15.7	Class Entry . . . . .	149
15.7.1	Methods . . . . .	150
15.7.2	Properties . . . . .	152
15.7.3	Instance Variables . . . . .	153
15.8	Class LocalFS . . . . .	153
15.8.1	Methods . . . . .	153
15.8.2	Properties . . . . .	154
15.9	Class FS . . . . .	154
15.9.1	Methods . . . . .	155
15.9.2	Properties . . . . .	157
15.10	Class DirNodeInfo . . . . .	157
15.10.1	Methods . . . . .	157
15.10.2	Properties . . . . .	158
15.10.3	Class Variables . . . . .	158
15.11	Class DirBuildInfo . . . . .	158
15.11.1	Methods . . . . .	158
15.11.2	Properties . . . . .	159
15.11.3	Class Variables . . . . .	159
15.12	Class Dir . . . . .	159
15.12.1	Methods . . . . .	160
15.12.2	Properties . . . . .	167
15.12.3	Instance Variables . . . . .	168
15.13	Class RootDir . . . . .	168
15.13.1	Methods . . . . .	168
15.13.2	Properties . . . . .	170
15.13.3	Instance Variables . . . . .	171
15.14	Class FileInfo . . . . .	171
15.14.1	Methods . . . . .	171
15.14.2	Properties . . . . .	172
15.14.3	Class Variables . . . . .	172
15.15	Class FileBuildInfo . . . . .	173
15.15.1	Methods . . . . .	173
15.15.2	Properties . . . . .	174
15.15.3	Class Variables . . . . .	174
15.16	Class File . . . . .	174
15.16.1	Methods . . . . .	175
15.16.2	Properties . . . . .	181
15.16.3	Class Variables . . . . .	182
15.16.4	Instance Variables . . . . .	182
15.17	Class FileFinder . . . . .	183
15.17.1	Methods . . . . .	183
15.17.2	Properties . . . . .	184
<b>16</b>	<b>Module SCons.Node.Python</b>	<b>185</b>
16.1	Variables . . . . .	185

16.2	Class ValueNodeInfo . . . . .	185
16.2.1	Methods . . . . .	185
16.2.2	Properties . . . . .	186
16.2.3	Class Variables . . . . .	186
16.3	Class ValueBuildInfo . . . . .	186
16.3.1	Methods . . . . .	187
16.3.2	Properties . . . . .	187
16.3.3	Class Variables . . . . .	187
16.4	Class Value . . . . .	187
16.4.1	Methods . . . . .	188
16.4.2	Properties . . . . .	190
<b>17</b>	<b>Module SCons.PathList</b>	<b>191</b>
17.1	Functions . . . . .	191
17.2	Variables . . . . .	191
<b>18</b>	<b>Module SCons.SConf</b>	<b>192</b>
18.1	Functions . . . . .	192
18.2	Variables . . . . .	194
18.3	Class SConfWarning . . . . .	195
18.3.1	Methods . . . . .	195
18.3.2	Properties . . . . .	195
18.4	Class SConfError . . . . .	196
18.4.1	Methods . . . . .	196
18.4.2	Properties . . . . .	196
18.5	Class ConfigureDryRunError . . . . .	197
18.5.1	Methods . . . . .	197
18.5.2	Properties . . . . .	197
18.6	Class ConfigureCacheError . . . . .	198
18.6.1	Methods . . . . .	198
18.6.2	Properties . . . . .	198
18.7	Class SConfBuildInfo . . . . .	199
18.7.1	Methods . . . . .	199
18.7.2	Properties . . . . .	199
18.7.3	Class Variables . . . . .	200
18.8	Class Streamer . . . . .	200
18.8.1	Methods . . . . .	200
18.8.2	Properties . . . . .	201
18.9	Class SConfBuildTask . . . . .	201
18.9.1	Methods . . . . .	201
18.9.2	Properties . . . . .	202
18.10	Class SConfBase . . . . .	203
18.10.1	Methods . . . . .	203
18.10.2	Properties . . . . .	205
18.11	Class CheckContext . . . . .	206
18.11.1	Methods . . . . .	206
18.11.2	Properties . . . . .	207
<b>19</b>	<b>Module SCons.SConsign</b>	<b>208</b>
19.1	Functions . . . . .	208
19.2	Variables . . . . .	208
19.3	Class SConsignEntry . . . . .	209

19.3.1	Methods . . . . .	209
19.3.2	Properties . . . . .	209
19.3.3	Class Variables . . . . .	209
19.4	Class Base . . . . .	210
19.4.1	Methods . . . . .	210
19.4.2	Properties . . . . .	211
19.5	Class DB . . . . .	211
19.5.1	Methods . . . . .	211
19.5.2	Properties . . . . .	211
19.6	Class Dir . . . . .	212
19.6.1	Methods . . . . .	212
19.6.2	Properties . . . . .	212
19.7	Class DirFile . . . . .	213
19.7.1	Methods . . . . .	213
19.7.2	Properties . . . . .	213
19.8	Class DB . . . . .	214
19.8.1	Methods . . . . .	214
19.8.2	Properties . . . . .	214
<b>20</b>	<b>Package SCons.Scanner</b>	<b>215</b>
20.1	Modules . . . . .	215
20.2	Functions . . . . .	215
20.3	Variables . . . . .	215
20.4	Class FindPathDirs . . . . .	216
20.4.1	Methods . . . . .	216
20.4.2	Properties . . . . .	216
20.5	Class Base . . . . .	216
20.5.1	Methods . . . . .	217
20.5.2	Properties . . . . .	219
20.6	Class Selector . . . . .	219
20.6.1	Methods . . . . .	222
20.6.2	Properties . . . . .	223
20.7	Class Current . . . . .	223
20.7.1	Methods . . . . .	225
20.7.2	Properties . . . . .	226
20.8	Class Classic . . . . .	226
20.8.1	Methods . . . . .	228
20.8.2	Properties . . . . .	229
20.9	Class ClassicCPP . . . . .	229
20.9.1	Methods . . . . .	230
20.9.2	Properties . . . . .	230
<b>21</b>	<b>Module SCons.Scanner.C</b>	<b>231</b>
21.1	Functions . . . . .	231
21.2	Variables . . . . .	231
21.3	Class SConsCPPScanner . . . . .	231
21.3.1	Methods . . . . .	232
21.3.2	Properties . . . . .	232
21.4	Class SConsCPPScannerWrapper . . . . .	233
21.4.1	Methods . . . . .	233
21.4.2	Properties . . . . .	233



<b>22 Module SCons.Scanner.D</b>	<b>234</b>
22.1 Functions . . . . .	234
22.2 Variables . . . . .	234
22.3 Class D . . . . .	234
22.3.1 Methods . . . . .	236
22.3.2 Properties . . . . .	237
<b>23 Module SCons.Scanner.Dir</b>	<b>238</b>
23.1 Functions . . . . .	238
23.2 Variables . . . . .	238
<b>24 Module SCons.Scanner.Fortran</b>	<b>240</b>
24.1 Functions . . . . .	240
24.2 Variables . . . . .	240
24.3 Class F90Scanner . . . . .	240
24.3.1 Methods . . . . .	243
24.3.2 Properties . . . . .	244
<b>25 Module SCons.Scanner.IDL</b>	<b>245</b>
25.1 Functions . . . . .	245
25.2 Variables . . . . .	245
<b>26 Module SCons.Scanner.LaTeX</b>	<b>246</b>
26.1 Functions . . . . .	246
26.2 Variables . . . . .	246
26.3 Class FindENVPPathDirs . . . . .	246
26.3.1 Methods . . . . .	247
26.3.2 Properties . . . . .	247
26.4 Class LaTeX . . . . .	247
26.4.1 Methods . . . . .	250
26.4.2 Properties . . . . .	251
26.4.3 Class Variables . . . . .	251
<b>27 Module SCons.Scanner.Prog</b>	<b>253</b>
27.1 Functions . . . . .	253
27.2 Variables . . . . .	253
<b>28 Module SCons.Scanner.RC</b>	<b>254</b>
28.1 Functions . . . . .	254
28.2 Variables . . . . .	254
<b>29 Package SCons.Script</b>	<b>255</b>
29.1 Modules . . . . .	255
29.2 Functions . . . . .	255
29.3 Variables . . . . .	255
29.4 Class TargetList . . . . .	262
29.4.1 Methods . . . . .	262
29.4.2 Properties . . . . .	263
29.4.3 Class Variables . . . . .	263
<b>30 Module SCons.Script.Interactive</b>	<b>264</b>
30.1 Functions . . . . .	264
30.2 Variables . . . . .	264

30.3 Class SConsInteractiveCmd . . . . .	264
30.3.1 Methods . . . . .	265
30.3.2 Class Variables . . . . .	266
<b>31 Module SCons.Script.Main . . . . .</b>	<b>267</b>
31.1 Functions . . . . .	267
31.2 Variables . . . . .	268
31.3 Class SConsPrintHelpException . . . . .	269
31.3.1 Methods . . . . .	269
31.3.2 Properties . . . . .	269
31.4 Class Progressor . . . . .	270
31.4.1 Methods . . . . .	270
31.4.2 Properties . . . . .	270
31.4.3 Class Variables . . . . .	270
31.5 Class BuildTask . . . . .	271
31.5.1 Methods . . . . .	271
31.5.2 Properties . . . . .	273
31.5.3 Class Variables . . . . .	273
31.6 Class CleanTask . . . . .	273
31.6.1 Methods . . . . .	274
31.6.2 Properties . . . . .	275
31.7 Class QuestionTask . . . . .	275
31.7.1 Methods . . . . .	276
31.7.2 Properties . . . . .	277
31.8 Class TreePrinter . . . . .	277
31.8.1 Methods . . . . .	277
31.8.2 Properties . . . . .	277
31.9 Class FakeOptionParser . . . . .	278
31.9.1 Methods . . . . .	278
31.9.2 Properties . . . . .	278
31.9.3 Class Variables . . . . .	278
31.10 Class Stats . . . . .	278
31.10.1 Methods . . . . .	279
31.10.2 Properties . . . . .	279
31.11 Class CountStats . . . . .	279
31.11.1 Methods . . . . .	279
31.11.2 Properties . . . . .	280
31.12 Class MemStats . . . . .	280
31.12.1 Methods . . . . .	280
31.12.2 Properties . . . . .	280
<b>32 Module SCons.Script.SConscript' . . . . .</b>	<b>281</b>
32.1 Functions . . . . .	281
32.2 Variables . . . . .	282
32.3 Class SConscriptReturn . . . . .	282
32.3.1 Methods . . . . .	282
32.3.2 Properties . . . . .	283
32.4 Class Frame . . . . .	283
32.4.1 Methods . . . . .	283
32.4.2 Properties . . . . .	283
32.5 Class SConsEnvironment . . . . .	284
32.5.1 Methods . . . . .	284

32.5.2	Properties . . . . .	285
32.6	Class DefaultEnvironmentCall . . . . .	286
32.6.1	Methods . . . . .	286
32.6.2	Properties . . . . .	286
<b>33</b>	<b>Module SCons.Sig</b>	<b>287</b>
33.1	Variables . . . . .	287
33.2	Class MD5Null . . . . .	287
33.2.1	Methods . . . . .	287
33.2.2	Properties . . . . .	288
33.3	Class TimeStampNull . . . . .	288
33.3.1	Methods . . . . .	288
33.3.2	Properties . . . . .	288
<b>34</b>	<b>Module SCons.Subst</b>	<b>290</b>
34.1	Functions . . . . .	290
34.2	Variables . . . . .	291
34.3	Class Literal . . . . .	292
34.3.1	Methods . . . . .	292
34.3.2	Properties . . . . .	293
34.4	Class SpecialAttrWrapper . . . . .	293
34.4.1	Methods . . . . .	293
34.4.2	Properties . . . . .	294
34.5	Class CmdStringHolder . . . . .	294
34.5.1	Methods . . . . .	294
34.5.2	Properties . . . . .	295
34.5.3	Class Variables . . . . .	295
34.6	Class NLWrapper . . . . .	296
34.6.1	Methods . . . . .	296
34.6.2	Properties . . . . .	296
34.7	Class Targets_or_Sources . . . . .	297
34.7.1	Methods . . . . .	297
34.7.2	Properties . . . . .	298
34.7.3	Class Variables . . . . .	298
34.8	Class Target_or_Source . . . . .	299
34.8.1	Methods . . . . .	299
34.8.2	Properties . . . . .	299
34.9	Class NullNodeList . . . . .	300
34.9.1	Methods . . . . .	300
34.9.2	Properties . . . . .	300
<b>35</b>	<b>Module SCons.Taskmaster</b>	<b>301</b>
35.1	Functions . . . . .	301
35.2	Variables . . . . .	301
35.3	Class Stats . . . . .	302
35.3.1	Methods . . . . .	302
35.3.2	Properties . . . . .	302
35.4	Class Task . . . . .	303
35.4.1	Methods . . . . .	303
35.4.2	Properties . . . . .	307
35.5	Class AlwaysTask . . . . .	307
35.5.1	Methods . . . . .	308

35.5.2 Properties . . . . .	308
35.6 Class OutOfDateTask . . . . .	308
35.6.1 Methods . . . . .	309
35.6.2 Properties . . . . .	309
35.7 Class Taskmaster . . . . .	309
35.7.1 Methods . . . . .	309
35.7.2 Properties . . . . .	311
<b>36 Module SCons.Util</b>	<b>312</b>
36.1 Functions . . . . .	312
36.2 Variables . . . . .	318
36.3 Class NodeList . . . . .	320
36.3.1 Methods . . . . .	320
36.3.2 Properties . . . . .	321
36.3.3 Class Variables . . . . .	321
36.4 Class DisplayEngine . . . . .	321
36.4.1 Methods . . . . .	322
36.4.2 Properties . . . . .	322
36.4.3 Class Variables . . . . .	322
36.5 Class Proxy . . . . .	322
36.5.1 Methods . . . . .	323
36.5.2 Properties . . . . .	323
36.6 Class Delegate . . . . .	324
36.6.1 Methods . . . . .	324
36.6.2 Properties . . . . .	324
36.7 Class _NoError . . . . .	324
36.7.1 Methods . . . . .	325
36.7.2 Properties . . . . .	325
36.8 Class WindowsError . . . . .	325
36.8.1 Methods . . . . .	325
36.8.2 Properties . . . . .	326
36.9 Class CLVar . . . . .	327
36.9.1 Methods . . . . .	327
36.9.2 Properties . . . . .	328
36.9.3 Class Variables . . . . .	328
36.10Class OrderedDict . . . . .	328
36.10.1Methods . . . . .	329
36.10.2Class Variables . . . . .	330
36.11Class Selector . . . . .	330
36.11.1Methods . . . . .	330
36.11.2Class Variables . . . . .	330
36.12Class LogicalLines . . . . .	331
36.12.1Methods . . . . .	331
36.12.2Properties . . . . .	331
36.13Class UniqueList . . . . .	332
36.13.1Methods . . . . .	332
36.13.2Properties . . . . .	335
36.13.3Class Variables . . . . .	335
36.14Class Unbuffered . . . . .	335
36.14.1Methods . . . . .	336
36.14.2Properties . . . . .	336
36.15Class Null . . . . .	336

36.15.1 Methods . . . . .	336
36.15.2 Properties . . . . .	337
36.16 Class NullSeq . . . . .	338
36.16.1 Methods . . . . .	338
36.16.2 Properties . . . . .	338
<b>37 Package SCons.Variables</b>	<b>339</b>
37.1 Modules . . . . .	339
37.2 Variables . . . . .	339
37.3 Class Variables . . . . .	339
37.3.1 Methods . . . . .	340
37.3.2 Properties . . . . .	342
37.3.3 Class Variables . . . . .	342
<b>38 Module SCons.Variables.BoolVariable'</b>	<b>343</b>
38.1 Functions . . . . .	343
<b>39 Module SCons.Variables.EnumVariable'</b>	<b>344</b>
39.1 Functions . . . . .	345
<b>40 Module SCons.Variables.ListVariable'</b>	<b>346</b>
40.1 Functions . . . . .	346
<b>41 Module SCons.Variables.PackageVariable'</b>	<b>347</b>
41.1 Functions . . . . .	347
<b>42 Module SCons.Variables.PathVariable'</b>	<b>348</b>
42.1 Variables . . . . .	349
<b>43 Module SCons.Warnings</b>	<b>350</b>
43.1 Functions . . . . .	350
43.2 Variables . . . . .	351
43.3 Class Warning . . . . .	351
43.3.1 Methods . . . . .	352
43.3.2 Properties . . . . .	352
43.4 Class WarningOnByDefault . . . . .	352
43.4.1 Methods . . . . .	353
43.4.2 Properties . . . . .	353
43.5 Class TargetNotBuiltWarning . . . . .	353
43.5.1 Methods . . . . .	353
43.5.2 Properties . . . . .	354
43.6 Class CacheWriteErrorWarning . . . . .	354
43.6.1 Methods . . . . .	354
43.6.2 Properties . . . . .	355
43.7 Class CorruptSConsignWarning . . . . .	355
43.7.1 Methods . . . . .	355
43.7.2 Properties . . . . .	356
43.8 Class DependencyWarning . . . . .	356
43.8.1 Methods . . . . .	356
43.8.2 Properties . . . . .	356
43.9 Class DevelopmentVersionWarning . . . . .	357
43.9.1 Methods . . . . .	357
43.9.2 Properties . . . . .	357

43.10	Class DuplicateEnvironmentWarning . . . . .	358
43.10.1	Methods . . . . .	358
43.10.2	Properties . . . . .	358
43.11	Class FutureReservedVariableWarning . . . . .	359
43.11.1	Methods . . . . .	359
43.11.2	Properties . . . . .	359
43.12	Class LinkWarning . . . . .	360
43.12.1	Methods . . . . .	360
43.12.2	Properties . . . . .	360
43.13	Class MisleadingKeywordsWarning . . . . .	361
43.13.1	Methods . . . . .	361
43.13.2	Properties . . . . .	361
43.14	Class MissingSConscriptWarning . . . . .	362
43.14.1	Methods . . . . .	362
43.14.2	Properties . . . . .	362
43.15	Class NoMD5ModuleWarning . . . . .	363
43.15.1	Methods . . . . .	363
43.15.2	Properties . . . . .	363
43.16	Class NoMetaclassSupportWarning . . . . .	364
43.16.1	Methods . . . . .	364
43.16.2	Properties . . . . .	364
43.17	Class NoObjectCountWarning . . . . .	365
43.17.1	Methods . . . . .	365
43.17.2	Properties . . . . .	365
43.18	Class NoParallelSupportWarning . . . . .	366
43.18.1	Methods . . . . .	366
43.18.2	Properties . . . . .	366
43.19	Class ReservedVariableWarning . . . . .	367
43.19.1	Methods . . . . .	367
43.19.2	Properties . . . . .	367
43.20	Class StackSizeWarning . . . . .	368
43.20.1	Methods . . . . .	368
43.20.2	Properties . . . . .	368
43.21	Class VisualCMissingWarning . . . . .	369
43.21.1	Methods . . . . .	369
43.21.2	Properties . . . . .	369
43.22	Class VisualVersionMismatch . . . . .	370
43.22.1	Methods . . . . .	370
43.22.2	Properties . . . . .	370
43.23	Class VisualStudioMissingWarning . . . . .	371
43.23.1	Methods . . . . .	371
43.23.2	Properties . . . . .	371
43.24	Class FortranCxxMixWarning . . . . .	372
43.24.1	Methods . . . . .	372
43.24.2	Properties . . . . .	372
43.25	Class FutureDeprecatedWarning . . . . .	373
43.25.1	Methods . . . . .	373
43.25.2	Properties . . . . .	373
43.26	Class DeprecatedWarning . . . . .	374
43.26.1	Methods . . . . .	374
43.26.2	Properties . . . . .	374

43.27	Class MandatoryDeprecatedWarning . . . . .	375
43.27.1	Methods . . . . .	375
43.27.2	Properties . . . . .	375
43.28	Class PythonVersionWarning . . . . .	376
43.28.1	Methods . . . . .	376
43.28.2	Properties . . . . .	376
43.29	Class DeprecatedSourceCodeWarning . . . . .	377
43.29.1	Methods . . . . .	377
43.29.2	Properties . . . . .	377
43.30	Class DeprecatedBuildDirWarning . . . . .	378
43.30.1	Methods . . . . .	378
43.30.2	Properties . . . . .	378
43.31	Class TaskmasterNeedsExecuteWarning . . . . .	379
43.31.1	Methods . . . . .	379
43.31.2	Properties . . . . .	379
43.32	Class DeprecatedCopyWarning . . . . .	380
43.32.1	Methods . . . . .	380
43.32.2	Properties . . . . .	380
43.33	Class DeprecatedOptionsWarning . . . . .	381
43.33.1	Methods . . . . .	381
43.33.2	Properties . . . . .	381
43.34	Class DeprecatedSourceSignaturesWarning . . . . .	382
43.34.1	Methods . . . . .	382
43.34.2	Properties . . . . .	382
43.35	Class DeprecatedTargetSignaturesWarning . . . . .	383
43.35.1	Methods . . . . .	383
43.35.2	Properties . . . . .	383
43.36	Class DeprecatedDebugOptionsWarning . . . . .	384
43.36.1	Methods . . . . .	384
43.36.2	Properties . . . . .	384
43.37	Class DeprecatedSigModuleWarning . . . . .	385
43.37.1	Methods . . . . .	385
43.37.2	Properties . . . . .	385
43.38	Class DeprecatedBuilderKeywordsWarning . . . . .	386
43.38.1	Methods . . . . .	386
43.38.2	Properties . . . . .	386
<b>44</b>	<b>Module SCons.cpp . . . . .</b>	<b>387</b>
44.1	Functions . . . . .	387
44.2	Variables . . . . .	387
44.3	Class FunctionEvaluator . . . . .	388
44.3.1	Methods . . . . .	388
44.3.2	Properties . . . . .	388
44.4	Class PreProcessor . . . . .	388
44.4.1	Methods . . . . .	389
44.4.2	Properties . . . . .	393
44.5	Class DumbPreProcessor . . . . .	393
44.5.1	Methods . . . . .	393
44.5.2	Properties . . . . .	393
<b>45</b>	<b>Module SCons.dblite . . . . .</b>	<b>395</b>
45.1	Functions . . . . .	395

---

45.2 Variables . . . . .	395
45.3 Class dblite . . . . .	395
45.3.1 Methods . . . . .	395
45.3.2 Properties . . . . .	396
<b>46 Module SCons.exitfuncs</b>	<b>397</b>
46.1 Functions . . . . .	397
46.2 Variables . . . . .	397



# 1 Package SCons

SCons

The main package for the SCons software construction utility. **Version:** 2.4.1

**Date:** 2015/11/09 03:25:05

## 1.1 Modules

- **Action:** SCons.Action  
(Section 2, p. 5)
- **Builder:** SCons.Builder  
(Section 3, p. 18)
- **CacheDir:** CacheDir support  
(Section 4, p. 30)
- **Conftest:** SCons.Conftest  
(Section 5, p. 33)
- **Debug:** SCons.Debug  
(Section 6, p. 38)
- **Defaults:** SCons.Defaults  
(Section 7, p. 40)
- **Environment:** SCons.Environment  
(Section 8, p. 45)
- **Errors:** SCons.Errors  
(Section 9, p. 74)
- **Executor:** SCons.Executor  
(Section 10, p. 82)
- **Job:** SCons.Job  
(Section 11, p. 94)
- **Memoize:** Memoizer  
(Section 12, p. 102)
- **Node:** SCons.Node  
(Section 13, p. 108)
  - **Alias:** scons.Node.Alias  
(Section 14, p. 131)
  - **FS:** scons.Node.FS  
(Section 15, p. 137)
  - **Python:** scons.Node.Python  
(Section 16, p. 185)
- **PathList:** SCons.PathList  
(Section 17, p. 191)
- **SConf:** SCons.SConf  
(Section 18, p. 192)
- **SConsign:** SCons.SConsign  
(Section 19, p. 208)
- **Scanner:** SCons.Scanner  
(Section 20, p. 215)
  - **C:** SCons.Scanner.C  
(Section 21, p. 231)
  - **D:** SCons.Scanner.D

- (Section 22, p. 234)
  - **Dir** (Section 23, p. 238)
  - **Fortran**: SCons.Scanner.Fortran  
(Section 24, p. 240)
  - **IDL**: SCons.Scanner.IDL  
(Section 25, p. 245)
  - **LaTeX**: SCons.Scanner.LaTeX  
(Section 26, p. 246)
  - **Prog** (Section 27, p. 253)
  - **RC**: SCons.Scanner.RC  
(Section 28, p. 254)
- **Script**: SCons.Script  
(Section 29, p. 255)
  - **Interactive**: SCons interactive mode  
(Section 30, p. 264)
  - **Main**: SCons.Script  
(Section 31, p. 267)
  - **SConscript'**: SCons.Script.SConscript  
(Section 32, p. 281)
- **Sig**: Place-holder for the old SCons.Sig module hierarchy  
(Section 33, p. 287)
- **Subst**: SCons.Subst  
(Section 34, p. 290)
- **Taskmaster**: Generic Taskmaster module for the SCons build engine.  
(Section 35, p. 301)
- **Util**: SCons.Util  
(Section 36, p. 312)
- **Variables**: engine.SCons.Variables  
(Section 37, p. 339)
  - **BoolVariable** (Section ??, p. ??)
  - **BoolVariable'**: engine.SCons.Variables.BoolVariable  
(Section 38, p. 343)
  - **EnumVariable** (Section ??, p. ??)
  - **EnumVariable'**: engine.SCons.Variables.EnumVariable  
(Section 39, p. 344)
  - **ListVariable** (Section ??, p. ??)
  - **ListVariable'**: engine.SCons.Variables.ListVariable  
(Section 40, p. 346)
  - **PackageVariable** (Section ??, p. ??)
  - **PackageVariable'**: engine.SCons.Variables.PackageVariable  
(Section 41, p. 347)
  - **PathVariable** (Section ??, p. ??)
  - **PathVariable'**: SCons.Variables.PathVariable  
(Section 42, p. 348)
- **Warnings**: SCons.Warnings  
(Section 43, p. 350)
- **cpp**: SCons C Pre-Processor module  
(Section 44, p. 387)
- **dblite** (Section 45, p. 395)
- **exitfuncs**: SCons.exitfuncs  
(Section 46, p. 397)

## 1.2 Variables

Name	Description
__build__	<b>Value:</b> 'rel_2.4.1:3453:73fef3ea0b0'
__buildsys__	<b>Value:</b> 'ubuntu1404-32bit'
__developer__	<b>Value:</b> 'bdbaddog'
__package__	<b>Value:</b> 'SCons'
__revision__	<b>Value:</b> 'src/engine/SCons/__init__.py rel_2.4.1:3453:73fef3ea0b0...

## 2 Module SCons.Action

### SCons.Action

This encapsulates information about executing any sort of action that can build one or more target Nodes (typically files) from one or more source Nodes (also typically files) given a specific Environment.

The base class here is ActionBase. The base class supplies just a few OO utility methods and some generic methods for displaying information about an Action in response to the various commands that control printing.

A second-level base class is `_ActionAction`. This extends ActionBase by providing the methods that can be used to show and perform an action. True Action objects will subclass `_ActionAction`; Action factory class objects will subclass ActionBase.

The heavy lifting is handled by subclasses for the different types of actions we might execute:

CommandAction CommandGeneratorAction FunctionAction ListAction

The subclasses supply the following public interface methods used by other modules:

**\_\_call\_\_()** THE public interface, "calling" an Action object executes the command or Python function. This also takes care of printing a pre-substitution command for debugging purposes.

**get\_contents()** Fetches the "contents" of an Action for signature calculation plus the varlist. This is what gets MD5 checksummed to decide if a target needs to be rebuilt because its action changed.

**genstring()** Returns a string representation of the Action *without* command substitution, but allows a CommandGeneratorAction to generate the right action based on the specified target, source and env. This is used by the Signature subsystem (through the Executor) to obtain an (imprecise) representation of the Action operation for informative purposes.

Subclasses also supply the following methods for internal use within this module:

**\_\_str\_\_()** Returns a string approximation of the Action; no variable substitution is performed.

**execute()** The internal method that really, truly, actually handles the execution of a command or Python function. This is used so that the `__call__()` methods can take care of displaying any pre-substitution representations, and *then* execute an action without worrying about the specific Actions involved.

**get\_presig()** Fetches the "contents" of a subclass for signature calculation. The varlist is added to this to produce the Action's contents.

**strfunction()** Returns a substituted string representation of the Action. This is used by the `_ActionAction.show()` command to display the command/function that will be executed to generate the target(s).

There is a related independent ActionCaller class that looks like a regular Action, and which serves as a wrapper for arbitrary functions that we want to let the user specify the arguments to now, but actually execute later (when an out-of-date check determines that it's needed to be executed, for example). Objects of this class are returned by an ActionFactory class that provides a `__call__()` method as a convenient way

for wrapping up the functions.

## 2.1 Functions

**rfile**(*n*)

**default\_exitstatfunc**(*s*)

**remove\_set\_lineno\_codes**(*x*)

**Action**(*act*, \**args*, \*\**kw*)

A factory for action objects.

**get\_default\_ENV**(*env*)

## 2.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Action.py rel_2.4.1:3453:73fef3ea0b0 2...
<code>print_actions</code>	<b>Value:</b> 1
<code>execute_actions</code>	<b>Value:</b> 1
<code>print_actions_presub</code>	<b>Value:</b> 0
<code>SET_LINENO</code>	<b>Value:</b> <code>dis.SET_LINENO</code>
<code>HAVE_ARGUMENT</code>	<b>Value:</b> <code>dis.HAVE_ARGUMENT</code>
<code>strip_quotes</code>	<b>Value:</b> <code>re.compile(r'^[\'"](.*)[\'"]\$')</code>
<code>default_ENV</code>	<b>Value:</b> None
<code>__package__</code>	<b>Value:</b> 'SCons'

## 2.3 Class *ActionBase*

object — **SCons.Action.ActionBase**

**Known Subclasses:** *SCons.Action.\_ActionAction*, *SCons.Action.CommandGeneratorAction*, *SCons.Action.ListAction*

Base class for all types of action objects that can be held by other objects (Builders, Executors, etc.) This provides the common methods for manipulating and combining those actions.

### 2.3.1 Methods

<code>__cmp__(self, other)</code>
-----------------------------------

<code>no_batch_key(self, env, target, source)</code>
--

<code>batch_key(self, env, target, source)</code>
---

<code>genstring(self, target, source, env)</code>
---

<code>get_contents(self, target, source, env)</code>
--

<code>__add__(self, other)</code>
-----------------------------------

<code>__radd__(self, other)</code>
------------------------------------

<code>presub_lines(self, env)</code>
--------------------------------------

<code>get_varlist(self, target, source, env, executor=None)</code>
--

<code>get_targets(self, env, executor)</code>
---

Returns the type of targets (\$TARGETS, \$CHANGED_TARGETS) used by this action.
---

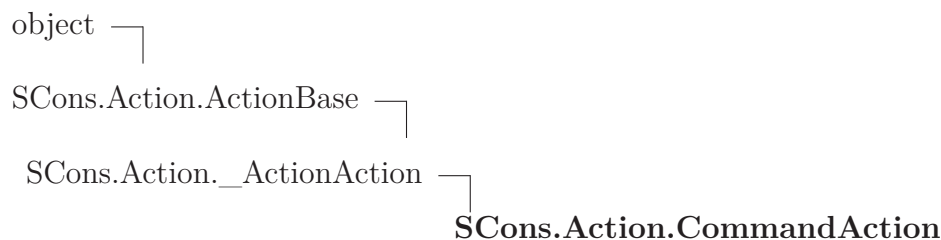
#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`,  
`__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`,  
`__sizeof__()`, `__str__()`, `__subclasshook__()`

### 2.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 2.4 Class *CommandAction*



**Known Subclasses:** *SCons.Action.LazyAction*

Class for command-execution actions.

### 2.4.1 Methods

**`__init__(self, cmd, **kw)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature Overrides:  
`object.__init__` `exitit`(inherited documentation)

**`__str__(self)`**

`str(x)` Overrides: `object.__str__` `exitit`(inherited documentation)

**`process(self, target, source, env, executor=None)`**

**`strfunction(self, target, source, env, executor=None)`**

**`execute(self, target, source, env, executor=None)`**

Execute a command action.

This will handle lists of commands as well as individual commands, because construction variable substitution may turn a single "command" into a list. This means that this class can actually handle lists of commands, even though that's not how we use it externally.

```
get_presig(self, target, source, env, executor=None)
```

Return the signature contents of this action’s command line.

This strips \$(-\$) and everything in between the string, since those parts don’t affect signatures.

```
get_implicit_deps(self, target, source, env, executor=None)
```

**Inherited from `SCons.Action._ActionAction`**

```
__call__(), print_cmd_line()
```

**Inherited from `SCons.Action.ActionBase` (Section 2.3)**

```
__add__(), __cmp__(), __radd__(), batch_key(), genstring(), get_contents(),
get_targets(), get_varlist(), no_batch_key(), presub_lines()
```

**Inherited from object**

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()
```

#### 2.4.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

## 2.5 Class `CommandGeneratorAction`



**Known Subclasses:** `SCons.Action.LazyAction`

Class for command-generator actions.



### 2.5.1 Methods

**\_\_init\_\_**(*self*, *generator*, *kw*)

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature Overrides: `object.__init__` `exitit`(inherited documentation)

**\_\_str\_\_**(*self*)

`str(x)` Overrides: `object.__str__` `exitit`(inherited documentation)

**batch\_key**(*self*, *env*, *target*, *source*)

Overrides: `SCons.Action.ActionBase.batch_key`

**genstring**(*self*, *target*, *source*, *env*, *executor*=None)

Overrides: `SCons.Action.ActionBase.genstring`

**\_\_call\_\_**(*self*, *target*, *source*, *env*, *exitstatfunc*=<class 'SCons.Action.\_null'>, *presub*=<class 'SCons.Action.\_null'>, *show*=<class 'SCons.Action.\_null'>, *execute*=<class 'SCons.Action.\_null'>, *chdir*=<class 'SCons.Action.\_null'>, *executor*=None)

**get\_presig**(*self*, *target*, *source*, *env*, *executor*=None)

Return the signature contents of this action's command line.

This strips `$(-$)` and everything in between the string, since those parts don't affect signatures.

**get\_implicit\_deps**(*self*, *target*, *source*, *env*, *executor*=None)

**get\_varlist**(*self*, *target*, *source*, *env*, *executor*=None)

Overrides: `SCons.Action.ActionBase.get_varlist`

**get\_targets**(*self*, *env*, *executor*)

Returns the type of targets (`$TARGETS`, `$CHANGED_TARGETS`) used by this action. Overrides: `SCons.Action.ActionBase.get_targets` `exitit`(inherited documentation)

***Inherited from SCons.Action.ActionBase(Section 2.3)***

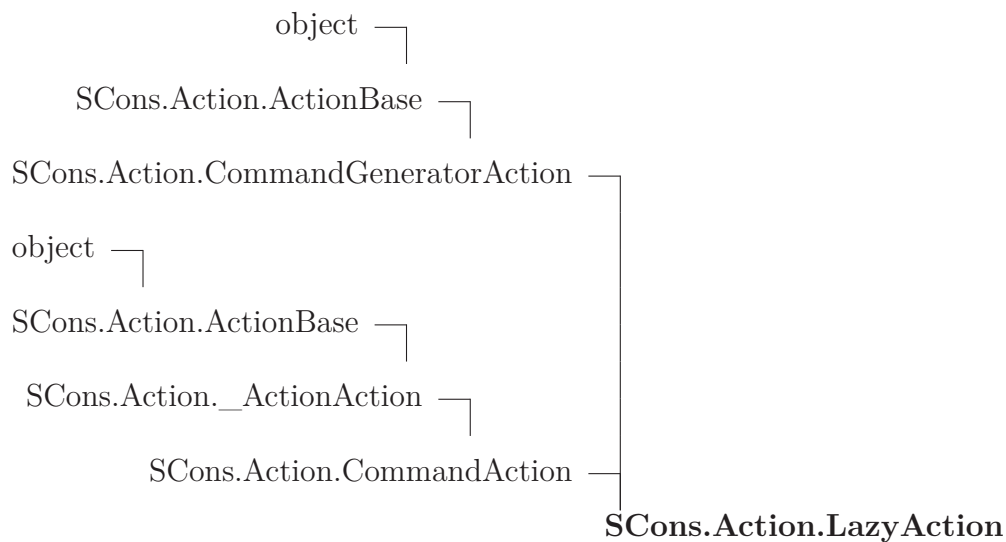
`__add__()`, `__cmp__()`, `__radd__()`, `get_contents()`, `no_batch_key()`, `pre_sub_lines()`

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

**2.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**2.6 Class LazyAction****2.6.1 Methods**

<code>__init__(self, var, kw)</code>
<code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)

<code>get_parent_class(self, env)</code>
--

<code>__call__(self, target, source, env, *args, **kw)</code>
---

Overrides: <code>SCons.Action._ActionAction.__call__</code>
---

<code>get_presig(self, target, source, env)</code>
--

Return the signature contents of this action's command line.
--

This strips \$(-\$) and everything in between the string, since those parts don't affect signatures. Overrides: <code>SCons.Action.CommandAction.get_presig</code> <code>exitit</code> (inherited documentation)
---

<code>get_varlist(self, target, source, env, executor=None)</code>
--

Overrides: <code>SCons.Action.ActionBase.get_varlist</code>
---

***Inherited from `SCons.Action.CommandGeneratorAction` (Section 2.5)***

`__str__()`, `batch_key()`, `genstring()`, `get_implicit_deps()`, `get_targets()`

***Inherited from `SCons.Action.CommandAction` (Section 2.4)***

`execute()`, `process()`, `strfunction()`

***Inherited from `SCons.Action._ActionAction`***

`print_cmd_line()`

***Inherited from `SCons.Action.ActionBase` (Section 2.3)***

`__add__()`, `__cmp__()`, `__radd__()`, `get_contents()`, `no_batch_key()`, `pre_sub_lines()`

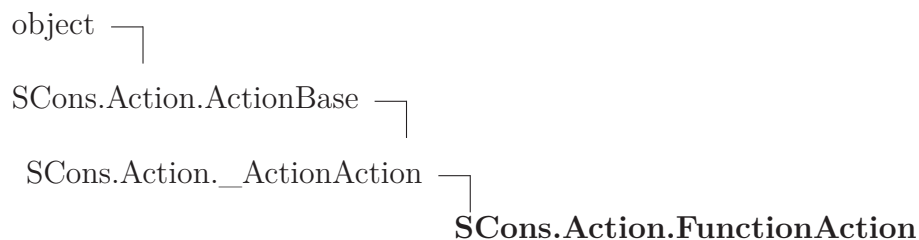
***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

## 2.6.2 Properties

Name	Description
<code>__class__</code>	<i>Inherited from object</i>

## 2.7 Class *FunctionAction*



Class for Python function actions.

### 2.7.1 Methods

```
__init__(self, execfunction, kw)
```

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature Overrides:  
*object*.**\_\_init\_\_** `exitit`(inherited documentation)

```
function_name(self)
```

```
strfunction(self, target, source, env, executor=None)
```

```
__str__(self)
```

`str(x)` Overrides: *object*.**\_\_str\_\_** `exitit`(inherited documentation)

```
execute(self, target, source, env, executor=None)
```

```
get_presig(self, target, source, env)
```

Return the signature contents of this callable action.

```
get_implicit_deps(self, target, source, env)
```

*Inherited from SCons.Action.\_ActionAction*

```
__call__(self), print_cmd_line(self)
```

*Inherited from SCons.Action.ActionBase*(Section 2.3)

```
__add__(self), __cmp__(self), __radd__(self), batch_key(self), genstring(self), get_contents(self),  

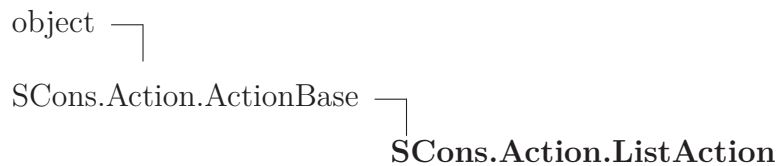
get_targets(self), get_varlist(self), no_batch_key(self), presub_lines(self)
```

***Inherited from object***

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()
```

**2.7.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**2.8 Class ListAction**

Class for lists of other actions.

**2.8.1 Methods**

<b><code>__init__(self, actionlist)</code></b> x. <code>__init__</code> (...) initializes x; see help(type(x)) for signature Overrides: object. <code>__init__</code> extit(inherited documentation)
<b><code>genstring(self, target, source, env)</code></b> Overrides: SCons.Action.ActionBase.genstring
<b><code>__str__(self)</code></b> str(x) Overrides: object. <code>__str__</code> extit(inherited documentation)
<b><code>presub_lines(self, env)</code></b> Overrides: SCons.Action.ActionBase.presub_lines

```
get_presig(self, target, source, env)
```

Return the signature contents of this action list.

Simple concatenation of the signatures of the elements.

```
__call__(self, target, source, env, exitstatfunc=<class
'SCons.Action._null'>, presub=<class 'SCons.Action._null'>,
show=<class 'SCons.Action._null'>, execute=<class
'SCons.Action._null'>, chdir=<class 'SCons.Action._null'>,
executor=None)
```

```
get_implicit_deps(self, target, source, env)
```

```
get_varlist(self, target, source, env, executor=None)
```

Overrides: SCons.Action.ActionBase.get\_varlist

### *Inherited from SCons.Action.ActionBase(Section 2.3)*

```
__add__(), __cmp__(), __radd__(), batch_key(), get_contents(), get_targets(),
no_batch_key()
```

### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()
```

## 2.8.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 2.9 Class ActionCaller

```
object └─ SCons.Action.ActionCaller
```

A class for delaying calling an Action function with specific (positional and keyword) arguments until the Action is actually executed.

This class looks to the rest of the world like a normal Action object, but what it's really doing is hanging on to the arguments until we have a target, source and env to use for the expansion.

### 2.9.1 Methods

```
__init__(self, parent, args, kw)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides: object.\_\_init\_\_ exitit(inherited documentation)

```
get_contents(self, target, source, env)
```

```
subst(self, s, target, source, env)
```

```
subst_args(self, target, source, env)
```

```
subst_kw(self, target, source, env)
```

```
__call__(self, target, source, env, executor=None)
```

```
strfunction(self, target, source, env)
```

```
__str__(self)
```

str(x) Overrides: object.\_\_str\_\_ exitit(inherited documentation)

### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()
```

### 2.9.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 2.10 Class ActionFactory



A factory class that will wrap up an arbitrary function as an SCons-executable Action object.

The real heavy lifting here is done by the ActionCaller class. We just collect the (positional and keyword) arguments that we're called with and give them to the ActionCaller object we create, so it can hang onto them until it needs them.

### 2.10.1 Methods

```
__init__(self, actfunc, strfunc, convert=<function <lambda> at
0xb6b48c34>)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides:  
object.\_\_init\_\_ extit(inherited documentation)

```
__call__(self, *args, **kw)
```

### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 2.10.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	



### 3 Module *SCons.Builder*

*SCons.Builder*

Builder object subsystem.

A Builder object is a callable that encapsulates information about how to execute actions to create a target Node (file) from source Nodes (files), and how to create those dependencies for tracking.

The main entry point here is the `Builder()` factory method. This provides a procedural interface that creates the right underlying Builder object based on the keyword arguments supplied and the types of the arguments.

The goal is for this external interface to be simple enough that the vast majority of users can create new Builders as necessary to support building new types of files in their configurations, without having to dive any deeper into this subsystem.

The base class here is `BuilderBase`. This is a concrete base class which does, in fact, represent the Builder objects that we (or users) create.

There is also a proxy that looks like a Builder:

`CompositeBuilder`

This proxies for a Builder with an action that is actually a dictionary that knows how to map file suffixes to a specific action. This is so that we can invoke different actions (compilers, compile options) for different flavors of source files.

Builders and their proxies have the following public interface methods used by other modules:

`--call--()`

THE public interface. Calling a Builder object (with the use of internal helper methods) sets up the target and source dependencies, appropriate mapping to a specific action, and the environment manipulation necessary for overridden construction variable. This also takes care of warning about possible mistakes in keyword arguments.

`add_emitter()`

Adds an emitter for a specific file suffix, used by some Tool modules to specify that (for example) a yacc invocation on a .y can create a .h *and* a .c file.

`add_action()`

Adds an action for a specific file suffix, heavily used by Tool modules to add their specific action(s) for turning a source file into an object file to the global static and shared object file Builders.

There are the following methods for internal use within this module:

`_execute()`

The internal method that handles the heavily lifting when a Builder is called. This is used so that the `__call__()` methods can set up warning about possible mistakes in keyword-argument overrides, and *then* execute all of the steps necessary so that the warnings only occur once.

`get_name()`

Returns the Builder's name within a specific Environment, primarily used to try to return helpful information in error messages.

`adjust_suffix()`

`get_prefix()`

`get_suffix()`

`get_src_suffix()`

`set_src_suffix()`

Miscellaneous stuff for handling the prefix and suffix manipulation we use in turning source file names into target file names.

### 3.1 Functions

<code>match_splitext(<i>path</i>, <i>suffixes</i>=[])</code>
--

<code>Builder(**<i>kw</i>)</code>
-----------------------------------

A factory for builder objects.
--------------------------------

**is\_a\_Builder**(*obj*)

"Returns True if the specified obj is one of our Builder classes.

The test is complicated a bit by the fact that CompositeBuilder is a proxy, not a subclass of BuilderBase.

## 3.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Builder.py rel_2.4.1:3453:73fef3ea0b0 ...
<code>misleading_keywords</code>	<b>Value:</b> {'sources': 'source', 'targets': 'target'}
<code>__package__</code>	<b>Value:</b> 'SCons'

## 3.3 Class DictCmdGenerator

UserDict.UserDict

SCons.Util.OrderedDict

SCons.Util.Selector

**SCons.Builder.DictCmdGenerator**

This is a callable class that can be used as a command generator function. It holds on to a dictionary mapping file suffixes to Actions. It uses that dictionary to return the proper action based on the file suffix of the source file.

### 3.3.1 Methods

**\_\_init\_\_**(*self*, *dict*=None, *source\_ext\_match*=1)

Overrides: UserDict.UserDict.\_\_init\_\_

**src\_suffixes**(*self*)

<b>add_action</b> ( <i>self, suffix, action</i> )
---

Add a suffix-action pair to the mapping.
--

<b>__call__</b> ( <i>self, target, source, env, for_signature</i> )
---

Overrides: SCons.Util.Selector.__call__
---

**Inherited from SCons.Util.OrderedDict(Section 36.10)**

\_\_delitem\_\_(), \_\_setitem\_\_(), clear(), copy(), items(), keys(), popitem(), setdefault(), update(), values()

**Inherited from UserDict.UserDict**

\_\_cmp\_\_(), \_\_contains\_\_(), \_\_getitem\_\_(), \_\_len\_\_(), \_\_repr\_\_(), fromkeys(), get(), has\_key(), iteritems(), iterkeys(), itervalues(), pop()

### 3.3.2 Class Variables

Name	Description
<i>Inherited from UserDict.UserDict</i>	
__hash__	

## 3.4 Class CallableSelector

UserDict.UserDict └─

SCons.Util.OrderedDict └─

SCons.Util.Selector └─

**SCons.Builder.CallableSelector**

A callable dictionary that will, in turn, call the value it finds if it can.

### 3.4.1 Methods

<b>__call__</b> ( <i>self, env, source</i> )
--

Overrides: SCons.Util.Selector.__call__
---

**Inherited from SCons.Util.OrderedDict(Section 36.10)**

`__delitem__()`, `__init__()`, `__setitem__()`, `clear()`, `copy()`, `items()`, `keys()`,  
`popitem()`, `setdefault()`, `update()`, `values()`

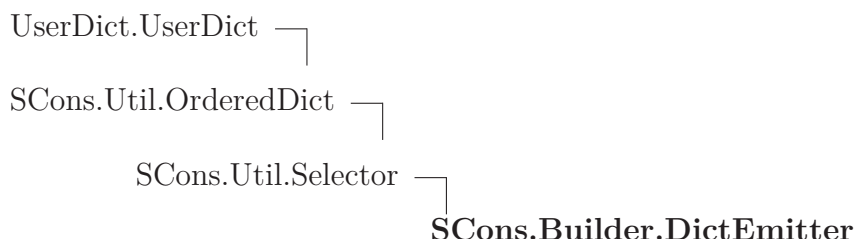
***Inherited from UserDict.UserDict***

`__cmp__()`, `__contains__()`, `__getitem__()`, `__len__()`, `__repr__()`, `fromkeys()`,  
`get()`, `has_key()`, `iteritems()`, `iterkeys()`, `itervalues()`, `pop()`

### 3.4.2 Class Variables

Name	Description
<i>Inherited from UserDict.UserDict</i>	
<code>__hash__</code>	

## 3.5 Class DictEmitter



A callable dictionary that maps file suffixes to emitters. When called, it finds the right emitter in its dictionary for the suffix of the first source file, and calls that emitter to get the right lists of targets and sources to return. If there's no emitter for the suffix in its dictionary, the original target and source are returned.

### 3.5.1 Methods

<code>__call__(self, target, source, env)</code>
Overrides: SCons.Util.Selector. <code>__call__</code>

***Inherited from SCons.Util.OrderedDict(Section 36.10)***

`__delitem__()`, `__init__()`, `__setitem__()`, `clear()`, `copy()`, `items()`, `keys()`,  
`popitem()`, `setdefault()`, `update()`, `values()`

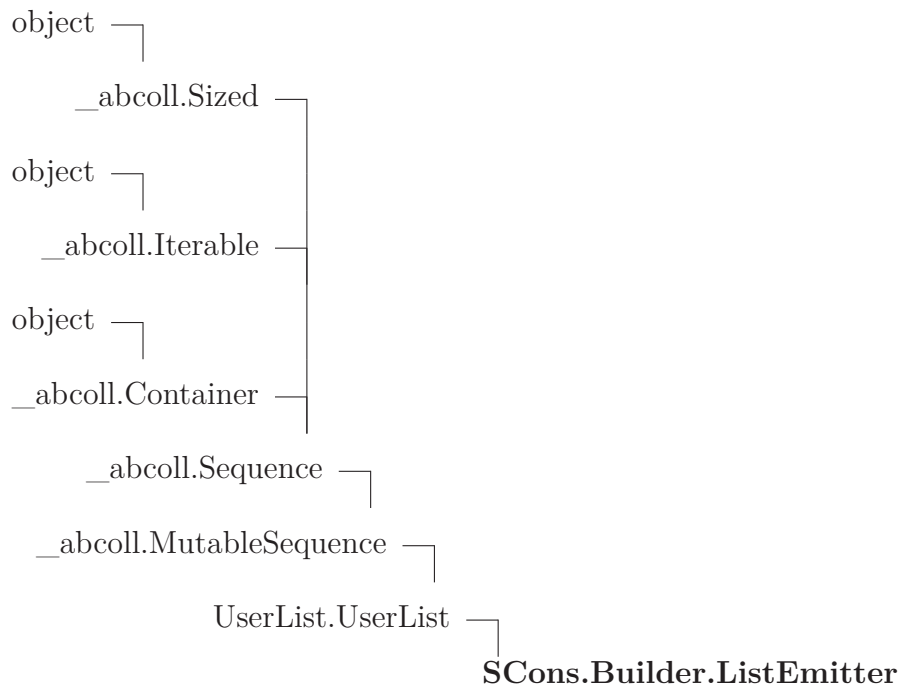
***Inherited from UserDict.UserDict***

`__cmp__()`, `__contains__()`, `__getitem__()`, `__len__()`, `__repr__()`, `fromkeys()`,  
`get()`, `has_key()`, `iteritems()`, `iterkeys()`, `itervalues()`, `pop()`

### 3.5.2 Class Variables

Name	Description
<i>Inherited from UserDict.UserDict</i>	
<code>__hash__</code>	

## 3.6 Class ListEmitter



A callable list of emitters that calls each in sequence, returning the result.

### 3.6.1 Methods

<code>__call__(self, target, source, env)</code>
--

*Inherited from UserList.UserList*

`__add__()`, `__cmp__()`, `__contains__()`, `__delitem__()`, `__delslice__()`,  
`__eq__()`, `__ge__()`, `__getitem__()`, `__getslice__()`, `__gt__()`, `__iadd__()`,  
`__imul__()`, `__init__()`, `__le__()`, `__len__()`, `__lt__()`, `__mul__()`, `__ne__()`,  
`__radd__()`, `__repr__()`, `__rmul__()`, `__setitem__()`, `__setslice__()`, `append()`, `count()`, `extend()`, `index()`, `insert()`, `pop()`, `remove()`, `reverse()`, `sort()`

*Inherited from \_abcoll.Sequence*

`__iter__()`, `__reversed__()`

*Inherited from `__abcoll.Sized`*

`__subclasshook__()`

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`,  
`__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__str__()`

### 3.6.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

### 3.6.3 Class Variables

Name	Description
<i>Inherited from <code>UserList.UserList</code></i> <code>__abstractmethods__</code> , <code>__hash__</code>	

## 3.7 Class OverrideWarner

UserDict.UserDict —  
**SCons.Builder.OverrideWarner**

A class for warning about keyword arguments that we use as overrides in a Builder call.

This class exists to handle the fact that a single Builder call can actually invoke multiple builders. This class only emits the warnings once, no matter how many Builders are invoked.

### 3.7.1 Methods

<code>__init__(self, dict)</code> Overrides: UserDict.UserDict.__init__
<code>warn(self)</code>

*Inherited from `UserDict.UserDict`*

```
__cmp__(), __contains__(), __delitem__(), __getitem__(), __len__(),
__repr__(), __setitem__(), clear(), copy(), fromkeys(), get(), has_key(), items(),
iteritems(), iterkeys(), itervalues(), keys(), pop(), popitem(), setdefault(), update(),
values()
```

### 3.7.2 Class Variables

Name	Description
<i>Inherited from UserDict.UserDict</i>	
__hash__	

## 3.8 Class EmitterProxy



This is a callable class that can act as a Builder emitter. It holds on to a string that is a key into an Environment dictionary, and will look there at actual build time to see if it holds a callable. If so, we will call that as the actual emitter.

### 3.8.1 Methods

```
__init__(self, var)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides:  
object.\_\_init\_\_ exitit(inherited documentation)

```
__call__(self, target, source, env)
```

```
__cmp__(self, other)
```

#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 3.8.2 Properties



Name	Description
<i>Inherited from object</i> __class__	

### 3.9 Class BuilderBase



Base class for Builders, objects that create output nodes (files) from input nodes (files).

#### 3.9.1 Methods

```
__init__(self, action=None, prefix='', suffix='', src_suffix='',
target_factory=None, source_factory=None, target_scanner=None,
source_scanner=None, emitter=None, multi=0, env=None, single_source=0,
name=None, chdir=<class 'SCons.Builder._Null'>, is_explicit=1,
src_builder=None, ensure_suffix=False, **overrides)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides:  
object.\_\_init\_\_ extit(inherited documentation)

```
__nonzero__(self)
```

```
get_name(self, env)
```

Attempts to get the name of the Builder.

Look at the BUILDERS variable of env, expecting it to be a dictionary containing this Builder, and return the key of the dictionary. If there's no key, then return a directly-configured name (if there is one) or the name of the class (by default).

```
__cmp__(self, other)
```

```
splitext(self, path, env=None)
```

```
__call__(self, env, target=None, source=None, chdir=<class  
'SCons.Builder._Null'>, **kw)
```

```
adjust__suffix(self, suff)
```

```
get__prefix(self, env, sources=[])
```

```
set__suffix(self, suffix)
```

```
get__suffix(self, env, sources=[])
```

```
set__src__suffix(self, src_suffix)
```

```
get__src__suffix(self, env)
```

Get the first `src_suffix` in the list of `src_suffixes`.

```
add__emitter(self, suffix, emitter)
```

Add a suffix-emitter mapping to this Builder.

This assumes that emitter has been initialized with an appropriate dictionary type, and will throw a `TypeError` if not, so the caller is responsible for knowing that this is an appropriate method to call for the Builder in question.

```
add__src__builder(self, builder)
```

Add a new Builder to the list of `src_builders`.

This requires wiping out cached values so that the computed lists of source suffixes get re-calculated.

```
src__builder__sources(self, env, source, overwarn={})
```

<b>get_src_builders</b> ( <i>self</i> , <i>env</i> )
--

Returns the list of source Builders for this Builder.

This exists mainly to look up Builders referenced as strings in the 'BUILDER' variable of the construction environment and cache the result.

<b>subst_src_suffixes</b> ( <i>self</i> , <i>env</i> )
--

The suffix list may contain construction variable expansions, so we have to evaluate the individual strings. To avoid doing this over and over, we memoize the results for each construction environment.

<b>src_suffixes</b> ( <i>self</i> , <i>env</i> )
--

Returns the list of source suffixes for all src\_builders of this Builder.

This is essentially a recursive descent of the src\_builder "tree." (This value isn't cached because there may be changes in a src\_builder many levels deep that we can't see.)

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

### 3.9.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

### 3.10 Class CompositeBuilder



A Builder Proxy whose main purpose is to always have a DictCmdGenerator as its action, and to provide access to the DictCmdGenerator's `add_action()` method.

#### 3.10.1 Methods

```
__init__(self, builder, cmdgen)
```

Wrap an object as a Proxy object. Overrides: `object.__init__`. extit(inherited documentation)

```
__call__(...)
```

A Python Descriptor class that delegates attribute fetches to an underlying wrapped subject of a Proxy. Typical use:

```
class Foo(Proxy): __str__ = Delegate('__str__')
```

```
add_action(self, suffix, action)
```

*Inherited from SCons.Util.Proxy (Section 36.5)*

```
__cmp__(), __getattr__(), get()
```

*Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

#### 3.10.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 4 Module **SCons.CacheDir**

CacheDir support

### 4.1 Functions

<b>CacheRetrieveFunc</b> ( <i>target, source, env</i> )
---

<b>CacheRetrieveString</b> ( <i>target, source, env</i> )
---

<b>CachePushFunc</b> ( <i>target, source, env</i> )
---

### 4.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/CacheDir.py rel_2.4.1:3453:73fef3d3ea0b0...
<code>__doc__</code>	<b>Value:</b> ...
<code>cache_enabled</code>	<b>Value:</b> True
<code>cache_debug</code>	<b>Value:</b> False
<code>cache_force</code>	<b>Value:</b> False
<code>cache_show</code>	<b>Value:</b> False
<code>cache_readonly</code>	<b>Value:</b> False
<code>CacheRetrieve</code>	<b>Value:</b> SCons.Action.Action(CacheRetrieveFunc, CacheRetrieveString)
<code>CacheRetrieveSilent</code>	<b>Value:</b> SCons.Action.Action(CacheRetrieveFunc, None)
<code>CachePush</code>	<b>Value:</b> SCons.Action.Action(CachePushFunc, None)
<code>__package__</code>	<b>Value:</b> 'SCons'

### 4.3 Class **CacheDir**

```

object └─ SCons.CacheDir.CacheDir

```

## 4.3.1 Methods

**\_\_init\_\_**(*self*, *path*)

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature Overrides: `object.__init__` `exitit`(inherited documentation)

**CacheDebug**(*self*, *fmt*, *target*, *cachefile*)

**is\_enabled**(*self*)

**is\_readonly**(*self*)

**cachepath**(*self*, *node*)

**retrieve**(*self*, *node*)

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in `built()`.

Note that there's a special trick here with the execute flag (one that's not normally done for other actions). Basically if the user requested a `no_exec` (`-n`) build, then `SCons.Action.execute_actions` is set to 0 and when any action is called, it does its showing but then just returns zero instead of actually calling the action execution operation. The problem for caching is that if the file does NOT exist in cache then the `CacheRetrieveString` won't return anything to show for the task, but the `Action.__call__` won't call `CacheRetrieveFunc`; instead it just returns zero, which makes the code below think that the file *was* successfully retrieved from the cache, therefore it doesn't do any subsequent building. However, the `CacheRetrieveString` didn't print anything because it didn't actually exist in the cache, and no more build actions will be performed, so the user just sees nothing. The fix is to tell `Action.__call__` to always execute the `CacheRetrieveFunc` and then have the latter explicitly check `SCons.Action.execute_actions` itself.

**push**(*self*, *node*)

**push\_if\_forced**(*self*, *node*)

*Inherited from object*

**\_\_delattr\_\_**(), **\_\_format\_\_**(), **\_\_getattr\_\_**(), **\_\_hash\_\_**(), **\_\_new\_\_**(),

`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

#### 4.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 5 Module **SCons.Conftest**

SCons.Conftest

Autoconf-like configuration support; low level implementation of tests.

### 5.1 Functions

**CheckBuilder**(*context*, *text*=None, *language*=None)

Configure check to see if the compiler works. Note that this uses the current value of compiler and linker flags, make sure \$CFLAGS, \$CPPFLAGS and \$LIBS are set correctly. "language" should be "C" or "C++" and is used to select the compiler. Default is "C". "text" may be used to specify the code to be build. Returns an empty string for success, an error message for failure.

**CheckCC**(*context*)

Configure check for a working C compiler.

This checks whether the C compiler, as defined in the \$CC construction variable, can compile a C source file. It uses the current \$CCCOM value too, so that it can test against non working flags.

**CheckSHCC**(*context*)

Configure check for a working shared C compiler.

This checks whether the C compiler, as defined in the \$SHCC construction variable, can compile a C source file. It uses the current \$SHCCCOM value too, so that it can test against non working flags.



**CheckCXX**(*context*)

Configure check for a working CXX compiler.

This checks whether the CXX compiler, as defined in the \$CXX construction variable, can compile a CXX source file. It uses the current \$CXXCOM value too, so that it can test against non working flags.

**CheckSHCXX**(*context*)

Configure check for a working shared CXX compiler.

This checks whether the CXX compiler, as defined in the \$SHCXX construction variable, can compile a CXX source file. It uses the current \$SHCXXCOM value too, so that it can test against non working flags.

**CheckFunc**(*context*, *function\_name*, *header=***None**, *language=***None**)

Configure check for a function "*function\_name*". "*language*" should be "C" or "C++" and is used to select the compiler. Default is "C". Optional "*header*" can be defined to define a function prototype, include a header file or anything else that comes before `main()`. Sets `HAVE_`*function\_name* in `context.havedict` according to the result. Note that this uses the current value of compiler and linker flags, make sure \$CFLAGS, \$CPPFLAGS and \$LIBS are set correctly. Returns an empty string for success, an error message for failure.

**CheckHeader**(*context*, *header\_name*, *header=***None**, *language=***None**, *include\_quotes=***None**)

Configure check for a C or C++ header file "*header\_name*". Optional "*header*" can be defined to do something before including the header file (unusual, supported for consistency). "*language*" should be "C" or "C++" and is used to select the compiler. Default is "C". Sets `HAVE_`*header\_name* in `context.havedict` according to the result. Note that this uses the current value of compiler and linker flags, make sure \$CFLAGS and \$CPPFLAGS are set correctly. Returns an empty string for success, an error message for failure.

---

**CheckType**(*context*, *type\_name*, *fallback*=None, *header*=None, *language*=None)

---

Configure check for a C or C++ type "*type\_name*". Optional "*header*" can be defined to include a header file. "*language*" should be "C" or "C++" and is used to select the compiler. Default is "C". Sets HAVE\_*type\_name* in *context*.havedict according to the result. Note that this uses the current value of compiler and linker flags, make sure \$CFLAGS, \$CPPFLAGS and \$LIBS are set correctly. Returns an empty string for success, an error message for failure.

---

**CheckTypeSize**(*context*, *type\_name*, *header*=None, *language*=None, *expect*=None)

---

This check can be used to get the size of a given type, or to check whether the type is of expected size.

**Arguments:**

- **type** (**str**)  
the type to check
- **includes** (**sequence**)  
list of headers to include in the test code before testing the type
- **language** (**str**)  
'C' or 'C++'
- **expect** (**int**)  
if given, will test whether the type has the given number of bytes.  
If not given, will automatically find the size.

**Returns:**

- status** (**int**)  
0 if the check failed, or the found size of the type if the check succeeded.

**CheckDeclaration**(*context*, *symbol*, *includes*=None, *language*=None)

Checks whether symbol is declared.

Use the same test as autoconf, that is test whether the symbol is defined as a macro or can be used as an r-value.

**Arguments:****symbol** (**str**)

the symbol to check

**includes** (**str**)

Optional "header" can be defined to include a header file.

**language** (**str**)

only C and C++ supported.

**Returns:****status** (**bool**)

True if the check failed, False if succeeded.

**CheckLib**(*context*, *libs*, *func\_name*=None, *header*=None, *extra\_libs*=None, *call*=None, *language*=None, *autoadd*=1, *append*=True)

Configure check for a C or C++ libraries "libs". Searches through the list of libraries, until one is found where the test succeeds. Tests if "func\_name" or "call" exists in the library. Note: if it exists in another library the test succeeds anyway! Optional "header" can be defined to include a header file. If not given a default prototype for "func\_name" is added. Optional "extra\_libs" is a list of library names to be added after "lib\_name" in the build command. To be used for libraries that "lib\_name" depends on. Optional "call" replaces the call to "func\_name" in the test code. It must consist of complete C statements, including a trailing ";". Both "func\_name" and "call" arguments are optional, and in that case, just linking against the libs is tested. "language" should be "C" or "C++" and is used to select the compiler. Default is "C". Note that this uses the current value of compiler and linker flags, make sure \$CFLAGS, \$CPPFLAGS and \$LIBS are set correctly. Returns an empty string for success, an error message for failure.

**CheckProg**(*context*, *prog\_name*)

Configure check for a specific program.

Check whether program *prog\_name* exists in path. If it is found, returns the path for it, otherwise returns *None*.

## 5.2 Variables

Name	Description
LogInputFiles	<b>Value:</b> 1
LogErrorMessages	<b>Value:</b> 1
__package__	<b>Value:</b> 'SCons'

## 6 Module SCons.Debug

SCons.Debug

Code for debugging SCons internal things. Shouldn't be needed by most users.

### 6.1 Functions

```
logInstanceCreation(instance, name=None)
```

```
string_to_classes(s)
```

```
fetchLoggedInstances(classes='*')
```

```
countLoggedInstances(classes, file=sys.stderr)
```

```
listLoggedInstances(classes, file=sys.stderr)
```

```
dumpLoggedInstances(classes, file=sys.stderr)
```

```
memory()
```

```
caller_stack()
```

```
caller_trace(back=0)
```

```
dump_caller_counts(file=sys.stderr)
```

```
func_shorten(func_tuple)
```

```
Trace(msg, file=None, mode='w', tstamp=None)
```

Write a trace message to a file. Whenever a file is specified, it becomes the default for the next call to Trace().

### 6.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Debug.py rel_2.4.1:3453:73fef3ea0b0 20...
<code>track_instances</code>	<b>Value:</b> False
<code>tracked_classes</code>	<b>Value:</b> {}
<code>caller_bases</code>	<b>Value:</b> {}
<code>caller_dicts</code>	<b>Value:</b> {}
<code>shorten_list</code>	<b>Value:</b> [('/scons/SCons/', 1), ('/src/engine/SCons/', 1), ('usr/...
<code>TraceFP</code>	<b>Value:</b> {}
<code>TraceDefault</code>	<b>Value:</b> '/dev/tty'
<code>TimeStampDefault</code>	<b>Value:</b> None
<code>StartTime</code>	<b>Value:</b> 1447068420.46
<code>PreviousTime</code>	<b>Value:</b> 1447068420.46
<code>__package__</code>	<b>Value:</b> 'SCons'

## 7 Module *SCons.Defaults*

### *SCons.Defaults*

Builders and other things for the local site. Here's where we'll duplicate the functionality of *autoconf* until we move it into the installation procedure or use something like *qmconf*.

The code that reads the registry to find MSVC components was borrowed from *distutils.msvccompiler*.

### 7.1 Functions

**DefaultEnvironment**(*\*args, \*\*kw*)

Initial public entry point for creating the default construction Environment.

After creating the environment, we overwrite our name (*DefaultEnvironment*) with the *\_fetch\_DefaultEnvironment()* function, which more efficiently returns the initialized default construction environment without checking for its existence.

(This function still exists with its *\_default\_check* because someone else (*cough* *Script/\_\_init\_\_.py* *cough*) may keep a reference to this function. So we can't use the fully functional idiom of having the name originally be a something that *only* creates the construction environment and then overwrites the name.)

**StaticObjectEmitter**(*target, source, env*)

**SharedObjectEmitter**(*target, source, env*)

**SharedFlagChecker**(*source, target, env*)

**get\_paths\_str**(*dest*)

**chmod\_func**(*dest, mode*)

**chmod\_strfunc**(*dest, mode*)

**copy\_\_func**(*dest*, *src*, *symlinks*=True)

If symlinks (is true), then a symbolic link will be shallow copied and recreated as a symbolic link; otherwise, copying a symbolic link will be equivalent to copying the symbolic link's final target regardless of symbolic link depth.

**delete\_\_func**(*dest*, *must\_exist*=0)

**delete\_\_strfunc**(*dest*, *must\_exist*=0)

**mkdir\_\_func**(*dest*)

**move\_\_func**(*dest*, *src*)

**touch\_\_func**(*dest*)

**processDefines**(*defs*)

process defines, resolving strings, lists, dictionaries, into a list of strings

## 7.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Defaults.py rel_2.4.1:3453:73fef3ea0b0...
SharedCheck	<b>Value:</b> SCons.Action.Action(SharedFlagChecker, None)
CScan	<b>Value:</b> SCons.Defaults.CScan
DScan	<b>Value:</b> SCons.Tool.DScanner
LaTeXScan	<b>Value:</b> SCons.Tool.LaTeXScanner
ObjSourceScan	<b>Value:</b> SCons.Tool.SourceFileScanner
ProgScan	<b>Value:</b> SCons.Tool.ProgramScanner
DirScanner	<b>Value:</b> SCons.Defaults.DirScanner
DirEntryScanner	<b>Value:</b> SCons.Scanner.Dir.DirEntryScanner()
CAction	<b>Value:</b> SCons.Action.Action("\$CCCOM", "\$CCCOMSTR")

*continued on next page*



Name	Description
ShCAction	<b>Value:</b> SCons.Action.Action("\$SHCCCOM", "\$SHCCCOMSTR")
CXXAction	<b>Value:</b> SCons.Action.Action("\$CXXCOM", "\$CXXCOMSTR")
ShCXXAction	<b>Value:</b> SCons.Action.Action("\$SHCXXCOM", "\$SHCXXCOMSTR")
DAction	<b>Value:</b> SCons.Action.Action("\$DCOM", "\$DCOMSTR")
ShDAction	<b>Value:</b> SCons.Action.Action("\$SHDCOM", "\$SHDCOMSTR")
ASAction	<b>Value:</b> SCons.Action.Action("\$ASCOM", "\$ASCOMSTR")
ASPPAction	<b>Value:</b> SCons.Action.Action("\$ASPPCOM", "\$ASPPCOMSTR")
LinkAction	<b>Value:</b> SCons.Action.Action("\$LINKCOM", "\$LINKCOMSTR")
ShLinkAction	<b>Value:</b> SCons.Action.Action("\$SHLINKCOM", "\$SHLINKCOMSTR")
LdModuleLinkAction	<b>Value:</b> SCons.Action.Action("\$LDMODULECOM", "\$LDMODULECOMSTR")
Chmod	<b>Value:</b> SCons.Defaults.Chmod
Copy	<b>Value:</b> SCons.Defaults.Copy
Delete	<b>Value:</b> SCons.Defaults.Delete
Mkdir	<b>Value:</b> SCons.Defaults.Mkdir
Move	<b>Value:</b> SCons.Defaults.Move
Touch	<b>Value:</b> SCons.Defaults.Touch
ConstructionEnvironment	<b>Value:</b> {'BUILDERS': {}, 'CONFIGUREDIRENTRY': '#/.sconf_temp', 'CONFIG...}
__package__	<b>Value:</b> 'SCons'

### 7.3 Class NullCmdGenerator

object —  
**SCons.Defaults.NullCmdGenerator**

This is a callable class that can be used in place of other command generators if you don't want them to do anything.

The `__call__` method for this class simply returns the thing you instantiated it with.

Example usage: `env["DO_NOthing"] = NullCmdGenerator env["LINKCOM"] = "${DO_NOthing('L`

```
$SOURCES $TARGET'})}"
```

### 7.3.1 Methods

```
__init__(self, cmd)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides:  
object.\_\_init\_\_ exitit(inherited documentation)

```
__call__(self, target, source, env, for_signature=None)
```

#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 7.3.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 7.4 Class Variable\_Method\_Caller

```
object └─ SCons.Defaults.Variable_Method_Caller
```

A class for finding a construction variable on the stack and calling one of its methods.

We use this to support "construction variables" in our string eval()s that actually stand in for methods--specifically, use of "RDirs" in call to \_\_concat that should actually execute the "TARGET.RDirs" method. (We used to support this by creating a little "build dictionary" that mapped RDirs to the method, but this got in the way of Memoizing construction environments, because we had to create new environment objects to hold the variables.)

### 7.4.1 Methods

<code>__init__(self, variable, method)</code>
---

<code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> extit(inherited documentation)
--

<code>__call__(self, *args, **kw)</code>
--

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

### 7.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 8 Module *SCons.Environment*

### *SCons.Environment*

Base class for construction Environments. These are the primary objects used to communicate dependency and construction information to the build engine.

Keyword arguments supplied when the construction Environment is created are construction variables used to initialize the Environment

### 8.1 Functions

<code>alias_builder(<i>env</i>, <i>target</i>, <i>source</i>)</code>
--

<code>apply_tools(<i>env</i>, <i>tools</i>, <i>toolpath</i>)</code>
---

<code>copy_non_reserved_keywords(<i>dict</i>)</code>
--

<code>is_valid_construction_var(<i>varstr</i>)</code>
---

Return if the specified string is a legitimate construction variable.
---

<code>default_decide_source(<i>dependency</i>, <i>target</i>, <i>prev_ni</i>)</code>
--

<code>default_decide_target(<i>dependency</i>, <i>target</i>, <i>prev_ni</i>)</code>
--

<code>default_copy_from_cache(<i>src</i>, <i>dst</i>)</code>
--

<code>NoSubstitutionProxy(<i>subject</i>)</code>
--

### 8.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Environment.py rel_2.4.1:3453:73fe3d3ea...
<code>CleanTargets</code>	<b>Value:</b> {}
<code>CalculatorArgs</code>	<b>Value:</b> {}
<code>AliasBuilder</code>	<b>Value:</b> <code>SCons.Builder.Builder(action=alias_builder, target_facto...</code>

*continued on next page*

Name	Description
reserved_construction_var_names	<b>Value:</b> ['CHANGED_SOURCES', 'CHANGED_TARGETS', 'SOURCE', 'SOURCES...']
future_reserved_construction_var_names	<b>Value:</b> []
__package__	<b>Value:</b> 'SCons'

### 8.3 Class MethodWrapper

object —  
**SCons.Environment.MethodWrapper**

**Known Subclasses:** SCons.Environment.BuilderWrapper

A generic Wrapper class that associates a method (which can actually be any callable) with an object. As part of creating this MethodWrapper object an attribute with the specified (by default, the name of the supplied method) is added to the underlying object. When that new "method" is called, our `__call__()` method adds the object as the first argument, simulating the Python behavior of supplying "self" on method calls.

We hang on to the name by which the method was added to the underlying base class so that we can provide a method to "clone" ourselves onto a new underlying object being copied (without which we wouldn't need to save that info).

#### 8.3.1 Methods

**\_\_init\_\_**(*self*, *object*, *method*, *name*=None)

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides:  
object.**\_\_init\_\_** extit(inherited documentation)

**\_\_call\_\_**(*self*, \**args*, \*\**kwargs*)

**clone**(*self*, *new\_object*)

Returns an object that re-binds the underlying "method" to the specified new object.

*Inherited from object*

```

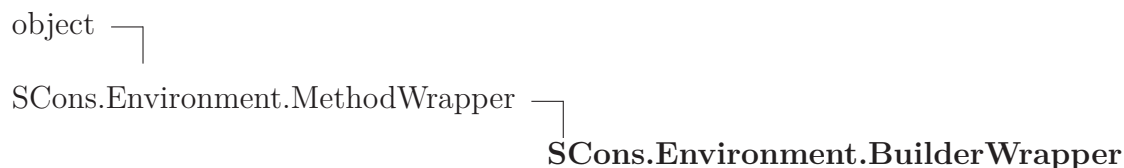
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

```

### 8.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 8.4 Class BuilderWrapper



A MethodWrapper subclass that that associates an environment with a Builder.

This mainly exists to wrap the `__call__()` function so that all calls to Builders can have their argument lists massaged in the same way (treat a lone argument as the source, treat two arguments as target then source, make sure both target and source are lists) without having to have cut-and-paste code to do it.

As a bit of obsessive backwards compatibility, we also intercept attempts to get or set the "env" or "builder" attributes, which were the names we used before we put the common functionality into the MethodWrapper base class. We'll keep this around for a while in case people shipped Tool modules that reached into the wrapper (like the Tool/qt.py module does, or did). There shouldn't be a lot attribute fetching or setting on these, so a little extra work shouldn't hurt.

### 8.4.1 Methods

<pre> __call__(self, target=None, source=&lt;class 'SCons.Environment._Null'&gt;, *args, **kw) </pre> <p>Overrides: SCons.Environment.MethodWrapper.__call__</p>
<pre> __repr__(self) </pre> <p>repr(x) Overrides: object.__repr__ extit(inherited documentation)</p>

```
__str__(self)
```

```
str(x) Overrides: object.__str__ extit(inherited documentation)
```

```
__getattr__(self, name)
```

```
__setattr__(self, name, value)
```

```
x.__setattr__('name', value) <==> x.name = value Overrides:
object.__setattr__ extit(inherited documentation)
```

*Inherited from SCons.Environment.MethodWrapper(Section 8.3)*

```
__init__(), clone()
```

*Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

#### 8.4.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

### 8.5 Class BuilderDict

```
UserDict.UserDict └─ SCons.Environment.BuilderDict
```

This is a dictionary-like class used by an Environment to hold the Builders. We need to do this because every time someone changes the Builders in the Environment's BUILDERS dictionary, we must update the Environment's attributes.

#### 8.5.1 Methods

```
__init__(self, dict, env)
```

```
Overrides: UserDict.UserDict.__init__
```

```
__semi_deepcopy__(self)
```

<code>__setitem__(self, item, val)</code>
---

Overrides: UserDict.UserDict.__setitem__
--

<code>__delitem__(self, item)</code>
--------------------------------------

Overrides: UserDict.UserDict.__delitem__
--

<code>update(self, dict)</code>
---------------------------------

Overrides: UserDict.UserDict.update
-------------------------------------

### ***Inherited from UserDict.UserDict***

`__cmp__()`, `__contains__()`, `__getitem__()`, `__len__()`, `__repr__()`, `clear()`, `copy()`, `fromkeys()`, `get()`, `has_key()`, `items()`, `iteritems()`, `iterkeys()`, `itervalues()`, `keys()`, `pop()`, `popitem()`, `setdefault()`, `values()`

#### **8.5.2 Class Variables**

Name	Description
<i>Inherited from UserDict.UserDict</i>	
<code>__hash__</code>	

## **8.6 Class SubstitutionEnvironment**

object —  
**SCons.Environment.SubstitutionEnvironment**

### **Known Subclasses:** SCons.Environment.Base

Base class for different flavors of construction environments.

This class contains a minimal set of methods that handle contruction variable expansion and conversion of strings to Nodes, which may or may not be actually useful as a stand-alone class. Which methods ended up in this class is pretty arbitrary right now. They're basically the ones which we've empirically determined are common to the different construction environment subclasses, and most of the others that use or touch the underlying dictionary of construction variables.

Eventually, this class should contain all the methods that we determine are necessary for a "minimal" interface to the build engine. A full "native Python" SCons environment has gotten pretty heavyweight with all of the methods and Tools and construction variables we've jammed in there, so it would be nice to have a lighter weight alternative for interfaces that don't need all of the bells and whistles. (At some point, we'll also probably rename this



class "Base," since that more reflects what we want this class to become, but because we've released comments that tell people to subclass `Environment.Base` to create their own flavors of construction environment, we'll save that for a future refactoring when this class actually becomes useful.)

### 8.6.1 Methods

```
__init__(self, **kw)
```

Initialization of an underlying `SubstitutionEnvironment` class. Overrides: `object.__init__`

```
__cmp__(self, other)
```

```
__delitem__(self, key)
```

```
__getitem__(self, key)
```

```
__setitem__(self, key, value)
```

```
get(self, key, default=None)
```

Emulates the `get()` method of dictionaries.

```
has_key(self, key)
```

```
__contains__(self, key)
```

```
items(self)
```

```
arg2nodes(self, args, node_factory=<class 'SCons.Environment._Null'>,
lookup_list=<class 'SCons.Environment._Null'>, **kw)
```

```
gvars(self)
```

```
lvars(self)
```

```
subst(self, string, raw=0, target=None, source=None, conv=None,  
executor=None)
```

Recursively interpolates construction variables from the Environment into the specified string, returning the expanded result. Construction variables are specified by a \$ prefix in the string and begin with an initial underscore or alphabetic character followed by any number of underscores or alphanumeric characters. The construction variable names may be surrounded by curly braces to separate the name from trailing characters.

```
subst__kw(self, kw, raw=0, target=None, source=None)
```

```
subst__list(self, string, raw=0, target=None, source=None, conv=None,  
executor=None)
```

Calls through to `SCons.Subst.scons_subst_list()`. See the documentation for that function.

```
subst__path(self, path, target=None, source=None)
```

Substitute a path list, turning EntryProxies into Nodes and leaving Nodes (and other objects) as-is.

```
subst__target__source(self, string, raw=0, target=None, source=None,  
conv=None, executor=None)
```

Recursively interpolates construction variables from the Environment into the specified string, returning the expanded result. Construction variables are specified by a \$ prefix in the string and begin with an initial underscore or alphabetic character followed by any number of underscores or alphanumeric characters. The construction variable names may be surrounded by curly braces to separate the name from trailing characters.

```
backtick(self, command)
```

**AddMethod**(*self*, *function*, *name*=None)

Adds the specified function as a method of this construction environment with the specified name. If the name is omitted, the default name is the name of the function itself.

**RemoveMethod**(*self*, *function*)

Removes the specified function's MethodWrapper from the added\_methods list, so we don't re-bind it when making a clone.

**Override**(*self*, *overrides*)

Produce a modified environment whose variables are overridden by the overrides dictionaries. "overrides" is a dictionary that will override the variables of this environment.

This function is much more efficient than Clone() or creating a new Environment because it doesn't copy the construction environment dictionary, it just wraps the underlying construction environment, and doesn't even create a wrapper object if there are no overrides.

**ParseFlags**(*self*, \**flags*)

Parse the set of flags and return a dict with the flags placed in the appropriate entry. The flags are treated as a typical set of command-line flags for a GNU-like toolchain and used to populate the entries in the dict immediately below. If one of the flag strings begins with a bang (exclamation mark), it is assumed to be a command and the rest of the string is executed; the result of that evaluation is then added to the dict.

**MergeFlags**(*self*, *args*, *unique*=1, *dict*=None)

Merge the dict in args into the construction variables of this env, or the passed-in dict. If args is not a dict, it is converted into a dict using ParseFlags. If unique is not set, the flags are appended rather than merged.

*Inherited from object*

```

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

```

### 8.6.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 8.7 Class Base

object

SCons.Environment.SubstitutionEnvironment

**SCons.Environment.Base**

**Known Subclasses:** SCons.Environment.OverrideEnvironment, SCons.Script.SConscript.SConsEnvironment

Base class for "real" construction Environments. These are the primary objects used to communicate dependency and construction information to the build engine.

Keyword arguments supplied when the construction Environment is created are construction variables used to initialize the Environment.

### 8.7.1 Methods

**Action**(*self*, \**args*, \*\**kw*)

**AddPostAction**(*self*, *files*, *action*)

**AddPreAction**(*self*, *files*, *action*)

**Alias**(*self*, *target*, *source*=[], *action*=None, \*\**kw*)

**AlwaysBuild**(*self*, \**targets*)

**Append**(*self*, *\*\*kw*)

Append values to existing construction variables in an Environment.

**AppendENVPath**(*self*, *name*, *newpath*, *envname*='ENV', *sep*=':', *delete\_existing*=1)

Append path elements to the path 'name' in the 'ENV' dictionary for this environment. Will only add any particular path once, and will normpath and normcase all paths to help assure this. This can also handle the case where the env variable is a list instead of a string.

If delete\_existing is 0, a newpath which is already in the path will not be moved to the end (it will be left where it is).

**AppendUnique**(*self*, *delete\_existing*=0, *\*\*kw*)

Append values to existing construction variables in an Environment, if they're not already there. If delete\_existing is 1, removes existing values first, so values move to end.

**BuildDir**(*self*, *\*args*, *\*\*kw*)

**Builder**(*self*, *\*\*kw*)

**CacheDir**(*self*, *path*)

**Clean**(*self*, *targets*, *files*)

**Clone**(*self*, *tools*=[], *toolpath*=None, *parse\_flags*=None, *\*\*kw*)

Return a copy of a construction Environment. The copy is like a Python "deep copy"--that is, independent copies are made recursively of each objects--except that a reference is copied when an object is not deep-copyable (like a function). There are no references to any mutable objects in the original Environment.

**Command**(*self*, *target*, *source*, *action*, \*\**kw*)

Builds the supplied target files from the supplied source files using the supplied action. Action may be any type that the Builder constructor will accept for an action.

**Configure**(*self*, \**args*, \*\**kw*)**Copy**(*self*, \**args*, \*\**kw*)**Decider**(*self*, *function*)**Depends**(*self*, *target*, *dependency*)

Explicitly specify that 'target's depend on 'dependency'.

**Detect**(*self*, *progs*)

Return the first available program in progs.

**Dictionary**(*self*, \**args*)**Dir**(*self*, *name*, \**args*, \*\**kw*)**Dump**(*self*, *key*=None)

Using the standard Python pretty printer, return the contents of the scons build environment as a string.

If the key passed in is anything other than None, then that will be used as an index into the build environment dictionary and whatever is found there will be fed into the pretty printer. Note that this key is case sensitive.

**Entry**(*self*, *name*, \**args*, \*\**kw*)**Environment**(*self*, \*\**kw*)

**Execute**(*self*, *action*, \**args*, \*\**kw*)

Directly execute an action through an Environment

**File**(*self*, *name*, \**args*, \*\**kw*)

**FindFile**(*self*, *file*, *dirs*)

**FindInstalledFiles**(*self*)

returns the list of all targets of the Install and InstallAs Builder.

**FindIxes**(*self*, *paths*, *prefix*, *suffix*)

Search a list of paths for something that matches the prefix and suffix.

paths - the list of paths or nodes. prefix - construction variable for the prefix.  
suffix - construction variable for the suffix.

**FindSourceFiles**(*self*, *node*='.'')

returns a list of all source files.

**Flatten**(*self*, *sequence*)

**GetBuildPath**(*self*, *files*)

**Glob**(*self*, *pattern*, *ondisk*=True, *source*=False, *strings*=False,  
*exclude*=None)

**Ignore**(*self*, *target*, *dependency*)

Ignore a dependency.

**Literal**(*self*, *string*)

**Local**(*self*, \**targets*)

**NoCache**(*self*, \**targets*)

Tags a target so that it will not be cached

**NoClean**(*self*, \**targets*)

Tags a target so that it will not be cleaned by -c

**ParseConfig**(*self*, *command*, *function*=None, *unique*=1)

Use the specified function to parse the output of the command in order to modify the current environment. The 'command' can be a string or a list of strings representing a command and its arguments. 'Function' is an optional argument that takes the environment, the output of the command, and the unique flag. If no function is specified, MergeFlags, which treats the output as the result of a typical 'X-config' command (i.e. gtk-config), will merge the output into the appropriate variables.

**ParseDepends**(*self*, *filename*, *must\_exist*=None, *only\_one*=0)

Parse a mkdep-style file for explicit dependencies. This is completely abusable, and should be unnecessary in the "normal" case of proper SCons configuration, but it may help make the transition from a Make hierarchy easier for some people to swallow. It can also be genuinely useful when using a tool that can write a .d file, but for which writing a scanner would be too complicated.

**Platform**(*self*, *platform*)

**Precious**(*self*, \**targets*)

**Prepend**(*self*, \*\**kw*)

Prepend values to existing construction variables in an Environment.



---

**PrependENVPath**(*self*, *name*, *newpath*, *envname*='ENV', *sep*=':',  
*delete\_existing*=1)

---

Prepend path elements to the path 'name' in the 'ENV' dictionary for this environment. Will only add any particular path once, and will normpath and normcase all paths to help assure this. This can also handle the case where the env variable is a list instead of a string.

If delete\_existing is 0, a newpath which is already in the path will not be moved to the front (it will be left where it is).

---

**PrependUnique**(*self*, *delete\_existing*=0, *\*\*kw*)

---

Prepend values to existing construction variables in an Environment, if they're not already there. If delete\_existing is 1, removes existing values first, so values move to front.

---

**Pseudo**(*self*, *\*targets*)

---



---

**Replace**(*self*, *\*\*kw*)

---

Replace existing construction variables in an Environment with new construction variables and/or values.

---

**ReplaceIxes**(*self*, *path*, *old\_prefix*, *old\_suffix*, *new\_prefix*, *new\_suffix*)

---

Replace old\_prefix with new\_prefix and old\_suffix with new\_suffix.

env - Environment used to interpolate variables. path - the path that will be modified. old\_prefix - construction variable for the old prefix. old\_suffix - construction variable for the old suffix. new\_prefix - construction variable for the new prefix. new\_suffix - construction variable for the new suffix.

---

**Repository**(*self*, *\*dirs*, *\*\*kw*)

---

**Requires**(*self*, *target*, *prerequisite*)

Specify that 'prerequisite' must be built before 'target', (but 'target' does not actually depend on 'prerequisite' and need not be rebuilt if it changes).

**SConsignFile**(*self*, *name*='.sconsign', *dbm\_module*=None)

**Scanner**(*self*, \**args*, \*\**kw*)

**SetDefault**(*self*, \*\**kw*)

**SideEffect**(*self*, *side\_effect*, *target*)

Tell scons that side\_effects are built as side effects of building targets.

**SourceCode**(*self*, *entry*, *builder*)

Arrange for a source code builder for (part of) a tree.

**SourceSignatures**(*self*, *type*)

**Split**(*self*, *arg*)

This function converts a string or list into a list of strings or Nodes. This makes things easier for users by allowing files to be specified as a white-space separated list to be split.

The input rules are:

- A single string containing names separated by spaces. These will be split apart at the spaces.
- A single Node instance
- A list containing either strings or Node instances. Any strings in the list are not split at spaces.

In all cases, the function returns a list of Nodes and strings.

**TargetSignatures**(*self*, *type*)

**Tool**(*self*, *tool*, *toolpath*=None, \*\**kw*)

**Value**(*self*, *value*, *built\_value*=None)

**VariantDir**(*self*, *variant\_dir*, *src\_dir*, *duplicate*=1)

**WhereIs**(*self*, *prog*, *path*=None, *pathext*=None, *reject*=[])

Find prog in the path.

**\_\_init\_\_**(*self*, *platform*=None, *tools*=None, *toolpath*=None, *variables*=None, *parse\_flags*=None, *\*\*kw*)

Initialization of a basic SCons construction environment, including setting up special construction variables like BUILDER, PLATFORM, etc., and searching for and applying available Tools.

Note that we do *not* call the underlying base class (SubstitutionEnvironment) initialization, because we need to initialize things in a very specific order that doesn't work with the much simpler base class initialization. Overrides: object.\_\_init\_\_

**get\_\_CacheDir**(*self*)

**get\_\_builder**(*self*, *name*)

Fetch the builder with the specified name from the environment.

**get\_\_factory**(*self*, *factory*, *default*='File')

Return a factory function for creating Nodes for this construction environment.

**get\_\_scanner**(*self*, *skey*)

Find the appropriate scanner given a key (usually a file suffix).

**get\_\_src\_sig\_type**(*self*)

```
get_tgt_sig_type(self)
```

```
scanner_map_delete(self, kw=None)
```

Delete the cached scanner map (if we need to).

### *Inherited from SCons.Environment.SubstitutionEnvironment(Section 8.6)*

AddMethod(), MergeFlags(), Override(), ParseFlags(), RemoveMethod(), \_\_cmp\_\_(), \_\_contains\_\_(), \_\_delitem\_\_(), \_\_getitem\_\_(), \_\_setitem\_\_(), arg2nodes(), backtick(), get(), gvars(), has\_key(), items(), lvars(), subst(), subst\_kw(), subst\_list(), subst\_path(), subst\_target\_source()

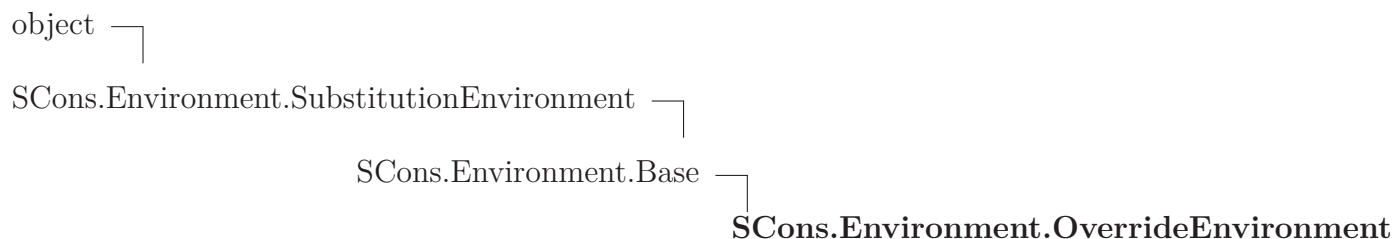
### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

#### 8.7.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 8.8 Class OverrideEnvironment



A proxy that overrides variables in a wrapped construction environment by returning values from an overrides dictionary in preference to values from the underlying subject environment.

This is a lightweight (I hope) proxy that passes through most use of attributes to the underlying Environment.Base class, but has just enough additional methods defined to act like a real construction environment with overridden values. It can wrap either a Base construction environment, or another OverrideEnvironment, which can in turn nest arbitrary OverrideEnvironments...

Note that we do *not* call the underlying base class (*SubstitutionEnvironment*) initialization, because we get most of those from proxying the attributes of the subject construction environment. But because we subclass *SubstitutionEnvironment*, this class also has inherited *arg2nodes()* and *subst\*()* methods; those methods can't be proxied because they need *this* object's methods to fetch the values from the overrides dictionary.

### 8.8.1 Methods

**`__init__(self, subject, overrides={})`**

Initialization of a basic SCons construction environment, including setting up special construction variables like *BUILDER*, *PLATFORM*, etc., and searching for and applying available Tools.

Note that we do *not* call the underlying base class (*SubstitutionEnvironment*) initialization, because we need to initialize things in a very specific order that doesn't work with the much simpler base class initialization. Overrides: *object.\_\_init\_\_* *exitit*(inherited documentation)

**`__getattr__(self, name)`**

**`__setattr__(self, name, value)`**

*x.\_\_setattr\_\_('name', value) <==> x.name = value* Overrides: *object.\_\_setattr\_\_* *exitit*(inherited documentation)

**`__getitem__(self, key)`**

Overrides: *SCons.Environment.SubstitutionEnvironment.\_\_getitem\_\_*

**`__setitem__(self, key, value)`**

Overrides: *SCons.Environment.SubstitutionEnvironment.\_\_setitem\_\_*

**`__delitem__(self, key)`**

Overrides: *SCons.Environment.SubstitutionEnvironment.\_\_delitem\_\_*

**`get(self, key, default=None)`**

Emulates the *get()* method of dictionaries. Overrides: *SCons.Environment.SubstitutionEnvironment.get*

**has\_key**(*self*, *key*)

Overrides: *SCons.Environment.SubstitutionEnvironment*.has\_key

**\_\_contains\_\_**(*self*, *key*)

Overrides: *SCons.Environment.SubstitutionEnvironment*.\_\_contains\_\_

**Dictionary**(*self*)

Emulates the items() method of dictionaries. Overrides:  
*SCons.Environment.Base.Dictionary*

**items**(*self*)

Emulates the items() method of dictionaries. Overrides:  
*SCons.Environment.SubstitutionEnvironment.items*

**gvars**(*self*)

Overrides: *SCons.Environment.SubstitutionEnvironment.gvars*

**lvars**(*self*)

Overrides: *SCons.Environment.SubstitutionEnvironment.lvars*

**Replace**(*self*, \*\**kw*)

Replace existing construction variables in an Environment with new construction variables and/or values. Overrides:  
*SCons.Environment.Base.Replace* extit(inherited documentation)

### *Inherited from SCons.Environment.Base(Section 8.9)*

Action(), AddPostAction(), AddPreAction(), Alias(), AlwaysBuild(), Append(), AppendENVPPath(), AppendUnique(), BuildDir(), Builder(), CacheDir(), Clean(), Clone(), Command(), Configure(), Copy(), Decider(), Depends(), Detect(), Dir(), Dump(), Entry(), Environment(), Execute(), File(), FindFile(), FindInstalledFiles(), FindIdxes(), FindSourceFiles(), Flatten(), GetBuildPath(), Glob(), Ignore(), Literal(), Local(), NoCache(), NoClean(), ParseConfig(), ParseDepends(), Platform(), Precious(), Prepend(), PrependENVPPath(), PrependUnique(), Pseudo(), ReplaceIdxes(), Repository(), Requires(), SConsignFile(), Scanner(), SetDefault(), SideEffect(), SourceCode(), SourceSignatures(), Split(), TargetSignatures(), Tool(), Value(), VariantDir(), WhereIs(), get\_CacheDir(), get\_builder(), get\_factory(),

get\_scanner(), get\_src\_sig\_type(), get\_tgt\_sig\_type(), scanner\_map\_delete()

***Inherited from SCons.Environment.SubstitutionEnvironment(Section 8.6)***

AddMethod(), MergeFlags(), Override(), ParseFlags(), RemoveMethod(), \_\_cmp\_\_(),  
arg2nodes(), backtick(), subst(), subst\_kw(), subst\_list(), subst\_path(), subst\_target\_source()

***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
\_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_sub-  
classhook\_\_()

### 8.8.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

## 8.9 Class Base

object └─

SCons.Environment.SubstitutionEnvironment └─  
SCons.Environment.Base

**Known Subclasses:** SCons.Environment.OverrideEnvironment, SCons.Script.SConscript.SConsEnvironment

Base class for "real" construction Environments. These are the primary objects used to communicate dependency and construction information to the build engine.

Keyword arguments supplied when the construction Environment is created are construction variables used to initialize the Environment.

### 8.9.1 Methods

**Action**(*self*, \**args*, \*\**kw*)

**AddPostAction**(*self*, *files*, *action*)

**AddPreAction**(*self*, *files*, *action*)

```
Alias(self, target, source=[], action=None, **kw)
```

```
AlwaysBuild(self, *targets)
```

```
Append(self, **kw)
```

Append values to existing construction variables in an Environment.

```
AppendENVPath(self, name, newpath, envname='ENV', sep=':',  
delete_existing=1)
```

Append path elements to the path 'name' in the 'ENV' dictionary for this environment. Will only add any particular path once, and will normpath and normcase all paths to help assure this. This can also handle the case where the env variable is a list instead of a string.

If delete\_existing is 0, a newpath which is already in the path will not be moved to the end (it will be left where it is).

```
AppendUnique(self, delete_existing=0, **kw)
```

Append values to existing construction variables in an Environment, if they're not already there. If delete\_existing is 1, removes existing values first, so values move to end.

```
BuildDir(self, *args, **kw)
```

```
Builder(self, **kw)
```

```
CacheDir(self, path)
```

```
Clean(self, targets, files)
```



**Clone**(*self*, *tools*=[], *toolpath*=None, *parse\_flags*=None, \*\**kw*)

Return a copy of a construction Environment. The copy is like a Python "deep copy"--that is, independent copies are made recursively of each objects--except that a reference is copied when an object is not deep-copyable (like a function). There are no references to any mutable objects in the original Environment.

**Command**(*self*, *target*, *source*, *action*, \*\**kw*)

Builds the supplied target files from the supplied source files using the supplied action. Action may be any type that the Builder constructor will accept for an action.

**Configure**(*self*, \**args*, \*\**kw*)

**Copy**(*self*, \**args*, \*\**kw*)

**Decider**(*self*, *function*)

**Depends**(*self*, *target*, *dependency*)

Explicitly specify that 'target's depend on 'dependency'.

**Detect**(*self*, *progs*)

Return the first available program in progs.

**Dictionary**(*self*, \**args*)

**Dir**(*self*, *name*, \**args*, \*\**kw*)

**Dump**(*self*, *key*=None)

Using the standard Python pretty printer, return the contents of the scons build environment as a string.

If the key passed in is anything other than None, then that will be used as an index into the build environment dictionary and whatever is found there will be fed into the pretty printer. Note that this key is case sensitive.

**Entry**(*self*, *name*, \**args*, \*\**kw*)

**Environment**(*self*, \*\**kw*)

**Execute**(*self*, *action*, \**args*, \*\**kw*)

Directly execute an action through an Environment

**File**(*self*, *name*, \**args*, \*\**kw*)

**FindFile**(*self*, *file*, *dirs*)

**FindInstalledFiles**(*self*)

returns the list of all targets of the Install and InstallAs Builder.

**FindIxes**(*self*, *paths*, *prefix*, *suffix*)

Search a list of paths for something that matches the prefix and suffix.

paths - the list of paths or nodes. prefix - construction variable for the prefix.  
suffix - construction variable for the suffix.

**FindSourceFiles**(*self*, *node*='.' )

returns a list of all source files.

**Flatten**(*self*, *sequence*)

**GetBuildPath**(*self*, *files*)

**Glob**(*self*, *pattern*, *ondisk*=True, *source*=False, *strings*=False, *exclude*=None)

**Ignore**(*self*, *target*, *dependency*)

Ignore a dependency.

**Literal**(*self*, *string*)

**Local**(*self*, \**targets*)

**NoCache**(*self*, \**targets*)

Tags a target so that it will not be cached

**NoClean**(*self*, \**targets*)

Tags a target so that it will not be cleaned by -c

**ParseConfig**(*self*, *command*, *function*=None, *unique*=1)

Use the specified function to parse the output of the command in order to modify the current environment. The 'command' can be a string or a list of strings representing a command and its arguments. 'Function' is an optional argument that takes the environment, the output of the command, and the unique flag. If no function is specified, MergeFlags, which treats the output as the result of a typical 'X-config' command (i.e. gtk-config), will merge the output into the appropriate variables.

**ParseDepends**(*self*, *filename*, *must\_exist*=None, *only\_one*=0)

Parse a mkdep-style file for explicit dependencies. This is completely abusable, and should be unnecessary in the "normal" case of proper SCons configuration, but it may help make the transition from a Make hierarchy easier for some people to swallow. It can also be genuinely useful when using a tool that can write a .d file, but for which writing a scanner would be too complicated.

**Platform**(*self*, *platform*)

**Precious**(*self*, *\*targets*)

**Prepend**(*self*, *\*\*kw*)

Prepend values to existing construction variables in an Environment.

**PrependENVPath**(*self*, *name*, *newpath*, *envname*='ENV', *sep*=':', *delete\_existing*=1)

Prepend path elements to the path 'name' in the 'ENV' dictionary for this environment. Will only add any particular path once, and will normpath and normcase all paths to help assure this. This can also handle the case where the env variable is a list instead of a string.

If delete\_existing is 0, a newpath which is already in the path will not be moved to the front (it will be left where it is).

**PrependUnique**(*self*, *delete\_existing*=0, *\*\*kw*)

Prepend values to existing construction variables in an Environment, if they're not already there. If delete\_existing is 1, removes existing values first, so values move to front.

**Pseudo**(*self*, *\*targets*)

**Replace**(*self*, *\*\*kw*)

Replace existing construction variables in an Environment with new construction variables and/or values.

**ReplaceIxes**(*self*, *path*, *old\_prefix*, *old\_suffix*, *new\_prefix*, *new\_suffix*)

Replace *old\_prefix* with *new\_prefix* and *old\_suffix* with *new\_suffix*.

*env* - Environment used to interpolate variables. *path* - the path that will be modified. *old\_prefix* - construction variable for the old prefix. *old\_suffix* - construction variable for the old suffix. *new\_prefix* - construction variable for the new prefix. *new\_suffix* - construction variable for the new suffix.

**Repository**(*self*, *\*dirs*, *\*\*kw*)

**Requires**(*self*, *target*, *prerequisite*)

Specify that 'prerequisite' must be built before 'target', (but 'target' does not actually depend on 'prerequisite' and need not be rebuilt if it changes).

**SConsignFile**(*self*, *name*='.sconsign', *dbm\_module*=None)

**Scanner**(*self*, *\*args*, *\*\*kw*)

**SetDefault**(*self*, *\*\*kw*)

**SideEffect**(*self*, *side\_effect*, *target*)

Tell scons that *side\_effects* are built as side effects of building targets.

**SourceCode**(*self*, *entry*, *builder*)

Arrange for a source code builder for (part of) a tree.

**SourceSignatures**(*self*, *type*)

**Split**(*self*, *arg*)

This function converts a string or list into a list of strings or Nodes. This makes things easier for users by allowing files to be specified as a white-space separated list to be split.

The input rules are:

- A single string containing names separated by spaces. These will be split apart at the spaces.
- A single Node instance
- A list containing either strings or Node instances. Any strings in the list are not split at spaces.

In all cases, the function returns a list of Nodes and strings.

**TargetSignatures**(*self*, *type*)

**Tool**(*self*, *tool*, *toolpath*=None, *\*\*kw*)

**Value**(*self*, *value*, *built\_value*=None)

**VariantDir**(*self*, *variant\_dir*, *src\_dir*, *duplicate*=1)

**WhereIs**(*self*, *prog*, *path*=None, *pathext*=None, *reject*=[])

Find prog in the path.

**\_\_init\_\_**(*self*, *platform*=None, *tools*=None, *toolpath*=None, *variables*=None, *parse\_flags*=None, *\*\*kw*)

Initialization of a basic SCons construction environment, including setting up special construction variables like BUILDER, PLATFORM, etc., and searching for and applying available Tools.

Note that we do *not* call the underlying base class (SubstitutionEnvironment) initialization, because we need to initialize things in a very specific order that doesn't work with the much simpler base class initialization. Overrides: object.\_\_init\_\_

**get\_\_CacheDir**(*self*)

<code>get_builder(self, name)</code>
--------------------------------------

Fetch the builder with the specified name from the environment.
---

<code>get_factory(self, factory, default='File')</code>
---

Return a factory function for creating Nodes for this construction environment.
---

<code>get_scanner(self, skey)</code>
--------------------------------------

Find the appropriate scanner given a key (usually a file suffix).
---

<code>get_src_sig_type(self)</code>
-------------------------------------

<code>get_tgt_sig_type(self)</code>
-------------------------------------

<code>scanner_map_delete(self, kw=None)</code>
--

Delete the cached scanner map (if we need to).
--

***Inherited from *SCons.Environment.SubstitutionEnvironment*(Section 8.6)***

AddMethod(), MergeFlags(), Override(), ParseFlags(), RemoveMethod(), \_\_cmp\_\_(), \_\_contains\_\_(), \_\_delitem\_\_(), \_\_getitem\_\_(), \_\_setitem\_\_(), arg2nodes(), backtick(), get(), gvars(), has\_key(), items(), lvars(), subst(), subst\_kw(), subst\_list(), subst\_path(), subst\_target\_source()

***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

### 8.9.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

*continued on next page*

Name	Description
------	-------------



## 9 Module SCons.Errors

SCons.Errors

This file contains the exception classes used to handle internal and user errors in SCons.

### 9.1 Functions

**convert\_to\_BuildError**(*status*, *exc\_info*=None)

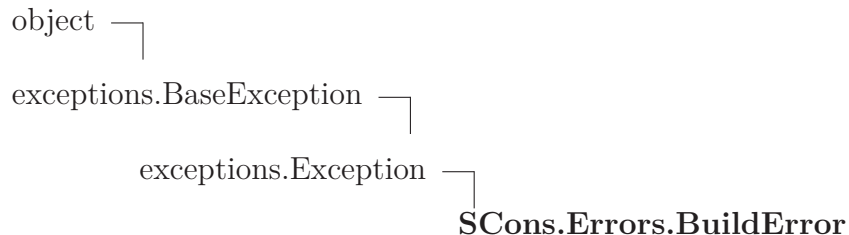
Convert any return code a BuildError Exception.

‘status’ can either be a return code or an Exception. The buildError.status we set here will normally be used as the exit status of the "scons" process.

### 9.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Errors.py rel_2.4.1:3453:73fef3ea0b0 2...
__package__	<b>Value:</b> 'SCons'

### 9.3 Class BuildError



Errors occuring while building.

BuildError have the following attributes:

Information about the cause of the build error:

-----

errstr : a description of the error message

`status` : the return code of the action that caused the build error. Must be set to a non-zero value even if the build error is not due to an action returning a non-zero returned code.

`exitstatus` : SCons exit status due to this build error. Must be nonzero unless due to an explicit `Exit()` call. Not always the same as `status`, since actions return a status code that should be respected, but SCons typically exits with 2 irrespective of the return value of the failed action.

`filename` : The name of the file or directory that caused the build error. Set to `None` if no files are associated with this error. This might be different from the target being built. For example, failure to create the directory in which the target file will appear. It can be `None` if the error is not due to a particular filename.

`exc_info` : Info about exception that caused the build error. Set to `(None, None, None)` if this build error is not due to an exception.

Information about the cause of the location of the error:  
-----

`node` : the error occurred while building this target node(s)

`executor` : the executor that caused the build to fail (might be `None` if the build failures is not due to the executor failing)

`action` : the action that caused the build to fail (might be `None` if the build failures is not due to the an action failure)

`command` : the command line for the action that caused the build to fail (might be `None` if the build failures is not due to the an action failure)

### 9.3.1 Methods

```
__init__(self, node=None, errstr='Unknown error', status=2,
        exitstatus=2, filename=None, executor=None, action=None, command=None,
        exc_info=(None, None, None))
```

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides:  
object.**\_\_init\_\_** **exitit**(inherited documentation)

```
__str__(self)
```

str(x) Overrides: object.**\_\_str\_\_** **exitit**(inherited documentation)

*Inherited from exceptions.Exception*

```
__new__()
```

*Inherited from exceptions.BaseException*

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(),
__repr__(), __setattr__(), __setstate__(), __unicode__()
```

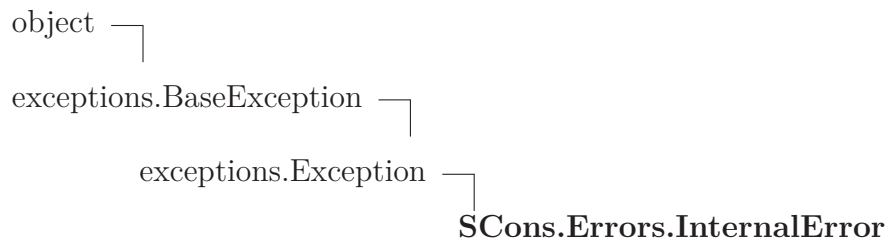
*Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

### 9.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<b>__class__</b>	

## 9.4 Class *InternalError*



### 9.4.1 Methods

*Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

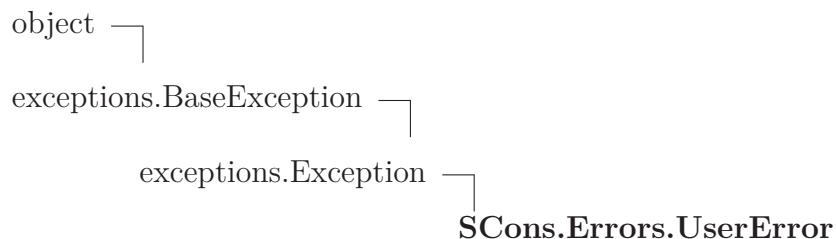
*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 9.4.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 9.5 Class `UserError`



**Known Subclasses:** `SCons.SConf.SConfError`, `SCons.Warnings.Warning`

### 9.5.1 Methods

*Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

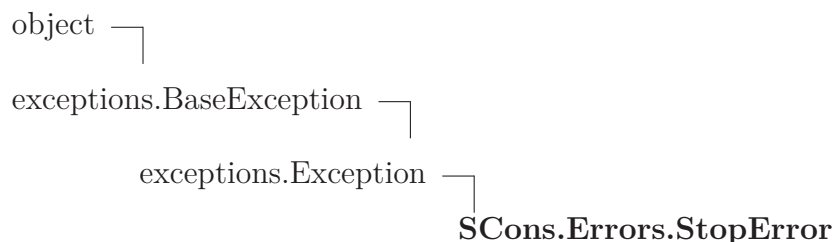
### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

## 9.5.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	args, message
<i>Inherited from object</i>	<code>__class__</code>

## 9.6 Class `StopError`



### 9.6.1 Methods

#### *Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

#### *Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

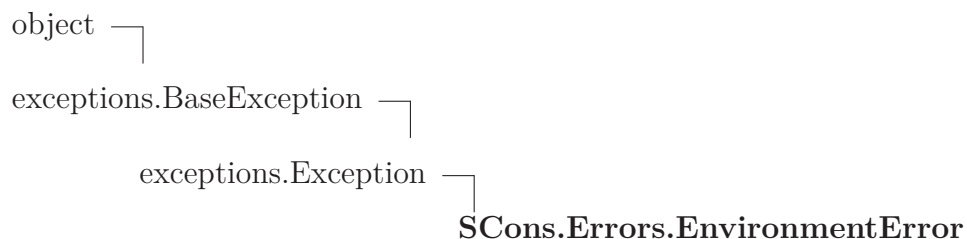
### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 9.6.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

## 9.7 Class EnvironmentError



### 9.7.1 Methods

*Inherited from exceptions.Exception*

\_\_init\_\_(), \_\_new\_\_()

*Inherited from exceptions.BaseException*

\_\_delattr\_\_(), \_\_getattr\_\_(), \_\_getitem\_\_(), \_\_getslice\_\_(), \_\_reduce\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_setstate\_\_(), \_\_str\_\_(), \_\_unicode\_\_()

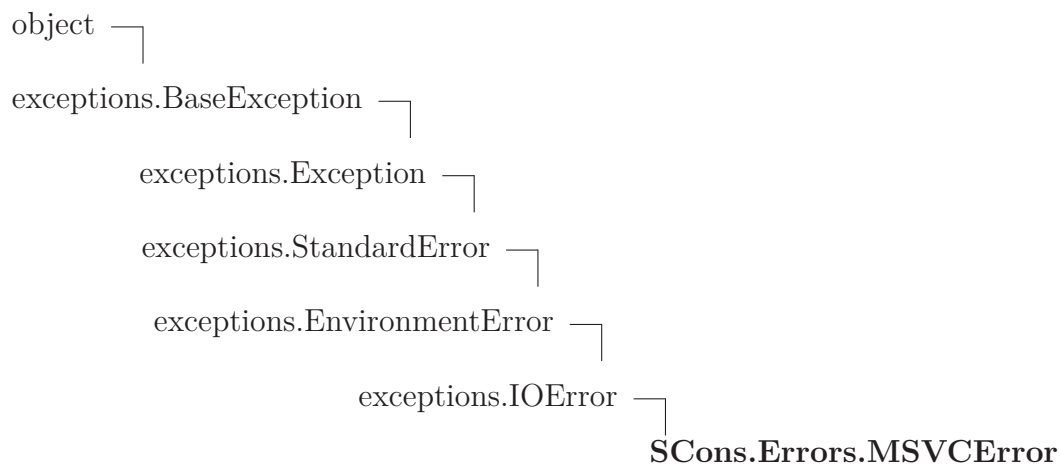
*Inherited from object*

\_\_format\_\_(), \_\_hash\_\_(), \_\_reduce\_ex\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

### 9.7.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

## 9.8 Class MSVCErrors



### 9.8.1 Methods

#### *Inherited from exceptions.IOError*

`__init__()`, `__new__()`

#### *Inherited from exceptions.EnvironmentError*

`__reduce__()`, `__str__()`

#### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

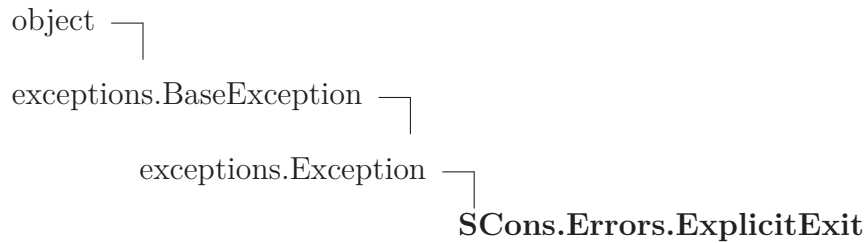
#### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 9.8.2 Properties

Name	Description
<i>Inherited from exceptions.EnvironmentError</i> <code>errno</code> , <code>filename</code> , <code>strerror</code>	
<i>Inherited from exceptions.BaseException</i> <code>args</code> , <code>message</code>	
<i>Inherited from object</i> <code>__class__</code>	

## 9.9 Class *ExplicitExit*



### 9.9.1 Methods

**`__init__`**(*self*, *node*=None, *status*=None, \**args*)

*x*.**`__init__`**(...) initializes *x*; see `help(type(x))` for signature Overrides:  
*object*.**`__init__`** `exitit`(inherited documentation)

*Inherited from exceptions.Exception*

**`__new__`**()

*Inherited from exceptions.BaseException*

**`__delattr__`**(), **`__getattr__`**(), **`__getitem__`**(), **`__getslice__`**(), **`__reduce__`**(), **`__repr__`**(), **`__setattr__`**(), **`__setstate__`**(), **`__str__`**(), **`__unicode__`**()

*Inherited from object*

**`__format__`**(), **`__hash__`**(), **`__reduce_ex__`**(), **`__sizeof__`**(), **`__subclasshook__`**()

### 9.9.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
	<i>args</i> , <i>message</i>
<i>Inherited from object</i>	
<b><code>__class__</code></b>	



## 10 Module *SCons.Executor*

*SCons.Executor*

A module for executing actions with specific lists of target and source Nodes.

### 10.1 Functions

**rfile**(*node*)

A function to return the results of a Node's `rfile()` method, if it exists, and the Node itself otherwise (if it's a Value Node, e.g.).

**execute\_nothing**(*obj*, *target*, *kw*)

**execute\_action\_list**(*obj*, *target*, *kw*)

Actually execute the action list.

**execute\_actions\_str**(*obj*)

**execute\_null\_str**(*obj*)

**GetBatchExecutor**(*key*)

**AddBatchExecutor**(*key*, *executor*)

**get\_NullEnvironment**()

Use singleton pattern for Null Environments.

### 10.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Executor.py rel_2.4.1:3453:73fef3ea0b0...

*continued on next page*

Name	Description
nullenv	<b>Value:</b> None
__package__	<b>Value:</b> 'SCons'

### 10.3 Class Batch



Remembers exact association between targets and sources of executor.

#### 10.3.1 Methods

<code>__init__(self, targets=[], sources=[])</code>
x. <code>__init__</code> (...) initializes x; see <code>help(type(x))</code> for signature Overrides: object. <code>__init__</code> <code>exitit</code> (inherited documentation)

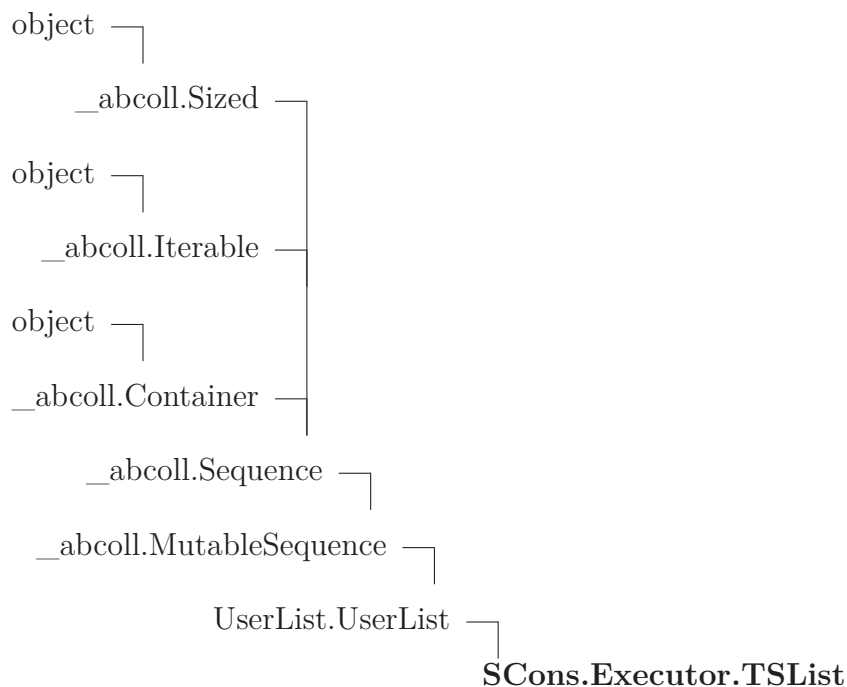
#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

#### 10.3.2 Properties

Name	Description
sources	
targets	
<i>Inherited from object</i>	
__class__	

## 10.4 Class **TSList**



A class that implements \$TARGETS or \$SOURCES expansions by wrapping an executor Method. This class is used in the `Executor.lvars()` to delay creation of `NodeList` objects until they're needed.

Note that we subclass `collections.UserList` purely so that the `is_Sequence()` function will identify an object of this class as a list during variable expansion. We're not really using any `collections.UserList` methods in practice.

### 10.4.1 Methods

**\_\_init\_\_**(*self*, *func*)

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature Overrides:  
*object*.**\_\_init\_\_** `extit`(inherited documentation)

**\_\_getattr\_\_**(*self*, *attr*)

**\_\_getitem\_\_**(*self*, *i*)

Overrides: `_abcoll.Sequence.__getitem__`

<code>__getslice__(self, i, j)</code>
---------------------------------------

Overrides: UserList.UserList.__getslice__
---

<code>__str__(self)</code>
----------------------------

str(x) Overrides: object.__str__ exitit(inherited documentation)
--

<code>__repr__(self)</code>
-----------------------------

repr(x) Overrides: object.__repr__ exitit(inherited documentation)
--

### ***Inherited from UserList.UserList***

`__add__()`, `__cmp__()`, `__contains__()`, `__delitem__()`, `__delslice__()`,  
`__eq__()`, `__ge__()`, `__gt__()`, `__iadd__()`, `__imul__()`, `__le__()`, `__len__()`,  
`__lt__()`, `__mul__()`, `__ne__()`, `__radd__()`, `__rmul__()`, `__setitem__()`,  
`__setslice__()`, `append()`, `count()`, `extend()`, `index()`, `insert()`, `pop()`, `remove()`,  
`reverse()`, `sort()`

### ***Inherited from \_\_abcoll.Sequence***

`__iter__()`, `__reversed__()`

### ***Inherited from \_\_abcoll.Sized***

`__subclasshook__()`

### ***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`,  
`__reduce_ex__()`, `__setattr__()`, `__sizeof__()`

## **10.4.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## **10.4.3 Class Variables**

Name	Description
<i>Inherited from UserList.UserList</i>	
<code>__abstractmethods__</code> , <code>__hash__</code>	

## 10.5 Class TSOBJect



A class that implements \$TARGET or \$SOURCE expansions by wrapping an Executor method.

### 10.5.1 Methods

**\_\_init\_\_**(*self*, *func*)

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides: object.**\_\_init\_\_** extit(inherited documentation)

**\_\_getattr\_\_**(*self*, *attr*)

**\_\_str\_\_**(*self*)

str(x) Overrides: object.**\_\_str\_\_** extit(inherited documentation)

**\_\_repr\_\_**(*self*)

repr(x) Overrides: object.**\_\_repr\_\_** extit(inherited documentation)

### *Inherited from object*

**\_\_delattr\_\_**(*self*, *attr*), **\_\_format\_\_**(*self*, *format\_spec*), **\_\_getattr\_\_**(*self*, *attr*), **\_\_hash\_\_**(*self*), **\_\_new\_\_**(*cls*, *args*, *kwargs*), **\_\_reduce\_\_**(*self*), **\_\_reduce\_ex\_\_**(*self*, *proto*), **\_\_setattr\_\_**(*self*, *attr*, *value*), **\_\_sizeof\_\_**(*self*), **\_\_subclasshook\_\_**(*self*, *subclass*)

### 10.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<b>__class__</b>	

## 10.6 Class Executor



A class for controlling instances of executing an action.

This largely exists to hold a single association of an action, environment, list of environment override dictionaries, targets and sources for later processing as needed.

### 10.6.1 Methods

```
__init__(self, action, env=None, overridelist=[{}], targets=[],
sources=[], builder_kw={})
```

*x*.**\_\_init\_\_**(...) initializes *x*; see help(type(*x*)) for signature Overrides:  
object.**\_\_init\_\_** **exitit**(inherited documentation)

```
get_lvars(self)
```

```
get_action_targets(self)
```

```
set_action_list(self, action)
```

```
get_action_list(self)
```

```
get_all_targets(self)
```

Returns all targets for all batches of this Executor.

```
get_all_sources(self)
```

Returns all sources for all batches of this Executor.

```
get_all_children(self)
```

Returns all unique children (dependencies) for all batches of this Executor.

The Taskmaster can recognize when it's already evaluated a Node, so we don't have to make this list unique for its intended canonical use case, but we expect there to be a lot of redundancy (long lists of batched .cc files #including the same .h files over and over), so removing the duplicates once up front should save the Taskmaster a lot of work.

**get\_all\_prerequisites**(*self*)

Returns all unique (order-only) prerequisites for all batches of this Executor.

**get\_action\_side\_effects**(*self*)

Returns all side effects for all batches of this Executor used by the underlying Action.

**get\_build\_env**(*self*)

Fetch or create the appropriate build Environment for this Executor.

**get\_build\_scanner\_path**(*self*, *scanner*)

Fetch the scanner path for this executor's targets and sources.

**get\_kw**(*self*, *kw*={})**\_\_call\_\_**(*self*, *target*, *\*\*kw*)**cleanup**(*self*)**add\_sources**(*self*, *sources*)

Add source files to this Executor's list. This is necessary for "multi" Builders that can be called repeatedly to build up a source file list for a given target.

**get\_sources**(*self*)

**add\_batch**(*self*, *targets*, *sources*)

Add pair of associated target and source to this Executor's list. This is necessary for "batch" Builders that can be called repeatedly to build up a list of matching target and source files that will be used in order to update multiple target files at once from multiple corresponding source files, for tools like MSVC that support it.

**prepare**(*self*)

Preparatory checks for whether this Executor can go ahead and (try to) build its targets.

**add\_pre\_action**(*self*, *action*)**add\_post\_action**(*self*, *action*)**\_\_str\_\_**(*self*)

str(x) Overrides: object.\_\_str\_\_ exitit(inherited documentation)

**nullify**(*self*)**get\_contents**(*self*)

Fetch the signature contents. This is the main reason this class exists, so we can compute this once and cache it regardless of how many target or source Nodes there are.

**get\_timestamp**(*self*)

Fetch a time stamp for this Executor. We don't have one, of course (only files do), but this is the interface used by the timestamp module.

**scan\_targets**(*self*, *scanner*)**scan\_sources**(*self*, *scanner*)



**scan**(*self*, *scanner*, *node\_list*)

Scan a list of this Executor's files (targets or sources) for implicit dependencies and update all of the targets with them. This essentially short-circuits an N\*M scan of the sources for each individual target, which is a hell of a lot more efficient.

**get\_unignored\_sources**(*self*, *node*, *ignore*=())

**get\_implicit\_deps**(*self*)

Return the executor's implicit dependencies, i.e. the nodes of the commands to be executed.

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__subclasshook__()`

### 10.6.2 Properties

Name	Description
<code>action_list</code>	
<code>batches</code>	
<code>builder_kw</code>	
<code>env</code>	
<code>lvars</code>	
<code>overridelist</code>	
<code>post_actions</code>	
<code>pre_actions</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 10.7 Class NullEnvironment



### 10.7.1 Methods

<code>get_CacheDir(self)</code>
---------------------------------

*Inherited from SCons.Util.Null(Section 36.15)*

`__call__()`, `__delattr__()`, `__getattr__()`, `__init__()`, `__new__()`, `__nonzero__()`,  
`__repr__()`, `__setattr__()`

*Inherited from object*

`__format__()`, `__getattribute__()`, `__hash__()`, `__reduce__()`, `__reduce_ex__()`,  
`__sizeof__()`, `__str__()`, `__subclasshook__()`

### 10.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 10.8 Class Null



A null Executor, with a null build Environment, that does nothing when the rest of the methods call it.

This might be able to disappear when we refactor things to disassociate Builders from Nodes entirely, so we're not going to worry about unit tests for this--at least for now.

### 10.8.1 Methods

`__init__(self, *args, **kw)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature Overrides:  
`object.__init__` extit(inherited documentation)

`get_build_env(self)`

`get_build_scanner_path(self)`

`cleanup(self)`

`prepare(self)`

`get_unignored_sources(self, *args, **kw)`

`get_action_targetes(self)`

`get_action_list(self)`

`get_all_targetes(self)`

`get_all_sources(self)`

`get_all_children(self)`

`get_all_prerequisites(self)`

`get_action_side_effects(self)`

`__call__(self, *args, **kw)`

`get_contents(self)`

`add_pre_action(self, action)`

`add_post_action(self, action)`

<code>set_action_list(self, action)</code>
--

*Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 10.8.2 Properties

Name	Description
action_list	
batches	
builder_kw	
env	
lvars	
overridelist	
post_actions	
pre_actions	
<i>Inherited from object</i>	
__class__	

## 11 Module SCons.Job

### SCons.Job

This module defines the Serial and Parallel classes that execute tasks to complete a build. The Jobs class provides a higher level interface to start, stop, and wait on jobs.

#### 11.1 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Job.py rel_2.4.1:3453:73fef3ea0b0 2015...
<code>explicit_stack_size</code>	<b>Value:</b> None
<code>default_stack_size</code>	<b>Value:</b> 256
<code>interrupt_msg</code>	<b>Value:</b> 'Build interrupted.'
<code>__package__</code>	<b>Value:</b> 'SCons'

#### 11.2 Class InterruptState



##### 11.2.1 Methods

```
__init__(self)
```

`x.__init__(...)` initializes x; see `help(type(x))` for signature Overrides:  
`object.__init__` `exitit`(inherited documentation)

```
set(self)
```

```
__call__(self)
```

##### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 11.2.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

## 11.3 Class Jobs



An instance of this class initializes N jobs, and provides methods for starting, stopping, and waiting on all N jobs.

### 11.3.1 Methods

<b>__init__</b> ( <i>self</i> , <i>num</i> , <i>taskmaster</i> ) <hr/> <p>create 'num' jobs using the given taskmaster.</p> <p>If 'num' is 1 or less, then a serial job will be used, otherwise a parallel job with 'num' worker threads will be used.</p> <p>The 'num_jobs' attribute will be set to the actual number of jobs allocated. If more than one job is requested but the Parallel class can't do it, it gets reset to 1. Wrapping interfaces that care should check the value of 'num_jobs' after initialization. Overrides: object.__init__</p>
<b>run</b> ( <i>self</i> , <i>postfunc</i> =<function <lambda> at 0xb69e7994>) <hr/> <p>Run the jobs.</p> <p>postfunc() will be invoked after the jobs has run. It will be invoked even if the jobs are interrupted by a keyboard interrupt (well, in fact by a signal such as either SIGINT, SIGTERM or SIGHUP). The execution of postfunc() is protected against keyboard interrupts and is guaranteed to run to completion.</p>

<b>were_interrupted</b> ( <i>self</i> )
---

Returns whether the jobs were interrupted by a signal.
--

### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

#### 11.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 11.4 Class Serial

```
object └─ SCons.Job.Serial
```

This class is used to execute tasks in series, and is more efficient than Parallel, but is only appropriate for non-parallel builds. Only one instance of this class should be in existence at a time.

This class is not thread safe.

#### 11.4.1 Methods

<b>__init__</b> ( <i>self</i> , <i>taskmaster</i> )
---

Create a new serial job given a taskmaster.
---

The taskmaster's next_task() method should return the next task that needs to be executed, or None if there are no more tasks. The taskmaster's executed() method will be called for each task when it is successfully executed or failed() will be called if it failed to execute (e.g. execute() raised an exception). Overrides: object.__init__
---

**start**(*self*)

Start the job. This will begin pulling tasks from the taskmaster and executing them, and return when there are no more tasks. If a task fails to execute (i.e. `execute()` raises an exception), then the job will stop.

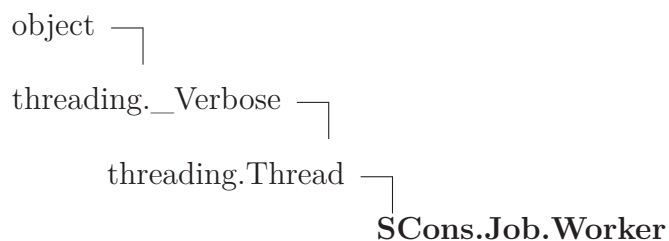
### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

#### 11.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 11.5 Class Worker



A worker thread waits on a task to be posted to its request queue, dequeues the task, executes it, and posts a tuple including the task and a boolean indicating whether the task executed successfully.



### 11.5.1 Methods

**\_\_init\_\_**(*self, requestQueue, resultsQueue, interrupted*)

This constructor should always be called with keyword arguments. Arguments are:

*group* should be None; reserved for future extension when a ThreadGroup class is implemented.

*target* is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

*args* is the argument tuple for the target invocation. Defaults to ().

*kwargs* is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread. Overrides: object.\_\_init\_\_ exitit(inherited documentation)

**run**(*self*)

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively. Overrides: threading.Thread.run exitit(inherited documentation)

#### *Inherited from threading.Thread*

\_\_repr\_\_(), getName(), isAlive(), isDaemon(), is\_alive(), join(), setDaemon(), setName(), start()

#### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

### 11.5.2 Properties

Name	Description
<i>Inherited from threading.Thread</i> daemon, ident, name	
<i>Inherited from object</i> __class__	

## 11.6 Class ThreadPool



This class is responsible for spawning and managing worker threads.

### 11.6.1 Methods

<b>__init__</b> ( <i>self, num, stack_size, interrupted</i> ) <hr/> <p>Create the request and reply queues, and 'num' worker threads.</p> <p>One must specify the stack size of the worker threads. The stack size is specified in kilobytes. Overrides: object.__init__</p>
<b>put</b> ( <i>self, task</i> ) <hr/> <p>Put task into request queue.</p>
<b>get</b> ( <i>self</i> ) <hr/> <p>Remove and return a result tuple from the results queue.</p>
<b>preparation_failed</b> ( <i>self, task</i> ) <hr/>

**cleanup**(*self*)

Shuts down the thread pool, giving each worker thread a chance to shut down gracefully.

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

#### 11.6.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

## 11.7 Class Parallel

object —  
**SCons.Job.Parallel**

This class is used to execute tasks in parallel, and is somewhat less efficient than Serial, but is appropriate for parallel builds.

This class is thread safe.

### 11.7.1 Methods

<b><code>__init__</code></b> ( <i>self</i> , <i>taskmaster</i> , <i>num</i> , <i>stack_size</i> )
<p>Create a new parallel job given a taskmaster.</p> <p>The taskmaster's <code>next_task()</code> method should return the next task that needs to be executed, or <code>None</code> if there are no more tasks. The taskmaster's <code>executed()</code> method will be called for each task when it is successfully executed or <code>failed()</code> will be called if the task failed to execute (i.e. <code>execute()</code> raised an exception).</p> <p>Note: calls to taskmaster are serialized, but calls to <code>execute()</code> on distinct tasks are not serialized, because that is the whole point of parallel jobs: they can execute multiple tasks simultaneously. Overrides: <code>object.__init__</code></p>
<b><code>start</code></b> ( <i>self</i> )
<p>Start the job. This will begin pulling tasks from the taskmaster and executing them, and return when there are no more tasks. If a task fails to execute (i.e. <code>execute()</code> raises an exception), then the job will stop.</p>

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

### 11.7.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

## 12 Module SCons.Memoize

### Memoizer

A decorator-based implementation to count hits and misses of the computed values that various methods cache in memory.

Use of this modules assumes that wrapped methods be coded to cache their values in a consistent way. In particular, it requires that the class uses a dictionary named "\_memo" to store the cached values.

Here is an example of wrapping a method that returns a computed value, with no input parameters:

```
@SCons.Memoize.CountMethodCall
def foo(self):

    try:
        return self._memo['foo']
    except KeyError:
        pass

    result = self.compute_foo_value()

    self._memo['foo'] = result

    return result
```

# Memoization  
# Memoization  
# Memoization  
# Memoization  
  
# Memoization

Here is an example of wrapping a method that will return different values based on one or more input arguments:

```
def _bar_key(self, argument):
    return argument

@SCons.Memoize.CountDictCall(_bar_key)
def bar(self, argument):

    memo_key = argument
    try:
        memo_dict = self._memo['bar']
    except KeyError:
        memo_dict = {}
        self._memo['dict'] = memo_dict
    else:
```

# Memoization  
# Memoization  
# Memoization  
# Memoization  
# Memoization  
# Memoization

```

        try:
            return memo_dict[memo_key]
        except KeyError:
            pass

    result = self.compute_bar_value(argument)

    memo_dict[memo_key] = result

    return result

```

Deciding what to cache is tricky, because different configurations can have radically different performance tradeoffs, and because the tradeoffs involved are often so non-obvious. Consequently, deciding whether or not to cache a given method will likely be more of an art than a science, but should still be based on available data from this module. Here are some VERY GENERAL guidelines about deciding whether or not to cache return values from a method that's being called a lot:

- The first question to ask is, "Can we change the calling code so this method isn't called so often?" Sometimes this can be done by changing the algorithm. Sometimes the *\*caller\** should be memoized, not the method you're looking at.
- The memoized function should be timed with multiple configurations to make sure it doesn't inadvertently slow down some other configuration.
- When memoizing values based on a dictionary key composed of input arguments, you don't need to use all of the arguments if some of them don't affect the return values.

## 12.1 Functions

<b>Dump</b> ( <i>title</i> =None)
Dump the hit/miss count for all the counters collected so far.

<b>EnableMemoization</b> ()
-----------------------------

**CountMethodCall(*fn*)**

Decorator for counting memoizer hits/misses while retrieving a simple value in a class method. It wraps the given method *fn* and uses a `CountValue` object to keep track of the caching statistics. Wrapping gets enabled by calling `EnableMemoization()`.

**CountDictCall(*keyfunc*)**

Decorator for counting memoizer hits/misses while accessing dictionary values with a key-generating function. Like `CountMethodCall` above, it wraps the given method *fn* and uses a `CountDict` object to keep track of the caching statistics. The dict-key function *keyfunc* has to get passed in the decorator call and gets stored in the `CountDict` instance. Wrapping gets enabled by calling `EnableMemoization()`.

## 12.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Memoize.py rel_2.4.1:3453:73fef3ea0b0 ...
<code>__doc__</code>	<b>Value:</b> ""Memoi...
<code>use_memoizer</code>	<b>Value:</b> None
<code>CounterList</code>	<b>Value:</b> {}
<code>__package__</code>	<b>Value:</b> None

## 12.3 Class Counter

object —  
    **SCons.Memoize.Counter**

**Known Subclasses:** `SCons.Memoize.CountDict`, `SCons.Memoize.CountValue`

Base class for counting memoization hits and misses.

We expect that the initialization in a matching decorator will fill in the correct class name and method name that represents the name of the function being counted.

### 12.3.1 Methods

```
__init__(self, cls_name, method_name)
```

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides:  
object.**\_\_init\_\_**

```
key(self)
```

```
display(self)
```

```
__cmp__(self, other)
```

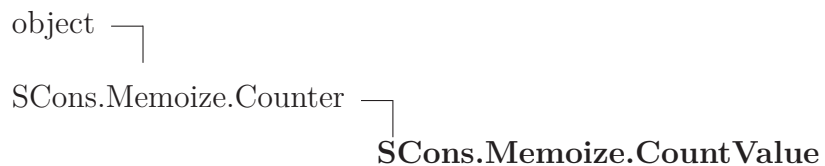
#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),  
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),  
__str__(), __subclasshook__()
```

### 12.3.2 Properties

Name	Description
<i>Inherited from object</i> <b>__class__</b>	

## 12.4 Class CountValue



A counter class for simple, atomic memoized values.

A CountValue object should be instantiated in a decorator for each of the class's methods that memoizes its return value by simply storing the return value in its `__memo` dictionary.



### 12.4.1 Methods

**count**(*self*, \**args*, \*\**kw*)

Counts whether the memoized value has already been set (a hit) or not (a miss).

*Inherited from SCons.Memoize.Counter(Section 12.3)*

\_\_cmp\_\_(), \_\_init\_\_(), display(), key()

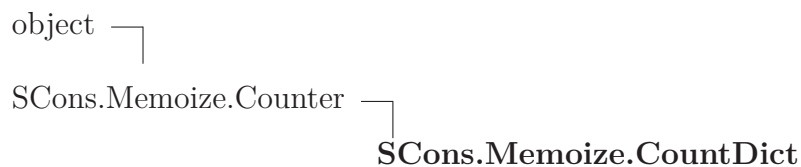
*Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
\_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
\_\_str\_\_(), \_\_subclasshook\_\_()

### 12.4.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 12.5 Class CountDict



A counter class for memoized values stored in a dictionary, with keys based on the method's input arguments.

A CountDict object is instantiated in a decorator for each of the class's methods that memoizes its return value in a dictionary, indexed by some key that can be computed from one or more of its input arguments.

### 12.5.1 Methods

<code>__init__(self, cls_name, method_name, keymaker)</code>
--

<code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code>
--

<code>count(self, *args, **kw)</code>
---------------------------------------

Counts whether the computed key value is already present in the memoization dictionary (a hit) or not (a miss).
---

***Inherited from SCons.Memoize.Counter(Section 12.3)***

`__cmp__()`, `display()`, `key()`

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

### 12.5.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

## 13 Package SCons.Node

SCons.Node

The Node package for the SCons software construction utility.

This is, in many ways, the heart of SCons.

A Node is where we encapsulate all of the dependency information about any thing that SCons can build, or about any thing which SCons can use to build some other thing. The canonical "thing," of course, is a file, but a Node can also represent something remote (like a web page) or something completely abstract (like an Alias).

Each specific type of "thing" is specifically represented by a subclass of the Node base class: Node.FS.File for files, Node.Alias for aliases, etc. Dependency information is kept here in the base class, and information specific to files/aliases/etc. is in the subclass. The goal, if we've done this correctly, is that any type of "thing" should be able to depend on any other type of "thing."

### 13.1 Modules

- **Alias:** `scons.Node.Alias`  
(Section 14, p. 131)
- **FS:** `scons.Node.FS`  
(Section 15, p. 137)
- **Python:** `scons.Node.Python`  
(Section 16, p. 185)

### 13.2 Functions

<code>classname(obj)</code>
-----------------------------

<code>Annotate(node)</code>
-----------------------------

<code>is_derived_none(node)</code>
------------------------------------

<code>is_derived_node(node)</code>
------------------------------------

Returns true if this node is derived (i.e. built).
--

<code>exists_none(node)</code>
--------------------------------

```
exists_always(node)
```

```
exists_base(node)
```

```
exists_entry(node)
```

Return if the Entry exists. Check the file system to see what we should turn into first. Assume a file if there's no directory.

```
exists_file(node)
```

```
rexists_none(node)
```

```
rexists_node(node)
```

```
rexists_base(node)
```

```
get_contents_none(node)
```

```
get_contents_entry(node)
```

Fetch the contents of the entry. Returns the exact binary contents of the file.

```
get_contents_dir(node)
```

Return content signatures and names of all our children separated by new-lines. Ensure that the nodes are sorted.

```
get_contents_file(node)
```

```
target_from_source_none(node, prefix, suffix, splitext)
```

```
target_from_source_base(node, prefix, suffix, splitext)
```

```
changed_since_last_build_node(node, target, prev_ni)
```

Must be overridden in a specific subclass to return True if this Node (a dependency) has changed since the last time it was used to build the specified target. *prev\_ni* is this Node's state (for example, its file timestamp, length, maybe content signature) as of the last time the target was built.

Note that this method is called through the dependency, not the target, because a dependency Node must be able to use its own logic to decide if it changed. For example, File Nodes need to obey if we're configured to use timestamps, but Python Value Nodes never use timestamps and always use the content. If this method were called through the target, then each Node's implementation of this method would have to have more complicated logic to handle all the different Node types on which it might depend.

```
changed_since_last_build_alias(node, target, prev_ni)
```

```
changed_since_last_build_entry(node, target, prev_ni)
```

```
changed_since_last_build_state_changed(node, target, prev_ni)
```

```
decide_source(node, target, prev_ni)
```

```
decide_target(node, target, prev_ni)
```

```
changed_since_last_build_python(node, target, prev_ni)
```

```
store_info_pass(node)
```

```
store_info_file(node)
```

```
get_children(node, parent)
```

```
ignore_cycle(node, stack)
```

```
do_nothing(node, parent)
```

### 13.3 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Node/__init__.py rel_2.4.1:3453:73fef3d3...
<code>print_duplicate</code>	<b>Value:</b> 0
<code>no_state</code>	<b>Value:</b> 0
<code>pending</code>	<b>Value:</b> 1
<code>executing</code>	<b>Value:</b> 2
<code>up_to_date</code>	<b>Value:</b> 3
<code>executed</code>	<b>Value:</b> 4
<code>failed</code>	<b>Value:</b> 5
<code>StateString</code>	<b>Value:</b> {0: 'no_state', 1: 'pending', 2: 'executing', 3: 'up_to_d...
<code>implicit_cache</code>	<b>Value:</b> 0
<code>implicit_deps_unchanged</code>	<b>Value:</b> 0
<code>implicit_deps_changed</code>	<b>Value:</b> 0
<code>interactive</code>	<b>Value:</b> False
<code>do_store_info</code>	<b>Value:</b> True
<code>store_info_map</code>	<b>Value:</b> {0: store_info_pass, 1: store_info_file}
<code>arg2nodes_lookups</code>	<b>Value:</b> [<bound method AliasNameSpace.lookup of {}>]
<code>__package__</code>	<b>Value:</b> 'SCons.Node'

### 13.4 Class NodeInfoBase



**Known Subclasses:** SCons.Node.Alias.AliasNodeInfo, SCons.Node.FS.DirNodeInfo, SCons.Node.FS.FileNodeInfo, SCons.Node.Python.ValueNodeInfo

The generic base class for signature information for a Node.

Node subclasses should subclass NodeInfoBase to provide their own logic for dealing with their own Node-specific signature information.

### 13.4.1 Methods

**\_\_getstate\_\_**(*self*)

Return all fields that shall be pickled. Walk the slots in the class hierarchy and add those to the state dictionary. If a '\_\_\_dict\_\_\_' slot is available, copy all entries to the dictionary. Also include the version id, which is fixed for all instances of a class.

**\_\_setstate\_\_**(*self*, *state*)

Restore the attributes from a pickled state. The version is discarded.

**convert**(*self*, *node*, *val*)

**format**(*self*, *field\_list*=None, *names*=0)

**merge**(*self*, *other*)

Merge the fields of another object into this object. Already existing information is overwritten by the other instance's data. WARNING: If a '\_\_\_dict\_\_\_' slot is added, it should be updated instead of replaced.

**update**(*self*, *node*)

#### *Inherited from object*

\_\_\_delattr\_\_\_(), \_\_\_format\_\_\_(), \_\_\_getattr\_\_\_(), \_\_\_hash\_\_\_(), \_\_\_init\_\_\_(),  
\_\_\_new\_\_\_(), \_\_\_reduce\_\_\_(), \_\_\_reduce\_ex\_\_\_(), \_\_\_repr\_\_\_(), \_\_\_setattr\_\_\_(),  
\_\_\_sizeof\_\_\_(), \_\_\_str\_\_\_(), \_\_\_subclasshook\_\_\_()

### 13.4.2 Properties

Name	Description
<i>Inherited from object</i> ___class___	

### 13.4.3 Class Variables

Name	Description
current_version_id	<b>Value:</b> 2

### 13.5 Class BuildInfoBase



**Known Subclasses:** SCons.Node.Alias.AliasBuildInfo, SCons.Node.FS.DirBuildInfo, SCons.Node.FS.FileBuildInfo, SCons.Node.Python.ValueBuildInfo

The generic base class for build information for a Node.

This is what gets stored in a .sconsign file for each target file. It contains a NodeInfo instance for this node (signature information that's specific to the type of Node) and direct attributes for the generic build stuff we have to track: sources, explicit dependencies, implicit dependencies, and action information.

### 13.5.1 Methods

<b><code>__getstate__</code></b> <i>(self)</i>	
Return all fields that shall be pickled. Walk the slots in the class hierarchy and add those to the state dictionary. If a ' <code>__dict__</code> ' slot is available, copy all entries to the dictionary. Also include the version id, which is fixed for all instances of a class.	
<b><code>__init__</code></b> <i>(self)</i>	
x. <code>__init__</code> (...) initializes x; see <code>help(type(x))</code> for signature Overrides: object. <code>__init__</code> <code>__init__(inherited documentation)</code>	
<b><code>__setstate__</code></b> <i>(self, state)</i>	
Restore the attributes from a pickled state.	



**merge**(*self*, *other*)

Merge the fields of another object into this object. Already existing information is overwritten by the other instance's data. WARNING: If a '\_\_\_dict\_\_\_' slot is added, it should be updated instead of replaced.

### *Inherited from object*

\_\_\_delattr\_\_\_(), \_\_\_format\_\_\_(), \_\_\_getattr\_\_\_(), \_\_\_hash\_\_\_(), \_\_\_new\_\_\_(),  
\_\_\_reduce\_\_\_(), \_\_\_reduce\_ex\_\_\_(), \_\_\_repr\_\_\_(), \_\_\_setattr\_\_\_(), \_\_\_sizeof\_\_\_(),  
\_\_\_str\_\_\_(), \_\_\_subclasshook\_\_\_()

### 13.5.2 Properties

Name	Description
bact	
bactsig	
bdepends	
bdependsigs	
bimplicit	
bimplicitigs	
bsources	
bsourcesigs	
<i>Inherited from object</i>	
___class___	

### 13.5.3 Class Variables

Name	Description
current_version_id	<b>Value:</b> 2

## 13.6 Class Node

object —  
    **SCons.Node.Node**

**Known Subclasses:** SCons.Node.Alias.Alias, SCons.Node.FS.Base, SCons.Node.Python.Value

The base Node class, for entities that we know how to build, or use to build other Nodes.

### 13.6.1 Methods

**Decider**(*self*, *function*)

**GetTag**(*self*, *key*)

Return a user-defined tag.

**Tag**(*self*, *key*, *value*)

Add a user-defined tag.

**\_\_init\_\_**(*self*)

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides: object.\_\_init\_\_ extit(inherited documentation)

**add\_dependency**(*self*, *depend*)

Adds dependencies.

**add\_ignore**(*self*, *depend*)

Adds dependencies to ignore.

**add\_prerequisite**(*self*, *prerequisite*)

Adds prerequisites

**add\_source**(*self*, *source*)

Adds sources.

**add\_to\_implicit**(*self*, *deps*)

**add\_to\_waiting\_parents**(*self*, *node*)

Returns the number of nodes added to our waiting parents list: 1 if we add a unique waiting parent, 0 if not. (Note that the returned values are intended to be used to increment a reference count, so don't think you can "clean up" this function by using True and False instead...)

**add\_to\_waiting\_s\_e**(*self*, *node*)**add\_wkid**(*self*, *wkid*)

Add a node to the list of kids waiting to be evaluated

**all\_children**(*self*, *scan*=1)

Return a list of all the node's direct children.

**alter\_targets**(*self*)

Return a list of alternate targets for this Node.

**build**(*self*, *\*\*kw*)

Actually build the node.

This is called by the Taskmaster after it's decided that the Node is out-of-date and must be rebuilt, and after the prepare() method has gotten everything, uh, prepared.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in built().

**builder\_set**(*self*, *builder*)

**built**(*self*)

Called just after this node is successfully built.

**changed**(*self*, *node=None*, *allowcache=False*)

Returns if the node is up-to-date with respect to the BuildInfo stored last time it was built. The default behavior is to compare it against our own previously stored BuildInfo, but the stored BuildInfo from another Node (typically one in a Repository) can be used instead.

Note that we now *always* check every dependency. We used to short-circuit the check by returning as soon as we detected any difference, but we now rely on checking every dependency to make sure that any necessary Node information (for example, the content signature of an `#included .h` file) is updated.

The `allowcache` option was added for supporting the early release of the executor/builder structures, right after a File target was built. When set to true, the return value of this `changed` method gets cached for File nodes. Like this, the executor isn't needed any longer for subsequent calls to `changed()`.

@see: `FS.File.changed()`, `FS.File.release_target_info()`

**children**(*self*, *scan=1*)

Return a list of the node's direct children, minus those that are ignored by this node.

**children\_are\_up\_to\_date**(*self*)

Alternate check for whether the Node is current: If all of our children were up-to-date, then this Node was up-to-date, too.

The `SCons.Node.Alias` and `SCons.Node.Python.Value` subclasses rebind their `current()` method to this method.

**clear**(*self*)

Completely clear a Node of all its cached state (so that it can be re-evaluated by interfaces that do continuous integration builds).

**clear\_memoized\_values**(*self*)**del\_binfo**(*self*)

Delete the build info from this node.

**disambiguate**(*self*, *must\_exist*=None)**env\_set**(*self*, *env*, *safe*=0)**executor\_cleanup**(*self*)

Let the executor clean up any cached information.

**exists**(*self*)

Does this node exists?

**explain**(*self*)**for\_signature**(*self*)

Return a string representation of the Node that will always be the same for this particular Node, no matter what. This is by contrast to the `__str__()` method, which might, for instance, return a relative path for a file Node. The purpose of this method is to generate a value to be used in signature calculation for the command line used to build a target, and we use this method instead of `str()` to avoid unnecessary rebuilds. This method does not need to return something that would actually work in a command line; it can return any kind of nonsense, so long as it does not change.

**get\_abspath(*self*)**

Return an absolute path to the Node. This will return simply str(Node) by default, but for Node types that have a concept of relative path, this might return something different.

**get\_binfo(*self*)**

Fetch a node's build information.

node - the node whose sources will be collected  
cache - alternate node to use for the signature  
returns - the build signature

This no longer handles the recursive descent of the node's children's signatures. We expect that they're already built and updated by someone else, if that's what's wanted.

**get\_build\_env(*self*)**

Fetch the appropriate Environment to build this node.

**get\_build\_scanner\_path(*self*, *scanner*)**

Fetch the appropriate scanner path for this node.

**get\_builder(*self*, *default\_builder*=None)**

Return the set builder, or a specified default value

**get\_cachedir\_csig(*self*)****get\_contents(*self*)**

Fetch the contents of the entry.

**get\_csig(*self*)**

**get\_env**(*self*)**get\_env\_scanner**(*self*, *env*, *kw*={})**get\_executor**(*self*, *create*=1)

Fetch the action executor for this node. Create one if there isn't already one, and requested to do so.

**get\_found\_includes**(*self*, *env*, *scanner*, *path*)

Return the scanned include lines (implicit dependencies) found in this node.

The default is no implicit dependencies. We expect this method to be overridden by any subclass that can be scanned for implicit dependencies.

**get\_implicit\_deps**(*self*, *env*, *scanner*, *path*)

Return a list of implicit dependencies for this node.

This method exists to handle recursive invocation of the scanner on the implicit dependencies returned by the scanner, if the scanner's recursive flag says that we should.

**get\_ninfo**(*self*)**get\_source\_scanner**(*self*, *node*)

Fetch the source scanner for the specified node

NOTE: "self" is the target being built, "node" is the source file for which we want to fetch the scanner.

Implies self.has\_builder() is true; again, expect to only be called from locations where this is already verified.

This function may be called very often; it attempts to cache the scanner found to improve performance.

**get\_state**(*self*)**get\_stored\_implicit**(*self*)

Fetch the stored implicit dependencies

**get\_stored\_info**(*self*)**get\_string**(*self*, *for\_signature*)

This is a convenience function designed primarily to be used in command generators (i.e., `CommandGeneratorActions` or `Environment` variables that are callable), which are called with a `for_signature` argument that is nonzero if the command generator is being called to generate a signature for the command line, which determines if we should rebuild or not.

Such command generators should use this method in preference to `str(Node)` when converting a `Node` to a string, passing in the `for_signature` parameter, such that we will call `Node.for_signature()` or `str(Node)` properly, depending on whether we are calculating a signature or actually constructing a command line.

**get\_subst\_proxy**(*self*)

This method is expected to return an object that will function exactly like this `Node`, except that it implements any additional special features that we would like to be in effect for `Environment` variable substitution. The principle use is that some `Nodes` would like to implement a `__getattr__()` method, but putting that in the `Node` type itself has a tendency to kill performance. We instead put it in a proxy and return it from this method. It is legal for this method to return `self` if no new functionality is needed for `Environment` substitution.

**get\_suffix**(*self*)**get\_target\_scanner**(*self*)



**has\_builder(*self*)**

Return whether this Node has a builder or not.

In Boolean tests, this turns out to be a *lot* more efficient than simply examining the builder attribute directly ("if node.builder: ..."). When the builder attribute is examined directly, it ends up calling `__getattr__` for both the `__len__` and `__nonzero__` attributes on instances of our Builder Proxy class(es), generating a bazillion extra calls and slowing things down immensely.

**has\_explicit\_builder(*self*)**

Return whether this Node has an explicit builder

This allows an internal Builder created by SCons to be marked non-explicit, so that it can be overridden by an explicit builder that the user supplies (the canonical example being directories).

**is\_derived(*self*)**

Returns true if this node is derived (i.e. built).

This should return true only for nodes whose path should be in the variant directory when `duplicate=0` and should contribute their build signatures when they are used as source files to other derived files. For example: source with source builders are not derived in this sense, and hence should not return true.

**is\_literal(*self*)**

Always pass the string representation of a Node to the command interpreter literally.

**is\_up\_to\_date(*self*)**

Default check for whether the Node is current: unknown Node subtypes are always out of date, so they will always get built.

**make\_ready**(*self*)

Get a Node ready for evaluation.

This is called before the Taskmaster decides if the Node is up-to-date or not. Overriding this method allows for a Node subclass to be disambiguated if necessary, or for an implicit source builder to be attached.

**missing**(*self*)**multiple\_side\_effect\_has\_builder**(*self*)

Return whether this Node has a builder or not.

In Boolean tests, this turns out to be a *lot* more efficient than simply examining the builder attribute directly ("if node.builder: ..."). When the builder attribute is examined directly, it ends up calling `__getattr__` for both the `__len__` and `__nonzero__` attributes on instances of our Builder Proxy class(es), generating a bazillion extra calls and slowing things down immensely.

**new\_binfo**(*self*)**new\_ninfo**(*self*)**postprocess**(*self*)

Clean up anything we don't need to hang onto after we've been built.

**prepare(*self*)**

Prepare for this Node to be built.

This is called after the Taskmaster has decided that the Node is out-of-date and must be rebuilt, but before actually calling the method to build the Node.

This default implementation checks that explicit or implicit dependencies either exist or are derived, and initializes the BuildInfo structure that will hold the information about how this node is, uh, built.

(The existence of source files is checked separately by the Executor, which aggregates checks for all of the targets built by a specific action.)

Overriding this method allows for for a Node subclass to remove the underlying file from the file system. Note that subclass methods should call this base class method to get the child check and the BuildInfo structure.

**push\_to\_cache(*self*)**

Try to push a node into a cache

**release\_target\_info(*self*)**

Called just after this node has been marked up-to-date or was built completely.

This is where we try to release as many target node infos as possible for clean builds and update runs, in order to minimize the overall memory consumption.

By purging attributes that aren't needed any longer after a Node (=File) got built, we don't have to care that much how many KBytes a Node actually requires...as long as we free the memory shortly afterwards.

@see: built() and File.release\_target\_info()

**remove(*self*)**

Remove this Node: no-op by default.

**render\_include\_tree**(*self*)

Return a text representation, suitable for displaying to the user, of the include tree for the sources of this node.

**reset\_executor**(*self*)

Remove cached executor; forces recompute when needed.

**retrieve\_from\_cache**(*self*)

Try to retrieve the node's content from a cache

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in `built()`.

Returns true if the node was successfully retrieved.

**rexists**(*self*)

Does this node exist locally or in a repository?

**scan**(*self*)

Scan this node's dependents for implicit dependencies.

**scanner\_key**(*self*)**select\_scanner**(*self*, *scanner*)

Selects a scanner for this Node.

This is a separate method so it can be overridden by Node subclasses (specifically, Node.FS.Dir) that *must* use their own Scanner and don't select one the Scanner.Selector that's configured for the target.

```
set_always_build(self, always_build=1)
```

Set the Node's `always_build` value.

```
set_executor(self, executor)
```

Set the action executor for this node.

```
set_explicit(self, is_explicit)
```

```
set_nocache(self, nocache=1)
```

Set the Node's `nocache` value.

```
set_noclean(self, noclean=1)
```

Set the Node's `noclean` value.

```
set_precious(self, precious=1)
```

Set the Node's `precious` value.

```
set_pseudo(self, pseudo=True)
```

Set the Node's `precious` value.

```
set_specific_source(self, source)
```

```
set_state(self, state)
```

```
visited(self)
```

Called just after this node has been visited (with or without a build).

***Inherited from object***

```

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

```

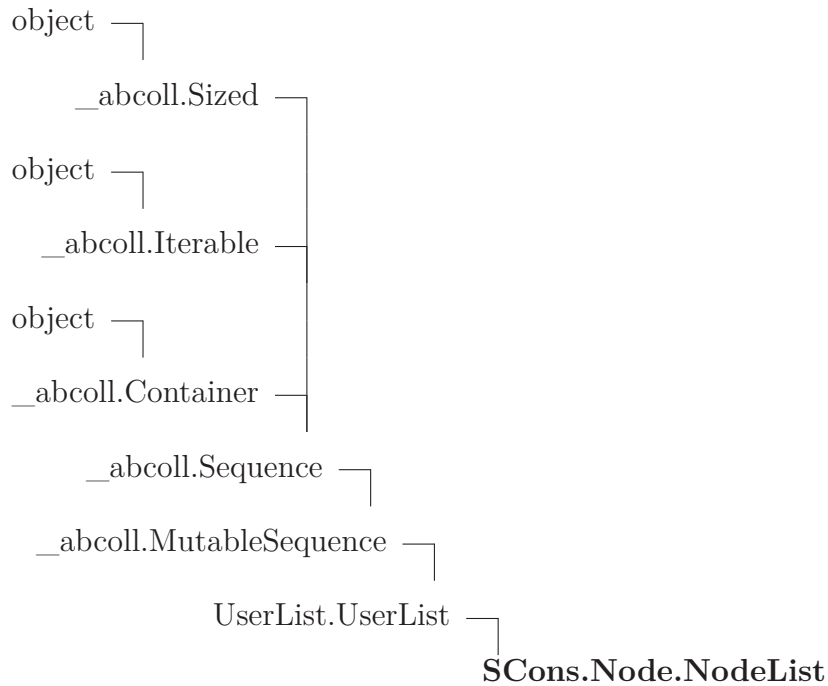
**13.6.2 Properties**

Name	Description
always_build	
attributes	
binfo	
builder	
cached	
changed_since_last_built	
depends	
depends_set	
env	
executor	
ignore	
ignore_set	
implicit	
implicit_set	
includes	
is_explicit	
linked	
ninfo	
nocache	
noclean	
precious	
prerequisites	
pseudo	
ref_count	
side_effect	
side_effects	
sources	
sources_set	
state	
store_info	
waiting_parents	
waiting_s_e	
wkids	
<i>Inherited from object</i>	

*continued on next page*

Name	Description
<code>__class__</code>	

### 13.7 Class *NodeList*



#### 13.7.1 Methods

<code>__str__(self)</code>
<code>str(x)</code> Overrides: <code>object.__str__</code> <code>exitit</code> (inherited documentation)

#### *Inherited from UserList.UserList*

`__add__()`, `__cmp__()`, `__contains__()`, `__delitem__()`, `__delslice__()`, `__eq__()`, `__ge__()`, `__getitem__()`, `__getslice__()`, `__gt__()`, `__iadd__()`, `__imul__()`, `__init__()`, `__le__()`, `__len__()`, `__lt__()`, `__mul__()`, `__ne__()`, `__radd__()`, `__repr__()`, `__rmul__()`, `__setitem__()`, `__setslice__()`, `append()`, `count()`, `extend()`, `index()`, `insert()`, `pop()`, `remove()`, `reverse()`, `sort()`

#### *Inherited from \_\_abcoll.Sequence*

`__iter__()`, `__reversed__()`

#### *Inherited from \_\_abcoll.Sized*

`__subclasshook__()`

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`,  
`__reduce_ex__()`, `__setattr__()`, `__sizeof__()`

### 13.7.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

### 13.7.3 Class Variables

Name	Description
<i>Inherited from UserList.UserList</i> <code>__abstractmethods__</code> , <code>__hash__</code>	

## 13.8 Class Walker

object —  
**SCons.Node.Walker**

An iterator for walking a Node tree.

This is depth-first, children are visited before the parent. The Walker object can be initialized with any node, and returns the next node on the descent with each `get_next()` call. 'kids\_func' is an optional function that will be called to get the children of a node instead of calling 'children'. 'cycle\_func' is an optional function that will be called when a cycle is detected.

This class does not get caught in node cycles caused, for example, by C header file include loops.



### 13.8.1 Methods

```
__init__(self, node, kids_func=<function get_children at
0xb69f4684>, cycle_func=<function ignore_cycle at 0xb69f46bc>,
eval_func=<function do_nothing at 0xb69f46f4>)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides:  
object.\_\_init\_\_ extit(inherited documentation)

```
get_next(self)
```

Return the next node for this walk of the tree.

This function is intentionally iterative, not recursive, to sidestep any issues of  
stack size limitations.

```
is_done(self)
```

#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 13.8.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 14 Module *SCons.Node.Alias*

*scons.Node.Alias*

Alias nodes.

This creates a hash of global Aliases (dummy targets).

### 14.1 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> <code>'src/engine/SCons/Node/Alias.py rel_2.4.1:3453:73fef3ea0...</code>
<code>default_ans</code>	<b>Value:</b> <code>{}</code>
<code>__package__</code>	<b>Value:</b> <code>'SCons.Node'</code>

### 14.2 Class *AliasNameSpace*

*UserDict.UserDict* —  
                                   *SCons.Node.Alias.AliasNameSpace*

#### 14.2.1 Methods

<b><i>Alias</i></b> ( <i>self</i> , <i>name</i> , ** <i>kw</i> )
--

<b><i>lookup</i></b> ( <i>self</i> , <i>name</i> , ** <i>kw</i> )
---

***Inherited from UserDict.UserDict***

`__cmp__()`, `__contains__()`, `__delitem__()`, `__getitem__()`, `__init__()`,  
`__len__()`, `__repr__()`, `__setitem__()`, `clear()`, `copy()`, `fromkeys()`, `get()`, `has_key()`,  
`items()`, `iteritems()`, `iterkeys()`, `itervalues()`, `keys()`, `pop()`, `popitem()`, `setdefault()`,  
`update()`, `values()`

#### 14.2.2 Class Variables

Name	Description
<i>Inherited from UserDict.UserDict</i> <code>__hash__</code>	

### 14.3 Class *AliasNodeInfo*



The generic base class for signature information for a Node.

Node subclasses should subclass *NodeInfoBase* to provide their own logic for dealing with their own Node-specific signature information.

#### 14.3.1 Methods

<b><code>str__to__node(self, s)</code></b>
--

<b><code>__getstate__(self)</code></b>
--

<p>Return all fields that shall be pickled. Walk the slots in the class hierarchy and add those to the state dictionary. If a '<code>__dict__</code>' slot is available, copy all entries to the dictionary. Also include the version id, which is fixed for all instances of a class. Overrides: <i>SCons.Node.NodeInfoBase.__getstate__</i></p>
---

<b><code>__setstate__(self, state)</code></b>
---

<p>Restore the attributes from a pickled state. Overrides: <i>SCons.Node.NodeInfoBase.__setstate__</i></p>
--

*Inherited from SCons.Node.NodeInfoBase(Section 13.4)*

`convert()`, `format()`, `merge()`, `update()`

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`,  
`__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`,  
`__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 14.3.2 Properties

Name	Description
<code>csig</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

### 14.3.3 Class Variables

Name	Description
<code>current_version_id</code>	<b>Value:</b> 2
<code>field_list</code>	<b>Value:</b> ['csig']

## 14.4 Class *AliasBuildInfo*



The generic base class for build information for a Node.

This is what gets stored in a .sconsign file for each target file. It contains a *NodeInfo* instance for this node (signature information that's specific to the type of Node) and direct attributes for the generic build stuff we have to track: sources, explicit dependencies, implicit dependencies, and action information.

### 14.4.1 Methods

*Inherited from SCons.Node.BuildInfoBase(Section 13.5)*

`__getstate__()`, `__init__()`, `__setstate__()`, `merge()`

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

### 14.4.2 Properties

Name	Description
<i>Inherited from SCons.Node.BuildInfoBase (Section 13.5)</i>	

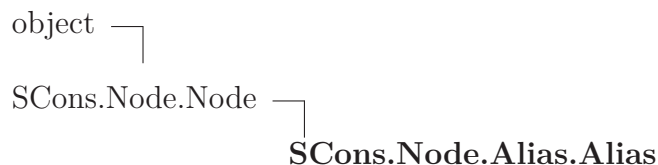
*continued on next page*

Name	Description
bact, bactsig, bdepends, bdependsigs, bimplicit, bimplicitsigs, bsources, bsourcesigs	
<i>Inherited from object</i>	
__class__	

#### 14.4.3 Class Variables

Name	Description
current_version_id	<b>Value:</b> 2

### 14.5 Class Alias



#### 14.5.1 Methods

**\_\_init\_\_**(*self*, *name*)

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides: object.**\_\_init\_\_** extit(inherited documentation)

**str\_for\_display**(*self*)

**\_\_str\_\_**(*self*)

str(x) Overrides: object.**\_\_str\_\_** extit(inherited documentation)

**make\_ready**(*self*)

Get a Node ready for evaluation.

This is called before the Taskmaster decides if the Node is up-to-date or not. Overriding this method allows for a Node subclass to be disambiguated if necessary, or for an implicit source builder to be attached. Overrides: SCons.Node.Node.**make\_ready** extit(inherited documentation)

**really\_\_build**(*self*, \*\**kw*)

Actually build the node.

This is called by the Taskmaster after it's decided that the Node is out-of-date and must be rebuilt, and after the prepare() method has gotten everything, uh, prepared.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in built().

**is\_\_up\_\_to\_\_date**(*self*)

Alternate check for whether the Node is current: If all of our children were up-to-date, then this Node was up-to-date, too.

The SCons.Node.Alias and SCons.Node.Python.Value subclasses rebind their current() method to this method. Overrides:  
SCons.Node.Node.is\_\_up\_\_to\_\_date

**is\_\_under**(*self*, *dir*)**get\_\_contents**(*self*)

The contents of an alias is the concatenation of the content signatures of all its sources. Overrides: SCons.Node.Node.get\_\_contents

**sconsign**(*self*)

An Alias is not recorded in .sconsign files

**build**(*self*)

A "builder" for aliases. Overrides: SCons.Node.Node.build

**convert**(*self*)

**get\_csig(*self*)**

Generate a node's content signature, the digested signature of its content.

node - the node cache - alternate node to use for the signature cache returns - the content signature Overrides: SCons.Node.Node.get\_csig

### ***Inherited from SCons.Node.Node (Section 13.6)***

Decider(), GetTag(), Tag(), add\_dependency(), add\_ignore(), add\_prerequisite(), add\_source(), add\_to\_implicit(), add\_to\_waiting\_parents(), add\_to\_waiting\_s\_e(), add\_wkid(), all\_children(), alter\_targets(), builder\_set(), built(), changed(), children(), children\_are\_up\_to\_date(), clear(), clear\_memoized\_values(), del\_binfo(), disambiguate(), env\_set(), executor\_cleanup(), exists(), explain(), for\_signature(), get\_abspath(), get\_binfo(), get\_build\_env(), get\_build\_scanner\_path(), get\_builder(), get\_cachedir\_csig(), get\_env(), get\_env\_scanner(), get\_executor(), get\_found\_includes(), get\_implicit\_deps(), get\_ninfo(), get\_source\_scanner(), get\_state(), get\_stored\_implicit(), get\_stored\_info(), get\_string(), get\_subst\_proxy(), get\_suffix(), get\_target\_scanner(), has\_builder(), has\_explicit\_builder(), is\_derived(), is\_literal(), missing(), multiple\_side\_effect\_has\_builder(), new\_binfo(), new\_ninfo(), postprocess(), prepare(), push\_to\_cache(), release\_target\_info(), remove(), render\_include\_tree(), reset\_executor(), retrieve\_from\_cache(), reexists(), scan(), scanner\_key(), select\_scanner(), set\_always\_build(), set\_executor(), set\_explicit(), set\_nocache(), set\_noclean(), set\_precious(), set\_pseudo(), set\_specific\_source(), set\_state(), visited()

### ***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

#### **14.5.2 Properties**

Name	Description
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	
always_build, attributes, binfo, builder, cached, changed_since_last_build, depends, depends_set, env, executor, ignore, ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo, nocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect, side_effects, sources, sources_set, state, store_info, waiting_parents, waiting_s_e, wkids	
<i>Inherited from object</i>	
__class__	

## 15 Module **SCons.Node.FS**

`scons.Node.FS`

File system nodes.

These Nodes represent the canonical external objects that people think of when they think of building software: files and directories.

This holds a "default\_fs" variable that should be initialized with an FS that can be used by scripts or modules looking for the canonical default.

### 15.1 Functions

<code>sconsign__none(<i>node</i>)</code>
--

<code>sconsign__dir(<i>node</i>)</code>
---

Return the .sconsign file info for this directory, creating it first if necessary.
--

<code>save__strings(<i>val</i>)</code>
--

<code>initialize__do__splitdrive()</code>
---

<code>needs__normpath__match(...)</code>
--

<code>match(string[, pos[, endpos]]) --&gt; match object or None. Matches zero or more characters at the beginning of the string</code>
---

<code>set__duplicate(<i>duplicate</i>)</code>
---

<code>LinkFunc(<i>target, source, env</i>)</code>
---

<code>LocalString(<i>target, source, env</i>)</code>
--

<code>UnlinkFunc(<i>target, source, env</i>)</code>
---

<code>MkdirFunc(<i>target, source, env</i>)</code>
--



```
get_MkdirBuilder()
```

```
get_DefaultSCCSBuilder()
```

```
get_DefaultRCSBuilder()
```

```
do_diskcheck_match(node, predicate, errorfmt)
```

```
ignore_diskcheck_match(node, predicate, errorfmt)
```

```
do_diskcheck_rcs(node, name)
```

```
ignore_diskcheck_rcs(node, name)
```

```
do_diskcheck_sccs(node, name)
```

```
ignore_diskcheck_sccs(node, name)
```

```
set_diskcheck(list)
```

```
diskcheck_types()
```

```
has_glob_magic(s)
```

```
get_default_fs()
```

**find\_file**(*filename*, *paths*, *verbose*=None)

`find_file(str, [Dir()]) -> [nodes]`

*filename* - a filename to find

*paths* - a list of directory path *\*nodes\** to search in. Can be represented as a list, a tuple, or a callable that is called with no arguments and returns the list or tuple.

*returns* - the node created from the found file.

Find a node corresponding to either a derived file or a file that exists already.

Only the first file found is returned, and none is returned if no file is found.

**invalidate\_node\_memos**(*targets*)

Invalidate the memoized values of all Nodes (files or directories) that are associated with the given entries. Has been added to clear the cache of nodes affected by a direct execution of an action (e.g. Delete/Copy/Chmod). Existing Node caches become inconsistent if the action is run through `Execute()`. The argument *targets* can be a single Node object or filename, or a sequence of Nodes/filenames.

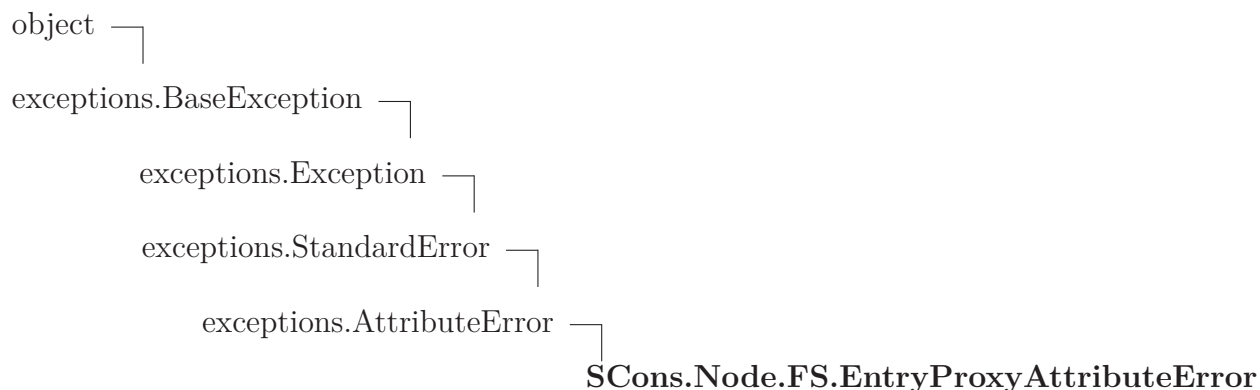
## 15.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Node/FS.py rel_2.4.1:3453:73fef3ea0b0 ...
<code>print_duplicate</code>	<b>Value:</b> 0
<code>default_max_drift</code>	<b>Value:</b> 172800
<code>Save_Strings</code>	<b>Value:</b> None
<code>do_splitdrive</code>	<b>Value:</b> False
<code>needs_normpath_check</code>	<b>Value:</b> <code>re.compile(r'(?x).*// (.*/)?\.\.(?:/ \$) \./ .*\.(?:/ \$)')</code>
<code>Valid_Duplicates</code>	<b>Value:</b> ['hard-soft-copy', 'soft-hard-copy', 'hard-copy', 'soft-c...

*continued on next page*

Name	Description
Link_Funcs	<b>Value:</b> []
Link	<b>Value:</b> SCons.Action.Action(LinkFunc, None)
LocalCopy	<b>Value:</b> SCons.Action.Action(LinkFunc, LocalString)
Unlink	<b>Value:</b> SCons.Action.Action(UnlinkFunc, None)
Mkdir	<b>Value:</b> SCons.Action.Action(MkdirFunc, None, presub= None)
MkdirBuilder	<b>Value:</b> None
DefaultSCCSBuilder	<b>Value:</b> None
DefaultRCSBuilder	<b>Value:</b> None
diskcheck_match	<b>Value:</b> DiskChecker('match', do_diskcheck_match, ignore_diskcheck...
diskcheck_rcs	<b>Value:</b> DiskChecker('rcs', do_diskcheck_rcs, ignore_diskcheck_rcs)
diskcheck_sccs	<b>Value:</b> DiskChecker('sccs', do_diskcheck_sccs, ignore_diskcheck_s...
diskcheckers	<b>Value:</b> [diskcheck_match, diskcheck_rcs, diskcheck_sccs,]
node_bwcomp	<b>Value:</b> {'abspath': <unbound method Base.get_abspath>, 'labspath'...
glob_magic_check	<b>Value:</b> re.compile(r'[\*\?\[]')
default_fs	<b>Value:</b> None
OS_SEP	<b>Value:</b> '/'
UNC_PREFIX	<b>Value:</b> '//'
__package__	<b>Value:</b> 'SCons.Node'
has_unc	<b>Value:</b> False
os_sep_is_slash	<b>Value:</b> True

### 15.3 Class `EntryProxyAttributeError`



An `AttributeError` subclass for recording and displaying the name of the underlying `Entry` involved in an `AttributeError` exception.

#### 15.3.1 Methods

```
__init__(self, entry_proxy, attribute)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature Overrides:  
`object.__init__` `exitit`(inherited documentation)

```
__str__(self)
```

`str(x)` Overrides: `object.__str__` `exitit`(inherited documentation)

*Inherited from `exceptions.AttributeError`*

```
__new__()
```

*Inherited from `exceptions.BaseException`*

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(),
__repr__(), __setattr__(), __setstate__(), __unicode__()
```

*Inherited from `object`*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

#### 15.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	

*continued on next page*

Name	Description
args, message	
<i>Inherited from object</i>	
__class__	

## 15.4 Class *DiskChecker*



### 15.4.1 Methods

<b>__init__</b> ( <i>self</i> , <i>type</i> , <i>do</i> , <i>ignore</i> )
x. <b>__init__</b> (...) initializes x; see help(type(x)) for signature Overrides: object. <b>__init__</b> extit(inherited documentation)

<b>__call__</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
--

<b>set</b> ( <i>self</i> , <i>list</i> )
--

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

### 15.4.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 15.5 Class EntryProxy



### 15.5.1 Methods

**\_\_str\_\_**(...)

A Python Descriptor class that delegates attribute fetches to an underlying wrapped subject of a Proxy. Typical use:

```
class Foo(Proxy): __str__ = Delegate('__str__')
```

Overrides: object.\_\_str\_\_

**\_\_getattr\_\_**(self, name)

Retrieve an attribute from the wrapped object. If the named attribute doesn't exist, AttributeError is raised. Overrides: SCons.Util.Proxy.\_\_getattr\_\_  
 extit(inherited documentation)

*Inherited from SCons.Util.Proxy(Section 36.5)*

\_\_cmp\_\_(), \_\_init\_\_(), get()

*Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattribute\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_subclasshook\_\_()

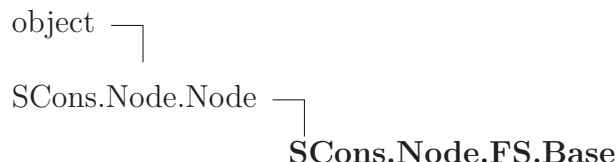
### 15.5.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

### 15.5.3 Class Variables

Name	Description
dictSpecialAttrs	<b>Value:</b> {"base": <code>__get_base_path</code> , "posix": <code>__get_posix_path</code> , "win...

## 15.6 Class Base



**Known Subclasses:** SCons.Node.FS.Dir, SCons.Node.FS.Entry, SCons.Node.FS.File

A generic class for file system entries. This class is for when we don't know yet whether the entry being looked up is a file or a directory. Instances of this class can morph into either Dir or File objects by a later, more precise lookup.

Note: this class does not define `__cmp__` and `__hash__` for efficiency reasons. SCons does a lot of comparing of Node.FS.{Base,Entry,File,Dir} objects, so those operations must be as fast as possible, which means we want to use Python's built-in object identity comparisons.

### 15.6.1 Methods

**`__init__(self, name, directory, fs)`**

Initialize a generic Node.FS.Base object.

Call the superclass initialization, take care of setting up our relative and absolute paths, identify our parent directory, and indicate that this node should use signatures. Overrides: `object.__init__`

**`str_for_display(self)`**

**`must_be_same(self, klass)`**

This node, which already existed, is being looked up as the specified klass. Raise an exception if it isn't.

**`get_dir(self)`**

**get\_suffix**(*self*)

Overrides: SCons.Node.Node.get\_suffix

**rfile**(*self*)

**\_\_getattr\_\_**(*self*, *attr*)

Together with the `node_bwcomp` dict defined below, this method provides a simple backward compatibility layer for the Node attributes 'abspath', 'labspath', 'path', 'tpath', 'suffix' and 'path\_elements'. These Node attributes used to be directly available in v2.3 and earlier, but have been replaced by getter methods that initialize the single variables lazily when required, in order to save memory. The redirection to the getters lets older Tools and SConstruct continue to work without any additional changes, fully transparent to the user. Note, that `__getattr__` is only called as fallback when the requested attribute can't be found, so there should be no speed performance penalty involved for standard builds.

**\_\_str\_\_**(*self*)

A Node.FS.Base object's string representation is its path name. Overrides: object.\_\_str\_\_

**rstr**(*self*)

A Node.FS.Base object's string representation is its path name.

**stat**(*self*)

**exists**(*self*)

Does this node exists? Overrides: SCons.Node.Node.exists exitit(inherited documentation)

**rexists**(*self*)

Does this node exist locally or in a repository? Overrides: SCons.Node.Node.rexists exitit(inherited documentation)



**getmtime**(*self*)**getsize**(*self*)**isdir**(*self*)**isfile**(*self*)**islink**(*self*)**is\_\_under**(*self*, *dir*)**set\_\_local**(*self*)**srcnode**(*self*)

If this node is in a build path, return the node corresponding to its source file. Otherwise, return `ourselves`.

**get\_\_path**(*self*, *dir*=None)

Return path relative to the current working directory of the `Node.FS.Base` object that owns us.

**set\_\_src\_\_builder**(*self*, *builder*)

Set the source code builder for this node.

**src\_\_builder**(*self*)

Fetch the source code builder for this node.

If there isn't one, we cache the source code builder specified for the directory (which in turn will cache the value from its parent directory, and so on up to the file system root).

**get\_abstractmethod(*self*)**

Get the absolute path of the file. Overrides: SCons.Node.Node.get\_abstractmethod

**get\_labspath(*self*)**

Get the absolute path of the file.

**get\_internal\_path(*self*)****get\_tpath(*self*)****get\_path\_elements(*self*)****for\_signature(*self*)**

Return a string representation of the Node that will always be the same for this particular Node, no matter what. This is by contrast to the `__str__()` method, which might, for instance, return a relative path for a file Node. The purpose of this method is to generate a value to be used in signature calculation for the command line used to build a target, and we use this method instead of `str()` to avoid unnecessary rebuilds. This method does not need to return something that would actually work in a command line; it can return any kind of nonsense, so long as it does not change. Overrides: SCons.Node.Node.for\_signature extit(inherited documentation)

**get\_subst\_proxy(*self*)**

This method is expected to return an object that will function exactly like this Node, except that it implements any additional special features that we would like to be in effect for Environment variable substitution. The principle use is that some Nodes would like to implement a `__getattr__()` method, but putting that in the Node type itself has a tendency to kill performance. We instead put it in a proxy and return it from this method. It is legal for this method to return `self` if no new functionality is needed for Environment substitution. Overrides: SCons.Node.Node.get\_subst\_proxy extit(inherited documentation)

**target\_from\_source**(*self*, *prefix*, *suffix*, *splittest*=<function splittest at 0xb6ba6f44>)

Generates a target entry that corresponds to this entry (usually a source file) with the specified prefix and suffix.

Note that this method can be overridden dynamically for generated files that need different behavior. See Tool/swig.py for an example.

**Rfindalldirs**(*self*, *pathlist*)

Return all of the directories for a given path list, including corresponding "backing" directories in any repositories.

The Node lookups are relative to this Node (typically a directory), so memoizing result saves cycles from looking up the same path for each target in a given directory.

**RDirs**(*self*, *pathlist*)

Search for a list of directories in the Repository list.

**rentry**(*self*)

### ***Inherited from SCons.Node.Node(Section 13.6)***

Decider(), GetTag(), Tag(), add\_dependency(), add\_ignore(), add\_prerequisite(), add\_source(), add\_to\_implicit(), add\_to\_waiting\_parents(), add\_to\_waiting\_source(), add\_wkid(), all\_children(), alter\_targets(), build(), builder\_set(), built(), changed(), children(), children\_are\_up\_to\_date(), clear(), clear\_memoized\_values(), del\_binfo(), disambiguate(), env\_set(), executor\_cleanup(), explain(), get\_binfo(), get\_build\_env(), get\_build\_scanner\_path(), get\_builder(), get\_cachedir\_csig(), get\_contents(), get\_csig(), get\_env(), get\_env\_scanner(), get\_executor(), get\_found\_includes(), get\_implicit\_deps(), get\_ninfo(), get\_source\_scanner(), get\_state(), get\_stored\_implicit(), get\_stored\_info(), get\_string(), get\_target\_scanner(), has\_builder(), has\_explicit\_builder(), is\_derived(), is\_literal(), is\_up\_to\_date(), make\_ready(), missing(), multiple\_side\_effect\_has\_built(), new\_binfo(), new\_ninfo(), postprocess(), prepare(), push\_to\_cache(), release\_target\_info(), remove(), render\_include\_tree(), reset\_executor(), retrieve\_from\_cache(), scan(), scanner\_key(), select\_scanner(), set\_always\_build(), set\_executor(), set\_explicit(), set\_nocache(), set\_noclean(), set\_precious(), set\_pseudo(), set\_specific\_source(), set\_state(), visited()

***Inherited from object***

```

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()

```

**15.6.2 Properties**

Name	Description
cwd	
dir	
duplicate	
sbuilder	
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	
always_build, attributes, binfo, builder, cached, changed_since_last_build, depends, depends_set, env, executor, ignore, ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo, nocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect, side_effects, sources, sources_set, state, store_info, waiting_parents, waiting_s_e, wkids	
<i>Inherited from object</i>	
__class__	

**15.6.3 Instance Variables**

Name	Description
name	
fs	

**15.7 Class Entry**

```

object └─
SCons.Node.Node └─
    SCons.Node.FS.Base └─
        SCons.Node.FS.Entry

```

This is the class for generic Node.FS entries--that is, things that could be a File or a Dir, but we're just not sure yet. Consequently, the methods in this class really exist just to transform their associated object into the right class when the time comes, and then call the

same-named method in the transformed class.

### 15.7.1 Methods

**\_\_init\_\_**(*self*, *name*, *directory*, *fs*)

Initialize a generic Node.FS.Base object.

Call the superclass initialization, take care of setting up our relative and absolute paths, identify our parent directory, and indicate that this node should use signatures. Overrides: object.\_\_init\_\_ exitit(inherited documentation)

**diskcheck\_\_match**(*self*)

**disambiguate**(*self*, *must\_exist*=None)

Overrides: SCons.Node.Node.disambiguate

**rfile**(*self*)

We're a generic Entry, but the caller is actually looking for a File at this point, so morph into one. Overrides: SCons.Node.FS.Base.rfile

**scanner\_\_key**(*self*)

Overrides: SCons.Node.Node.scanner\_\_key

**get\_\_contents**(*self*)

Fetch the contents of the entry. Returns the exact binary contents of the file. Overrides: SCons.Node.Node.get\_\_contents

**get\_\_text\_\_contents**(*self*)

Fetch the decoded text contents of a Unicode encoded Entry.

Since this should return the text contents from the file system, we check to see into what sort of subclass we should morph this Entry.

**must\_be\_same**(*self*, *klass*)

Called to make sure a Node is a Dir. Since we're an Entry, we can morph into one. Overrides: SCons.Node.FS.Base.must\_be\_same

**exists**(*self*)

Does this node exists? Overrides: SCons.Node.Node.exists extit(inherited documentation)

**rel\_path**(*self*, *other*)

**new\_ninfo**(*self*)

Overrides: SCons.Node.Node.new\_ninfo

**get\_subst\_proxy**(*self*)

This method is expected to return an object that will function exactly like this Node, except that it implements any additional special features that we would like to be in effect for Environment variable substitution. The principle use is that some Nodes would like to implement a `__getattr__()` method, but putting that in the Node type itself has a tendency to kill performance. We instead put it in a proxy and return it from this method. It is legal for this method to return self if no new functionality is needed for Environment substitution. Overrides: SCons.Node.Node.get\_subst\_proxy extit(inherited documentation)

### *Inherited from SCons.Node.FS.Base(Section 15.6)*

RDirs(), Rfindalldirs(), `__getattr__()`, `__str__()`, `for_signature()`, `get_abspath()`, `get_dir()`, `get_internal_path()`, `get_labspath()`, `get_path()`, `get_path_elements()`, `get_suffix()`, `get_tpath()`, `getmtime()`, `getsize()`, `is_under()`, `isdir()`, `isfile()`, `islink()`, `rentry()`, `rexists()`, `rstr()`, `set_local()`, `set_src_builder()`, `src_builder()`, `srcnode()`, `stat()`, `str_for_display()`, `target_from_source()`

### *Inherited from SCons.Node.Node(Section 13.6)*

Decider(), GetTag(), Tag(), `add_dependency()`, `add_ignore()`, `add_prerequisite()`, `add_source()`, `add_to_implicit()`, `add_to_waiting_parents()`, `add_to_waiting_s_e()`, `add_wkid()`, `all_children()`, `alter_targets()`, `build()`, `builder_set()`, `built()`, `changed()`, `children()`, `children_are_up_to_date()`, `clear()`, `clear_memoized_values()`, `del_binfo()`, `env_set()`, `executor_cleanup()`, `explain()`, `get_binfo()`, `get_build_env()`, `get_build_scanner_path()`, `get_builder()`, `get_cachedir_csig()`, `get_csig()`, `get_env()`, `get_env_scanner()`,

get\_executor(), get\_found\_includes(), get\_implicit\_deps(), get\_ninfo(), get\_source\_scanner(),  
 get\_state(), get\_stored\_implicit(), get\_stored\_info(), get\_string(), get\_target\_scanner(),  
 has\_builder(), has\_explicit\_builder(), is\_derived(), is\_literal(), is\_up\_to\_date(),  
 make\_ready(), missing(), multiple\_side\_effect\_has\_builder(), new\_binfo(), post-  
 process(), prepare(), push\_to\_cache(), release\_target\_info(), remove(), render\_include\_tree(),  
 reset\_executor(), retrieve\_from\_cache(), scan(), select\_scanner(), set\_always\_build(),  
 set\_executor(), set\_explicit(), set\_nocache(), set\_noclean(), set\_precious(), set\_pseudo(),  
 set\_specific\_source(), set\_state(), visited()

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_subclasshook\_\_()

### 15.7.2 Properties

Name	Description
cachedir_csig	
cachesig	
contentsig	
dirname	
entries	
on_disk_entries	
rcs_dir	
released_target_info	
repositories	
root	
scanner_paths	
sccs_dir	
searched	
srcdir	
variant_dirs	
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i>	
cwd, dir, duplicate, sbuilder	
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	
always_build, attributes, binfo, builder, cached, changed_since_last_build, depends, depends_set, env, executor, ignore, ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo, nocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect, side_effects, sources, sources_set, state, store_info, waiting_parents, waiting_s_e, wkids	
<i>Inherited from object</i>	
__class__	

### 15.7.3 Instance Variables

Name	Description
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i> fs, name	

## 15.8 Class LocalFS

object —  
SCons.Node.FS.LocalFS

**Known Subclasses:** SCons.Node.FS.FS

### 15.8.1 Methods

**chmod**(*self*, *path*, *mode*)

**copy**(*self*, *src*, *dst*)

**copy2**(*self*, *src*, *dst*)

**exists**(*self*, *path*)

**getmtime**(*self*, *path*)

**getsize**(*self*, *path*)

**isdir**(*self*, *path*)

**isfile**(*self*, *path*)

**link**(*self*, *src*, *dst*)

**lstat**(*self*, *path*)

**listdir**(*self*, *path*)

**makedirs**(*self*, *path*)



mkdir(*self*, *path*)

rename(*self*, *old*, *new*)

stat(*self*, *path*)

symlink(*self*, *src*, *dst*)

open(*self*, *path*)

unlink(*self*, *path*)

islink(*self*, *path*)

readlink(*self*, *file*)

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_init\_\_(),  
 \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(),  
 \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

#### 15.8.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 15.9 Class FS



## 15.9.1 Methods

---

**\_\_init\_\_**(*self*, *path*=None)

---

Initialize the Node.FS subsystem.

The supplied path is the top of the source tree, where we expect to find the top-level build file. If no path is supplied, the current directory is the default.

The path argument must be a valid absolute path. Overrides:  
object.\_\_init\_\_

---

**set\_SConstruct\_dir**(*self*, *dir*)

---



---

**get\_max\_drift**(*self*)

---



---

**set\_max\_drift**(*self*, *max\_drift*)

---



---

**getcwd**(*self*)

---



---

**chdir**(*self*, *dir*, *change\_os\_dir*=0)

---

Change the current working directory for lookups. If *change\_os\_dir* is true, we will also change the "real" cwd to match.

---

**get\_root**(*self*, *drive*)

---

Returns the root directory for the specified drive, creating it if necessary.

---

**Entry**(*self*, *name*, *directory*=None, *create*=1)

---

Look up or create a generic Entry node with the specified name. If the name is a relative path (begins with ./, ../, or a file name), then it is looked up relative to the supplied directory node, or to the top level directory of the FS (supplied at construction time) if no directory is supplied.

**File**(*self*, *name*, *directory*=None, *create*=1)

Look up or create a File node with the specified name. If the name is a relative path (begins with ./, ../, or a file name), then it is looked up relative to the supplied directory node, or to the top level directory of the FS (supplied at construction time) if no directory is supplied.

This method will raise TypeError if a directory is found at the specified path.

**Dir**(*self*, *name*, *directory*=None, *create*=True)

Look up or create a Dir node with the specified name. If the name is a relative path (begins with ./, ../, or a file name), then it is looked up relative to the supplied directory node, or to the top level directory of the FS (supplied at construction time) if no directory is supplied.

This method will raise TypeError if a normal file is found at the specified path.

**VariantDir**(*self*, *variant\_dir*, *src\_dir*, *duplicate*=1)

Link the supplied variant directory to the source directory for purposes of building files.

**Repository**(*self*, \**dirs*)

Specify Repository directories to search.

**variant\_dir\_target\_climb**(*self*, *orig*, *dir*, *tail*)

Create targets in corresponding variant directories

Climb the directory tree, and look up path names relative to any linked variant directories we find.

Even though this loops and walks up the tree, we don't memoize the return value because this is really only used to process the command-line targets.

```
Glob(self, pathname, ondisk=True, source=True, strings=False,
      exclude=None, cwd=None)
```

Globs

This is mainly a shim layer

### *Inherited from SCons.Node.FS.LocalFS(Section 15.8)*

chmod(), copy(), copy2(), exists(), getmtime(), getsize(), isdir(), isfile(), islink(),  
link(), listdir(), lstat(), makedirs(), mkdir(), open(), readlink(), rename(), stat(),  
symlink(), unlink()

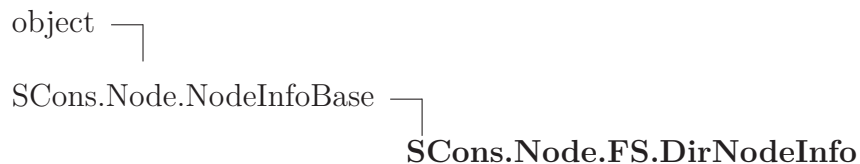
### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
\_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
\_\_str\_\_(), \_\_subclasshook\_\_()

#### 15.9.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 15.10 Class DirNodeInfo



The generic base class for signature information for a Node.

Node subclasses should subclass NodeInfoBase to provide their own logic for dealing with their own Node-specific signature information.

### 15.10.1 Methods

```
str_to_node(self, s)
```

***Inherited from SCons.Node.NodeInfoBase(Section 13.4)***

\_\_getstate\_\_(), \_\_setstate\_\_(), convert(), format(), merge(), update()

***Inherited from object***

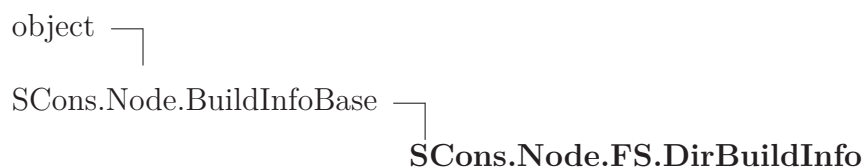
\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_init\_\_(),  
 \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(),  
 \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

**15.10.2 Properties**

Name	Description
<i>Inherited from object</i>	
__class__	

**15.10.3 Class Variables**

Name	Description
current_version_id	<b>Value:</b> 2
fs	<b>Value:</b> None

**15.11 Class DirBuildInfo**

The generic base class for build information for a Node.

This is what gets stored in a .sconsign file for each target file. It contains a NodeInfo instance for this node (signature information that's specific to the type of Node) and direct attributes for the generic build stuff we have to track: sources, explicit dependencies, implicit dependencies, and action information.

**15.11.1 Methods*****Inherited from SCons.Node.BuildInfoBase(Section 13.5)***

\_\_getstate\_\_(), \_\_init\_\_(), \_\_setstate\_\_(), merge()

***Inherited from object***

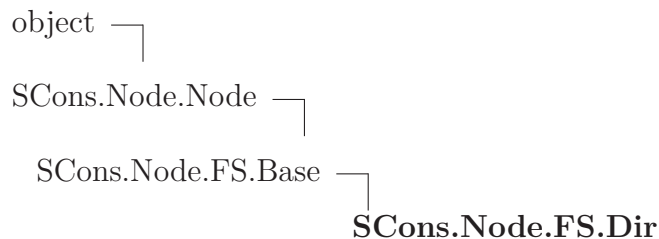
\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

**15.11.2 Properties**

Name	Description
<i>Inherited from SCons.Node.BuildInfoBase (Section 13.5)</i>	bact, bactsig, bdepends, bdependsigs, bimplicit, bimplicitsigs, bsources, bsourcesigs
<i>Inherited from object</i>	__class__

**15.11.3 Class Variables**

Name	Description
current_version_id	<b>Value:</b> 2

**15.12 Class Dir**

**Known Subclasses:** SCons.Node.FS.RootDir

A class for directories in a file system.

## 15.12.1 Methods

**\_\_init\_\_**(*self*, *name*, *directory*, *fs*)

Initialize a generic Node.FS.Base object.

Call the superclass initialization, take care of setting up our relative and absolute paths, identify our parent directory, and indicate that this node should use signatures. Overrides: object.\_\_init\_\_ exitit(inherited documentation)

**diskcheck\_\_match**(*self*)

**Entry**(*self*, *name*)

Looks up or creates an entry node named 'name' relative to this directory.

**Dir**(*self*, *name*, *create=True*)

Looks up or creates a directory node named 'name' relative to this directory.

**File**(*self*, *name*)

Looks up or creates a file node named 'name' relative to this directory.

**link**(*self*, *srcdir*, *duplicate*)

Set this directory as the variant directory for the supplied source directory.

**getRepositories**(*self*)

Returns a list of repositories for this directory.

**get\_\_all\_\_rdirs**(*self*)

**addRepository**(*self*, *dir*)

**up**(*self*)

**rel\_path**(*self*, *other*)

Return a path to "other" relative to this directory.

**get\_env\_scanner**(*self*, *env*, *kw*={})

Overrides: SCons.Node.Node.get\_env\_scanner

**get\_target\_scanner**(*self*)

Overrides: SCons.Node.Node.get\_target\_scanner

**get\_found\_includes**(*self*, *env*, *scanner*, *path*)

Return this directory's implicit dependencies.

We don't bother caching the results because the scan typically shouldn't be requested more than once (as opposed to scanning .h file contents, which can be requested as many times as the file is `#included` by other files).

Overrides: SCons.Node.Node.get\_found\_includes

**prepare**(*self*)

Prepare for this Node to be built.

This is called after the Taskmaster has decided that the Node is out-of-date and must be rebuilt, but before actually calling the method to build the Node.

This default implementation checks that explicit or implicit dependencies either exist or are derived, and initializes the BuildInfo structure that will hold the information about how this node is, uh, built.

(The existence of source files is checked separately by the Executor, which aggregates checks for all of the targets built by a specific action.)

Overriding this method allows for for a Node subclass to remove the underlying file from the file system. Note that subclass methods should call this base class method to get the child check and the BuildInfo structure.

Overrides: SCons.Node.Node.prepare extit(inherited documentation)



---

**build**(*self*, \*\**kw*)

---

A null "builder" for directories. Overrides: SCons.Node.Node.build

---

**multiple\_side\_effect\_has\_builder**(*self*)

Return whether this Node has a builder or not.

In Boolean tests, this turns out to be a *lot* more efficient than simply examining the builder attribute directly ("if node.builder: ..."). When the builder attribute is examined directly, it ends up calling `__getattr__` for both the `__len__` and `__nonzero__` attributes on instances of our Builder Proxy class(es), generating a bazillion extra calls and slowing things down immensely. Overrides: SCons.Node.Node.multiple\_side\_effect\_has\_builder  
extit(inherited documentation)

---

**alter\_targets**(*self*)

---

Return any corresponding targets in a variant directory. Overrides: SCons.Node.Node.alter\_targets

---

**scanner\_key**(*self*)

---

A directory does not get scanned. Overrides: SCons.Node.Node.scanner\_key

---

**get\_text\_contents**(*self*)

---

We already emit things in text, so just return the binary version.

---

**get\_contents**(*self*)

---

Return content signatures and names of all our children separated by new-lines. Ensure that the nodes are sorted. Overrides: SCons.Node.Node.get\_contents

**get\_csig**(*self*)

Compute the content signature for Directory nodes. In general, this is not needed and the content signature is not stored in the `DirNodeInfo`. However, if `get_contents` on a `Dir` node is called which has a child directory, the child directory should return the hash of its contents. Overrides: `SCons.Node.Node.get_csig`

**do\_duplicate**(*self*, *src*)**is\_up\_to\_date**(*self*)

If any child is not up-to-date, then this directory isn't, either. Overrides: `SCons.Node.Node.is_up_to_date`

**rdir**(*self*)**sconsign**(*self*)

Return the `.sconsign` file info for this directory.

**srcnode**(*self*)

`Dir` has a special need for `srcnode()`...if we have a `srcdir` attribute set, then that *is* our `srcnode`. Overrides: `SCons.Node.FS.Base.srcnode`

**get\_timestamp**(*self*)

Return the latest timestamp from among our children

**get\_abspath**(*self*)

Get the absolute path of the file. Overrides: `SCons.Node.Node.get_abspath`

**get\_labspath**(*self*)

Get the absolute path of the file. Overrides:  
SCons.Node.FS.Base.get\_labspath

**get\_internal\_path**(*self*)

Overrides: SCons.Node.FS.Base.get\_internal\_path

**get\_tpath**(*self*)

Overrides: SCons.Node.FS.Base.get\_tpath

**get\_path\_elements**(*self*)

Overrides: SCons.Node.FS.Base.get\_path\_elements

**entry\_abspath**(*self*, *name*)

**entry\_labspath**(*self*, *name*)

**entry\_path**(*self*, *name*)

**entry\_tpath**(*self*, *name*)

**entry\_exists\_on\_disk**(*self*, *name*)

Searches through the file/dir entries of the current directory, and returns True if a physical entry with the given name could be found.

@see reentry\_exists\_on\_disk

**reentry\_exists\_on\_disk**(*self*, *name*)

Searches through the file/dir entries of the current *and* all its remote directories (repos), and returns True if a physical entry with the given name could be found. The local directory (*self*) gets searched first, so repositories take a lower precedence regarding the searching order.

@see entry\_exists\_on\_disk

```
srcdir__list(self)
```

```
srcdir__duplicate(self, name)
```

```
srcdir__find__file(self, filename)
```

```
dir__on__disk(self, name)
```

```
file__on__disk(self, name)
```

```
walk(self, func, arg)
```

Walk this directory tree by calling the specified function for each directory in the tree.

This behaves like the `os.path.walk()` function, but for in-memory `Node.FS.Dir` objects. The function takes the same arguments as the functions passed to `os.path.walk()`:

```
func(arg, dirname, fnames)
```

Except that "dirname" will actually be the directory *Node*, not the string. The `'.'` and `'..'` entries are excluded from `fnames`. The `fnames` list may be modified in-place to filter the subdirectories visited or otherwise impose a specific order. The "arg" argument is always passed to `func()` and may be used in any way (or ignored, passing `None` is common).

```
glob(self, pathname, ondisk=True, source=False, strings=False,
      exclude=None)
```

Returns a list of Nodes (or strings) matching a specified pathname pattern.

Pathname patterns follow UNIX shell semantics: \* matches any-length strings of any characters, ? matches any character, and [] can enclose lists or ranges of characters. Matches do not span directory separators.

The matches take into account Repositories, returning local Nodes if a corresponding entry exists in a Repository (either an in-memory Node or something on disk).

By default, the glob() function matches entries that exist on-disk, in addition to in-memory Nodes. Setting the "ondisk" argument to False (or some other non-true value) causes the glob() function to only match in-memory Nodes. The default behavior is to return both the on-disk and in-memory Nodes.

The "source" argument, when true, specifies that corresponding source Nodes must be returned if you're globbing in a build directory (initialized with VariantDir()). The default behavior is to return Nodes local to the VariantDir().

The "strings" argument, when true, returns the matches as strings, not Nodes. The strings are path names relative to this directory.

The "exclude" argument, if not None, must be a pattern or a list of patterns following the same UNIX shell semantics. Elements matching a least one pattern of this list will be excluded from the result.

The underlying algorithm is adapted from the glob.glob() function in the Python library (but heavily modified), and uses fnmatch() under the covers.

### ***Inherited from SCons.Node.FS.Base(Section 15.6)***

```
RDirs(), Rfindalldirs(), __getattr__(), __str__(), exists(), for_signature(), get_dir(),
get_path(), get_subst_proxy(), get_suffix(), getmtime(), getsize(), is_under(),
isdir(), isfile(), islink(), must_be_same(), reentry(), reexists(), rfile(), rstr(), set_local(),
set_src_builder(), src_builder(), stat(), str_for_display(), target_from_source()
```

### ***Inherited from SCons.Node.Node(Section 13.6)***

```
Decider(), GetTag(), Tag(), add_dependency(), add_ignore(), add_prerequisite(),
add_source(), add_to_implicit(), add_to_waiting_parents(), add_to_waiting_s_e(),
add_wkid(), all_children(), builder_set(), built(), changed(), children(), children_are_up_to_date(),
clear(), clear_memoized_values(), del_binfo(), disambiguate(), env_set(), execu-
```

```

tor_cleanup(), explain(), get_binfo(), get_build_env(), get_build_scanner_path(),
get_builder(), get_cachedir_csig(), get_env(), get_executor(), get_implicit_deps(),
get_ninfo(), get_source_scanner(), get_state(), get_stored_implicit(), get_stored_info(),
get_string(), has_builder(), has_explicit_builder(), is_derived(), is_literal(), make_ready(),
missing(), new_binfo(), new_ninfo(), postprocess(), push_to_cache(), release_target_info(),
remove(), render_include_tree(), reset_executor(), retrieve_from_cache(), scan(),
select_scanner(), set_always_build(), set_executor(), set_explicit(), set_nocache(),
set_noclean(), set_precious(), set_pseudo(), set_specific_source(), set_state(),
visited()

```

### *Inherited from object*

```

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()

```

#### 15.12.2 Properties

Name	Description
cachedir_csig	
cachesig	
contentsig	
dirname	
entries	
on_disk_entries	
rcs_dir	
released_target_info	
repositories	
root	
scanner_paths	
sccs_dir	
searched	
srcdir	
variant_dirs	
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i>	
cwd, dir, duplicate, sbuilder	
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	
always_build, attributes, binfo, builder, cached, changed_since_last_build, depends, depends_set, env, executor, ignore, ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo, nocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect, side_effects, sources, sources_set, state, store_info, waiting_parents, waiting_s_e, wkids	
<i>Inherited from object</i>	

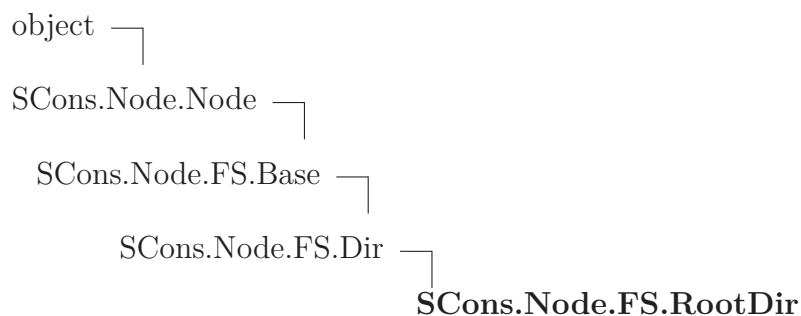
*continued on next page*

Name	Description
<code>__class__</code>	

### 15.12.3 Instance Variables

Name	Description
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i> <code>fs, name</code>	

## 15.13 Class RootDir



A class for the root directory of a file system.

This is the same as a Dir class, except that the path separator ('/' or '\') is actually part of the name, so we don't need to add a separator when creating the path names of entries within this directory.

### 15.13.1 Methods

<p><b><code>__init__</code></b>(<i>self, drive, fs</i>)</p> <p>Initialize a generic Node.FS.Base object.</p> <p>Call the superclass initialization, take care of setting up our relative and absolute paths, identify our parent directory, and indicate that this node should use signatures. Overrides: object.<code>__init__</code> extit(inherited documentation)</p>
---

**must\_be\_same**(*self*, *klass*)

This node, which already existed, is being looked up as the specified klass. Raise an exception if it isn't. Overrides: SCons.Node.FS.Base.must\_be\_same exitit(inherited documentation)

**\_\_str\_\_**(*self*)

A Node.FS.Base object's string representation is its path name. Overrides: object.\_\_str\_\_ exitit(inherited documentation)

**entry\_abspath**(*self*, *name*)

Overrides: SCons.Node.FS.Dir.entry\_abspath

**entry\_labspath**(*self*, *name*)

Overrides: SCons.Node.FS.Dir.entry\_labspath

**entry\_path**(*self*, *name*)

Overrides: SCons.Node.FS.Dir.entry\_path

**entry\_tpath**(*self*, *name*)

Overrides: SCons.Node.FS.Dir.entry\_tpath

**is\_under**(*self*, *dir*)

Overrides: SCons.Node.FS.Base.is\_under

**up**(*self*)

Overrides: SCons.Node.FS.Dir.up

**get\_dir**(*self*)

Overrides: SCons.Node.FS.Base.get\_dir

**src\_builder**(*self*)

Fetch the source code builder for this node.

If there isn't one, we cache the source code builder specified for the directory (which in turn will cache the value from its parent directory, and so on up to the file system root). Overrides: SCons.Node.FS.Base.src\_builder exitit(inherited documentation)



***Inherited from SCons.Node.FS.Dir(Section 15.12)***

Dir(), Entry(), File(), addRepository(), alter\_targets(), build(), dir\_on\_disk(), diskcheck\_match(), do\_duplicate(), entry\_exists\_on\_disk(), file\_on\_disk(), getRepositories(), get\_abspath(), get\_all\_rdirs(), get\_contents(), get\_csig(), get\_env\_scanner(), get\_found\_includes(), get\_internal\_path(), get\_labspath(), get\_path\_elements(), get\_target\_scanner(), get\_text\_contents(), get\_timestamp(), get\_tpath(), glob(), is\_up\_to\_date(), link(), multiple\_side\_effect\_has\_builder(), prepare(), rdir(), rel\_path(), reentry\_exists\_on\_disk(), scanner\_key(), sconsign(), srcdir\_duplicate(), srcdir\_find\_file(), srcdir\_list(), srcnode(), walk()

***Inherited from SCons.Node.FS.Base(Section 15.6)***

RDirs(), Rfindalldirs(), \_\_getattr\_\_(), exists(), for\_signature(), get\_path(), get\_subst\_proxy(), get\_suffix(), getmtime(), getsize(), isdir(), isfile(), islink(), reentry(), reexists(), rfile(), rstr(), set\_local(), set\_src\_builder(), stat(), str\_for\_display(), target\_from\_source()

***Inherited from SCons.Node.Node(Section 13.6)***

Decider(), GetTag(), Tag(), add\_dependency(), add\_ignore(), add\_prerequisite(), add\_source(), add\_to\_implicit(), add\_to\_waiting\_parents(), add\_to\_waiting\_s\_e(), add\_wkid(), all\_children(), builder\_set(), built(), changed(), children(), children\_are\_up\_to\_date(), clear(), clear\_memoized\_values(), del\_binfo(), disambiguate(), env\_set(), executor\_cleanup(), explain(), get\_binfo(), get\_build\_env(), get\_build\_scanner\_path(), get\_builder(), get\_cachedir\_csig(), get\_env(), get\_executor(), get\_implicit\_deps(), get\_ninfo(), get\_source\_scanner(), get\_state(), get\_stored\_implicit(), get\_stored\_info(), get\_string(), has\_builder(), has\_explicit\_builder(), is\_derived(), is\_literal(), make\_ready(), missing(), new\_binfo(), new\_ninfo(), postprocess(), push\_to\_cache(), release\_target\_info(), remove(), render\_include\_tree(), reset\_executor(), retrieve\_from\_cache(), scan(), select\_scanner(), set\_always\_build(), set\_executor(), set\_explicit(), set\_nocache(), set\_noclean(), set\_precious(), set\_pseudo(), set\_specific\_source(), set\_state(), visited()

***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

**15.13.2 Properties**

Name	Description
<i>Inherited from SCons.Node.FS.Dir (Section 15.12)</i>	
cachedir_csig, cachesig, contentsig, dirname, entries, on_disk_entries, rcs_dir, released_target_info, repositories, root, scanner_paths, sccs_dir, searched, srcdir, variant_dirs	

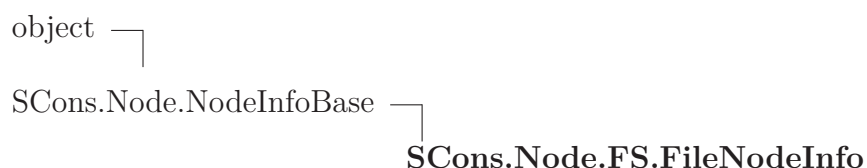
*continued on next page*

Name	Description
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i> cwd, dir, duplicate, sbuilder	
<i>Inherited from SCons.Node.Node (Section 13.6)</i> always_build, attributes, binfo, builder, cached, changed_since_last_build, depends, depends_set, env, executor, ignore, ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo, nocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect, side_effects, sources, sources_set, state, store_info, waiting_parents, waiting_s_e, wkids	
<i>Inherited from object</i> __class__	

### 15.13.3 Instance Variables

Name	Description
name	Filename with extension as it was specified when the object was created; to obtain filesystem path, use Python str() function
fs	Reference to parent Node.FS object

## 15.14 Class *FileInfo*



The generic base class for signature information for a Node.

Node subclasses should subclass NodeInfoBase to provide their own logic for dealing with their own Node-specific signature information.

### 15.14.1 Methods

<code>str_to_node(self, s)</code>
-----------------------------------

<b><code>__getstate__</code></b> ( <i>self</i> )
Return all fields that shall be pickled. Walk the slots in the class hierarchy and add those to the state dictionary. If a ' <code>__dict__</code> ' slot is available, copy all entries to the dictionary. Also include the version id, which is fixed for all instances of a class. Overrides: SCons.Node.NodeInfoBase. <code>__getstate__</code>
<b><code>__setstate__</code></b> ( <i>self</i> , <i>state</i> )
Restore the attributes from a pickled state. Overrides: SCons.Node.NodeInfoBase. <code>__setstate__</code>

***Inherited from SCons.Node.NodeInfoBase(Section 13.4)***

`convert()`, `format()`, `merge()`, `update()`

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`,  
`__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`,  
`__sizeof__()`, `__str__()`, `__subclasshook__()`

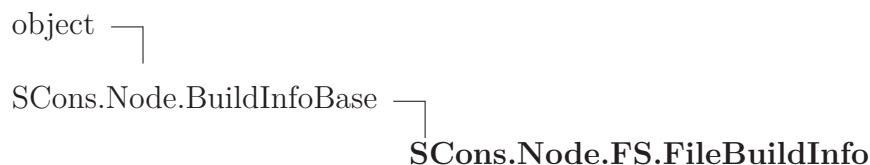
### 15.14.2 Properties

Name	Description
<code>csig</code>	
<code>size</code>	
<code>timestamp</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

### 15.14.3 Class Variables

Name	Description
<code>current_version_id</code>	<b>Value:</b> 2
<code>field_list</code>	<b>Value:</b> ['csig', 'timestamp', 'size']
<code>fs</code>	<b>Value:</b> None

## 15.15 Class FileBuildInfo



**Known Subclasses:** SCons.SConf.SConfBuildInfo

The generic base class for build information for a Node.

This is what gets stored in a .sconsign file for each target file. It contains a NodeInfo instance for this node (signature information that's specific to the type of Node) and direct attributes for the generic build stuff we have to track: sources, explicit dependencies, implicit dependencies, and action information.

### 15.15.1 Methods

**convert\_from\_sconsign**(*self*, *dir*, *name*)

Converts a newly-read FileBuildInfo object for in-SCons use

For normal up-to-date checking, we don't have any conversion to perform--but we're leaving this method here to make that clear.

**convert\_to\_sconsign**(*self*)

Converts this FileBuildInfo object for writing to a .sconsign file

This replaces each Node in our various dependency lists with its usual string representation: relative to the top-level SConstruct directory, or an absolute path if it's outside.

**format**(*self*, *names*=0)

```
prepare_dependencies(self)
```

Prepares a FileBuildInfo object for explaining what changed

The bsources, bdepends and bimplicit lists have all been stored on disk as paths relative to the top-level SConstruct directory. Convert the strings to actual Nodes (for use by the --debug=explain code and --implicit-cache).

**Inherited from SCons.Node.BuildInfoBase(Section 13.5)**

```
__getstate__(), __init__(), __setstate__(), merge()
```

**Inherited from object**

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

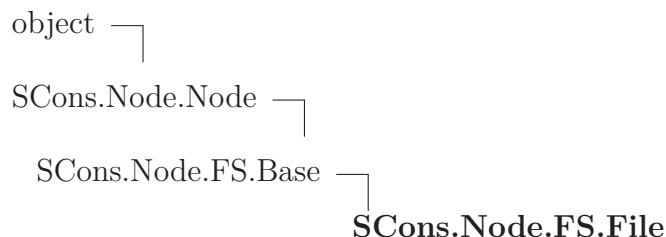
### 15.15.2 Properties

Name	Description
<i>Inherited from SCons.Node.BuildInfoBase (Section 13.5)</i>	bact, bactsig, bdepends, bdependsigns, bimplicit, bimplicitsigns, bsources, bsourcesigns
<i>Inherited from object</i>	__class__

### 15.15.3 Class Variables

Name	Description
current_version_id	<b>Value:</b> 2

## 15.16 Class File



A class for files in a file system.

### 15.16.1 Methods

**diskcheck\_\_match**(*self*)

**\_\_init\_\_**(*self, name, directory, fs*)

Initialize a generic Node.FS.Base object.

Call the superclass initialization, take care of setting up our relative and absolute paths, identify our parent directory, and indicate that this node should use signatures. Overrides: object.\_\_init\_\_ extit(inherited documentation)

**Entry**(*self, name*)

Create an entry node named 'name' relative to the directory of this file.

**Dir**(*self, name, create=True*)

Create a directory node named 'name' relative to the directory of this file.

**Dirs**(*self, pathlist*)

Create a list of directories relative to the SConscript directory of this file.

**File**(*self, name*)

Create a file node named 'name' relative to the directory of this file.

**scanner\_\_key**(*self*)

Overrides: SCons.Node.Node.scanner\_\_key

**get\_contents**(*self*)

Fetch the contents of the entry. Overrides: SCons.Node.Node.get\_contents  
extit(inherited documentation)

**get\_text\_contents**(*self*)**get\_content\_hash**(*self*)

Compute and return the MD5 hash for this file.

**get\_size**(*self*)**get\_timestamp**(*self*)**convert\_old\_entry**(*self*, *old\_entry*)**get\_stored\_info**(*self*)

Overrides: SCons.Node.Node.get\_stored\_info

**get\_stored\_implicit**(*self*)

Fetch the stored implicit dependencies Overrides:  
SCons.Node.Node.get\_stored\_implicit extit(inherited documentation)

**rel\_path**(*self*, *other*)**get\_found\_includes**(*self*, *env*, *scanner*, *path*)

Return the included implicit dependencies in this file. Cache results so we only scan the file once per path regardless of how many times this information is requested. Overrides: SCons.Node.Node.get\_found\_includes

**push\_to\_cache**(*self*)

Try to push the node into a cache Overrides:  
SCons.Node.Node.push\_to\_cache

**retrieve\_from\_cache**(*self*)

Try to retrieve the node's content from a cache

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in built().

Returns true if the node was successfully retrieved. Overrides:  
SCons.Node.Node.retrieve\_from\_cache

**visited**(*self*)

Called just after this node has been visited (with or without a build).

Overrides: SCons.Node.Node.visited extit(inherited documentation)

**release\_target\_info**(*self*)

Called just after this node has been marked up-to-date or was built completely.

This is where we try to release as many target node infos as possible for clean builds and update runs, in order to minimize the overall memory consumption.

We'd like to remove a lot more attributes like self.sources and self.sources\_set, but they might get used in a next build step. For example, during configuration the source files for a built \*.o file are used to figure out which linker to use for the resulting Program (gcc vs. g++)! That's why we check for the 'keep\_targetinfo' attribute, config Nodes and the Interactive mode just don't allow an early release of most variables.

In the same manner, we can't simply remove the self.attributes here. The smart linking relies on the shared flag, and some parts of the java Tool use it to transport information about nodes...

@see: built() and Node.release\_target\_info() Overrides:  
SCons.Node.Node.release\_target\_info

**find\_src\_builder**(*self*)



**has\_src\_builder(*self*)**

Return whether this Node has a source builder or not.

If this Node doesn't have an explicit source code builder, this is where we figure out, on the fly, if there's a transparent source code builder for it.

Note that if we found a source builder, we also set the `self.builder` attribute, so that all of the methods that actually *build* this file don't have to do anything different.

**alter\_targets(*self*)**

Return any corresponding targets in a variant directory. Overrides: `SCons.Node.Node.alter_targets`

**make\_ready(*self*)**

Get a Node ready for evaluation.

This is called before the Taskmaster decides if the Node is up-to-date or not. Overriding this method allows for a Node subclass to be disambiguated if necessary, or for an implicit source builder to be attached. Overrides: `SCons.Node.Node.make_ready` `exitit`(inherited documentation)

**prepare(*self*)**

Prepare for this file to be created. Overrides: `SCons.Node.Node.prepare`

**remove(*self*)**

Remove this file. Overrides: `SCons.Node.Node.remove`

**do\_duplicate(*self*, *src*)****exists(*self*)**

Does this node exists? Overrides: `SCons.Node.Node.exists` `exitit`(inherited documentation)

**get\_max\_drift\_csig**(*self*)

Returns the content signature currently stored for this node if it's been unmodified longer than the max\_drift value, or the max\_drift value is 0. Returns None otherwise.

**get\_csig**(*self*)

Generate a node's content signature, the digested signature of its content.

node - the node cache - alternate node to use for the signature cache returns - the content signature Overrides: SCons.Node.Node.get\_csig

**builder\_set**(*self*, *builder*)

Overrides: SCons.Node.Node.builder\_set

**built**(*self*)

Called just after this File node is successfully built.

Just like for 'release\_target\_info' we try to release some more target node attributes in order to minimize the overall memory consumption.

@see: release\_target\_info Overrides: SCons.Node.Node.built

**changed**(*self*, *node=None*, *allowcache=False*)

Returns if the node is up-to-date with respect to the BuildInfo stored last time it was built.

For File nodes this is basically a wrapper around Node.changed(), but we allow the return value to get cached after the reference to the Executor got released in release\_target\_info().

@see: Node.changed() Overrides: SCons.Node.Node.changed

**changed\_content**(*self*, *target*, *prev\_ni*)

**changed\_state**(*self*, *target*, *prev\_ni*)

**changed\_timestamp\_then\_content**(*self*, *target*, *prev\_ni*)

**changed\_timestamp\_newer**(*self*, *target*, *prev\_ni*)

**changed\_timestamp\_match**(*self*, *target*, *prev\_ni*)

**is\_up\_to\_date**(*self*)

Default check for whether the Node is current: unknown Node subtypes are always out of date, so they will always get built. Overrides: SCons.Node.Node.is\_up\_to\_date exitit(inherited documentation)

**rfile**(*self*)

Overrides: SCons.Node.FS.Base.rfile

**rstr**(*self*)

A Node.FS.Base object's string representation is its path name. Overrides: SCons.Node.FS.Base.rstr exitit(inherited documentation)

**get\_cachedir\_csig**(*self*)

Fetch a Node's content signature for purposes of computing another Node's cachesig.

This is a wrapper around the normal get\_csig() method that handles the somewhat obscure case of using CacheDir with the -n option. Any files that don't exist would normally be "built" by fetching them from the cache, but the normal get\_csig() method will try to open up the local file, which doesn't exist because the -n option meant we didn't actually pull the file from cachedir. But since the file *does* actually exist in the cachedir, we can use its contents for the csig. Overrides: SCons.Node.Node.get\_cachedir\_csig

**get\_contents\_sig**(*self*)

A helper method for get\_cachedir\_bsig.

It computes and returns the signature for this node's contents.

```
get_cachedir_bsig(self)
```

Return the signature for a cached file, including its children.

It adds the path of the cached file to the cache signature, because multiple targets built by the same action will all have the same build signature, and we have to differentiate them somehow.

***Inherited from SCons.Node.FS.Base(Section 15.6)***

RDirs(), Rfindalldirs(), \_\_getattr\_\_(), \_\_str\_\_(), for\_signature(), get\_abspath(), get\_dir(), get\_internal\_path(), get\_labspath(), get\_path(), get\_path\_elements(), get\_subst\_proxy(), get\_suffix(), get\_tpath(), getmtime(), getsize(), is\_under(), isdir(), isfile(), islink(), must\_be\_same(), reentry(), rexists(), set\_local(), set\_src\_builder(), src\_builder(), srcnode(), stat(), str\_for\_display(), target\_from\_source()

***Inherited from SCons.Node.Node(Section 13.6)***

Decider(), GetTag(), Tag(), add\_dependency(), add\_ignore(), add\_prerequisite(), add\_source(), add\_to\_implicit(), add\_to\_waiting\_parents(), add\_to\_waiting\_s\_e(), add\_wkid(), all\_children(), build(), children(), children\_are\_up\_to\_date(), clear(), clear\_memoized\_values(), del\_binfo(), disambiguate(), env\_set(), executor\_cleanup(), explain(), get\_binfo(), get\_build\_env(), get\_build\_scanner\_path(), get\_builder(), get\_env(), get\_env\_scanner(), get\_executor(), get\_implicit\_deps(), get\_ninfo(), get\_source\_scanner(), get\_state(), get\_string(), get\_target\_scanner(), has\_builder(), has\_explicit\_builder(), is\_derived(), is\_literal(), missing(), multiple\_side\_effect\_has\_builder(), new\_binfo(), new\_ninfo(), postprocess(), render\_include\_tree(), reset\_executor(), scan(), select\_scanner(), set\_always\_build(), set\_executor(), set\_explicit(), set\_nocache(), set\_noclean(), set\_precious(), set\_pseudo(), set\_specific\_source(), set\_state()

***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

**15.16.2 Properties**

Name	Description
cachedir_csig	
cachesig	
contentsig	
dirname	
entries	

*continued on next page*

Name	Description
on_disk_entries	
rcs_dir	
released_target_info	
repositories	
root	
scanner_paths	
sccs_dir	
searched	
srcdir	
variant_dirs	
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i>	
cwd, dir, duplicate, sbuilder	
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	
always_build, attributes, binfo, builder, cached, changed_since_last_build, depends, depends_set, env, executor, ignore, ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo, nocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect, side_effects, sources, sources_set, state, store_info, waiting_parents, waiting_s_e, wkids	
<i>Inherited from object</i>	
__class__	

### 15.16.3 Class Variables

Name	Description
md5_chunksize	<b>Value:</b> 64
convert_copy_attrs	<b>Value:</b> ['bsources', 'bimplicit', 'bdepends', 'bact', 'bactsig', ...]
convert_sig_attrs	<b>Value:</b> ['bsourcesigs', 'bimplicitsigs', 'bdependssigs']

### 15.16.4 Instance Variables

Name	Description
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i>	
fs, name	

## 15.17 Class FileFinder

```
object └─ SCons.Node.FS.FileFinder
```

### 15.17.1 Methods

```
__init__(self)
```

*x*.**\_\_init\_\_**(...) initializes *x*; see help(type(*x*)) for signature Overrides: object.**\_\_init\_\_** **exitit**(inherited documentation)

```
filedir_lookup(self, p, fd=None)
```

A helper method for **find\_file()** that looks up a directory for a file we're trying to find. This only creates the Dir Node if it exists on-disk, since if the directory doesn't exist we know we won't find any files in it... :-)

It would be more compact to just use this as a nested function with a default keyword argument (see the commented-out version below), but that doesn't work unless you have nested scopes, so we define it here just so this work under Python 1.5.2.

```
find_file(self, filename, paths, verbose=None)
```

```
find_file(str, [Dir()]) -> [nodes]
```

*filename* - a filename to find

*paths* - a list of directory path \*nodes\* to search in. Can be represented as a list, a tuple, or a callable that is called with no arguments and returns the list or tuple.

**returns** - the node created from the found file.

Find a node corresponding to either a derived file or a file that exists already.

Only the first file found is returned, and none is returned if no file is found.

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

**15.17.2 Properties**

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

## 16 Module SCons.Node.Python

scons.Node.Python

Python nodes.

### 16.1 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> <code>'src/engine/SCons/Node/Python.py rel_2.4.1:3453:73fef3ea...'</code>
<code>__package__</code>	<b>Value:</b> <code>'SCons.Node'</code>

### 16.2 Class ValueNodeInfo



The generic base class for signature information for a Node.

Node subclasses should subclass NodeInfoBase to provide their own logic for dealing with their own Node-specific signature information.

#### 16.2.1 Methods

<b><code>str_to_node(self, s)</code></b>
--

<b><code>__getstate__(self)</code></b>
--

<p>Return all fields that shall be pickled. Walk the slots in the class hierarchy and add those to the state dictionary. If a <code>'__dict__'</code> slot is available, copy all entries to the dictionary. Also include the version id, which is fixed for all instances of a class. Overrides: <code>SCons.Node.NodeInfoBase.__getstate__</code></p>
---



<code>__setstate__(self, state)</code>
Restore the attributes from a pickled state. Overrides: SCons.Node.NodeInfoBase.__setstate__

***Inherited from SCons.Node.NodeInfoBase(Section 13.4)***

convert(), format(), merge(), update()

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`,  
`__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`,  
`__sizeof__()`, `__str__()`, `__subclasshook__()`

### 16.2.2 Properties

Name	Description
csig	
<i>Inherited from object</i>	
<code>__class__</code>	

### 16.2.3 Class Variables

Name	Description
current_version_id	<b>Value:</b> 2
field_list	<b>Value:</b> ['csig']

## 16.3 Class ValueBuildInfo

object └

SCons.Node.BuildInfoBase └

**SCons.Node.Python.ValueBuildInfo**

The generic base class for build information for a Node.

This is what gets stored in a .sconsign file for each target file. It contains a NodeInfo instance for this node (signature information that's specific to the type of Node) and direct attributes for the generic build stuff we have to track: sources, explicit dependencies, implicit dependencies, and action information.

### 16.3.1 Methods

#### *Inherited from SCons.Node.BuildInfoBase(Section 13.5)*

`__getstate__()`, `__init__()`, `__setstate__()`, `merge()`

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

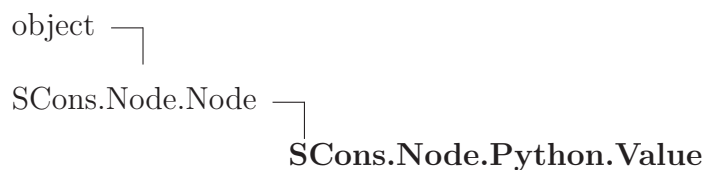
### 16.3.2 Properties

Name	Description
<i>Inherited from SCons.Node.BuildInfoBase (Section 13.5)</i>	
	bact, bactsig, bdepends, bdependsigns, bimplicit, bimplicitsigns, bsources, bsourcesigns
<i>Inherited from object</i>	
<code>__class__</code>	

### 16.3.3 Class Variables

Name	Description
<code>current_version_id</code>	<b>Value: 2</b>

## 16.4 Class Value



A class for Python variables, typically passed on the command line or generated by a script, but not from a file or some other source.

## 16.4.1 Methods

**\_\_init\_\_**(*self*, *value*, *built\_value*=None)

*x*.**\_\_init\_\_**(...) initializes *x*; see help(type(*x*)) for signature Overrides: object.**\_\_init\_\_** extit(inherited documentation)

**str\_for\_display**(*self*)

**\_\_str\_\_**(*self*)

str(*x*) Overrides: object.**\_\_str\_\_** extit(inherited documentation)

**make\_ready**(*self*)

Get a Node ready for evaluation.

This is called before the Taskmaster decides if the Node is up-to-date or not. Overriding this method allows for a Node subclass to be disambiguated if necessary, or for an implicit source builder to be attached. Overrides: SCons.Node.Node.make\_ready extit(inherited documentation)

**build**(*self*, \*\**kw*)

Actually build the node.

This is called by the Taskmaster after it's decided that the Node is out-of-date and must be rebuilt, and after the prepare() method has gotten everything, uh, prepared.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in built(). Overrides: SCons.Node.Node.build extit(inherited documentation)

**is\_up\_to\_date**(*self*)

Alternate check for whether the Node is current: If all of our children were up-to-date, then this Node was up-to-date, too.

The SCons.Node.Alias and SCons.Node.Python.Value subclasses rebind their current() method to this method. Overrides: SCons.Node.Node.is\_up\_to\_date

**is\_under**(*self*, *dir*)

**write**(*self*, *built\_value*)

Set the value of the node.

**read**(*self*)

Return the value. If necessary, the value is built.

**get\_text\_contents**(*self*)

By the assumption that the node.built\_value is a deterministic product of the sources, the contents of a Value are the concatenation of all the contents of its sources. As the value need not be built when get\_contents() is called, we cannot use the actual node.built\_value.

**get\_contents**(*self*)

By the assumption that the node.built\_value is a deterministic product of the sources, the contents of a Value are the concatenation of all the contents of its sources. As the value need not be built when get\_contents() is called, we cannot use the actual node.built\_value. Overrides:  
SCons.Node.Node.get\_contents

**changed\_since\_last\_build**(*self*, *target*, *prev\_ni*)

Overrides: SCons.Node.Node.changed\_since\_last\_build

**get\_csig**(*self*, *calc*=None)

Because we're a Python value node and don't have a real timestamp, we get to ignore the calculator and just use the value contents. Overrides:  
SCons.Node.Node.get\_csig

### ***Inherited from SCons.Node.Node(Section 13.6)***

Decider(), GetTag(), Tag(), add\_dependency(), add\_ignore(), add\_prerequisite(),

add\_source(), add\_to\_implicit(), add\_to\_waiting\_parents(), add\_to\_waiting\_s\_e(),  
 add\_wkid(), all\_children(), alter\_targets(), builder\_set(), built(), changed(), chil-  
 dren(), children\_are\_up\_to\_date(), clear(), clear\_memoized\_values(), del\_binfo(),  
 disambiguate(), env\_set(), executor\_cleanup(), exists(), explain(), for\_signature(),  
 get\_abspath(), get\_binfo(), get\_build\_env(), get\_build\_scanner\_path(), get\_builder(),  
 get\_cachedir\_csig(), get\_env(), get\_env\_scanner(), get\_executor(), get\_found\_includes(),  
 get\_implicit\_deps(), get\_ninfo(), get\_source\_scanner(), get\_state(), get\_stored\_implicit(),  
 get\_stored\_info(), get\_string(), get\_subst\_proxy(), get\_suffix(), get\_target\_scanner(),  
 has\_builder(), has\_explicit\_builder(), is\_derived(), is\_literal(), missing(), mul-  
 tiple\_side\_effect\_has\_builder(), new\_binfo(), new\_ninfo(), postprocess(), pre-  
 pare(), push\_to\_cache(), release\_target\_info(), remove(), render\_include\_tree(),  
 reset\_executor(), retrieve\_from\_cache(), reexists(), scan(), scanner\_key(), select\_scanner(),  
 set\_always\_build(), set\_executor(), set\_explicit(), set\_nocache(), set\_noclean(),  
 set\_precious(), set\_pseudo(), set\_specific\_source(), set\_state(), visited()

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_subclasshook\_\_()

#### 16.4.2 Properties

Name	Description
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	
always_build, attributes, binfo, builder, cached, depends, depends_set, env, executor, ignore, ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo, nocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect, side_effects, sources, sources_set, state, store_info, waiting_parents, waiting_s_e, wkids	
<i>Inherited from object</i>	
__class__	

## 17 Module SCons.PathList

### SCons.PathList

A module for handling lists of directory paths (the sort of things that get set as CPPPATH, LIBPATH, etc.) with as much caching of data and efficiency as we can while still keeping the evaluation delayed so that we Do the Right Thing (almost) regardless of how the variable is specified.

### 17.1 Functions

#### **node\_\_conv**(*obj*)

This is the "string conversion" routine that we have our substitutions use to return Nodes, not strings. This relies on the fact that an EntryProxy object has a get() method that returns the underlying Node that it wraps, which is a bit of architectural dependence that we might need to break or modify in the future in response to additional requirements.

#### **PathList**(*pathlist*)

Returns the cached \_\_PathList object for the specified pathlist, creating and caching a new object as necessary.

### 17.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/PathList.py rel_2.4.1:3453:73fef3ea0b0...
__doc__	<b>Value:</b> ""SCons.PathL...
TYPE_STRING_NO_SUBST	<b>Value:</b> 0
TYPE_STRING_SUBST	<b>Value:</b> 1
TYPE_OBJECT	<b>Value:</b> 2
__package__	<b>Value:</b> 'SCons'

## 18 Module SCons.SConf

SCons.SConf

Autoconf-like configuration support.

In other words, SConf allows to run tests on the build machine to detect capabilities of system and do some things based on result: generate config files, header files for C/C++, update variables in environment.

Tests on the build system can detect if compiler sees header files, if libraries are installed, if some command line options are supported etc.

### 18.1 Functions

<b>SetBuildType</b> ( <i>type</i> )
-------------------------------------

<b>SetCacheMode</b> ( <i>mode</i> )
-------------------------------------

Set the Configure cache mode. mode must be one of "auto", "force", or "cache".
--

<b>SetProgressDisplay</b> ( <i>display</i> )
--

Set the progress display to use (called from SCons.Script)
--

<b>NeedConfigHBuilder</b> ()
------------------------------

<b>CreateConfigHBuilder</b> ( <i>env</i> )
--

Called if necessary just before the building targets phase begins.
--

<b>SConf</b> (* <i>args</i> , ** <i>kw</i> )
--

<b>CheckFunc</b> ( <i>context</i> , <i>function_name</i> , <i>header</i> =None, <i>language</i> =None)
--

<b>CheckType</b> ( <i>context</i> , <i>type_name</i> , <i>includes</i> ='', <i>language</i> =None)
--

**CheckTypeSize**(*context*, *type\_name*, *includes*='', *language*=None, *expect*=None)

**CheckDeclaration**(*context*, *declaration*, *includes*='', *language*=None)

**createIncludesFromHeaders**(*headers*, *leaveLast*, *include\_quotes*='\"')

**CheckHeader**(*context*, *header*, *include\_quotes*='<>', *language*=None)

A test for a C or C++ header file.

**CheckCC**(*context*)

**CheckCXX**(*context*)

**CheckSHCC**(*context*)

**CheckSHCXX**(*context*)

**CheckCHheader**(*context*, *header*, *include\_quotes*='\"')

A test for a C header file.

**CheckCXXHeader**(*context*, *header*, *include\_quotes*='\"')

A test for a C++ header file.

**CheckLib**(*context*, *library*=None, *symbol*='main', *header*=None, *language*=None, *autoadd*=1)

A test for a library. See also `CheckLibWithHeader`. Note that `library` may also be `None` to test whether the given symbol compiles without flags.



**CheckLibWithHeader**(*context, libs, header, language, call=None, autoadd=1*)

Another (more sophisticated) test for a library. Checks, if library and header is available for language (may be 'C' or 'CXX'). Call maybe be a valid expression `__with__` a trailing ';'. As in `CheckLib`, we support `library=None`, to test if the call compiles without extra link flags.

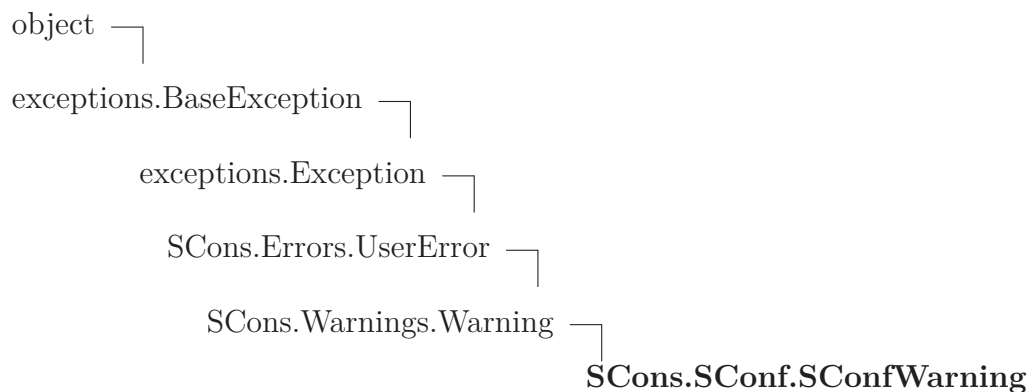
**CheckProg**(*context, prog\_name*)

Simple check if a program exists in the path. Returns the path for the application, or `None` if not found.

## 18.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/SConf.py rel_2.4.1:3453:73fef3ea0b0 20...
<code>build_type</code>	<b>Value:</b> None
<code>build_types</code>	<b>Value:</b> ['clean', 'help']
<code>dryrun</code>	<b>Value:</b> 0
<code>AUTO</code>	<b>Value:</b> 0
<code>FORCE</code>	<b>Value:</b> 1
<code>CACHE</code>	<b>Value:</b> 2
<code>cache_mode</code>	<b>Value:</b> 0
<code>progress_display</code>	<b>Value:</b> DisplayEngine()
<code>SConfFS</code>	<b>Value:</b> None
<code>sconf_global</code>	<b>Value:</b> None
<code>__package__</code>	<b>Value:</b> 'SCons'

### 18.3 Class SConfWarning



#### 18.3.1 Methods

##### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

##### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

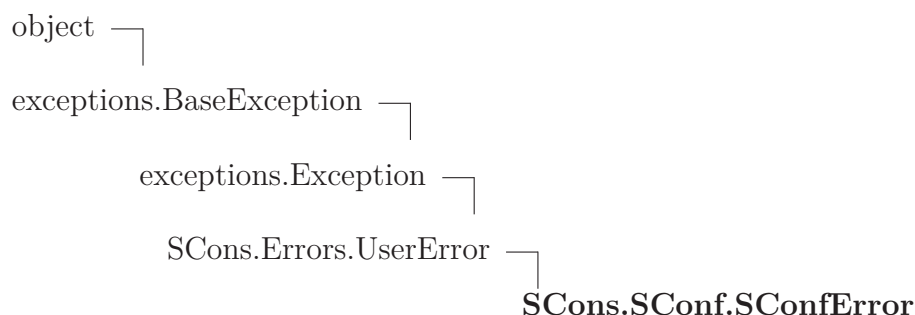
##### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 18.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 18.4 Class SConfError



**Known Subclasses:** SCons.SConf.ConfigureCacheError, SCons.SConf.ConfigureDryRunError

### 18.4.1 Methods

```
__init__(self, msg)
```

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides:  
object.**\_\_init\_\_** **exitit**(inherited documentation)

*Inherited from exceptions.Exception*

```
__new__()
```

*Inherited from exceptions.BaseException*

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(),  
__repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()
```

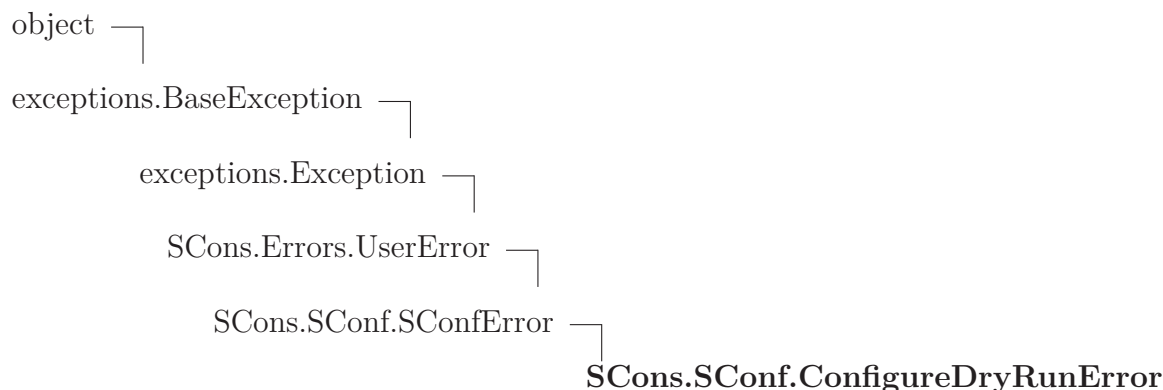
*Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

### 18.4.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<b>__class__</b>	

## 18.5 Class `ConfigureDryRunError`



Raised when a file or directory needs to be updated during a Configure process, but the user requested a dry-run

### 18.5.1 Methods

`__init__(self, target)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature Overrides:  
`object.__init__` `exitit`(inherited documentation)

*Inherited from `exceptions.Exception`*

`__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

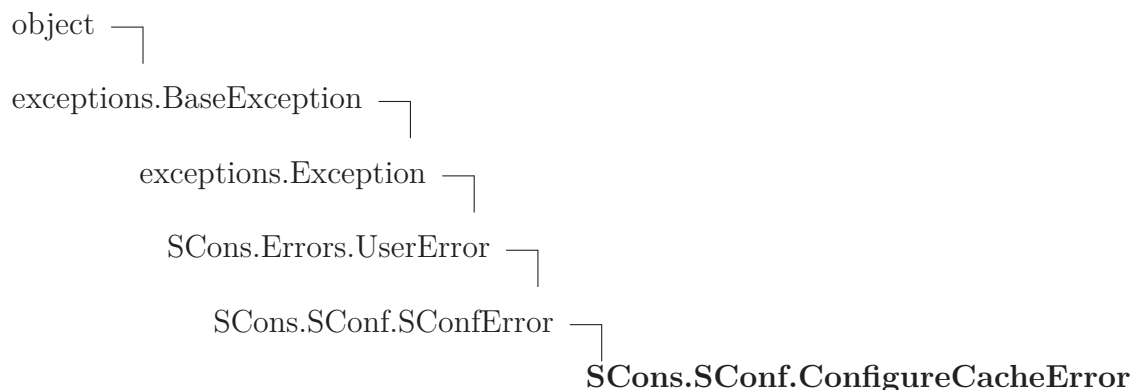
*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 18.5.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 18.6 Class `ConfigureCacheError`



Raised when a use explicitly requested the cache feature, but the test is run the first time.

### 18.6.1 Methods

**`__init__`**(*self*, *target*)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature Overrides:  
`object.__init__` `exitit`(inherited documentation)

*Inherited from `exceptions.Exception`*

`__new__`()

*Inherited from `exceptions.BaseException`*

`__delattr__`(), `__getattr__`(), `__getitem__`(), `__getslice__`(), `__reduce__`(), `__repr__`(), `__setattr__`(), `__setstate__`(), `__str__`(), `__unicode__`()

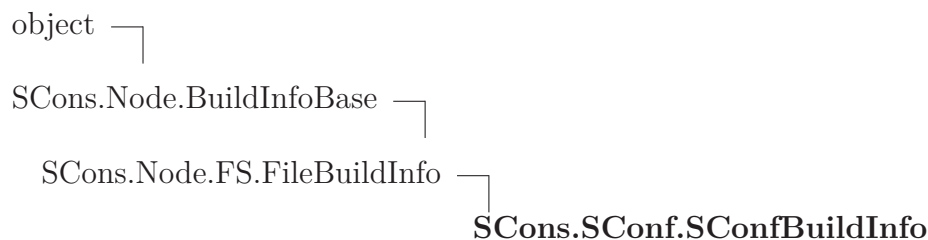
*Inherited from `object`*

`__format__`(), `__hash__`(), `__reduce_ex__`(), `__sizeof__`(), `__subclasshook__`()

### 18.6.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
args, message	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 18.7 Class SConfBuildInfo



Special build info for targets of configure tests. Additional members are result (did the builder succeed last time?) and string, which contains messages of the original build phase.

### 18.7.1 Methods

```
__init__(self)
```

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides:  
object.**\_\_init\_\_** **exitit**(inherited documentation)

```
set_build_result(self, result, string)
```

*Inherited from SCons.Node.FS.FileBuildInfo(Section 15.15)*

convert\_from\_sconsign(), convert\_to\_sconsign(), format(), prepare\_dependencies()

*Inherited from SCons.Node.BuildInfoBase(Section 13.5)*

**\_\_getstate\_\_**(), **\_\_setstate\_\_**(), merge()

*Inherited from object*

**\_\_delattr\_\_**(), **\_\_format\_\_**(), **\_\_getattr\_\_**(), **\_\_hash\_\_**(), **\_\_new\_\_**(),  
**\_\_reduce\_\_**(), **\_\_reduce\_ex\_\_**(), **\_\_repr\_\_**(), **\_\_setattr\_\_**(), **\_\_sizeof\_\_**(),  
**\_\_str\_\_**(), **\_\_subclasshook\_\_**()

### 18.7.2 Properties

Name	Description
result	
string	
<i>Inherited from SCons.Node.BuildInfoBase (Section 13.5)</i>	
bact, bactsig, bdepends, bdependsigns, bimplicit, bimplicitsigs, bsources, bsourcesigs	
<i>Inherited from object</i>	

*continued on next page*

Name	Description
<code>__class__</code>	

### 18.7.3 Class Variables

Name	Description
<i>Inherited from SCons.Node.FS.FileBuildInfo (Section 15.15)</i>	
<code>current_version_id</code>	

## 18.8 Class Streamer

object —  
**SCons.SConf.Streamer**

'Sniffer' for a file-like writable object. Similar to the unix tool tee.

### 18.8.1 Methods

**`__init__(self, orig)`**

x.`__init__`(...) initializes x; see help(type(x)) for signature Overrides:  
object.`__init__` extit(inherited documentation)

**`write(self, str)`**

**`writelines(self, lines)`**

**`getvalue(self)`**

Return everything written to orig since the Streamer was created.

**`flush(self)`**

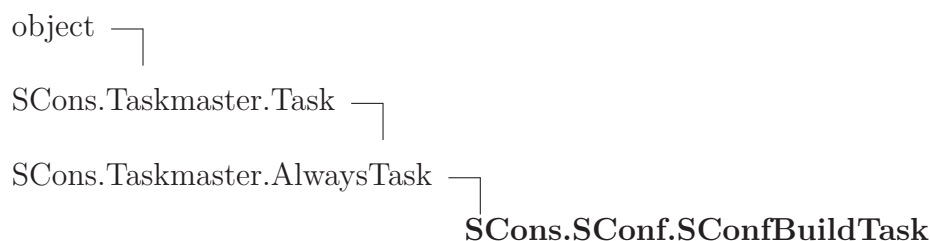
*Inherited from object*

`__delattr__`(), `__format__`(), `__getattr__`(), `__hash__`(), `__new__`(),  
`__reduce__`(), `__reduce_ex__`(), `__repr__`(), `__setattr__`(), `__sizeof__`(),  
`__str__`(), `__subclasshook__`()

### 18.8.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 18.9 Class SCons.SConf.SConfBuildTask



This is almost the same as SCons.Script.BuildTask. Handles SConfErrors correctly and knows about the current `cache_mode`.

### 18.9.1 Methods

**display**(*self*, *message*)

Hook to allow the calling interface to display a message.

This hook gets called as part of preparing a task for execution (that is, a Node to be built). As part of figuring out what Node should be built next, the actually target list may be altered, along with a message describing the alteration. The calling interface can subclass Task and provide a concrete implementation of this method to see those messages. Overrides: SCons.Taskmaster.Task.display exitit(inherited documentation)

**display\_cached\_string**(*self*, *bi*)

Logs the original builder messages, given the SConfBuildInfo instance bi.



**failed(*self*)**

Default action when a task fails: stop the build.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure().  
 Overrides: SCons.Taskmaster.Task.failed extit(inherited documentation)

**collect\_node\_states(*self*)****execute(*self*)**

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in prepare(), executed() or failed(). Overrides: SCons.Taskmaster.Task.execute extit(inherited documentation)

***Inherited from SCons.Taskmaster.AlwaysTask(Section 35.5)***

needs\_execute()

***Inherited from SCons.Taskmaster.Task(Section 35.4)***

\_\_init\_\_(), exc\_clear(), exc\_info(), exception\_set(), executed(), executed\_with\_callbacks(),  
 executed\_without\_callbacks(), fail\_continue(), fail\_stop(), get\_target(), make\_ready(),  
 make\_ready\_all(), make\_ready\_current(), postprocess(), prepare(), trace\_message()

***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

**18.9.2 Properties**

Name	Description
<i>Inherited from object</i>	
__class__	

## 18.10 Class SConfBase



This is simply a class to represent a configure context. After creating a SConf object, you can call any tests. After finished with your tests, be sure to call the `Finish()` method, which returns the modified environment. Some words about caching: In most cases, it is not necessary to cache Test results explicitly. Instead, we use the scons dependency checking mechanism. For example, if one wants to compile a test program (`SConf.TryLink`), the compiler is only called, if the program dependencies have changed. However, if the program could not be compiled in a former SConf run, we need to explicitly cache this error.

### 18.10.1 Methods

---

**\_\_init\_\_**(*self*, *env*, *custom\_tests*={}, *conf\_dir*='\$CONFIGUREDIR',  
*log\_file*='\$CONFIGURELOG', *config\_h*=None, *\_depth*=0)

---

Constructor. Pass additional tests in the *custom\_tests*-dictionary, e.g. *custom\_tests*={'CheckPrivate':MyPrivateTest}, where MyPrivateTest defines a custom test. Note also the *conf\_dir* and *log\_file* arguments (you may want to build tests in the VariantDir, not in the SourceDir) Overrides: `object.__init__`

---

**Finish**(*self*)

---

Call this method after finished with your tests: `env = sconf.Finish()`

---

**Define**(*self*, *name*, *value*=None, *comment*=None)

---

Define a pre processor symbol name, with the optional given value in the current config header.

If *value* is None (default), then `#define name` is written. If *value* is not none, then `#define name value` is written.

*comment* is a string which will be put as a C comment in the header, to explain the meaning of the value (appropriate C comments `/*` and `*/` will be put automatically.

**BuildNodes**(*self*, *nodes*)

Tries to build the given nodes immediately. Returns 1 on success, 0 on error.

**pspawn\_wrapper**(*self*, *sh*, *escape*, *cmd*, *args*, *env*)

Wrapper function for handling piped spawns.

This looks to the calling interface (in Action.py) like a "normal" spawn, but associates the call with the PSPAWN variable from the construction environment and with the streams to which we want the output logged. This gets slid into the construction environment as the SPAWN variable so Action.py doesn't have to know or care whether it's spawning a piped command or not.

**TryBuild**(*self*, *builder*, *text*=None, *extension*='')

Low level TryBuild implementation. Normally you don't need to call that - you can use TryCompile / TryLink / TryRun instead

**TryAction**(*self*, *action*, *text*=None, *extension*='')

Tries to execute the given action with optional source file contents <text> and optional source file extension <extension>, Returns the status (0 : failed, 1 : ok) and the contents of the output file.

**TryCompile**(*self*, *text*, *extension*)

Compiles the program given in text to an env.Object, using extension as file extension (e.g. '.c'). Returns 1, if compilation was successful, 0 otherwise. The target is saved in self.lastTarget (for further processing).

**TryLink**(*self*, *text*, *extension*)

Compiles the program given in text to an executable env.Program, using extension as file extension (e.g. '.c'). Returns 1, if compilation was successful, 0 otherwise. The target is saved in self.lastTarget (for further processing).

**TryRun**(*self*, *text*, *extension*)

Compiles and runs the program given in text, using extension as file extension (e.g. '.c'). Returns (1, outputStr) on success, (0, "") otherwise. The target (a file containing the program's stdout) is saved in self.lastTarget (for further processing).

**AddTest**(*self*, *test\_name*, *test\_instance*)

Adds test\_class to this SConf instance. It can be called with self.test\_name(...)

**AddTests**(*self*, *tests*)

Adds all the tests given in the tests dictionary to this SConf instance

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

### 18.10.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

## 18.11 Class *CheckContext*

object —  
**SCons.SConf.CheckContext**

Provides a context for configure tests. Defines how a test writes to the screen and log file.

A typical test is just a callable with an instance of *CheckContext* as first argument:

```
def CheckCustom(context, ...) context.Message('Checking my weird test ... ') ret = my-
WeirdTestFunction(...) context.Result(ret)
```

Often, *myWeirdTestFunction* will be one of *context.TryCompile/context.TryLink/context.TryRun*. The results of those are cached, for they are only rebuild, if the dependencies have changed.

### 18.11.1 Methods

<b>__init__</b> ( <i>self</i> , <i>sconf</i> )
--

<p>Constructor. Pass the corresponding <i>SConf</i> instance. Overrides:  <i>object.__init__</i></p>
--

<b>Message</b> ( <i>self</i> , <i>text</i> )
--

<p>Inform about what we are doing right now, e.g. 'Checking for SOMETHING ... '</p>
---

<b>Result</b> ( <i>self</i> , <i>res</i> )
--

<p>Inform about the result of the test. If <i>res</i> is not a string, displays 'yes' or 'no' depending on whether <i>res</i> is evaluated as true or false. The result is only displayed when <i>self.did_show_result</i> is not set.</p>
--

<b>TryBuild</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
--

<b>TryAction</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

<b>TryCompile</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
--

<b>TryLink</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

<b>TryRun</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
--

<b>__getattr__</b> ( <i>self</i> , <i>attr</i> )
--

<b>BuildProg</b> ( <i>self</i> , <i>text</i> , <i>ext</i> )
---

<b>CompileProg</b> ( <i>self</i> , <i>text</i> , <i>ext</i> )
---

<b>CompileSharedObject</b> ( <i>self</i> , <i>text</i> , <i>ext</i> )
---

<b>RunProg</b> ( <i>self</i> , <i>text</i> , <i>ext</i> )
---

<b>AppendLIBS</b> ( <i>self</i> , <i>lib_name_list</i> )
--

<b>PrependLIBS</b> ( <i>self</i> , <i>lib_name_list</i> )
---

<b>SetLIBS</b> ( <i>self</i> , <i>val</i> )
---

<b>Display</b> ( <i>self</i> , <i>msg</i> )
---

<b>Log</b> ( <i>self</i> , <i>msg</i> )
---

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

#### 18.11.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 19 Module SCons.SConsign

SCons.SConsign

Writing and reading information to the .sconsign file or files.

### 19.1 Functions

**corrupt\_dblite\_warning**(*filename*)

**Get\_DataBase**(*dir*)

**Reset**()

Reset global state. Used by unit tests that end up using SConsign multiple times to get a clean slate for each test.

**write**()

**File**(*name*, *dbm\_module*=None)

Arrange for all signatures to be stored in a global .sconsign.db\* file.

### 19.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/SConsign.py rel_2.4.1:3453:73fef3ea0b0...
sig_files	<b>Value:</b> []
DataBase	<b>Value:</b> {}
DB_Name	<b>Value:</b> '.sconsign'
DB_sync_list	<b>Value:</b> []
__package__	<b>Value:</b> 'SCons'

### 19.3 Class SConsignEntry



Wrapper class for the generic entry in a .sconsign file. The Node subclass populates it with attributes as it pleases.

XXX As coded below, we do expect a 'binfo' attribute to be added, but we'll probably generalize this in the next refactorings.

#### 19.3.1 Methods

```
__init__(self)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides:  
object.\_\_init\_\_ extit(inherited documentation)

```
convert_to_sconsign(self)
```

```
convert_from_sconsign(self, dir, name)
```

```
__getstate__(self)
```

```
__setstate__(self, state)
```

#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

#### 19.3.2 Properties

Name	Description
binfo	
ninfo	
<i>Inherited from object</i>	
__class__	

#### 19.3.3 Class Variables



Name	Description
current_version_id	Value: 2

## 19.4 Class Base



**Known Subclasses:** SCons.SConsign.DB, SCons.SConsign.Dir

This is the controlling class for the signatures for the collection of entries associated with a specific directory. The actual directory association will be maintained by a subclass that is specific to the underlying storage method. This class provides a common set of methods for fetching and storing the individual bits of information that make up signature entry.

### 19.4.1 Methods

**\_\_init\_\_**(*self*)

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides:  
object.**\_\_init\_\_** extit(inherited documentation)

**get\_entry**(*self*, *filename*)

Fetch the specified entry attribute.

**set\_entry**(*self*, *filename*, *obj*)

Set the entry.

**do\_not\_set\_entry**(*self*, *filename*, *obj*)

**store\_info**(*self*, *filename*, *node*)

**do\_not\_store\_info**(*self*, *filename*, *node*)

**merge**(*self*)

***Inherited from object***

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

**19.4.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**19.5 Class DB**

A Base subclass that reads and writes signature information from a global `.sconsign.db*` file--the actual file suffix is determined by the database module.

**19.5.1 Methods**

<code>__init__(self, dir)</code>
x. <code>__init__</code> (...) initializes x; see <code>help(type(x))</code> for signature Overrides: object. <code>__init__</code> <code>exitit</code> (inherited documentation)
<code>write(self, sync=1)</code>

***Inherited from SCons.SConsign.Base(Section 19.4)***

```
do_not_set_entry(), do_not_store_info(), get_entry(), merge(), set_entry(), store_info()
```

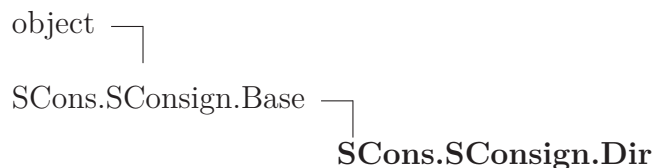
***Inherited from object***

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

**19.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
__class__	

## 19.6 Class Dir



**Known Subclasses:** SCons.SConsign.DirFile

### 19.6.1 Methods

<b>__init__</b> ( <i>self</i> , <i>fp</i> =None, <i>dir</i> =None)
fp - file pointer to read entries from Overrides: object.__init__

*Inherited from SCons.SConsign.Base(Section 19.4)*

do\_not\_set\_entry(), do\_not\_store\_info(), get\_entry(), merge(), set\_entry(), store\_info()

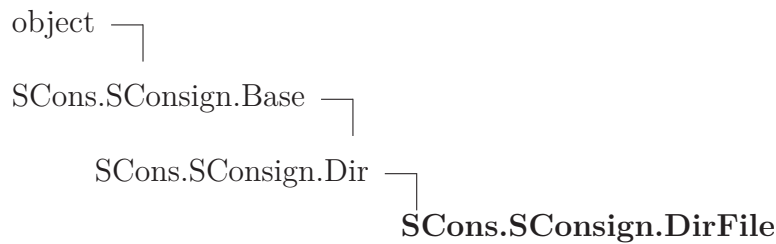
*Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

### 19.6.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 19.7 Class *DirFile*



Encapsulates reading and writing a per-directory `.sconsign` file.

### 19.7.1 Methods

<b><code>__init__(self, dir)</code></b>
dir - the directory for the file   Overrides: <code>object.__init__</code>

<b><code>write(self, sync=1)</code></b>
Write the <code>.sconsign</code> file to disk.
Try to write to a temporary file first, and rename it if we succeed. If we can't write to the temporary file, it's probably because the directory isn't writable (and if so, how did we build anything in this directory, anyway?), so try to write directly to the <code>.sconsign</code> file as a backup. If we can't rename, try to copy the temporary contents back to the <code>.sconsign</code> file. Either way, always try to remove the temporary file at the end.

*Inherited from `SCons.SConsign.Base`(Section 19.4)*

`do_not_set_entry()`, `do_not_store_info()`, `get_entry()`, `merge()`, `set_entry()`, `store_info()`

*Inherited from `object`*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 19.7.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 19.8 Class DB



A Base subclass that reads and writes signature information from a global .sconsign.db\* file--the actual file suffix is determined by the database module.

### 19.8.1 Methods

<b>__init__</b> ( <i>self</i> , <i>dir</i> )
x. <b>__init__</b> (...) initializes x; see help(type(x)) for signature Overrides: object. <b>__init__</b> extit(inherited documentation)
<b>write</b> ( <i>self</i> , <i>sync</i> =1)

*Inherited from SCons.SConsign.Base(Section 19.4)*

do\_not\_set\_entry(), do\_not\_store\_info(), get\_entry(), merge(), set\_entry(), store\_info()

*Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
\_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
\_\_str\_\_(), \_\_subclasshook\_\_()

### 19.8.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 20 Package SCons.Scanner

SCons.Scanner

The Scanner package for the SCons software construction utility.

### 20.1 Modules

- **C:** SCons.Scanner.C  
(Section 21, p. 231)
- **D:** SCons.Scanner.D  
(Section 22, p. 234)
- **Dir** (Section 23, p. 238)
- **Fortran:** SCons.Scanner.Fortran  
(Section 24, p. 240)
- **IDL:** SCons.Scanner.IDL  
(Section 25, p. 245)
- **LaTeX:** SCons.Scanner.LaTeX  
(Section 26, p. 246)
- **Prog** (Section 27, p. 253)
- **RC:** SCons.Scanner.RC  
(Section 28, p. 254)

### 20.2 Functions

**Scanner**(*function*, \**args*, \*\**kw*)

Public interface factory function for creating different types of Scanners based on the different types of "functions" that may be supplied.

TODO: Deprecate this some day. We've moved the functionality inside the Base class and really don't need this factory function any more. It was, however, used by some of our Tool modules, so the call probably ended up in various people's custom modules patterned on SCons code.

### 20.3 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Scanner/__init__.py rel_2.4.1:3453:73fe...

*continued on next page*

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'SCons.Scanner'</code>

## 20.4 Class FindPathDirs



A class to bind a specific **\*PATH** variable name to a function that will return all of the **\*path** directories.

### 20.4.1 Methods

<code>__init__(self, variable)</code>
x. <code>__init__</code> (...) initializes x; see help(type(x)) for signature Overrides: object. <code>__init__</code> extit(inherited documentation)
<code>__call__(self, env, dir=None, target=None, source=None, argument=None)</code>

### Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

### 20.4.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

## 20.5 Class Base



**Known Subclasses:** SCons.Scanner.Current, SCons.Scanner.Selector, SCons.Scanner.LaTeX.LaTeX

The base class for dependency scanners. This implements straightforward, single-pass scanning of a single file.

### 20.5.1 Methods

**\_\_call\_\_**(*self*, *node*, *env*, *path*=())

This method scans a single object. 'node' is the node that will be passed to the scanner function, and 'env' is the environment that will be passed to the scanner function. A list of direct dependency nodes for the specified node will be returned.

**\_\_cmp\_\_**(*self*, *other*)

**\_\_hash\_\_**(*self*)

hash(x) Overrides: object.\_\_hash\_\_ extit(inherited documentation)



---

```
init__(self, function, name='NONE', argument=<class
'SCons.Scanner._Null'>, keys=<class 'SCons.Scanner._Null'>,
path_function=None, node_class=<class 'SCons.Node.FS.Base'>,
node_factory=None, scan_check=None, recursive=None)
```

---

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the path\_function.

'keys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'keys' would be file suffixes.

'path\_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node\_class' - the class of Nodes which this scan will return. If node\_class is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node\_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected node\_class objects.

'scan\_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being #include lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function's first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the path\_function, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

```
s = Scanner(my_scanner_function)
```

```
s = Scanner(function = my_scanner_function)
```

```
__str__(self)
```

```
str(x) Overrides: object.__str__ extit(inherited documentation)
```

```
add_scanner(self, skey, scanner)
```

```
add_skey(self, skey)
```

```
Add a skey to the list of skeys
```

```
get_skeys(self, env=None)
```

```
path(self, env, dir=None, target=None, source=None)
```

```
recurse_nodes(self, nodes)
```

```
select(self, node)
```

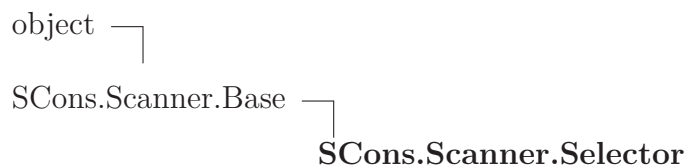
### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __new__(), __reduce__(),
__reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()
```

#### 20.5.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 20.6 Class Selector



A class for selecting a more specific scanner based on the scanner\_key() (suffix) for a specific Node.

TODO: This functionality has been moved into the inner workings of the Base class, and this

class will be deprecated at some point. (It was never exposed directly as part of the public interface, although it is used by the `Scanner()` factory function that was used by various Tool modules and therefore was likely a template for custom modules that may be out there.)



### 20.6.1 Methods

`__init__(self, dict, *args, **kw)`

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the `path_function`.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path\_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node\_class' - the class of Nodes which this scan will return. If `node_class` is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node\_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected `node_class` objects.

'scan\_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being `#include` lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function's first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the `path_function`, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

```
s = Scanner(my_scanner_function) 237
```

```
s = Scanner(function = my_scanner_function)
```

```
s = Scanner(function = my_scanner_function, argument = 'foo') Overrides:
```

```
__call__(self, node, env, path=())
```

This method scans a single object. 'node' is the node that will be passed to the scanner function, and 'env' is the environment that will be passed to the scanner function. A list of direct dependency nodes for the specified node will be returned. Overrides: SCons.Scanner.Base.\_\_call\_\_ extit(inherited documentation)

```
select(self, node)
```

Overrides: SCons.Scanner.Base.select

```
add_scanner(self, skey, scanner)
```

Overrides: SCons.Scanner.Base.add\_scanner

### *Inherited from SCons.Scanner.Base(Section 20.5)*

```
__cmp__(), __hash__(), __str__(), add_skey(), get_skeys(), path(), recurse_nodes()
```

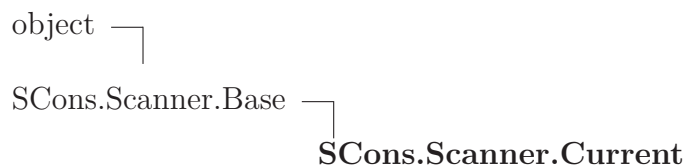
### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __new__(), __reduce__(),
__reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()
```

#### 20.6.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 20.7 Class Current



### **Known Subclasses:** SCons.Scanner.Classic

A class for scanning files that are source files (have no builder) or are derived files and are current (which implies that they exist, either locally or in a repository).



## 20.7.1 Methods

`__init__(self, *args, **kw)`

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the `path_function`.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path\_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node\_class' - the class of Nodes which this scan will return. If `node_class` is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node\_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected `node_class` objects.

'scan\_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being `#include` lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function's first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the `path_function`, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

```
s = Scanner(my_scanner_function)    240
```

```
s = Scanner(function = my_scanner_function)
```

```
s = Scanner(function = my_scanner_function, argument = 'foo') Overrides:
```



***Inherited from SCons.Scanner.Base(Section 20.5)***

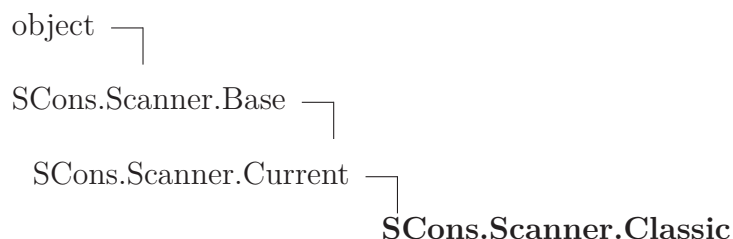
`__call__()`, `__cmp__()`, `__hash__()`, `__str__()`, `add_scanner()`, `add_skey()`,  
`get_skeys()`, `path()`, `recurse_nodes()`, `select()`

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`,  
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

**20.7.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**20.8 Class Classic**

**Known Subclasses:** `SCons.Scanner.ClassicCPP`, `SCons.Scanner.D.D`, `SCons.Scanner.Fortran.F90Scanner`

A Scanner subclass to contain the common logic for classic CPP-style include scanning, but which can be customized to use different regular expressions to find the includes.

Note that in order for this to work "out of the box" (without overriding the `find_include()` and `sort_key()` methods), the regular expression passed to the constructor must return the name of the include file in group 0.



### 20.8.1 Methods

**\_\_init\_\_**(*self*, *name*, *suffixes*, *path\_variable*, *regex*, *\*args*, *\*\*kw*)

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the `path_function`.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path\_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node\_class' - the class of Nodes which this scan will return. If `node_class` is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node\_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected `node_class` objects.

'scan\_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being `#include` lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function's first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the `path_function`, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

```
s = Scanner(my_scanner_function) 243
```

```
s = Scanner(function = my_scanner_function)
```

```
s = Scanner(function = my_scanner_function, argument = 'foo') Overrides:
```

```
find_include(self, include, source_dir, path)
```

```
find_include_names(self, node)
```

```
scan(self, node, path=())
```

```
sort_key(self, include)
```

*Inherited from **SCons.Scanner.Base**(Section 20.5)*

```
__call__(), __cmp__(), __hash__(), __str__(), add_scanner(), add_skey(),
get_skeys(), path(), recurse_nodes(), select()
```

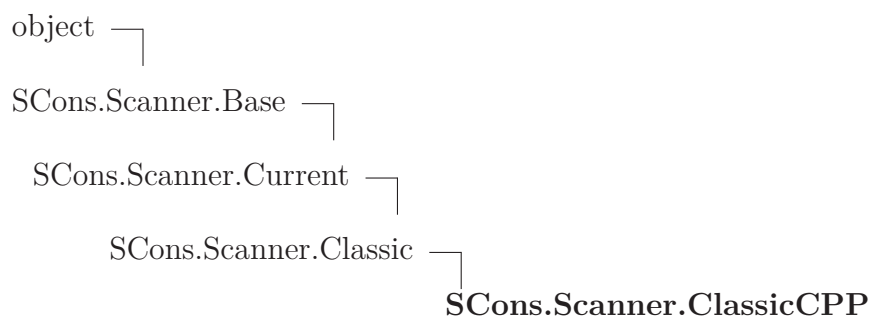
*Inherited from **object***

```
__delattr__(), __format__(), __getattr__(), __new__(), __reduce__(),
__reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()
```

## 20.8.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 20.9 Class **ClassicCPP**



A Classic Scanner subclass which takes into account the type of bracketing used to include the file, and uses classic CPP rules for searching for the files based on the bracketing.

Note that in order for this to work, the regular expression passed to the constructor must return the leading bracket in group 0, and the contained filename in group 1.

**20.9.1 Methods**

<b>find_include</b> ( <i>self</i> , <i>include</i> , <i>source_dir</i> , <i>path</i> )
--

Overrides: <i>SCons.Scanner.Classic.find_include</i>
--

<b>sort_key</b> ( <i>self</i> , <i>include</i> )
--

Overrides: <i>SCons.Scanner.Classic.sort_key</i>
--

***Inherited from SCons.Scanner.Classic(Section 20.8)***

`__init__()`, `find_include_names()`, `scan()`

***Inherited from SCons.Scanner.Base(Section 20.5)***

`__call__()`, `__cmp__()`, `__hash__()`, `__str__()`, `add_scanner()`, `add_skey()`,  
`get_skeys()`, `path()`, `recurse_nodes()`, `select()`

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`,  
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

**20.9.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 21 Module **SCons.Scanner.C**

**SCons.Scanner.C**

This module implements the dependency scanner for C/C++ code.

### 21.1 Functions

<b>dictify__CPPDEFINES</b> ( <i>env</i> )
---

<b>CScanner</b> ()
--------------------

Return a prototype Scanner instance for scanning source files that use the C pre-processor
--

### 21.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Scanner/C.py rel_2.4.1:3453:73fef3ea0b...
__package__	<b>Value:</b> 'SCons.Scanner'

### 21.3 Class **SConsCPPScanner**

object └─

SCons.cpp.PreProcessor └─

**SCons.Scanner.C.SConsCPPScanner**

SCons-specific subclass of the cpp.py module's processing.

We subclass this so that: 1) we can deal with files represented by Nodes, not strings; 2) we can keep track of the files that are missing.

### 21.3.1 Methods

**\_\_init\_\_**(*self*, \**args*, \*\**kw*)

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides: object.\_\_init\_\_ exitit(inherited documentation)

**initialize\_result**(*self*, *fname*)

Overrides: SCons.cpp.PreProcessor.initialize\_result

**finalize\_result**(*self*, *fname*)

Overrides: SCons.cpp.PreProcessor.finalize\_result

**find\_include\_file**(*self*, *t*)

Finds the #include file for a given preprocessor tuple. Overrides: SCons.cpp.PreProcessor.find\_include\_file exitit(inherited documentation)

**read\_file**(*self*, *file*)

Overrides: SCons.cpp.PreProcessor.read\_file

#### *Inherited from SCons.cpp.PreProcessor(Section 44.4)*

\_\_call\_\_(), all\_include(), do\_define(), do\_elif(), do\_else(), do\_endif(), do\_if(), do\_ifdef(), do\_ifndef(), do\_import(), do\_include(), do\_include\_next(), do\_nothing(), do\_undef(), eval\_expression(), process\_contents(), resolve\_include(), restore(), save(), sconscurrent\_file(), start\_handling\_includes(), stop\_handling\_includes(), tupleize()

#### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

### 21.3.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 21.4 Class SConsCPPScannerWrapper

object —  
**SCons.Scanner.C.SConsCPPScannerWrapper**

The SCons wrapper around a cpp.py scanner.

This is the actual glue between the calling conventions of generic SCons scanners, and the (subclass of) cpp.py class that knows how to look for #include lines with reasonably real C-preprocessor-like evaluation of #if/#ifdef/#else/#elif lines.

### 21.4.1 Methods

<b>__init__</b> ( <i>self</i> , <i>name</i> , <i>variable</i> )
---

x. <b>__init__</b> (...) initializes x; see help(type(x)) for signature Overrides: object. <b>__init__</b> extit(inherited documentation)
--

<b>__call__</b> ( <i>self</i> , <i>node</i> , <i>env</i> , <i>path</i> =())
---

<b>recurse__nodes</b> ( <i>self</i> , <i>nodes</i> )
--

<b>select</b> ( <i>self</i> , <i>node</i> )
---

### *Inherited from object*

<b>__delattr__</b> (), <b>__format__</b> (), <b>__getattr__</b> (), <b>__hash__</b> (), <b>__new__</b> (), <b>__reduce__</b> (), <b>__reduce_ex__</b> (), <b>__repr__</b> (), <b>__setattr__</b> (), <b>__sizeof__</b> (), <b>__str__</b> (), <b>__subclasshook__</b> ()
--

### 21.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<b>__class__</b>	



## 22 Module SCons.Scanner.D

SCons.Scanner.D

Scanner for the Digital Mars "D" programming language.

Coded by Andy Friesen 17 Nov 2003

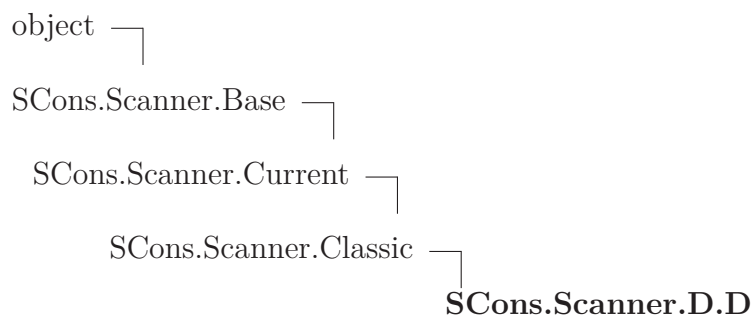
### 22.1 Functions

DScanner()
Return a prototype Scanner instance for scanning D source files

### 22.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Scanner/D.py rel_2.4.1:3453:73fef3ea0b...
__package__	<b>Value:</b> 'SCons.Scanner'

### 22.3 Class D





### 22.3.1 Methods

`__init__(self)`

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the `path_function`.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path\_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node\_class' - the class of Nodes which this scan will return. If `node_class` is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node\_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected `node_class` objects.

'scan\_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being `#include` lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function's first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the `path_function`, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

```
s = Scanner(my_scanner_function) 251
```

```
s = Scanner(function = my_scanner_function)
```

```
s = Scanner(function = my_scanner_function, argument = 'foo') Overrides:
```

<b>find_include</b> ( <i>self</i> , <i>include</i> , <i>source_dir</i> , <i>path</i> )
--

Overrides: SCons.Scanner.Classic.find_include
---

<b>find_include_names</b> ( <i>self</i> , <i>node</i> )
---

Overrides: SCons.Scanner.Classic.find_include_names
---

***Inherited from SCons.Scanner.Classic(Section 20.8)***

scan(), sort\_key()

***Inherited from SCons.Scanner.Base(Section 20.5)***

\_\_call\_\_(), \_\_cmp\_\_(), \_\_hash\_\_(), \_\_str\_\_(), add\_scanner(), add\_skey(),  
get\_skeys(), path(), recurse\_nodes(), select()

***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_new\_\_(), \_\_reduce\_\_(),  
\_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

### 22.3.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 23 Module *SCons.Scanner.Dir*

### 23.1 Functions

**only\_dirs**(*nodes*)

**DirScanner**(\*\**kw*)

Return a prototype Scanner instance for scanning directories for on-disk files

**DirEntryScanner**(\*\**kw*)

Return a prototype Scanner instance for "scanning" directory Nodes for their in-memory entries

**do\_not\_scan**(*k*)

**scan\_on\_disk**(*node*, *env*, *path*=())

Scans a directory for on-disk files and directories therein.

Looking up the entries will add these to the in-memory Node tree representation of the file system, so all we have to do is just that and then call the in-memory scanning function.

**scan\_in\_memory**(*node*, *env*, *path*=())

"Scans" a Node.FS.Dir for its in-memory entries.

### 23.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Scanner/Dir.py rel_2.4.1:3453:73fef3ea...
<code>skip_entry</code>	<b>Value:</b> {'.'': 1, '..': 1, '.sconsign': 1, '.sconsign.bak': 1, '.s...

*continued on next page*

Name	Description
skip_entry_list	<b>Value:</b> ['.', '..', '.sconsign', .sconsign.dblite', '.sconsign.d...
__package__	<b>Value:</b> 'SCons.Scanner'
skip	<b>Value:</b> '.sconsign.db'

## 24 Module SCons.Scanner.Fortran

SCons.Scanner.Fortran

This module implements the dependency scanner for Fortran code.

### 24.1 Functions

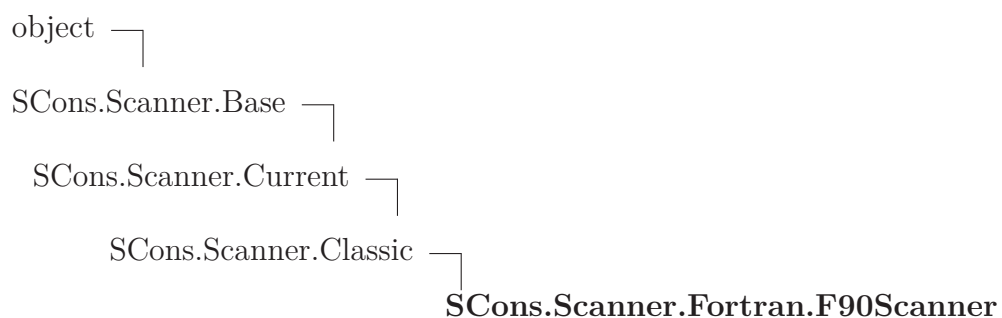
<b>FortranScan</b> ( <i>path_variable</i> ='FORTRANPATH')
---

Return a prototype Scanner instance for scanning source files for Fortran USE & INCLUDE statements
--

### 24.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Scanner/Fortran.py rel_2.4.1:3453:73fef...'
__package__	<b>Value:</b> 'SCons.Scanner'

### 24.3 Class F90Scanner



A Classic Scanner subclass for Fortran source files which takes into account both USE and INCLUDE statements. This scanner will work for both F77 and F90 (and beyond) compilers.

Currently, this scanner assumes that the include files do not contain USE statements. To enable the ability to deal with USE statements in include files, add logic right after the module names are found to loop over each include file, search for and locate each USE statement, and append each module name to the list of dependencies. Caching the search

results in a common dictionary somewhere so that the same include file is not searched multiple times would be a smart thing to do.





### 24.3.1 Methods

**\_\_init\_\_**(*self, name, suffixes, path\_variable, use\_regex, incl\_regex, def\_regex, \*args, \*\*kw*)

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the path\_function.

'keys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'keys' would be file suffixes.

'path\_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node\_class' - the class of Nodes which this scan will return. If node\_class is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node\_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected node\_class objects.

'scan\_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being #include lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function's first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the path\_function, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

258

```
s = Scanner(my_scanner_function)
```

```
s = Scanner(function = my_scanner_function)
```

```
scan(self, node, env, path=())
```

Overrides: SCons.Scanner.Classic.scan

***Inherited from SCons.Scanner.Classic(Section 20.8)***

find\_include(), find\_include\_names(), sort\_key()

***Inherited from SCons.Scanner.Base(Section 20.5)***

\_\_call\_\_(), \_\_cmp\_\_(), \_\_hash\_\_(), \_\_str\_\_(), add\_scanner(), add\_skey(),  
get\_skeys(), path(), recurse\_nodes(), select()

***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_new\_\_(), \_\_reduce\_\_(),  
\_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

### 24.3.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

## 25 Module SCons.Scanner.IDL

SCons.Scanner.IDL

This module implements the dependency scanner for IDL (Interface Definition Language) files.

### 25.1 Functions

<b>IDLScan()</b>
Return a prototype Scanner instance for scanning IDL source files

### 25.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Scanner/IDL.py rel_2.4.1:3453:73fef3ea...
__package__	<b>Value:</b> 'SCons.Scanner'

## 26 Module SCons.Scanner.LaTeX

SCons.Scanner.LaTeX

This module implements the dependency scanner for LaTeX code.

### 26.1 Functions

<b>modify__env__var</b> ( <i>env</i> , <i>var</i> , <i>abspath</i> )
--

<b>LaTeXScanner</b> ()
------------------------

Return a prototype Scanner instance for scanning LaTeX source files when built with latex.
--

<b>PDFLaTeXScanner</b> ()
---------------------------

Return a prototype Scanner instance for scanning LaTeX source files when built with pdflatex.
---

### 26.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Scanner/LaTeX.py rel_2.4.1:3453:73fed3...
TexGraphics	<b>Value:</b> ['.eps', '.ps']
LatexGraphics	<b>Value:</b> ['.pdf', '.png', '.jpg', '.gif', '.tif']
__package__	<b>Value:</b> 'SCons.Scanner'

### 26.3 Class FindENVPPathDirs

object —  
**SCons.Scanner.LaTeX.FindENVPPathDirs**

A class to bind a specific \*PATH variable name to a function that will return all of the

\*path directories.

### 26.3.1 Methods

```
__init__(self, variable)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides:  
object.\_\_init\_\_ exitit(inherited documentation)

```
__call__(self, env, dir=None, target=None, source=None, argument=None)
```

#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 26.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 26.4 Class LaTeX

object └─

SCons.Scanner.Base └─

**SCons.Scanner.LaTeX.LaTeX**

Class for scanning LaTeX files for included files.

Unlike most scanners, which use regular expressions that just return the included file name, this returns a tuple consisting of the keyword for the inclusion ("include", "includegraphics", "input", or "bibliography"), and then the file name itself.

Based on a quick look at LaTeX documentation, it seems that we should append .tex suffix for the "include" keywords, append .tex if there is no extension for the "input" keyword, and need to add .bib for the "bibliography" keyword that does not accept extensions by itself.

Finally, if there is no extension for an "includegraphics" keyword latex will append .ps or .eps to find the file, while pdftex may use .pdf, .jpg, .tif, .mps, or .png.

The actual subset and search order may be altered by DeclareGraphicsExtensions command. This complication is ignored. The default order corresponds to experimentation with teTeX

```
$ latex --version
pdfTeX 3.141592-1.21a-2.2 (Web2C 7.5.4)
kpathsea version 3.5.4
```

The order is:

```
['.eps', '.ps'] for latex
['.png', '.pdf', '.jpg', '.tif'].
```

Another difference is that the search path is determined by the type of the file being searched:

```
env['TEXINPUTS'] for "input" and "include" keywords
env['TEXINPUTS'] for "includegraphics" keyword
env['TEXINPUTS'] for "lstinputlisting" keyword
env['BIBINPUTS'] for "bibliography" keyword
env['BSTINPUTS'] for "bibliographystyle" keyword
env['INDEXSTYLE'] for "makeindex" keyword, no scanning support needed
just allows user to set it if needed.
```

FIXME: also look for the class or style in document[class|style]{}

FIXME: also look for the argument of bibliographystyle{}





### 26.4.1 Methods

**\_\_init\_\_**(*self*, *name*, *suffixes*, *graphics\_extensions*, \**args*, \*\**kw*)

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the `path_function`.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path\_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node\_class' - the class of Nodes which this scan will return. If `node_class` is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node\_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected `node_class` objects.

'scan\_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being `#include` lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function's first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the `path_function`, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

```
s = Scanner(my_scanner_function) 265
```

```
s = Scanner(function = my_scanner_function)
```

```
s = Scanner(function = my_scanner_function, argument = 'foo') Overrides:
```

<code>sort_key(self, include)</code>
--------------------------------------

<code>find_include(self, include, source_dir, path)</code>
--

<code>canonical_text(self, text)</code>
---

Standardize an input TeX-file contents.

**Currently:**

- removes comments, unwrapping comment-wrapped lines.

<code>scan(self, node)</code>
-------------------------------

<code>scan_recurse(self, node, path=())</code>
--

do a recursive scan of the top level target file This lets us search for included files based on the directory of the main file just as latex does

**Inherited from *SCons.Scanner.Base*(Section 20.5)**

`__call__()`, `__cmp__()`, `__hash__()`, `__str__()`, `add_scanner()`, `add_skey()`, `get_skeys()`, `path()`, `recurse_nodes()`, `select()`

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

#### 26.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

#### 26.4.3 Class Variables

Name	Description
<code>keyword_paths</code>	<b>Value:</b> {'addbibresource': 'BIBINPUTS', 'addglobalbib': 'BIBINPUT...}

*continued on next page*

Name	Description
env_variables	<b>Value:</b> ['INDEXSTYLE', 'BIBINPUTS', 'TEXINPUTS', 'BSTINPUTS']

## 27 Module *SCons.Scanner.Prog*

### 27.1 Functions

**ProgramScanner**(\*\**kw*)

Return a prototype Scanner instance for scanning executable files for static-lib dependencies

**scan**(*node*, *env*, *libpath*=())

This scanner scans program files for static-library dependencies. It will search the LIBPATH environment variable for libraries specified in the LIBS variable, returning any files it finds as dependencies.

### 27.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Scanner/Prog.py rel_2.4.1:3453:73fef3e...
print_find_libs	<b>Value:</b> None
__package__	<b>Value:</b> 'SCons.Scanner'

## 28 Module `SCons.Scanner.RC`

`SCons.Scanner.RC`

This module implements the dependency scanner for RC (Interface Definition Language) files.

### 28.1 Functions

<b>RCScan()</b>
Return a prototype Scanner instance for scanning RC source files

### 28.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> <code>'src/engine/SCons/Scanner/RC.py rel_2.4.1:3453:73fe3d3ea0...'</code>
<code>__package__</code>	<b>Value:</b> <code>'SCons.Scanner'</code>

## 29 Package SCons.Script

### SCons.Script

This file implements the `main()` function used by the `scons` script.

Architecturally, this *is* the `scons` script, and will likely only be called from the external "scons" wrapper. Consequently, anything here should not be, or be considered, part of the build engine. If it's something that we expect other software to want to use, it should go in some other module. If it's specific to the "scons" script invocation, it goes here.

### 29.1 Modules

- **Interactive:** SCons interactive mode  
(Section 30, p. 264)
- **Main:** SCons.Script  
(Section 31, p. 267)
- **SConscript':** SCons.Script.SConscript  
(Section 32, p. 281)

### 29.2 Functions

<b>HelpFunction</b> ( <i>text</i> , <i>append</i> =False)
---

<b>Variables</b> ( <i>files</i> =[], <i>args</i> ={})
---

<b>Options</b> ( <i>files</i> =[], <i>args</i> ={})
---

### 29.3 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Script/__init__.py rel_2.4.1:3453:73fef...
<code>start_time</code>	<b>Value:</b> 1447068421.53
<code>call_stack</code>	<b>Value:</b> []
<code>PathVariable</code>	<b>Value:</b> SCons.Variables.PathVariable
<code>PathOption</code>	<b>Value:</b> SCons.Options.PathOption
<code>Chmod</code>	<b>Value:</b> SCons.Defaults.Chmod
<code>Copy</code>	<b>Value:</b> SCons.Defaults.Copy
<code>Delete</code>	<b>Value:</b> SCons.Defaults.Delete

*continued on next page*

Name	Description
Mkdir	<b>Value:</b> SCons.Defaults.Mkdir
Move	<b>Value:</b> SCons.Defaults.Move
Touch	<b>Value:</b> SCons.Defaults.Touch
CScanner	<b>Value:</b> SCons.Defaults.CScan
DScanner	<b>Value:</b> SCons.Tool.DScanner
DirScanner	<b>Value:</b> SCons.Defaults.DirScanner
ProgramScanner	<b>Value:</b> SCons.Tool.ProgramScanner
SourceFileScanner	<b>Value:</b> SCons.Tool.SourceFileScanner
CScan	<b>Value:</b> SCons.Defaults.CScan
ARGUMENTS	<b>Value:</b> {}
ARGLIST	<b>Value:</b> []
BUILD_TARGETS	<b>Value:</b> []
COMMAND_LINE_TARGETS	<b>Value:</b> []
DEFAULT_TARGETS	<b>Value:</b> []
help_text	<b>Value:</b> None
sconscrip_reading	<b>Value:</b> 0
GlobalDefaultEnvironmentFunctions	<b>Value:</b> ['Default', 'EnsurePythonVersion', 'EnsureSConsVersion', ...]
GlobalDefaultBuilders	<b>Value:</b> ['CFile', 'CXXFile', 'DVI', 'Jar', 'Java', 'JavaH', 'Libr...]
SConscript	<b>Value:</b> _SConscript.DefaultEnvironmentCall('SConscript')
Command	<b>Value:</b> _SConscript.DefaultEnvironmentCall('Command', subst= 1)
AddPostAction	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...>
AddPreAction	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...>
Alias	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...>
AlwaysBuild	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...>
BuildDir	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...>

*continued on next page*

Name	Description
CFile	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
CXXFile	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
CacheDir	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Clean	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
DVI	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Decider	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Default	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Depends	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Dir	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
EnsurePythonVersion	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
EnsureSConsVersion	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Entry	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Execute	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Exit	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...

*continued on next page*



Name	Description
Export	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
File	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
FindFile	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
FindInstalledFiles	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
FindSourceFiles	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Flatten	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
GetBuildPath	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
GetLaunchDir	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Glob	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Help	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Ignore	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Import	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Install	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
InstallAs	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...

*continued on next page*

Name	Description
InstallVersionedLib	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Jar	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Java	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
JavaH	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Library	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Literal	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Local	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
M4	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
MSVSPProject	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
NoCache	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
NoClean	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Object	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
PCH	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
PDF	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...

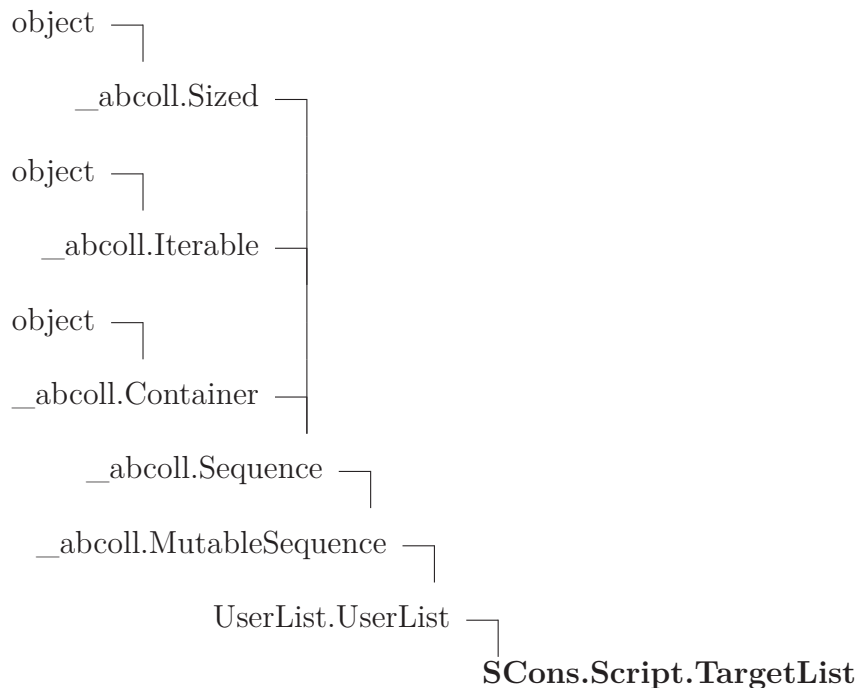
*continued on next page*

Name	Description
Package	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
ParseDepends	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
PostScript	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Precious	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Program	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
RES	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
RMIC	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Repository	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Requires	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SConscriptChdir	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SConsignFile	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SharedLibrary	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SharedObject	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SideEffect	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...

*continued on next page*

Name	Description
SourceCode	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SourceSignatures	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Split	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
StaticLibrary	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
StaticObject	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Tag	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Tar	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
TargetSignatures	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
TypeLibrary	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Value	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
VariantDir	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Zip	<b>Value:</b> <SCons.Script.SConscript.DefaultEnvironmentCall object at...
__package__	<b>Value:</b> 'SCons.Script'

## 29.4 Class *TargetList*



### 29.4.1 Methods

#### *Inherited from UserList.UserList*

`__add__()`, `__cmp__()`, `__contains__()`, `__delitem__()`, `__delslice__()`, `__eq__()`, `__ge__()`, `__getitem__()`, `__getslice__()`, `__gt__()`, `__iadd__()`, `__imul__()`, `__init__()`, `__le__()`, `__len__()`, `__lt__()`, `__mul__()`, `__ne__()`, `__radd__()`, `__repr__()`, `__rmul__()`, `__setitem__()`, `__setslice__()`, `append()`, `count()`, `extend()`, `index()`, `insert()`, `pop()`, `remove()`, `reverse()`, `sort()`

#### *Inherited from \_\_abcoll.Sequence*

`__iter__()`, `__reversed__()`

#### *Inherited from \_\_abcoll.Sized*

`__subclasshook__()`

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__str__()`

**29.4.2 Properties**

Name	Description
<i>Inherited from object</i> __class__	

**29.4.3 Class Variables**

Name	Description
<i>Inherited from UserList.UserList</i> __abstractmethods__, __hash__	

## 30 Module *SCons.Script.Interactive*

SCons interactive mode

### 30.1 Functions

<code>interact(<i>fs</i>, <i>parser</i>, <i>options</i>, <i>targets</i>, <i>target_top</i>)</code>
--

### 30.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Script/Interactive.py rel_2.4.1:3453:73...'
<code>__doc__</code>	<b>Value:</b> ...
<code>__package__</code>	<b>Value:</b> 'SCons.Script'

### 30.3 Class *SConsInteractiveCmd*

```
cmd.Cmd └─ SCons.Script.Interactive.SConsInteractiveCmd
```

<code>build [TARGETS]</code>	Build the specified TARGETS and their dependencies. 'b' is a synonym.
<code>clean [TARGETS]</code>	Clean (remove) the specified TARGETS and their dependencies. 'c' is a synonym.
<code>exit</code>	Exit SCons interactive mode.
<code>help [COMMAND]</code>	Prints help for the specified COMMAND. 'h' and '?' are synonyms.
<code>shell [COMMANDLINE]</code>	Execute COMMANDLINE in a subshell. 'sh' and '!' are synonyms.
<code>version</code>	Prints SCons version information.

**30.3.1 Methods**

**\_\_init\_\_**(*self*, *\*\*kw*)

Instantiate a line-oriented interpreter framework.

The optional argument 'completekey' is the readline name of a completion key; it defaults to the Tab key. If completekey is not None and the readline module is available, command completion is done automatically. The optional arguments stdin and stdout specify alternate input and output file objects; if not specified, sys.stdin and sys.stdout are used. Overrides: cmd.Cmd.\_\_init\_\_ exitit(inherited documentation)

**default**(*self*, *argv*)

Called on an input line when the command prefix is not recognized.

If this method is not overridden, it prints an error message and returns. Overrides: cmd.Cmd.default exitit(inherited documentation)

**onecmd**(*self*, *line*)

Interpret the argument as though it had been typed in response to the prompt.

This may be overridden, but should not normally need to be; see the precmd() and postcmd() methods for useful execution hooks. The return value is a flag indicating whether interpretation of commands by the interpreter should stop. Overrides: cmd.Cmd.onecmd exitit(inherited documentation)

**do\_build**(*self*, *argv*)

build [TARGETS] Build the specified TARGETS and their dependencies. 'b' is a synonym.

**do\_clean**(*self*, *argv*)

clean [TARGETS] Clean (remove) the specified TARGETS and their dependencies. 'c' is a synonym.

**do\_EOF**(*self*, *argv*)



**do\_exit**(*self*, *argv*)

exit Exit SCons interactive mode.

**do\_help**(*self*, *argv*)

help [COMMAND] Prints help for the specified COMMAND. 'h' and '?' are synonyms. Overrides: cmd.Cmd.do\_help

**do\_shell**(*self*, *argv*)

shell [COMMANDLINE] Execute COMMANDLINE in a subshell. 'sh' and '!' are synonyms.

**do\_version**(*self*, *argv*)

version Prints SCons version information.

### *Inherited from cmd.Cmd*

cmdloop(), columnize(), complete(), complete\_help(), completedefault(), complete\_names(), emptyline(), get\_names(), parseline(), postcmd(), postloop(), precmd(), preloop(), print\_topics()

### 30.3.2 Class Variables

Name	Description
synonyms	<b>Value:</b> {'b': 'build', 'c': 'clean', 'h': 'help', 'scons': 'build...}
<i>Inherited from cmd.Cmd</i> doc_header, doc_leader, identchars, intro, lastcmd, misc_header, nohelp, prompt, ruler, undoc_header, use_rawinput	

## 31 Module *SCons.Script.Main*

### *SCons.Script*

This file implements the `main()` function used by the `scons` script.

Architecturally, this *is* the `scons` script, and will likely only be called from the external "scons" wrapper. Consequently, anything here should not be, or be considered, part of the build engine. If it's something that we expect other software to want to use, it should go in some other module. If it's specific to the "scons" script invocation, it goes here.

### 31.1 Functions

```
fetch_win32_parallel_msg()
```

```
revert_io()
```

```
Progress(*args, **kw)
```

```
GetBuildFailures()
```

```
python_version_string()
```

```
python_version_unsupported(version=sys.version_info(major=2,
minor=7, micro=6, releaselevel=...))
```

```
python_version_deprecated(version=sys.version_info(major=2,
minor=7, micro=6, releaselevel=...))
```

```
AddOption(*args, **kw)
```

```
GetOption(name)
```

```
SetOption(name, value)
```

```
PrintHelp(file=None)
```

**find\_deepest\_user\_frame(*tb*)**

Find the deepest stack frame that is not part of SCons.

Input is a "pre-processed" stack trace in the form returned by `traceback.extract_tb()` or `traceback.extract_stack()`

**test\_load\_all\_site\_scons\_dirs(*d*)**

**version\_string(*label, module*)**

**path\_string(*label, module*)**

**main()**

## 31.2 Variables

Name	Description
<code>unsupported_python_version</code>	<b>Value:</b> (2, 3, 0)
<code>deprecated_python_version</code>	<b>Value:</b> (2, 7, 0)
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Script/Main.py rel_2.4.1:3453:73fe3d3ea...
<code>display</code>	<b>Value:</b> DisplayEngine()
<code>progress_display</code>	<b>Value:</b> SCons.Util.DisplayEngine()
<code>first_command_start</code>	<b>Value:</b> None
<code>last_command_end</code>	<b>Value:</b> None
<code>ProgressObject</code>	<b>Value:</b> Null(0xB67475AC)
<code>print_objects</code>	<b>Value:</b> 0
<code>print_memoizer</code>	<b>Value:</b> 0
<code>print_stacktrace</code>	<b>Value:</b> 0
<code>print_time</code>	<b>Value:</b> 0
<code>sconscrip_time</code>	<b>Value:</b> 0
<code>cumulative_command_time</code>	<b>Value:</b> 0
<code>exit_status</code>	<b>Value:</b> 0
<code>this_build_status</code>	<b>Value:</b> 0
<code>num_jobs</code>	<b>Value:</b> None
<code>delayed_warnings</code>	<b>Value:</b> []
<code>OptionsParser</code>	<b>Value:</b> FakeOptionParser()

*continued on next page*

Name	Description
count_stats	<b>Value:</b> CountStats()
memory_stats	<b>Value:</b> MemStats()
__package__	<b>Value:</b> 'SCons.Script'

### 31.3 Class *SConsPrintHelpException*



#### 31.3.1 Methods

##### *Inherited from exceptions.Exception*

\_\_init\_\_(), \_\_new\_\_()

##### *Inherited from exceptions.BaseException*

\_\_delattr\_\_(), \_\_getattr\_\_(), \_\_getitem\_\_(), \_\_getslice\_\_(), \_\_reduce\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_setstate\_\_(), \_\_str\_\_(), \_\_unicode\_\_()

##### *Inherited from object*

\_\_format\_\_(), \_\_hash\_\_(), \_\_reduce\_ex\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

#### 31.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

## 31.4 Class Progressor



### 31.4.1 Methods

```
__init__(self, obj, interval=1, file=None, overwrite=False)
```

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides:  
object.**\_\_init\_\_** **exitit**(inherited documentation)

```
write(self, s)
```

```
erase__previous(self)
```

```
spinner(self, node)
```

```
string(self, node)
```

```
replace__string(self, node)
```

```
__call__(self, node)
```

### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),  
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),  
__str__(), __subclasshook__()
```

### 31.4.2 Properties

Name	Description
<i>Inherited from object</i> <b>__class__</b>	

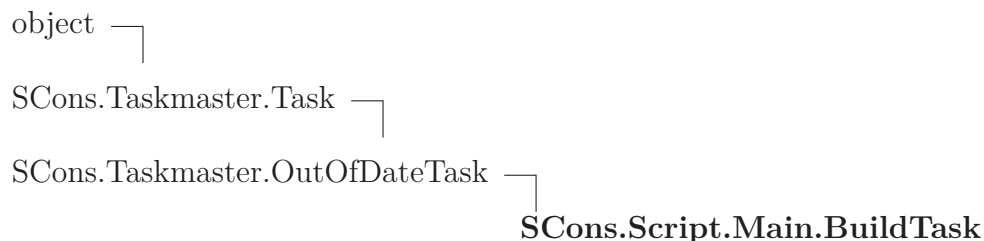
### 31.4.3 Class Variables

Name	Description
prev	<b>Value:</b> ''

*continued on next page*

Name	Description
count	<b>Value:</b> 0
target_string	<b>Value:</b> '\$TARGET'

### 31.5 Class BuildTask



An SCons build task.

#### 31.5.1 Methods

##### **display**(*self*, *message*)

Hook to allow the calling interface to display a message.

This hook gets called as part of preparing a task for execution (that is, a Node to be built). As part of figuring out what Node should be built next, the actually target list may be altered, along with a message describing the alteration. The calling interface can subclass Task and provide a concrete implementation of this method to see those messages. Overrides: SCons.Taskmaster.Task.display extit(inherited documentation)

##### **prepare**(*self*)

Called just before the task is executed.

This is mainly intended to give the target Nodes a chance to unlink underlying files and make all necessary directories before the Action is actually called to build the targets. Overrides: SCons.Taskmaster.Task.prepare extit(inherited documentation)

##### **needs\_execute**(*self*)

Returns True (indicating this Task should be executed) if this Task's target state indicates it needs executing, which has already been determined by an earlier up-to-date check. Overrides: SCons.Taskmaster.Task.needs\_execute

**execute(*self*)**

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in `prepare()`, `executed()` or `failed()`. Overrides: `SCons.Taskmaster.Task.execute` extit(inherited documentation)

**do\_failed(*self*, *status*=2)****executed(*self*)**

Called when the task has been successfully executed and the Taskmaster instance wants to call the Node's callback methods.

This may have been a do-nothing operation (to preserve build order), so we must check the node's state before deciding whether it was "built", in which case we call the appropriate Node method. In any event, we always call "visited()", which will handle any post-visit actions that must take place regardless of whether or not the target was an actual built target or a source Node. Overrides: `SCons.Taskmaster.Task.executed` extit(inherited documentation)

**failed(*self*)**

Default action when a task fails: stop the build.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using `Configure()`. Overrides: `SCons.Taskmaster.Task.failed` extit(inherited documentation)

**postprocess(*self*)**

Post-processes a task after it's been executed.

This examines all the targets just built (or not, we don't care if the build was successful, or even if there was no build because everything was up-to-date) to see if they have any waiting parent Nodes, or Nodes waiting on a common side effect, that can be put back on the candidates list. Overrides: `SCons.Taskmaster.Task.postprocess` extit(inherited documentation)

<b>make_ready(self)</b>
Make a task ready for execution Overrides: SCons.Taskmaster.Task.make_ready

***Inherited from SCons.Taskmaster.Task(Section 35.4)***

`__init__()`, `exc_clear()`, `exc_info()`, `exception_set()`, `executed_with_callbacks()`,  
`executed_without_callbacks()`, `fail_continue()`, `fail_stop()`, `get_target()`, `make_ready_all()`,  
`make_ready_current()`, `trace_message()`

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

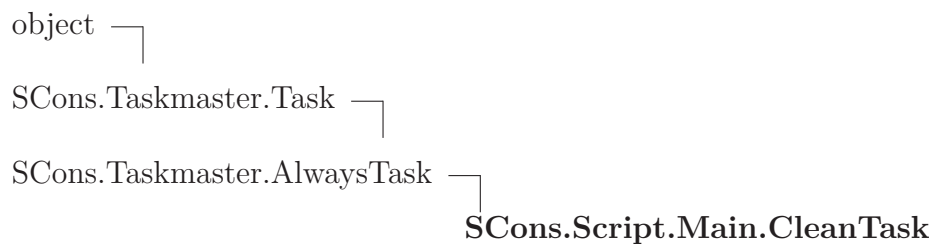
### 31.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 31.5.3 Class Variables

Name	Description
<code>progress</code>	<b>Value:</b> Null(0xB67475AC)

## 31.6 Class CleanTask



An SCons clean task.



### 31.6.1 Methods

**fs\_delete**(*self*, *path*, *pathstr*, *remove=True*)

**show**(*self*)

**remove**(*self*)

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in `prepare()`, `executed()` or `failed()`.

**execute**(*self*)

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in `prepare()`, `executed()` or `failed()`. Overrides: `SCons.Taskmaster.Task.execute` `exitit`(inherited documentation)

**executed**(*self*)

Called when the task has been successfully executed and the Taskmaster instance doesn't want to call the Node's callback methods. Overrides: `SCons.Taskmaster.Task.executed`

**make\_ready**(*self*)

Marks all targets in a task ready for execution.

This is used when the interface needs every target Node to be visited--the canonical example being the "scons -c" option. Overrides: `SCons.Taskmaster.Task.make_ready`

**prepare(*self*)**

Called just before the task is executed.

This is mainly intended to give the target Nodes a chance to unlink underlying files and make all necessary directories before the Action is actually called to build the targets. Overrides: SCons.Taskmaster.Task.prepare extit(inherited documentation)

**Inherited from SCons.Taskmaster.AlwaysTask(Section 35.5)**

needs\_execute()

**Inherited from SCons.Taskmaster.Task(Section 35.4)**

\_\_init\_\_(), display(), exc\_clear(), exc\_info(), exception\_set(), executed\_with\_callbacks(), executed\_without\_callbacks(), fail\_continue(), fail\_stop(), failed(), get\_target(), make\_ready\_all(), make\_ready\_current(), postprocess(), trace\_message()

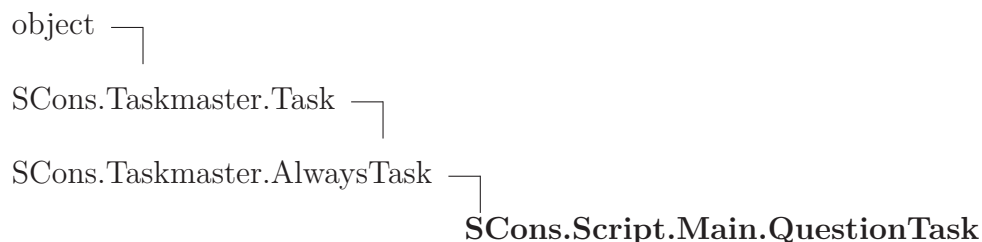
**Inherited from object**

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

### 31.6.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 31.7 Class QuestionTask



An SCons task for the -q (question) option.

**31.7.1 Methods****prepare(*self*)**

Called just before the task is executed.

This is mainly intended to give the target Nodes a chance to unlink underlying files and make all necessary directories before the Action is actually called to build the targets. Overrides: SCons.Taskmaster.Task.prepare extit(inherited documentation)

**execute(*self*)**

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in prepare(), executed() or failed(). Overrides: SCons.Taskmaster.Task.execute extit(inherited documentation)

**executed(*self*)**

Called when the task has been successfully executed and the Taskmaster instance wants to call the Node's callback methods.

This may have been a do-nothing operation (to preserve build order), so we must check the node's state before deciding whether it was "built", in which case we call the appropriate Node method. In any event, we always call "visited()", which will handle any post-visit actions that must take place regardless of whether or not the target was an actual built target or a source Node. Overrides: SCons.Taskmaster.Task.executed extit(inherited documentation)

***Inherited from SCons.Taskmaster.AlwaysTask(Section 35.5)***

needs\_execute()

***Inherited from SCons.Taskmaster.Task(Section 35.4)***

\_\_init\_\_(), display(), exc\_clear(), exc\_info(), exception\_set(), executed\_with\_callbacks(), executed\_without\_callbacks(), fail\_continue(), fail\_stop(), failed(), get\_target(), make\_ready(), make\_ready\_all(), make\_ready\_current(), postprocess(), trace\_message()

***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),

`__str__()`, `__subclasshook__()`

### 31.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 31.8 Class `TreePrinter`

```

object
 |
 |__SCons.Script.Main.TreePrinter

```

### 31.8.1 Methods

`__init__(self, derived=False, prune=False, status=False)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature Overrides:  
`object.__init__` `exitit`(inherited documentation)

`get_all_children(self, node)`

`get_derived_children(self, node)`

`display(self, t)`

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

### 31.8.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 31.9 Class FakeOptionParser



A do-nothing option parser, used for the initial OptionsParser variable.

During normal SCons operation, the OptionsParser is created right away by the main() function. Certain tests scripts however, can introspect on different Tool modules, the initialization of which can try to add a new, local option to an otherwise uninitialized OptionsParser object. This allows that introspection to happen without blowing up.

#### 31.9.1 Methods

<code>add_local_option(self, *args, **kw)</code>
--

*Inherited from object*

```

__delattr__(), __format__(), __getattr__(), __hash__(), __init__(),
__new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(),
__sizeof__(), __str__(), __subclasshook__()

```

#### 31.9.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

#### 31.9.3 Class Variables

Name	Description
values	<b>Value:</b> FakeOptionValues()

### 31.10 Class Stats



**Known Subclasses:** SCons.Script.Main.CountStats, SCons.Script.Main.MemStats

### 31.10.1 Methods

```
__init__(self)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature Overrides:  
`object.__init__` `exitit`(inherited documentation)

```
enable(self, outfp)
```

```
do_nothing(self, *args, **kw)
```

#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 31.10.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 31.11 Class CountStats



### 31.11.1 Methods

```
do_append(self, label)
```

```
do_print(self)
```

#### *Inherited from SCons.Script.Main.Stats(Section 31.10)*

```
__init__(), do_nothing(), enable()
```

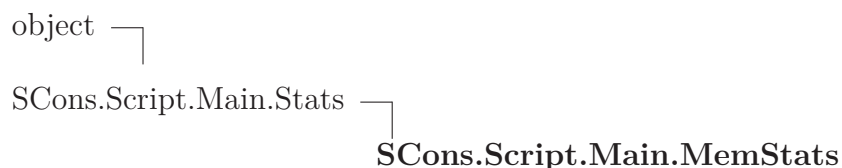
#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 31.11.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 31.12 Class MemStats



### 31.12.1 Methods

```
do_append(self, label)
```

```
do_print(self)
```

*Inherited from SCons.Script.Main.Stats(Section 31.10)*

```
__init__(), do_nothing(), enable()
```

*Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 31.12.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 32 Module *SCons.Script.SConscript*

*SCons.Script.SConscript*

This module defines the Python API provided to *SConscript* and *SConstruct* files.

### 32.1 Functions

**get\_calling\_namespaces()**

Return the locals and globals for the function that called into this module in the current call stack.

**compute\_exports(*exports*)**

Compute a dictionary of exports given one of the parameters to the *Export()* function or the *exports* argument to *SConscript()*.

**Return(\**vars*, \*\**kw*)**

**SConscript\_exception(*file=sys.stderr*)**

Print an exception stack trace just for the *SConscript* file(s). This will show users who have Python errors where the problem is, without cluttering the output with all of the internal calls leading up to where we exec the *SConscript*.

**annotate(*node*)**

Annotate a node with the stack frame describing the *SConscript* file and line number that created it.

**Configure(\**args*, \*\**kw*)**

**get\_DefaultEnvironmentProxy()**



**BuildDefaultGlobals()**

Create a dictionary containing all the default globals for SConstruct and SConscript files.

**32.2 Variables**

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Script/SConscript.py rel_2.4.1:3453:73f...'
<code>launch_dir</code>	<b>Value:</b> '/scons/as_scons'
<code>GlobalDict</code>	<b>Value:</b> None
<code>global_exports</code>	<b>Value:</b> {}
<code>sconscript_chdir</code>	<b>Value:</b> 1
<code>call_stack</code>	<b>Value:</b> []
<code>stack_bottom</code>	<b>Value:</b> '% Stack boTtom %'
<code>__package__</code>	<b>Value:</b> 'SCons.Script'

**32.3 Class SConscriptReturn****32.3.1 Methods**

*Inherited from exceptions.Exception*

`__init__()`, `__new__()`

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**32.3.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

**32.4 Class Frame**

object —  
**SCons.Script.SConscript'.Frame**

A frame on the SConstruct/SConscript call stack

**32.4.1 Methods**

<code>__init__(self, fs, exports, sconsript)</code>
x. <code>__init__</code> (...) initializes x; see <code>help(type(x))</code> for signature Overrides: object. <code>__init__</code> <code>exitit</code> (inherited documentation)

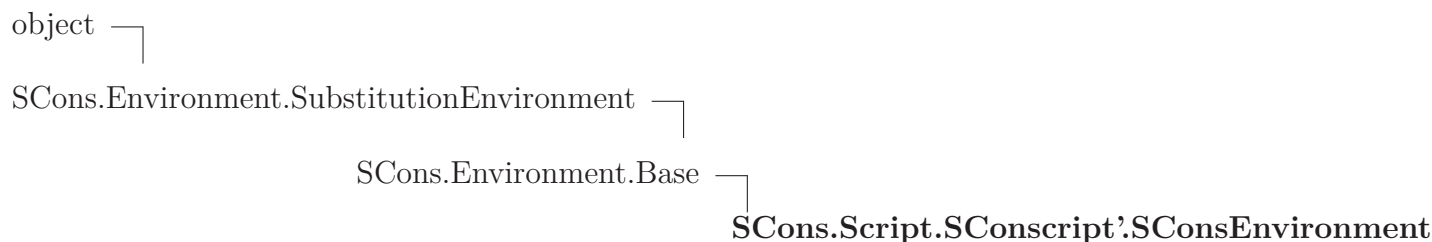
*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

**32.4.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 32.5 Class *SConsEnvironment*



An Environment subclass that contains all of the methods that are particular to the wrapper SCons interface and which aren't (or shouldn't be) part of the build engine itself.

Note that not all of the methods of this class have corresponding global functions, there are some private methods.

### 32.5.1 Methods

**Configure**(*self*, \**args*, \*\**kw*)

Overrides: *SCons.Environment.Base.Configure*

**Default**(*self*, \**targets*)

**EnsureSConsVersion**(*self*, *major*, *minor*, *revision*=0)

Exit abnormally if the SCons version is not late enough.

**EnsurePythonVersion**(*self*, *major*, *minor*)

Exit abnormally if the Python version is not late enough.

**Exit**(*self*, *value*=0)

**Export**(*self*, \**vars*, \*\**kw*)

**GetLaunchDir**(*self*)

**GetOption**(*self*, *name*)

<b>Help</b> ( <i>self</i> , <i>text</i> , <i>append=False</i> )
---

<b>Import</b> ( <i>self</i> , * <i>vars</i> )
---

<b>SConscript</b> ( <i>self</i> , * <i>ls</i> , ** <i>kw</i> )
--

<b>SConscriptChdir</b> ( <i>self</i> , <i>flag</i> )
--

<b>SetOption</b> ( <i>self</i> , <i>name</i> , <i>value</i> )
---

### ***Inherited from SCons.Environment.Base(Section 8.9)***

Action(), AddPostAction(), AddPreAction(), Alias(), AlwaysBuild(), Append(), AppendENVPath(), AppendUnique(), BuildDir(), Builder(), CacheDir(), Clean(), Clone(), Command(), Copy(), Decider(), Depends(), Detect(), Dictionary(), Dir(), Dump(), Entry(), Environment(), Execute(), File(), FindFile(), FindInstalledFiles(), FindIxes(), FindSourceFiles(), Flatten(), GetBuildPath(), Glob(), Ignore(), Literal(), Local(), NoCache(), NoClean(), ParseConfig(), ParseDepends(), Platform(), Precious(), Prepend(), PrependENVPath(), PrependUnique(), Pseudo(), Replace(), ReplaceIxes(), Repository(), Requires(), SConsignFile(), Scanner(), SetDefault(), SideEffect(), SourceCode(), SourceSignatures(), Split(), TargetSignatures(), Tool(), Value(), VariantDir(), WhereIs(), \_\_init\_\_(), get\_CacheDir(), get\_builder(), get\_factory(), get\_scanner(), get\_src\_sig\_type(), get\_tgt\_sig\_type(), scanner\_map\_delete()

### ***Inherited from SCons.Environment.SubstitutionEnvironment(Section 8.6)***

AddMethod(), MergeFlags(), Override(), ParseFlags(), RemoveMethod(), \_\_cmp\_\_(), \_\_contains\_\_(), \_\_delitem\_\_(), \_\_getitem\_\_(), \_\_setitem\_\_(), arg2nodes(), backtick(), get(), gvars(), has\_key(), items(), lvars(), subst(), subst\_kw(), subst\_list(), subst\_path(), subst\_target\_source()

### ***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

#### **32.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
__class__	

## 32.6 Class `DefaultEnvironmentCall`



A class that implements "global function" calls of Environment methods by fetching the specified method from the DefaultEnvironment's class. Note that this uses an intermediate proxy class instead of calling the DefaultEnvironment method directly so that the proxy can override the `subst()` method and thereby prevent expansion of construction variables (since from the user's point of view this was called as a global function, with no associated construction environment).

### 32.6.1 Methods

<code>__init__(self, method_name, subst=0)</code>
---

<code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)
---

<code>__call__(self, *args, **kw)</code>
--

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

### 32.6.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 33 Module SCons.Sig

Place-holder for the old SCons.Sig module hierarchy

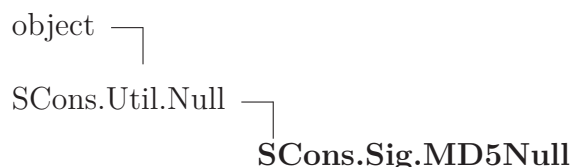
This is no longer used, but code out there (such as the NSIS module on the SCons wiki) may try to import SCons.Sig. If so, we generate a warning that points them to the line that caused the import, and don't die.

If someone actually tried to use the sub-modules or functions within the package (for example, SCons.Sig.MD5.signature()), then they'll still get an AttributeError, but at least they'll know where to start looking.

#### 33.1 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> <code>'src/engine/SCons/Sig.py rel_2.4.1:3453:73fef3ea0b0 2015...</code>
<code>__doc__</code>	<b>Value:</b> <code>"""Place-holder for the old SCons.Sig module hierar...</code>
<code>msg</code>	<b>Value:</b> <code>'The SCons.Sig module no longer exists.\n Remove the f...</code>
<code>default_calc</code>	<b>Value:</b> <code>None</code>
<code>default_module</code>	<b>Value:</b> <code>None</code>
<code>MD5</code>	<b>Value:</b> <code>MD5Null()</code>
<code>TimeStamp</code>	<b>Value:</b> <code>TimeStampNull()</code>
<code>__package__</code>	<b>Value:</b> <code>'SCons'</code>

#### 33.2 Class MD5Null



##### 33.2.1 Methods

<code>__repr__(self)</code>
<code>repr(x)</code> Overrides: <code>object.__repr__</code> <code>exitit</code> (inherited documentation)

*Inherited from SCons.Util.Null(Section 36.15)*

`__call__()`, `__delattr__()`, `__getattr__()`, `__init__()`, `__new__()`, `__nonzero__()`,  
`__setattr__()`

### *Inherited from object*

`__format__()`, `__getattribute__()`, `__hash__()`, `__reduce__()`, `__reduce_ex__()`,  
`__sizeof__()`, `__str__()`, `__subclasshook__()`

### 33.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 33.3 Class `TimeStampNull`



### 33.3.1 Methods

<code>__repr__(self)</code> <code>repr(x)</code> Overrides: <code>object.__repr__</code> <code>exitit</code> (inherited documentation)
---

### *Inherited from SCons.Util.Null(Section 36.15)*

`__call__()`, `__delattr__()`, `__getattr__()`, `__init__()`, `__new__()`, `__nonzero__()`,  
`__setattr__()`

### *Inherited from object*

`__format__()`, `__getattribute__()`, `__hash__()`, `__reduce__()`, `__reduce_ex__()`,  
`__sizeof__()`, `__str__()`, `__subclasshook__()`

### 33.3.2 Properties

Name	Description
<i>Inherited from object</i>	

*continued on next page*

Name	Description
__class__	



## 34 Module SCons.Subst

SCons.Subst

SCons string substitution.

### 34.1 Functions

<b>SetAllowableExceptions</b> (* <i>excepts</i> )
---

<b>raise__exception</b> ( <i>exception</i> , <i>target</i> , <i>s</i> )
---

<b>quote__spaces</b> ( <i>arg</i> )
-------------------------------------

Generic function for putting double quotes around any string that has white space in it.

<b>escape__list</b> ( <i>mylist</i> , <i>escape__func</i> )
---

Escape a list of arguments by running the specified `escape__func` on every object in the list that has an `escape()` method.

<b>subst__dict</b> ( <i>target</i> , <i>source</i> )
--

Create a dictionary for substitution of special construction variables.

This translates the following special arguments:

**target - the target (object or array of objects)**, used to generate the TARGET and TARGETS construction variables

**source - the source (object or array of objects)**, used to generate the SOURCES and SOURCE construction variables

```
scons__subst(strSubst, env, mode=1, target=None, source=None, gvars={},  
lvars={}, conv=None)
```

Expand a string or list containing construction variable substitutions.

This is the work-horse function for substitutions in file names and the like. The companion `scons__subst__list()` function (below) handles separating command lines into lists of arguments, so see that function if that's what you're looking for.

```
scons__subst__list(strSubst, env, mode=1, target=None, source=None,  
gvars={}, lvars={}, conv=None)
```

Substitute construction variables in a string (or list or other object) and separate the arguments into a command list.

The companion `scons__subst()` function (above) handles basic substitutions within strings, so see that function instead if that's what you're looking for.

```
scons__subst__once(strSubst, env, key)
```

Perform single (non-recursive) substitution of a single construction variable keyword.

This is used when setting a variable when copying or overriding values in an Environment. We want to capture (expand) the old value before we override it, so people can do things like:

```
env2 = env.Clone(CCFLAGS = '$CCFLAGS -g')
```

We do this with some straightforward, brute-force code here...

## 34.2 Variables

Name	Description
<code>__revision__</code>	<b>Value:</b> 'src/engine/SCons/Subst.py rel_2.4.1:3453:73fef3ea0b0 20...
<code>AllowableExceptions</code>	<b>Value:</b> (<type 'exceptions.IndexError'>, <type 'exceptions.NameEr...
<code>NullNodesList</code>	<b>Value:</b> Null(0xB6B41B6C)

*continued on next page*

Name	Description
SUBST_CMD	Value: 0
SUBST_RAW	Value: 1
SUBST_SIG	Value: 2
__package__	Value: 'SCons'

### 34.3 Class Literal

object —  
**SCons.Subst.Literal**

A wrapper for a string. If you use this object wrapped around a string, then it will be interpreted as literal. When passed to the command interpreter, all special characters will be escaped.

#### 34.3.1 Methods

**\_\_init\_\_**(*self*, *lstr*)

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides: object.**\_\_init\_\_** **exitit**(inherited documentation)

**\_\_str\_\_**(*self*)

str(x) Overrides: object.**\_\_str\_\_** **exitit**(inherited documentation)

**escape**(*self*, *escape\_func*)

**for\_signature**(*self*)

**is\_literal**(*self*)

**\_\_eq\_\_**(*self*, *other*)

**\_\_neq\_\_**(*self*, *other*)

#### *Inherited from object*

**\_\_delattr\_\_**(), **\_\_format\_\_**(), **\_\_getattr\_\_**(), **\_\_hash\_\_**(), **\_\_new\_\_**(),  
**\_\_reduce\_\_**(), **\_\_reduce\_ex\_\_**(), **\_\_repr\_\_**(), **\_\_setattr\_\_**(), **\_\_sizeof\_\_**(),  
**\_\_subclasshook\_\_**()

### 34.3.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

## 34.4 Class SpecialAttrWrapper



This is a wrapper for what we call a 'Node special attribute.' This is any of the attributes of a Node that we can reference from Environment variable substitution, such as \$TARGET.abspath or \$SOURCES[1].filebase. We implement the same methods as Literal so we can handle special characters, plus a for\_signature method, such that we can return some canonical string during signature calculation to avoid unnecessary rebuilds.

### 34.4.1 Methods

<b>__init__</b> ( <i>self</i> , <i>lstr</i> , <i>for_signature</i> =None)
The for_signature parameter, if supplied, will be the canonical string we return from for_signature(). Else we will simply return lstr. Overrides: object.__init__
<b>__str__</b> ( <i>self</i> )
str(x) Overrides: object.__str__ exitit(inherited documentation)
<b>escape</b> ( <i>self</i> , <i>escape_func</i> )
<b>for_signature</b> ( <i>self</i> )
<b>is_literal</b> ( <i>self</i> )

*Inherited from object*

```

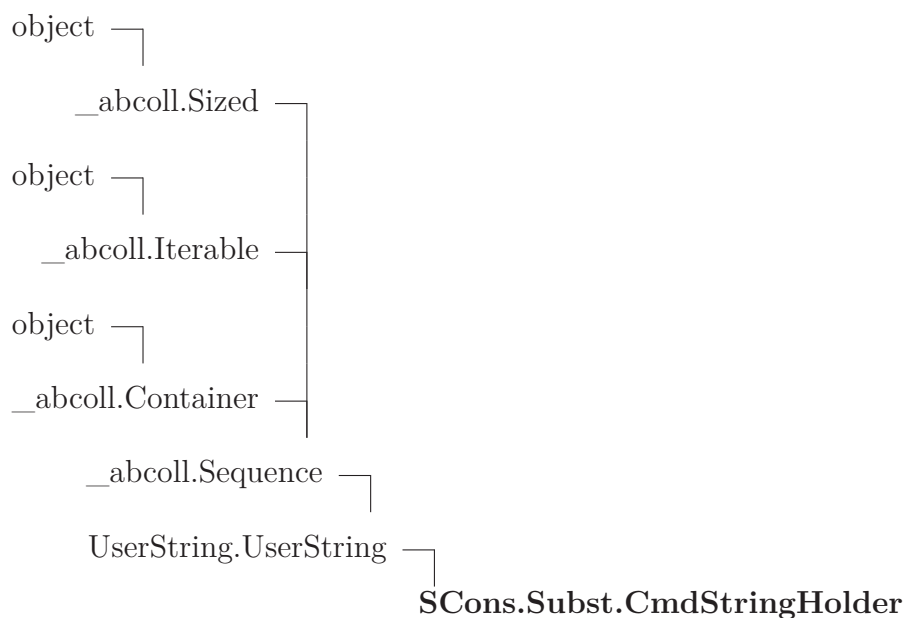
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()

```

### 34.4.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

## 34.5 Class `CmdStringHolder`



This is a special class used to hold strings generated by `scons_subst()` and `scons_subst_list()`. It defines a special method `escape()`. When passed a function with an escape algorithm for a particular platform, it will return the contained string with the proper escape sequences inserted.

### 34.5.1 Methods

<code>__init__(self, cmd, literal=None)</code>
--

<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)
---

<code>is_literal(self)</code>
-------------------------------

```
escape(self, escape_func, quote_func=<function quote_spaces at
0xb6b43144>)
```

Escape the string with the supplied function. The function is expected to take an arbitrary string, then return it with all special characters escaped and ready for passing to the command interpreter.

After calling this function, the next call to str() will return the escaped string.

### ***Inherited from UserString.UserString***

```
__add__(), __cmp__(), __complex__(), __contains__(), __float__(), __getitem__(),
__getslice__(), __hash__(), __int__(), __len__(), __long__(), __mod__(),
__mul__(), __radd__(), __repr__(), __rmul__(), __str__(), capitalize(),
center(), count(), decode(), encode(), endswith(), expandtabs(), find(), index(),
isalnum(), isalpha(), isdecimal(), isdigit(), islower(), isnumeric(), isspace(),
is-title(), isupper(), join(), ljust(), lower(), lstrip(), partition(), replace(), rfind(),
rindex(), rjust(), rpartition(), rsplit(),rstrip(), split(), splitlines(), startswith(),
strip(), swapcase(), title(), translate(), upper(), zfill()
```

### ***Inherited from \_\_abcoll.Sequence***

```
__iter__(), __reversed__()
```

### ***Inherited from \_\_abcoll.Sized***

```
__subclasshook__()
```

### ***Inherited from object***

```
__delattr__(), __format__(), __getattr__(), __new__(), __reduce__(),
__reduce_ex__(), __setattr__(), __sizeof__()
```

#### **34.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
__class__	

#### **34.5.3 Class Variables**

Name	Description
<i>Inherited from UserString.UserString</i>	
__abstractmethods__	

## 34.6 Class NLWrapper



A wrapper class that delays turning a list of sources or targets into a NodeList until it's needed. The specified function supplied when the object is initialized is responsible for turning raw nodes into proxies that implement the special attributes like `.abspath`, `.source`, etc. This way, we avoid creating those proxies just "in case" someone is going to use `$TARGET` or the like, and only go through the trouble if we really have to.

In practice, this might be a wash performance-wise, but it's a little cleaner conceptually...

### 34.6.1 Methods

```
__init__(self, list, func)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature Overrides:  
`object.__init__` `exitit`(inherited documentation)

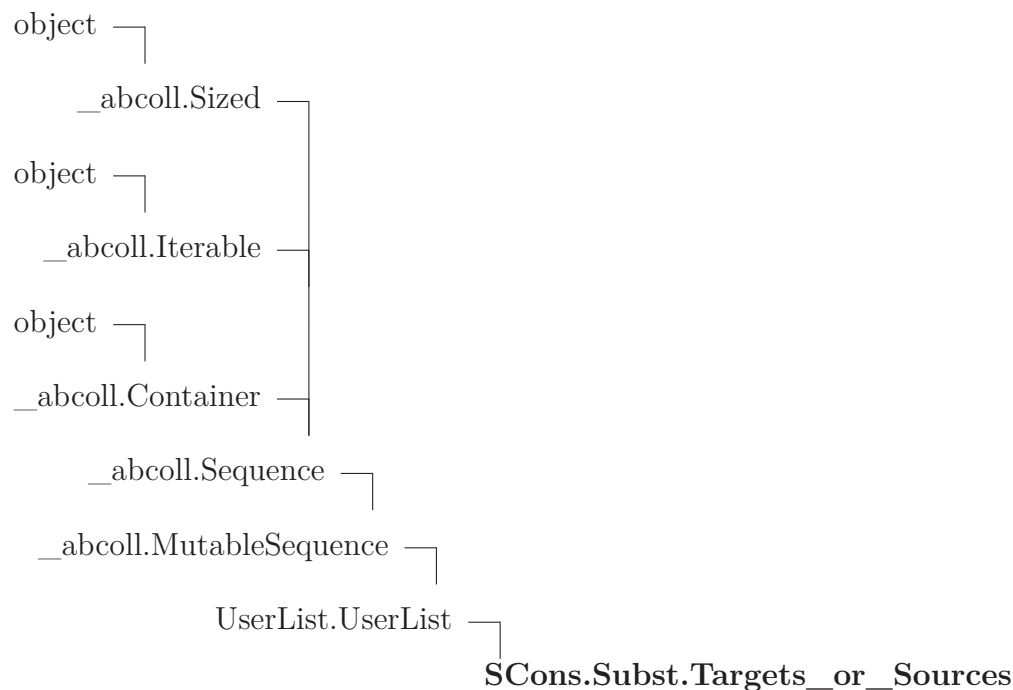
#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 34.6.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 34.7 Class `Targets_or_Sources`



A class that implements `$TARGETS` or `$SOURCES` expansions by in turn wrapping a `NLWrapper`. This class handles the different methods used to access the list, calling the `NLWrapper` to create proxies on demand.

Note that we subclass `collections.UserList` purely so that the `is_Sequence()` function will identify an object of this class as a list during variable expansion. We're not really using any `collections.UserList` methods in practice.

#### 34.7.1 Methods

**`__init__`**(*self*, *nl*)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature. Overrides: `object.__init__` extit(inherited documentation)

**`__getattr__`**(*self*, *attr*)

**`__getitem__`**(*self*, *i*)

Overrides: `_abcoll.Sequence.__getitem__`



<code>__getslice__(self, i, j)</code>
---------------------------------------

Overrides: UserList.UserList.__getslice__
---

<code>__str__(self)</code>
----------------------------

str(x) Overrides: object.__str__ exitit(inherited documentation)
--

<code>__repr__(self)</code>
-----------------------------

repr(x) Overrides: object.__repr__ exitit(inherited documentation)
--

### ***Inherited from UserList.UserList***

`__add__()`, `__cmp__()`, `__contains__()`, `__delitem__()`, `__delslice__()`,  
`__eq__()`, `__ge__()`, `__gt__()`, `__iadd__()`, `__imul__()`, `__le__()`, `__len__()`,  
`__lt__()`, `__mul__()`, `__ne__()`, `__radd__()`, `__rmul__()`, `__setitem__()`,  
`__setslice__()`, `append()`, `count()`, `extend()`, `index()`, `insert()`, `pop()`, `remove()`,  
`reverse()`, `sort()`

### ***Inherited from \_\_abcoll.Sequence***

`__iter__()`, `__reversed__()`

### ***Inherited from \_\_abcoll.Sized***

`__subclasshook__()`

### ***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`,  
`__reduce_ex__()`, `__setattr__()`, `__sizeof__()`

## **34.7.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## **34.7.3 Class Variables**

Name	Description
<i>Inherited from UserList.UserList</i>	
<code>__abstractmethods__</code> , <code>__hash__</code>	

## 34.8 Class `Target_or_Source`



A class that implements `$TARGET` or `$SOURCE` expansions by in turn wrapping a `NLWrapper`. This class handles the different methods used to access an individual proxy `Node`, calling the `NLWrapper` to create a proxy on demand.

### 34.8.1 Methods

<code>__init__(self, nl)</code>
---------------------------------

<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)
--

<code>__getattr__(self, attr)</code>
--------------------------------------

<code>__str__(self)</code>
----------------------------

<code>str(x)</code> Overrides: <code>object.__str__</code> <code>exitit</code> (inherited documentation)
--

<code>__repr__(self)</code>
-----------------------------

<code>repr(x)</code> Overrides: <code>object.__repr__</code> <code>exitit</code> (inherited documentation)
--

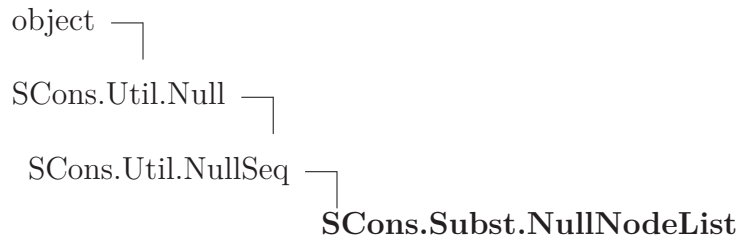
### *Inherited from `object`*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

### 34.8.2 Properties

Name	Description
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

### 34.9 Class *NullNodeList*



#### 34.9.1 Methods

<code>__call__(self, *args, **kwargs)</code>
--

Overrides: <i>SCons.Util.Null</i> . <code>__call__</code>
---

<code>__str__(self)</code>
----------------------------

<code>str(x)</code> Overrides: <i>object</i> . <code>__str__</code> <code>exitit</code> (inherited documentation)
---

*Inherited from SCons.Util.NullSeq(Section 36.16)*

`__delitem__()`, `__getitem__()`, `__iter__()`, `__len__()`, `__setitem__()`

*Inherited from SCons.Util.Null(Section 36.15)*

`__delattr__()`, `__getattr__()`, `__init__()`, `__new__()`, `__nonzero__()`, `__repr__()`,  
`__setattr__()`

*Inherited from object*

`__format__()`, `__getattribute__()`, `__hash__()`, `__reduce__()`, `__reduce_ex__()`,  
`__sizeof__()`, `__subclasshook__()`

#### 34.9.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 35 Module SCons.Taskmaster

Generic Taskmaster module for the SCons build engine.

This module contains the primary interface(s) between a wrapping user interface and the SCons build engine. There are two key classes here:

**Taskmaster** This is the main engine for walking the dependency graph and calling things to decide what does or doesn't need to be built.

**Task** This is the base class for allowing a wrapping interface to decide what does or doesn't actually need to be done. The intention is for a wrapping interface to subclass this as appropriate for different types of behavior it may need.

The canonical example is the SCons native Python interface, which has Task subclasses that handle its specific behavior, like printing "'foo' is up to date" when a top-level target doesn't need to be built, and handling the -c option by removing targets as its "build" action. There is also a separate subclass for suppressing this output when the -q option is used.

The Taskmaster instantiates a Task object for each (set of) target(s) that it decides need to be evaluated and/or built.

### 35.1 Functions

dump\_stats()

find\_cycle(*stack*, *visited*)

### 35.2 Variables

Name	Description
__doc__	Value: ...
__revision__	Value: 'src/engine/SCons/Taskmaster.py rel_2.4.1:3453:73fef3ea0...
StateString	Value: {0: 'no_state', 1: 'pending', 2: 'executing', 3: 'up_to_d...
NODE_NO_STATE	Value: 0
NODE_PENDING	Value: 1
NODE_EXECUTING	Value: 2
NODE_UP_TO_DATE	Value: 3
NODE_EXECUTED	Value: 4
NODE_FAILED	Value: 5

*continued on next page*

Name	Description
print_prepare	<b>Value:</b> 0
CollectStats	<b>Value:</b> None
StatsNodes	<b>Value:</b> []
fmt	<b>Value:</b> '%(considered)3d %(already_handled)3d %(problem)3d %(chil...
__package__	<b>Value:</b> 'SCons'

### 35.3 Class Stats

object —  
**SCons.Taskmaster.Stats**

A simple class for holding statistics about the disposition of a Node by the Taskmaster. If we're collecting statistics, each Node processed by the Taskmaster gets one of these attached, in which case the Taskmaster records its decision each time it processes the Node. (Ideally, that's just once per Node.)

#### 35.3.1 Methods

<b>__init__</b> ( <i>self</i> )
Instantiates a Taskmaster.Stats object, initializing all appropriate counters to zero. Overrides: object.__init__

#### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

#### 35.3.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

## 35.4 Class Task



**Known Subclasses:** SCons.Taskmaster.AlwaysTask, SCons.Taskmaster.OutOfDateTask

Default SCons build engine task.

This controls the interaction of the actual building of node and the rest of the engine.

This is expected to handle all of the normally-customizable aspects of controlling a build, so any given application *should* be able to do what it wants by sub-classing this class and overriding methods as appropriate. If an application needs to customize something by sub-classing Taskmaster (or some other build engine class), we should first try to migrate that functionality into this class.

Note that it's generally a good idea for sub-classes to call these methods explicitly to update state, etc., rather than roll their own interaction with Taskmaster from scratch.

### 35.4.1 Methods

```
__init__(self, tm, targets, top, node)
```

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides:  
object.**\_\_init\_\_** extit(inherited documentation)

```
trace_message(self, method, node, description='node')
```

```
display(self, message)
```

Hook to allow the calling interface to display a message.

This hook gets called as part of preparing a task for execution (that is, a Node to be built). As part of figuring out what Node should be built next, the actually target list may be altered, along with a message describing the alteration. The calling interface can subclass Task and provide a concrete implementation of this method to see those messages.

**prepare(*self*)**

Called just before the task is executed.

This is mainly intended to give the target Nodes a chance to unlink underlying files and make all necessary directories before the Action is actually called to build the targets.

**get\_target(*self*)**

Fetch the target being built or updated by this task.

**needs\_execute(*self*)****execute(*self*)**

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in prepare(), executed() or failed().

**executed\_without\_callbacks(*self*)**

Called when the task has been successfully executed and the Taskmaster instance doesn't want to call the Node's callback methods.

**executed\_with\_callbacks(*self*)**

Called when the task has been successfully executed and the Taskmaster instance wants to call the Node's callback methods.

This may have been a do-nothing operation (to preserve build order), so we must check the node's state before deciding whether it was "built", in which case we call the appropriate Node method. In any event, we always call "visited()", which will handle any post-visit actions that must take place regardless of whether or not the target was an actual built target or a source Node.

**executed(*self*)**

Called when the task has been successfully executed and the Taskmaster instance wants to call the Node's callback methods.

This may have been a do-nothing operation (to preserve build order), so we must check the node's state before deciding whether it was "built", in which case we call the appropriate Node method. In any event, we always call "visited()", which will handle any post-visit actions that must take place regardless of whether or not the target was an actual built target or a source Node.

**failed(*self*)**

Default action when a task fails: stop the build.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure().

**fail\_stop(*self*)**

Explicit stop-the-build failure.

This sets failure status on the target nodes and all of their dependent parent nodes.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure().

**fail\_continue(*self*)**

Explicit continue-the-build failure.

This sets failure status on the target nodes and all of their dependent parent nodes.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure().



---

**make\_ready\_all**(*self*)

Marks all targets in a task ready for execution.

This is used when the interface needs every target Node to be visited--the canonical example being the "scons -c" option.

---

**make\_ready\_current**(*self*)

Marks all targets in a task ready for execution if any target is not current.

This is the default behavior for building only what's necessary.

---

**make\_ready**(*self*)

Marks all targets in a task ready for execution if any target is not current.

This is the default behavior for building only what's necessary.

---

**postprocess**(*self*)

Post-processes a task after it's been executed.

This examines all the targets just built (or not, we don't care if the build was successful, or even if there was no build because everything was up-to-date) to see if they have any waiting parent Nodes, or Nodes waiting on a common side effect, that can be put back on the candidates list.

---

**exc\_info**(*self*)

Returns info about a recorded exception.

**exc\_clear**(*self*)

Clears any recorded exception.

This also changes the "exception\_raise" attribute to point to the appropriate do-nothing method.

**exception\_set**(*self*, *exception*=None)

Records an exception to be raised at the appropriate time.

This also changes the "exception\_raise" attribute to point to the method that will, in fact

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

#### 35.4.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

## 35.5 Class AlwaysTask

object └─

SCons.Taskmaster.Task └─

**SCons.Taskmaster.AlwaysTask**

**Known Subclasses:** SCons.SConf.SConfBuildTask, SCons.Script.Main.CleanTask, SCons.Script.Main.Q

### 35.5.1 Methods

**needs\_execute(*self*)**

Always returns True (indicating this Task should always be executed).

Subclasses that need this behavior (as opposed to the default of only executing Nodes that are out of date w.r.t. their dependencies) can use this as follows:

```
class MyTaskSubclass(SCons.Taskmaster.Task):
    needs_execute = SCons.Taskmaster.Task.execute_always
```

Overrides: SCons.Taskmaster.Task.needs\_execute

#### *Inherited from SCons.Taskmaster.Task(Section 35.4)*

`__init__()`, `display()`, `exc_clear()`, `exc_info()`, `exception_set()`, `execute()`, `executed()`, `executed_with_callbacks()`, `executed_without_callbacks()`, `fail_continue()`, `fail_stop()`, `failed()`, `get_target()`, `make_ready()`, `make_ready_all()`, `make_ready_current()`, `postprocess()`, `prepare()`, `trace_message()`

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 35.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 35.6 Class OutOfDateTask



**Known Subclasses:** SCons.Script.Main.BuildTask

### 35.6.1 Methods

**needs\_execute(*self*)**

Returns True (indicating this Task should be executed) if this Task's target state indicates it needs executing, which has already been determined by an earlier up-to-date check. Overrides: SCons.Taskmaster.Task.needs\_execute

*Inherited from SCons.Taskmaster.Task(Section 35.4)*

\_\_init\_\_(), display(), exc\_clear(), exc\_info(), exception\_set(), execute(), executed(), executed\_with\_callbacks(), executed\_without\_callbacks(), fail\_continue(), fail\_stop(), failed(), get\_target(), make\_ready(), make\_ready\_all(), make\_ready\_current(), postprocess(), prepare(), trace\_message()

*Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

### 35.6.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 35.7 Class Taskmaster

object —  
SCons.Taskmaster.Taskmaster

The Taskmaster for walking the dependency DAG.

### 35.7.1 Methods

**\_\_init\_\_(*self*, targets=[], tasker=None, order=None, trace=None)**

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides: object.\_\_init\_\_ extit(inherited documentation)

**find\_next\_candidate**(*self*)

Returns the next candidate Node for (potential) evaluation.

The candidate list (really a stack) initially consists of all of the top-level (command line) targets provided when the Taskmaster was initialized. While we walk the DAG, visiting Nodes, all the children that haven't finished processing get pushed on to the candidate list. Each child can then be popped and examined in turn for whether *their* children are all up-to-date, in which case a Task will be created for their actual evaluation and potential building.

Here is where we also allow candidate Nodes to alter the list of Nodes that should be examined. This is used, for example, when invoking SCons in a source directory. A source directory Node can return its corresponding build directory Node, essentially saying, "Hey, you really need to build this thing over here instead."

**no\_next\_candidate**(*self*)

Stops Taskmaster processing by not returning a next candidate.

Note that we have to clean-up the Taskmaster candidate list because the cycle detection depends on the fact all nodes have been processed somehow.

**trace\_message**(*self*, *message*)**trace\_node**(*self*, *node*)**next\_task**(*self*)

Returns the next task to be executed.

This simply asks for the next Node to be evaluated, and then wraps it in the specific Task subclass with which we were initialized.

```
will_not_build(self, nodes, node_func=<function <lambda> at 0xb69777d4>)
```

Perform clean-up about nodes that will never be built. Invokes a user defined function on all of these nodes (including all of their parents).

```
stop(self)
```

Stops the current build completely.

```
cleanup(self)
```

Check for dependency cycles.

### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 35.7.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

## 36 Module **SCons.Util**

SCons.Util

Various utility functions go here.

### 36.1 Functions

**dictify**(*keys*, *values*, *result*={})

**rightmost\_\_separator**(*path*, *sep*)

**containsAny**(*str*, *set*)

Check whether sequence *str* contains ANY of the items in *set*.

**containsAll**(*str*, *set*)

Check whether sequence *str* contains ALL of the items in *set*.

**containsOnly**(*str*, *set*)

Check whether sequence *str* contains ONLY items in *set*.

**splitext**(*path*)

Same as `os.path.splitext()` but faster.

**updrive**(*path*)

Make the drive letter (if any) upper case. This is useful because Windows is inconsistent on the case of the drive letter, which can cause inconsistencies when calculating command signatures.

**get\_environment\_var**(*varstr*)

Given a string, first determine if it looks like a reference to a single environment variable, like "\$FOO" or "\${FOO}". If so, return that variable with no decorations ("FOO"). If not, return None.

**render\_tree**(*root*, *child\_func*, *prune*=0, *margin*=[0], *visited*={})

Render a tree of nodes into an ASCII tree view.

*root* - the root node of the tree

*child\_func* - the function called to get the children of a node

*prune* - don't visit the same node twice

*margin* - the format of the left margin to use for children of root.

1 results in a pipe, and 0 results in no pipe.

*visited* - a dictionary of visited nodes in the current branch if not *prune*, or in the whole tree if *prune*.

**IDX**(*N*)

**print\_tree**(*root*, *child\_func*, *prune*=0, *showtags*=0, *margin*=[0], *visited*={})

Print a tree of nodes. This is like `render_tree`, except it prints lines directly instead of creating a string representation in memory, so that huge trees can be printed.

*root* - the root node of the tree

*child\_func* - the function called to get the children of a node

*prune* - don't visit the same node twice

*showtags* - print status information to the left of each node line

*margin* - the format of the left margin to use for children of root.

1 results in a pipe, and 0 results in no pipe.

*visited* - a dictionary of visited nodes in the current branch if not *prune*, or in the whole tree if *prune*.

**is\_Dict**(*obj*, *isinstance*=<built-in function isinstance>, *DictTypes*=dict, UserDict)

**is\_List**(*obj*, *isinstance*=<built-in function isinstance>, *ListTypes*=(<type 'list'>, <class 'UserList.UserList'>))



```
is_Sequence(obj, isinstance=<built-in function isinstance>,
SequenceTypes=(<type 'list'>, <type 'tuple'>, <class
'UserList.UserList'>))
```

```
is_Tuple(obj, isinstance=<built-in function isinstance>, tuple=<type
'tuple'>)
```

```
is_String(obj, isinstance=<built-in function isinstance>,
StringTypes=(<type 'str'>, <type 'unicode'>, <class
'UserString.UserS...>))
```

```
is_Scalar(obj, isinstance=<built-in function isinstance>,
StringTypes=(<type 'str'>, <type 'unicode'>, <class
'UserString.UserS...>, SequenceTypes=(<type 'list'>, <type
'tuple'>, <class 'UserList.UserList'>))
```

```
do_flatten(sequence, result, isinstance=<built-in function isinstance>,
StringTypes=(<type 'str'>, <type 'unicode'>, <class
'UserString.UserS...>, SequenceTypes=(<type 'list'>, <type
'tuple'>, <class 'UserList.UserList'>))
```

```
flatten(obj, isinstance=<built-in function isinstance>,
StringTypes=(<type 'str'>, <type 'unicode'>, <class
'UserString.UserS...>, SequenceTypes=(<type 'list'>, <type
'tuple'>, <class 'UserList.UserList'>), do_flatten=<function
do_flatten at 0xb6b3848c>)
```

Flatten a sequence to a non-nested list.

Flatten() converts either a single scalar or a nested sequence to a non-nested list. Note that flatten() considers strings to be scalars instead of sequences like Python would.

```
flatten_sequence(sequence, isinstance=<built-in function isinstance>,
StringTypes=(<type 'str'>, <type 'unicode'>, <class
'UserString.UserString'>, SequenceTypes=(<type 'list'>, <type
'tuple'>, <class 'UserList.UserList'>), do_flatten=<function
do_flatten at 0xb6b3848c>)
```

Flatten a sequence to a non-nested list.

Same as `flatten()`, but it does not handle the single scalar case. This is slightly more efficient when one knows that the sequence to flatten can not be a scalar.

```
to_String(s, isinstance=<built-in function isinstance>, str=<type
'str'>, UserString=<class 'UserString.UserString'>,
BaseStringTypes=(<type 'str'>, <type 'unicode'>))
```

```
to_String_for_subst(s, isinstance=<built-in function isinstance>,
str=<type 'str'>, to_String=<function to_String at 0xb6b38534>,
BaseStringTypes=(<type 'str'>, <type 'unicode'>),
SequenceTypes=(<type 'list'>, <type 'tuple'>, <class
'UserList.UserList'>), UserString=<class 'UserString.UserString'>)
```

```
to_String_for_signature(obj, to_String_for_subst=<function
to_String_for_subst at 0xb6b3856c>, AttributeError=<type
'exceptions.AttributeError'>)
```

```
semi_deepcopy_dict(x, exclude=[])
```

```
semi_deepcopy(x)
```

```
RegGetValue(root, key)
```

```
RegOpenKeyEx(root, key)
```

```
WhereIs(file, path=None, pathext=None, reject=[])
```

---

**PrependPath**(*oldpath*, *newpath*, *sep*=':', *delete\_existing*=1, *canonicalize*=None)

---

This prepends *newpath* elements to the given *oldpath*. Will only add any particular path once (leaving the first one it encounters and ignoring the rest, to preserve path order), and will `os.path.normpath` and `os.path.normcase` all paths to help assure this. This can also handle the case where the given *oldpath* variable is a list instead of a string, in which case a list will be returned instead of a string.

**Example:** Old Path: `"/foo/bar:/foo"` New Path: `"/biz/boom:/foo"` Result: `"/biz/boom:/foo:/foo/bar"`

If *delete\_existing* is 0, then adding a path that exists will not move it to the beginning; it will stay where it is in the list.

If *canonicalize* is not None, it is applied to each element of *newpath* before use.

---

**AppendPath**(*oldpath*, *newpath*, *sep*=':', *delete\_existing*=1, *canonicalize*=None)

---

This appends new path elements to the given old path. Will only add any particular path once (leaving the last one it encounters and ignoring the rest, to preserve path order), and will `os.path.normpath` and `os.path.normcase` all paths to help assure this. This can also handle the case where the given *oldpath* variable is a list instead of a string, in which case a list will be returned instead of a string.

**Example:** Old Path: `"/foo/bar:/foo"` New Path: `"/biz/boom:/foo"` Result: `"/foo/bar:/biz/boom:/foo"`

If *delete\_existing* is 0, then adding a path that exists will not move it to the end; it will stay where it is in the list.

If *canonicalize* is not None, it is applied to each element of *newpath* before use.

---

**get\_native\_path**(*path*)

---

Transforms an absolute path into a native path for the system. Non-Cygwin version, just leave the path alone.

**Split**(*arg*)**case\_sensitive\_suffixes**(*s1*, *s2*)**adjustixes**(*fname*, *pre*, *suf*, *ensure\_suffix=False*)**unique**(*s*)

Return a list of the elements in *s*, but without duplicates.

For example, `unique([1,2,3,1,2,3])` is some permutation of `[1,2,3]`, `unique("abcabc")` some permutation of `["a", "b", "c"]`, and `unique([[1, 2], [2, 3], [1, 2]])` some permutation of `[[2, 3], [1, 2]]`.

For best speed, all sequence elements should be hashable. Then `unique()` will usually work in linear time.

If not possible, the sequence elements should enjoy a total ordering, and if `list(s).sort()` doesn't raise `TypeError` it's assumed that they do enjoy a total ordering. Then `unique()` will usually work in  $O(N \cdot \log_2(N))$  time.

If that's not possible either, the sequence elements must support equality-testing. Then `unique()` will usually work in quadratic time.

**uniquer**(*seq*, *idfun=None*)**uniquer\_\_hashables**(*seq*)**make\_path\_relative**(*path*)

makes an absolute path name to a relative pathname.

**AddMethod**(*obj, function, name=None*)

Adds either a bound method to an instance or an unbound method to a class. If name is omitted the name of the specified function is used by default.

Example:

```
a = A()
def f(self, x, y):
    self.z = x + y
AddMethod(f, A, "add")
a.add(2, 4)
print a.z
AddMethod(lambda self, i: self.l[i], a, "listIndex")
print a.listIndex(5)
```

**RenameFunction**(*function, name*)

Returns a function identical to the specified function, but with the specified name.

**MD5signature**(*s*)**MD5filesignature**(*fname, chunksize=65536*)**MD5collect**(*signatures*)

Collects a list of signatures into an aggregate signature.

signatures - a list of signatures returns - the aggregate signature

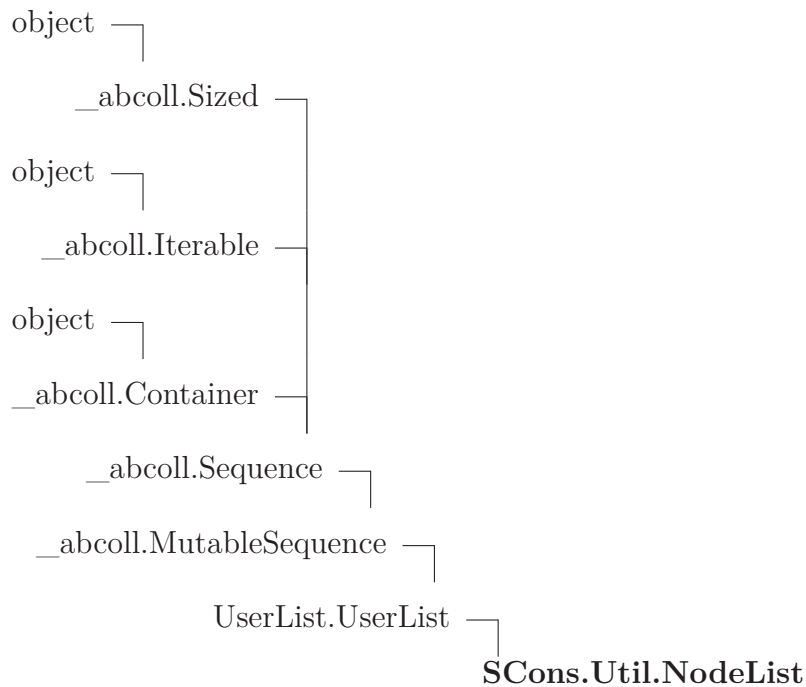
**silent\_intern**(*x*)

Perform sys.intern() on the passed argument and return the result. If the input is ineligible (e.g. a unicode string) the original argument is returned and no exception is thrown.

## 36.2 Variables

Name	Description
DictTypes	<b>Value:</b> dict, UserDict
ListTypes	<b>Value:</b> (<type 'list'>, <class 'UserList.UserList'>)
SequenceTypes	<b>Value:</b> (<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>)
StringTypes	<b>Value:</b> (<type 'str'>, <type 'unicode'>, <class 'UserString.UserS...>)
BaseStringTypes	<b>Value:</b> (<type 'str'>, <type 'unicode'>)
d	<b>Value:</b> {}
can_read_reg	<b>Value:</b> 0
hkey_mod	<b>Value:</b> win32con
RegEnumKey	<b>Value:</b> win32api.RegEnumKey
RegEnumValue	<b>Value:</b> win32api.RegEnumValue
RegQueryValueEx	<b>Value:</b> win32api.RegQueryValueEx
HKEY_CLASSES_ROOT	<b>Value:</b> None
HKEY_LOCAL_MACHINE	<b>Value:</b> None
HKEY_CURRENT_USER	<b>Value:</b> None
HKEY_USERS	<b>Value:</b> None
display	<b>Value:</b> DisplayEngine()
md5	<b>Value:</b> True
__package__	<b>Value:</b> 'SCons'

### 36.3 Class NodeList



This class is almost exactly like a regular list of Nodes (actually it can hold any object), with one important difference. If you try to get an attribute from this list, it will return that attribute from every item in the list. For example:

```
>>> someList = NodeList([ 'foo', 'bar' ])
>>> someList.strip()
[ 'foo', 'bar' ]
```

#### 36.3.1 Methods

<code>__nonzero__(self)</code>
--------------------------------

<code>__str__(self)</code> <code>str(x)</code> Overrides: <code>object.__str__</code> <code>exitit</code> (inherited documentation)
--

<code>__iter__(self)</code> Overrides: <code>_abcoll.Iterable.__iter__</code>
--

<code>__call__(self, *args, **kwargs)</code>
--

<code>__getattr__(self, name)</code>
--------------------------------------

**Inherited from *UserList.UserList***

`__add__()`, `__cmp__()`, `__contains__()`, `__delitem__()`, `__delslice__()`,  
`__eq__()`, `__ge__()`, `__getitem__()`, `__getslice__()`, `__gt__()`, `__iadd__()`,  
`__imul__()`, `__init__()`, `__le__()`, `__len__()`, `__lt__()`, `__mul__()`, `__ne__()`,  
`__radd__()`, `__repr__()`, `__rmul__()`, `__setitem__()`, `__setslice__()`, `ap-`  
`pend()`, `count()`, `extend()`, `index()`, `insert()`, `pop()`, `remove()`, `reverse()`, `sort()`

**Inherited from *\_\_abcoll.Sequence***`__reversed__()`**Inherited from *\_\_abcoll.Sized***`__subclasshook__()`**Inherited from *object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`,  
`__reduce_ex__()`, `__setattr__()`, `__sizeof__()`

**36.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**36.3.3 Class Variables**

Name	Description
<i>Inherited from UserList.UserList</i>	
<code>__abstractmethods__</code> , <code>__hash__</code>	

**36.4 Class DisplayEngine**

```

object └─
          SCons.Util.DisplayEngine

```



### 36.4.1 Methods

```
__call__(self, text, append_newline=1)
```

```
set_mode(self, mode)
```

#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __init__(),
__new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(),
__sizeof__(), __str__(), __subclasshook__()
```

### 36.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 36.4.3 Class Variables

Name	Description
<code>print_it</code>	<b>Value:</b> True

## 36.5 Class Proxy

```

object ┌
      │ SCons.Util.Proxy

```

**Known Subclasses:** SCons.Builder.CompositeBuilder, SCons.Node.FS.EntryProxy

A simple generic Proxy class, forwarding all calls to subject. So, for the benefit of the python newbie, what does this really mean? Well, it means that you can take an object, let's call it 'objA', and wrap it in this Proxy class, with a statement like this

```
proxyObj = Proxy(objA),
```

Then, if in the future, you do something like this

```
x = proxyObj.var1,
```

since Proxy does not have a 'var1' attribute (but presumably objA does), the request actually is equivalent to saying

```
x = objA.var1
```

Inherit from this class to create a Proxy.

Note that, with new-style classes, this does *not* work transparently for Proxy subclasses that use special `__*__()` method names, because those names are now bound to the class, not the individual instances. You now need to know in advance which `__*__()` method names you want to pass on to the underlying Proxy object, and specifically delegate their calls like this:

```
class Foo(Proxy): __str__ = Delegate('__str__')
```

### 36.5.1 Methods

<code>__init__(self, subject)</code>
Wrap an object as a Proxy object Overrides: object.__init__

<code>__getattr__(self, name)</code>
Retrieve an attribute from the wrapped object. If the named attribute doesn't exist, AttributeError is raised

<code>get(self)</code>
Retrieve the entire wrapped object

<code>__cmp__(self, other)</code>
-----------------------------------

#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 36.5.2 Properties

Name	Description
<code>__class__</code>	<i>Inherited from object</i>

### 36.6 Class Delegate



A Python Descriptor class that delegates attribute fetches to an underlying wrapped subject of a Proxy. Typical use:

```
class Foo(Proxy): __str__ = Delegate('__str__')
```

#### 36.6.1 Methods

```
__init__(self, attribute)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature Overrides:  
`object.__init__` extit(inherited documentation)

```
__get__(self, obj, cls)
```

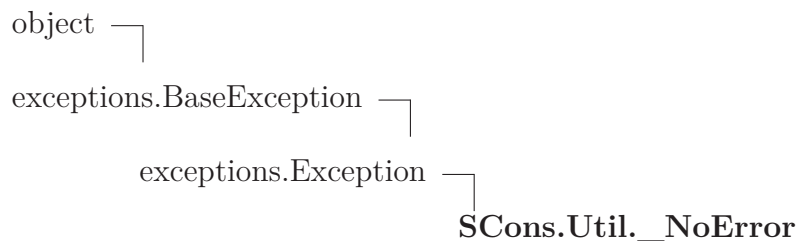
#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

#### 36.6.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 36.7 Class `__NoError`



### 36.7.1 Methods

*Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

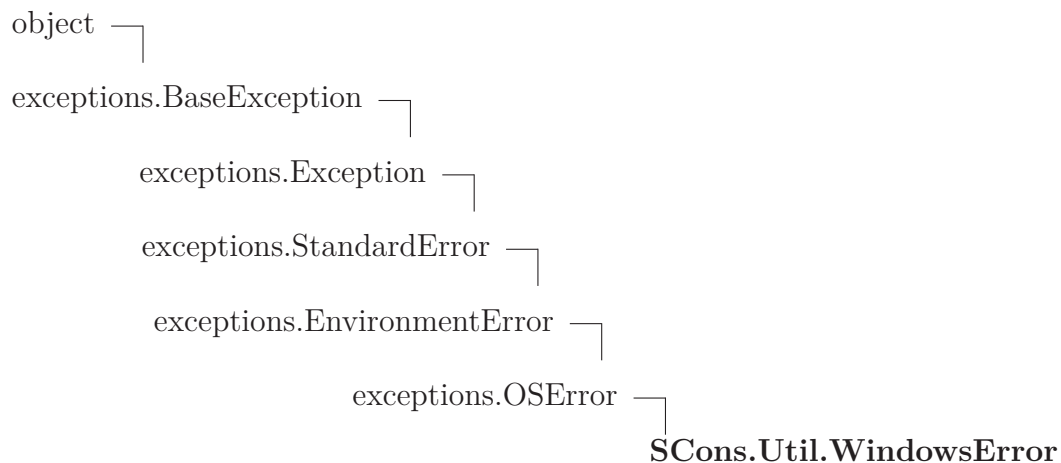
*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 36.7.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 36.8 Class `WindowsError`



### 36.8.1 Methods

*Inherited from `exceptions.OSError`*

`__init__()`, `__new__()`

*Inherited from exceptions.EnvironmentError*

`__reduce__()`, `__str__()`

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

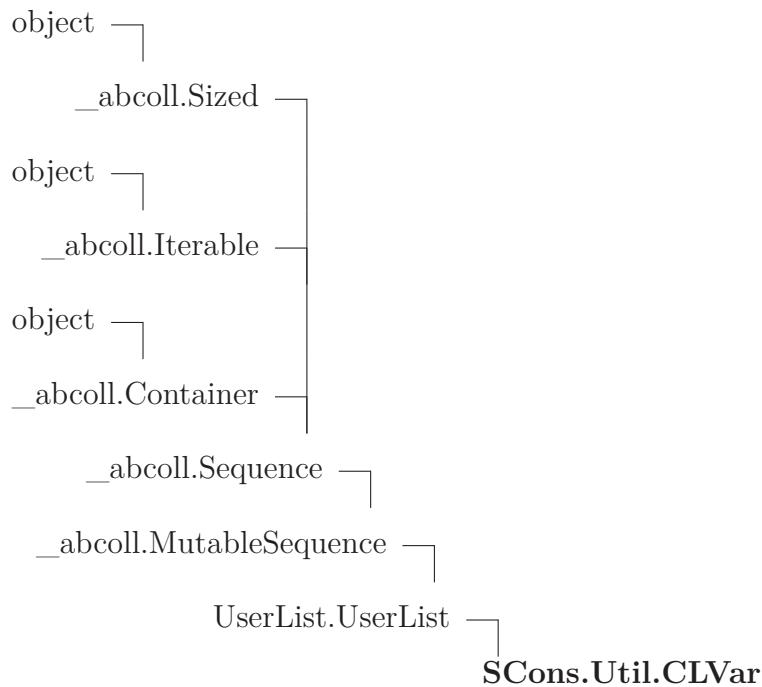
*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 36.8.2 Properties

Name	Description
<i>Inherited from exceptions.EnvironmentError</i> errno, filename, strerror	
<i>Inherited from exceptions.BaseException</i> args, message	
<i>Inherited from object</i> __class__	

### 36.9 Class CLVar



A class for command-line construction variables.

This is a list that uses Split() to split an initial string along white-space arguments, and similarly to split any strings that get added. This allows us to Do the Right Thing with Append() and Prepend() (as well as straight Python `foo = env['VAR'] + 'arg1 arg2'`) regardless of whether a user adds a list or a string to a command-line construction variable.

#### 36.9.1 Methods

**\_\_init\_\_**(*self*, *seq*=[])

*x*.\_\_init\_\_(...) initializes *x*; see help(type(*x*)) for signature Overrides: object.\_\_init\_\_ extit(inherited documentation)

**\_\_add\_\_**(*self*, *other*)

Overrides: UserList.UserList.\_\_add\_\_

**\_\_radd\_\_**(*self*, *other*)

Overrides: UserList.UserList.\_\_radd\_\_

**\_\_coerce\_\_**(*self*, *other*)

```
__str__(self)
```

```
str(x) Overrides: object.__str__ extit(inherited documentation)
```

### ***Inherited from UserList.UserList***

```
__cmp__(), __contains__(), __delitem__(), __delslice__(), __eq__(), __ge__(),
__getitem__(), __getslice__(), __gt__(), __iadd__(), __imul__(), __le__(),
__len__(), __lt__(), __mul__(), __ne__(), __repr__(), __rmul__(), __setitem__(),
__setslice__(), append(), count(), extend(), index(), insert(), pop(), remove(),
reverse(), sort()
```

### ***Inherited from \_\_abcoll.Sequence***

```
__iter__(), __reversed__()
```

### ***Inherited from \_\_abcoll.Sized***

```
__subclasshook__()
```

### ***Inherited from object***

```
__delattr__(), __format__(), __getattr__(), __new__(), __reduce__(),
__reduce_ex__(), __setattr__(), __sizeof__()
```

## **36.9.2 Properties**

Name	Description
<i>Inherited from object</i> __class__	

## **36.9.3 Class Variables**

Name	Description
<i>Inherited from UserList.UserList</i> __abstractmethods__, __hash__	

## **36.10 Class OrderedDict**

```
UserDict.UserDict └─ SCons.Util.OrderedDict
```

**Known Subclasses:** SCons.Util.Selector

**36.10.1 Methods****\_\_init\_\_**(*self*, *dict*=None)

Overrides: UserDict.UserDict.\_\_init\_\_

**\_\_delitem\_\_**(*self*, *key*)

Overrides: UserDict.UserDict.\_\_delitem\_\_

**\_\_setitem\_\_**(*self*, *key*, *item*)

Overrides: UserDict.UserDict.\_\_setitem\_\_

**clear**(*self*)

Overrides: UserDict.UserDict.clear

**copy**(*self*)

Overrides: UserDict.UserDict.copy

**items**(*self*)

Overrides: UserDict.UserDict.items

**keys**(*self*)

Overrides: UserDict.UserDict.keys

**popitem**(*self*)

Overrides: UserDict.UserDict.popitem

**setdefault**(*self*, *key*, *failobj*=None)

Overrides: UserDict.UserDict.setdefault

**update**(*self*, *dict*)

Overrides: UserDict.UserDict.update

**values**(*self*)

Overrides: UserDict.UserDict.values

***Inherited from UserDict.UserDict*****\_\_cmp\_\_**(*self*), **\_\_contains\_\_**(*self*, *key*), **\_\_getitem\_\_**(*self*, *key*), **\_\_len\_\_**(*self*), **\_\_repr\_\_**(*self*), **fromkeys**(*cls*, *keys*), **get**(*self*, *key*, *default*), **has\_key**(*self*, *key*), **iteritems**(*self*), **iterkeys**(*self*), **itervalues**(*self*), **pop**(*self*, *key*)



### 36.10.2 Class Variables

Name	Description
<i>Inherited from UserDict.UserDict</i>	
__hash__	

## 36.11 Class Selector



**Known Subclasses:** SCons.Builder.CallableSelector, SCons.Builder.DictCmdGenerator, SCons.Builder.DictEmitter

A callable ordered dictionary that maps file suffixes to dictionary values. We preserve the order in which items are added so that `get_suffix()` calls always return the first suffix added.

### 36.11.1 Methods

<code>__call__(self, env, source, ext=None)</code>
--

*Inherited from SCons.Util.OrderedDict(Section 36.10)*

`__delitem__()`, `__init__()`, `__setitem__()`, `clear()`, `copy()`, `items()`, `keys()`, `popitem()`, `setdefault()`, `update()`, `values()`

*Inherited from UserDict.UserDict*

`__cmp__()`, `__contains__()`, `__getitem__()`, `__len__()`, `__repr__()`, `fromkeys()`, `get()`, `has_key()`, `iteritems()`, `iterkeys()`, `itervalues()`, `pop()`

### 36.11.2 Class Variables

Name	Description
<i>Inherited from UserDict.UserDict</i>	
__hash__	

## 36.12 Class *LogicalLines*



### 36.12.1 Methods

<code>__init__(self, fileobj)</code>
--------------------------------------

<code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> extit(inherited documentation)
--

<code>readline(self)</code>
-----------------------------

<code>readlines(self)</code>
------------------------------

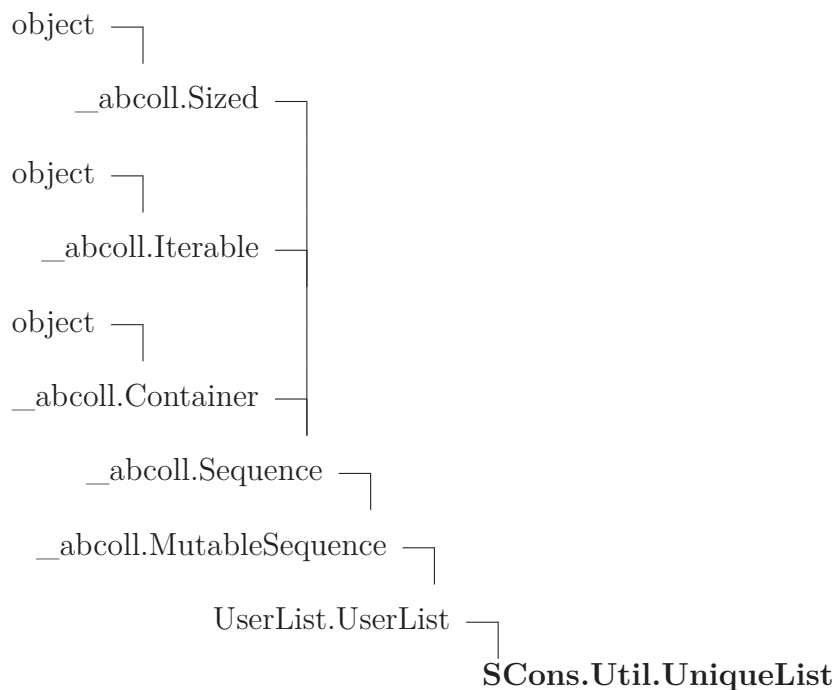
### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

### 36.12.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 36.13 Class **UniqueList**



#### 36.13.1 Methods

**\_\_init\_\_**(*self*, *seq*=[])

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature Overrides: `object.__init__` `exitit`(inherited documentation)

**\_\_lt\_\_**(*self*, *other*)

Overrides: `UserList.UserList.__lt__`

**\_\_le\_\_**(*self*, *other*)

Overrides: `UserList.UserList.__le__`

**\_\_eq\_\_**(*self*, *other*)

Overrides: `UserList.UserList.__eq__`

**\_\_ne\_\_**(*self*, *other*)

Overrides: `UserList.UserList.__ne__`

**\_\_gt\_\_**(*self*, *other*)

Overrides: UserList.UserList.\_\_gt\_\_

**\_\_ge\_\_**(*self*, *other*)

Overrides: UserList.UserList.\_\_ge\_\_

**\_\_cmp\_\_**(*self*, *other*)

Overrides: UserList.UserList.\_\_cmp\_\_

**\_\_len\_\_**(*self*)

Overrides: \_abcoll.Sized.\_\_len\_\_

**\_\_getitem\_\_**(*self*, *i*)

Overrides: \_abcoll.Sequence.\_\_getitem\_\_

**\_\_setitem\_\_**(*self*, *i*, *item*)

Overrides: \_abcoll.MutableSequence.\_\_setitem\_\_

**\_\_getslice\_\_**(*self*, *i*, *j*)

Overrides: UserList.UserList.\_\_getslice\_\_

**\_\_setslice\_\_**(*self*, *i*, *j*, *other*)

Overrides: UserList.UserList.\_\_setslice\_\_

**\_\_add\_\_**(*self*, *other*)

Overrides: UserList.UserList.\_\_add\_\_

**\_\_radd\_\_**(*self*, *other*)

Overrides: UserList.UserList.\_\_radd\_\_

**\_\_iadd\_\_**(*self*, *other*)

Overrides: \_abcoll.MutableSequence.\_\_iadd\_\_

**\_\_mul\_\_**(*self*, *other*)

Overrides: UserList.UserList.\_\_mul\_\_

**\_\_rmul\_\_**(*self*, *other*)

Overrides: `UserList.UserList.__rmul__`

**\_\_imul\_\_**(*self*, *other*)

Overrides: `UserList.UserList.__imul__`

**append**(*self*, *item*)

append object to the end of the sequence Overrides:  
`__abcoll.MutableSequence.append` extit(inherited documentation)

**insert**(*self*, *i*)

insert object before index Overrides: `__abcoll.MutableSequence.insert`  
 extit(inherited documentation)

**count**(*self*, *item*)

return number of occurrences of value **Return Value**  
 integer

Overrides: `__abcoll.Sequence.count` extit(inherited documentation)

**index**(*self*, *item*)

return first index of value. Raises `ValueError` if the value is not present.

**Return Value**

integer

Overrides: `__abcoll.Sequence.index` extit(inherited documentation)

**reverse**(*self*)

reverse *IN PLACE* Overrides: `__abcoll.MutableSequence.reverse`  
 extit(inherited documentation)

**sort**(*self*, *\*args*, *\*\*kws*)

Overrides: `UserList.UserList.sort`

**extend**(*self*, *other*)

extend sequence by appending elements from the iterable Overrides:  
`__abcoll.MutableSequence.extend` extit(inherited documentation)

***Inherited from UserList.UserList***

`__contains__()`, `__delitem__()`, `__delslice__()`, `__repr__()`, `pop()`, `remove()`

***Inherited from \_\_abcoll.Sequence***

`__iter__()`, `__reversed__()`

***Inherited from \_\_abcoll.Sized***

`__subclasshook__()`

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`,  
`__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__str__()`

**36.13.2 Properties**

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

**36.13.3 Class Variables**

Name	Description
<i>Inherited from UserList.UserList</i> <code>__abstractmethods__</code> , <code>__hash__</code>	

**36.14 Class Unbuffered**

object —  
**SCons.Util.Unbuffered**

A proxy class that wraps a file object, flushing after every write, and delegating everything else to the wrapped object.

**36.14.1 Methods**

`__init__(self, file)`

x.`__init__`(...) initializes x; see `help(type(x))` for signature Overrides:  
object.`__init__` extit(inherited documentation)

`write(self, arg)`

`__getattr__(self, attr)`

***Inherited from object***

`__delattr__`(), `__format__`(), `__getattr__`(), `__hash__`(), `__new__`(),  
`__reduce__`(), `__reduce_ex__`(), `__repr__`(), `__setattr__`(), `__sizeof__`(),  
`__str__`(), `__subclasshook__`()

**36.14.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**36.15 Class Null**

object —  
    **SCons.Util.Null**

**Known Subclasses:** SCons.Executor.NullEnvironment, SCons.Sig.MD5Null, SCons.Sig.TimeStampNull, SCons.Util.NullSeq

Null objects always and reliably "do nothing."

**36.15.1 Methods**

`__new__(cls, *args, **kwargs)`

**Return Value**

a new object with type S, a subtype of T

Overrides: object.`__new__` extit(inherited documentation)

```
__init__(self, *args, **kwargs)
```

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides:  
object.**\_\_init\_\_** extit(inherited documentation)

```
__call__(self, *args, **kwargs)
```

```
__repr__(self)
```

repr(x) Overrides: object.**\_\_repr\_\_** extit(inherited documentation)

```
__nonzero__(self)
```

```
__getattr__(self, name)
```

```
__setattr__(self, name, value)
```

x.**\_\_setattr\_\_**('name', value) <==> x.name = value Overrides:  
object.**\_\_setattr\_\_** extit(inherited documentation)

```
__delattr__(self, name)
```

x.**\_\_delattr\_\_**('name') <==> del x.name Overrides: object.**\_\_delattr\_\_**  
extit(inherited documentation)

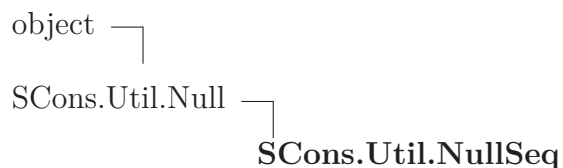
### *Inherited from object*

```
__format__(), __getattribute__(), __hash__(), __reduce__(), __reduce_ex__(),  
__sizeof__(), __str__(), __subclasshook__()
```

### 36.15.2 Properties

Name	Description
<i>Inherited from object</i> <b>__class__</b>	



**36.16 Class *NullSeq*****Known Subclasses:** *SCons.Subst.NullNodeList***36.16.1 Methods**`__len__(self)``__iter__(self)``__getitem__(self, i)``__delitem__(self, i)``__setitem__(self, i, v)`***Inherited from *SCons.Util.Null*(Section 36.15)***

`__call__()`, `__delattr__()`, `__getattr__()`, `__init__()`, `__new__()`, `__nonzero__()`,  
`__repr__()`, `__setattr__()`

***Inherited from *object****

`__format__()`, `__getattribute__()`, `__hash__()`, `__reduce__()`, `__reduce_ex__()`,  
`__sizeof__()`, `__str__()`, `__subclasshook__()`

**36.16.2 Properties**

Name	Description
<i>Inherited from <i>object</i></i>	
<code>__class__</code>	

## 37 Package SCons.Variables

engine.SCons.Variables

This file defines the Variables class that is used to add user-friendly customizable variables to an SCons build.

### 37.1 Modules

- **BoolVariable** (*Section ??, p. ??*)
- **BoolVariable'**: engine.SCons.Variables.BoolVariable  
(*Section 38, p. 343*)
- **EnumVariable** (*Section ??, p. ??*)
- **EnumVariable'**: engine.SCons.Variables.EnumVariable  
(*Section 39, p. 344*)
- **ListVariable** (*Section ??, p. ??*)
- **ListVariable'**: engine.SCons.Variables.ListVariable  
(*Section 40, p. 346*)
- **PackageVariable** (*Section ??, p. ??*)
- **PackageVariable'**: engine.SCons.Variables.PackageVariable  
(*Section 41, p. 347*)
- **PathVariable** (*Section ??, p. ??*)
- **PathVariable'**: SCons.Variables.PathVariable  
(*Section 42, p. 348*)

### 37.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Variables/__init__.py rel_2.4.1:3453:73...
__package__	<b>Value:</b> 'SCons.Variables'

### 37.3 Class Variables

object —  
SCons.Variables.Variables

## 37.3.1 Methods

---

```
__init__(self, files=[], args={}, is_global=1)
```

---

**files** - [optional] List of option configuration files to load

(backward compatibility) If a single string is passed it is automatically placed in a file list

Overrides: object.\_\_init\_\_

---

```
keys(self)
```

---

Returns the keywords for the options

---

```
Add(self, key, help='', default=None, validator=None, converter=None,
**kw)
```

---

Add an option.

**key** - the name of the variable, or a list or tuple of arguments

**help** - optional help text for the options

**default** - optional default value

**validator** - optional function that is called to validate the option's value  
 Called with (key, value, environment)

**converter** - optional function that is called to convert the option's value before  
 putting it in the environment.

---

**AddVariables**(*self*, \**optlist*)

Add a list of options.

Each list element is a tuple/list of arguments to be passed on to the underlying method for adding options.

Example:

```
opt.AddVariables(  
    ('debug', '', 0),  
    ('CC', 'The C compiler'),  
    ('VALIDATE', 'An option for testing validation', 'notset',  
     validator, None),  
)
```

**Update**(*self*, *env*, *args*=None)

Update an environment with the option variables.

*env* - the environment to update.

**UnknownVariables**(*self*)

Returns any options in the specified arguments lists that were not known, declared options in this object.

**Save**(*self*, *filename*, *env*)

Saves all the options in the given file. This file can then be used to load the options next run. This can be used to create an option cache file.

*filename* - Name of the file to save into *env* - the environment get the option values from

<b>GenerateHelpText</b> ( <i>self</i> , <i>env</i> , <i>sort</i> =None)
---

Generate the help text for the options.
---

<b>env</b> - an environment that is used to get the current values of the options.
--

<b>FormatVariableHelpText</b> ( <i>self</i> , <i>env</i> , <i>key</i> , <i>help</i> , <i>default</i> , <i>actual</i> , <i>aliases</i> =[])
--

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

### 37.3.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

### 37.3.3 Class Variables

Name	Description
instance	Holds all the options, updates the environment with the variables, and renders the help text. <b>Value:</b> None
format	<b>Value:</b> '\n%s: %s\n default: %s\n actual: %s\n'
format__	<b>Value:</b> '\n%s: %s\n default: %s\n actual: %s\n aliases: ...'

## 38 Module *SCons.Variables.BoolVariable*

`engine.SCons.Variables.BoolVariable`

This file defines the option type for SCons implementing true/false values.

Usage example:

```
opts = Variables()
opts.Add(BoolVariable('embedded', 'build for an embedded system', 0))
...
if env['embedded'] == 1:
    ...
```

### 38.1 Functions

<b><code>BoolVariable</code></b> <i>(key, help, default)</i>
The input parameters describe a boolean option, thus they are returned with the correct converter and validator appended. The 'help' text will be appended by '(yes no)' to show the valid values. The result is usable for input to <code>opts.Add()</code> .

## 39 Module SCons.Variables.EnumVariable'

`engine.SCons.Variables.EnumVariable`

This file defines the option type for SCons allowing only specified input-values.

Usage example:

```
opts = Variables()
opts.Add(EnumVariable('debug', 'debug output and symbols', 'no',
                      allowed_values=('yes', 'no', 'full'),
                      map={}, ignorecase=2))
...
if env['debug'] == 'full':
    ...
```

### 39.1 Functions

**EnumVariable**(*key*, *help*, *default*, *allowed\_values*, *map*={}, *ignorecase*=0)

The input parameters describe a option with only certain values allowed. They are returned with an appropriate converter and validator appended. The result is usable for input to `Variables.Add()`.

'key' and 'default' are the values to be passed on to `Variables.Add()`.

'help' will be appended by the allowed values automatically

'allowed\_values' is a list of strings, which are allowed as values for this option.

The 'map'-dictionary may be used for converting the input value into canonical values (eg. for aliases).

'ignorecase' defines the behaviour of the validator:

If `ignorecase == 0`, the validator/converter are case-sensitive.  
If `ignorecase == 1`, the validator/converter are case-insensitive.  
If `ignorecase == 2`, the validator/converter is case-insensitive and the converted value will always be lower-case.

The 'validator' tests whether the value is in the list of allowed values. The 'converter' converts input values according to the given 'map'-dictionary (unmapped input values are returned unchanged).



## 40 Module *SCons.Variables.ListVariable*

`engine.SCons.Variables.ListVariable`

This file defines the option type for SCons implementing 'lists'.

A 'list' option may either be 'all', 'none' or a list of names separated by comma. After the option has been processed, the option value holds either the named list elements, all list elements or no list elements at all.

Usage example:

```
list_of_libs = Split('x11 gl qt ical')

opts = Variables()
opts.Add(ListVariable('shared',
                      'libraries to build as shared libraries',
                      'all',
                      elems = list_of_libs))

...
for lib in list_of_libs:
    if lib in env['shared']:
        env.SharedObject(...)
    else:
        env.Object(...)
```

### 40.1 Functions

<b>ListVariable</b> ( <i>key, help, default, names, map={}</i> )
--

The input parameters describe a 'package list' option, thus they are returned with the correct converter and validator appended. The result is usable for input to `opts.Add()` .

A 'package list' option may either be 'all', 'none' or a list of package names (separated by space).

## 41 Module *SCons.Variables.PackageVariable*

`engine.SCons.Variables.PackageVariable`

This file defines the option type for SCons implementing 'package activation'.

To be used whenever a 'package' may be enabled/disabled and the package path may be specified.

Usage example:

Examples:

```
x11=no      (disables X11 support)
x11=yes     (will search for the package installation dir)
x11=/usr/local/X11 (will check this path for existence)
```

To replace autoconf's `--with-xxx=yyy`

```
opts = Variables()
opts.Add(PackageVariable('x11',
                        'use X11 installed here (yes = search some places',
                        'yes'))
...
if env['x11'] == True:
    dir = ... search X11 in some standard places ...
    env['x11'] = dir
if env['x11']:
    ... build with x11 ...
```

### 41.1 Functions

<b><code>PackageVariable(key, help, default, searchfunc=None)</code></b>
<p>The input parameters describe a 'package list' option, thus they are returned with the correct converter and validator appended. The result is usable for input to <code>opts.Add()</code> .</p> <p>A 'package list' option may either be 'all', 'none' or a list of package names (seperated by space).</p>

## 42 Module *SCons.Variables.PathVariable*

*SCons.Variables.PathVariable*

This file defines an option type for SCons implementing path settings.

To be used whenever a user-specified path override should be allowed.

Arguments to *PathVariable* are:

```
option-name  = name of this option on the command line (e.g. "prefix")
option-help  = help string for option
option-dflt  = default value for this option
validator    = [optional] validator for option value.  Predefined
               validators are:
```

```
PathAccept  -- accepts any path setting; no validation
PathIsDir   -- path must be an existing directory
PathIsDirCreate -- path must be a dir; will create
PathIsFile  -- path must be a file
PathExists  -- path must exist (any type) [default]
```

The validator is a function that is called and which should return `True` or `False` to indicate if the path is valid. The arguments to the validator function are: (key, val, env). The key is the name of the option, the val is the path specified for the option, and the env is the env to which the Options have been added.

Usage example:

Examples:

```
prefix=/usr/local
```

```
opts = Variables()
```

```
opts = Variables()
```

```
opts.Add(PathVariable('qtdir',
                      'where the root of Qt is installed',
                      qtdir, PathIsDir))
opts.Add(PathVariable('qt_includes',
                      'where the Qt includes are installed',
                      '$qtdir/includes', PathIsDirCreate))
opts.Add(PathVariable('qt_libraries',
```

```
'where the Qt library is installed',  
'$qtdir/lib'))
```

## 42.1 Variables

Name	Description
PathVariable	<b>Value:</b> <code>SCons.Variables.PathVariable</code>

## 43 Module **SCons.Warnings**

SCons.Warnings

This file implements the warnings framework for SCons.

### 43.1 Functions

<b>suppressWarningClass</b> ( <i>clazz</i> )
Suppresses all warnings that are of type <i>clazz</i> or derived from <i>clazz</i> .

<b>enableWarningClass</b> ( <i>clazz</i> )
Enables all warnings that are of type <i>clazz</i> or derived from <i>clazz</i> .

<b>warningAsException</b> ( <i>flag</i> =1)
Turn warnings into exceptions. Returns the old value of the flag.

<b>warn</b> ( <i>clazz</i> , * <i>args</i> )
--

**process\_warn\_strings**(*arguments*)

Process string specifications of enabling/disabling warnings, as passed to the --warn option or the SetOption('warn') function.

An argument to this option should be of the form <warning-class> or no-<warning-class>. The warning class is munged in order to get an actual class name from the classes above, which we need to pass to the {enable,disable}WarningClass() functions. The supplied <warning-class> is split on hyphens, each element is capitalized, then smushed back together. Then the string "Warning" is appended to get the class name.

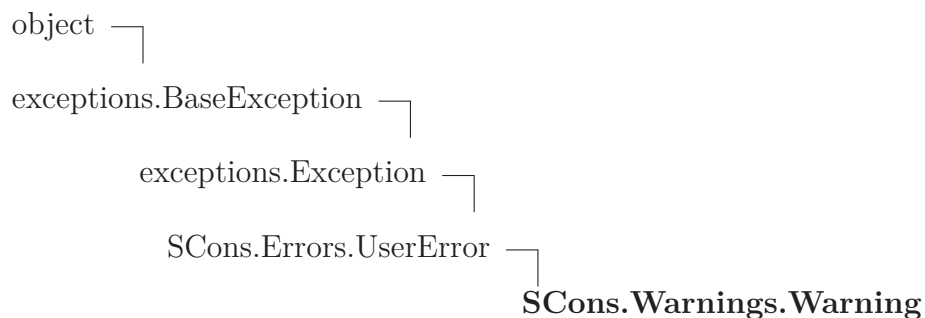
For example, 'deprecated' will enable the DeprecatedWarning class. 'no-dependency' will disable the DependencyWarning class.

As a special case, --warn=all and --warn=no-all will enable or disable (respectively) the base Warning class of all warnings.

## 43.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/Warnings.py rel_2.4.1:3453:73fef3ea0b0...
__package__	<b>Value:</b> 'SCons'

## 43.3 Class Warning



**Known Subclasses:** SCons.SConf.SConfWarning, SCons.Warnings.CacheWriteErrorWarning, SCons.Warnings.WarningOnByDefault, SCons.Warnings.DependencyWarning, SCons.Warnings.DeprecatedWarning, SCons.Warnings.FutureDeprecatedWarning, SCons.Warnings.TargetNotBuiltWarning, SCons.Warnings.VI

### 43.3.1 Methods

#### *Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

#### *Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

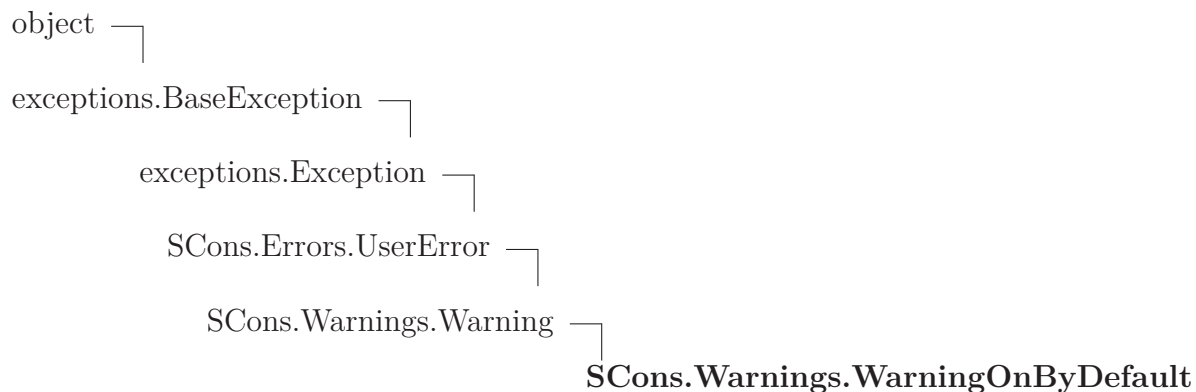
#### *Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 43.4 Class `WarningOnByDefault`



**Known Subclasses:** `SCons.Warnings.CorruptSConsignWarning`, `SCons.Warnings.DevelopmentVersionWarning`, `SCons.Warnings.DuplicateEnvironmentWarning`, `SCons.Warnings.LinkWarning`, `SCons.Warnings.FutureReleaseWarning`, `SCons.Warnings.MisleadingKeywordsWarning`, `SCons.Warnings.MissingSConscriptWarning`, `SCons.Warnings.NoMD5ModuleWarning`, `SCons.Warnings.NoMetaclassSupportWarning`, `SCons.Warnings.NoParallelSupportWarning`, `SCons.Warnings.ReservedVariableWarning`, `SCons.Warnings.UnexpectedWarning`, `SCons.Warnings.VisualCMissingWarning`, `SCons.Warnings.VisualVersionMismatchWarning`

#### 43.4.1 Methods

*Inherited from exceptions.Exception*

`__init__()`, `__new__()`

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

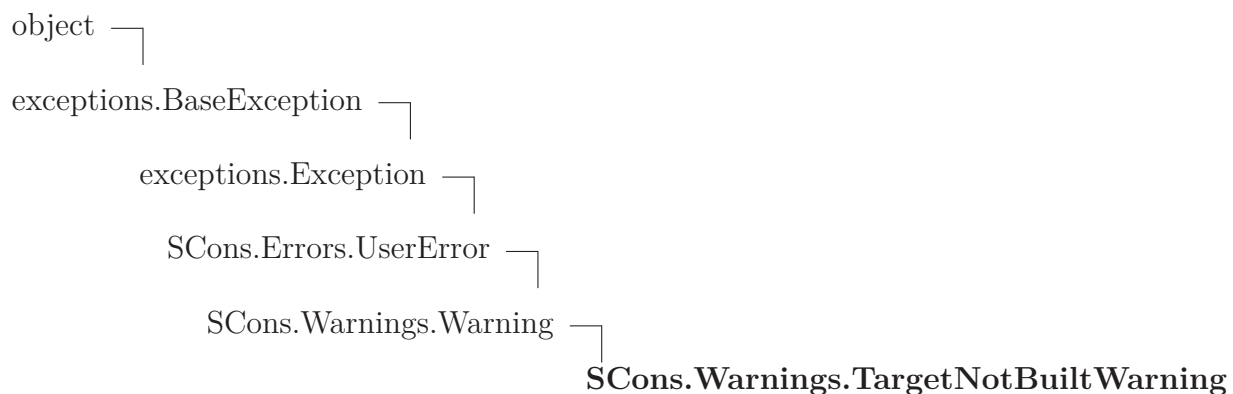
*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 43.4.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

### 43.5 Class TargetNotBuiltWarning



#### 43.5.1 Methods

*Inherited from exceptions.Exception*

`__init__()`, `__new__()`



***Inherited from `exceptions.BaseException`***

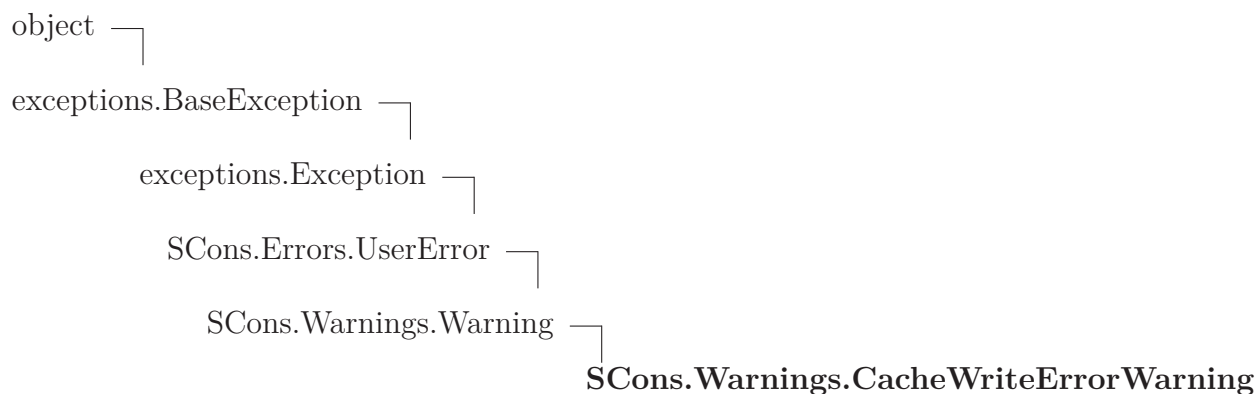
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**43.5.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

**43.6 Class `CacheWriteErrorWarning`****43.6.1 Methods*****Inherited from `exceptions.Exception`***

`__init__()`, `__new__()`

***Inherited from `exceptions.BaseException`***

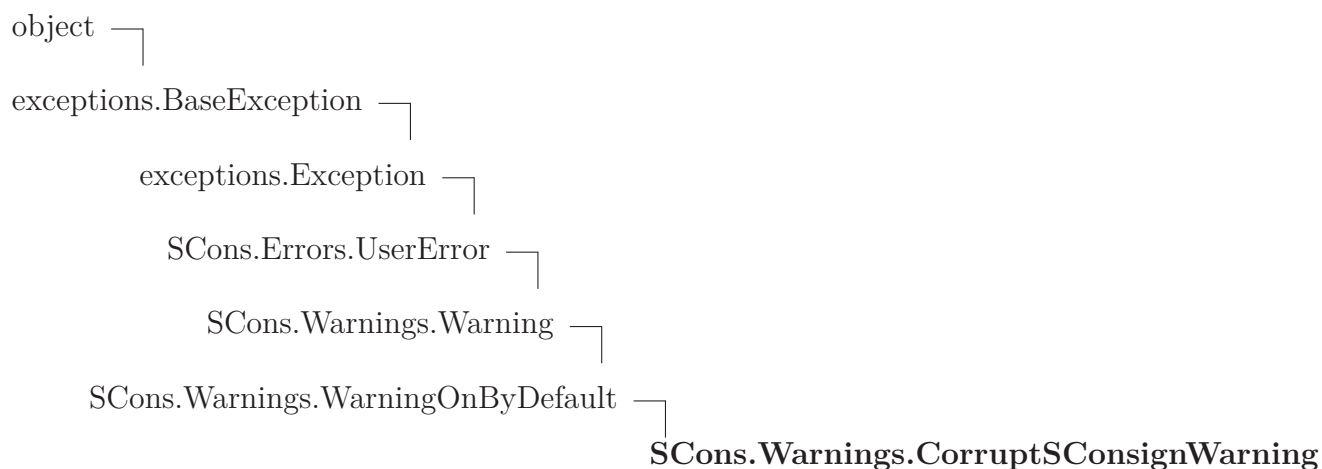
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**43.6.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**43.7 Class `CorruptSConsignWarning`****43.7.1 Methods*****Inherited from `exceptions.Exception`***

`__init__()`, `__new__()`

***Inherited from `exceptions.BaseException`***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

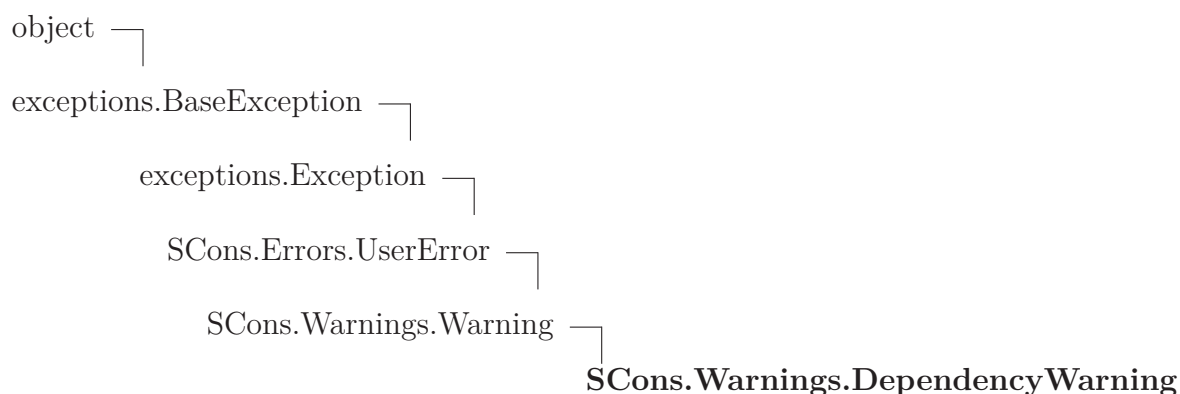
***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.7.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

## 43.8 Class DependencyWarning



### 43.8.1 Methods

#### *Inherited from exceptions.Exception*

\_\_init\_\_(), \_\_new\_\_()

#### *Inherited from exceptions.BaseException*

\_\_delattr\_\_(), \_\_getattr\_\_(), \_\_getitem\_\_(), \_\_getslice\_\_(), \_\_reduce\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_setstate\_\_(), \_\_str\_\_(), \_\_unicode\_\_()

#### *Inherited from object*

\_\_format\_\_(), \_\_hash\_\_(), \_\_reduce\_ex\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

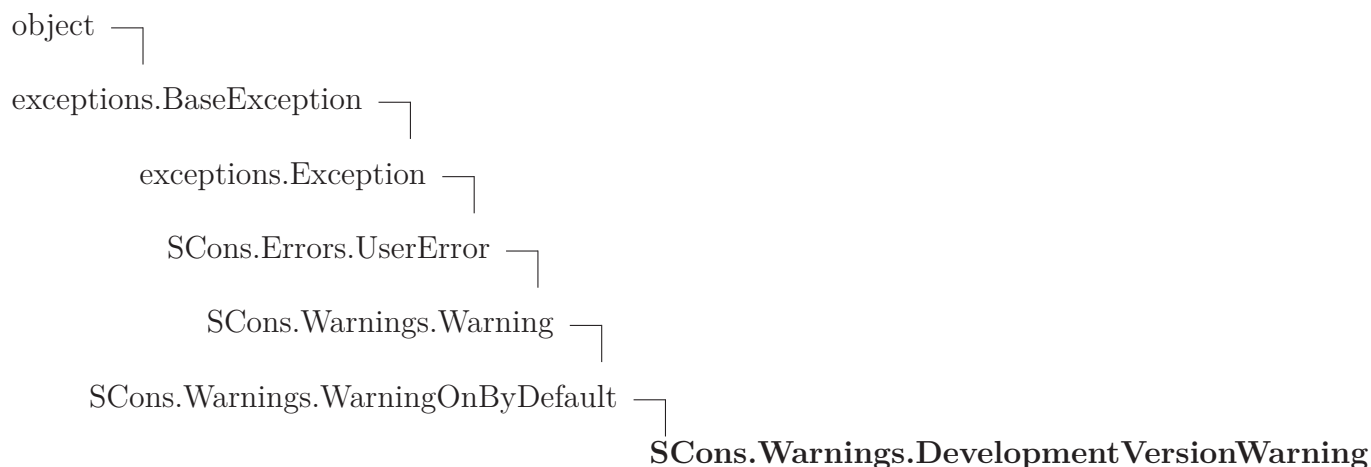
### 43.8.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	

*continued on next page*

Name	Description
args, message	
<i>Inherited from object</i>	
__class__	

## 43.9 Class DevelopmentVersionWarning



### 43.9.1 Methods

#### *Inherited from exceptions.Exception*

\_\_init\_\_(), \_\_new\_\_()

#### *Inherited from exceptions.BaseException*

\_\_delattr\_\_(), \_\_getattr\_\_(), \_\_getitem\_\_(), \_\_getslice\_\_(), \_\_reduce\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_setstate\_\_(), \_\_str\_\_(), \_\_unicode\_\_()

#### *Inherited from object*

\_\_format\_\_(), \_\_hash\_\_(), \_\_reduce\_ex\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

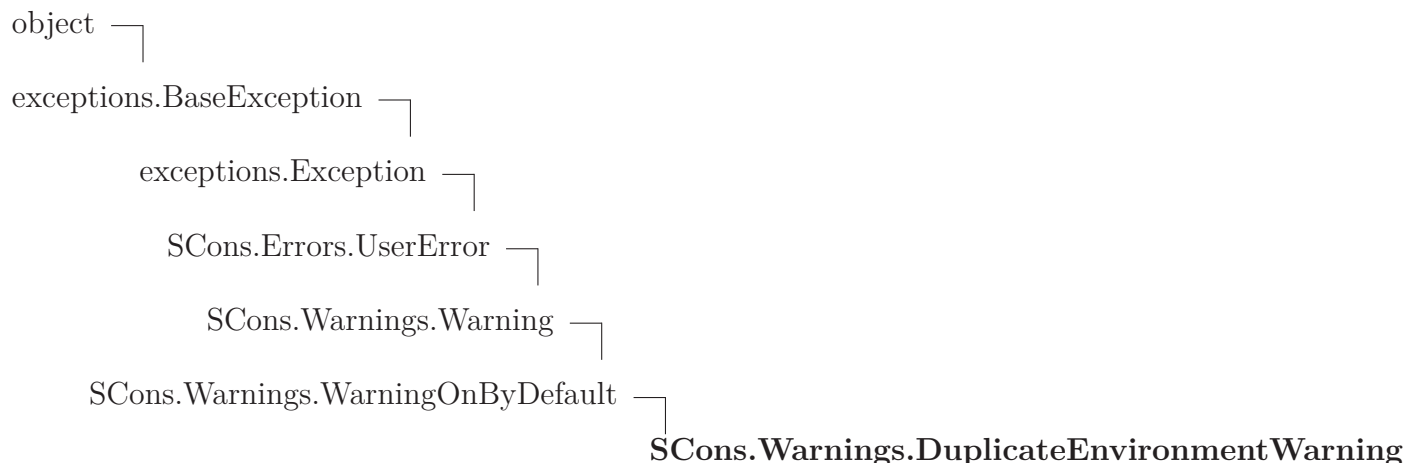
### 43.9.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	

*continued on next page*

Name	Description
<i>Inherited from object</i>	
__class__	

## 43.10 Class DuplicateEnvironmentWarning



### 43.10.1 Methods

#### *Inherited from exceptions.Exception*

\_\_init\_\_(), \_\_new\_\_()

#### *Inherited from exceptions.BaseException*

\_\_delattr\_\_(), \_\_getattr\_\_(), \_\_getitem\_\_(), \_\_getslice\_\_(), \_\_reduce\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_setstate\_\_(), \_\_str\_\_(), \_\_unicode\_\_()

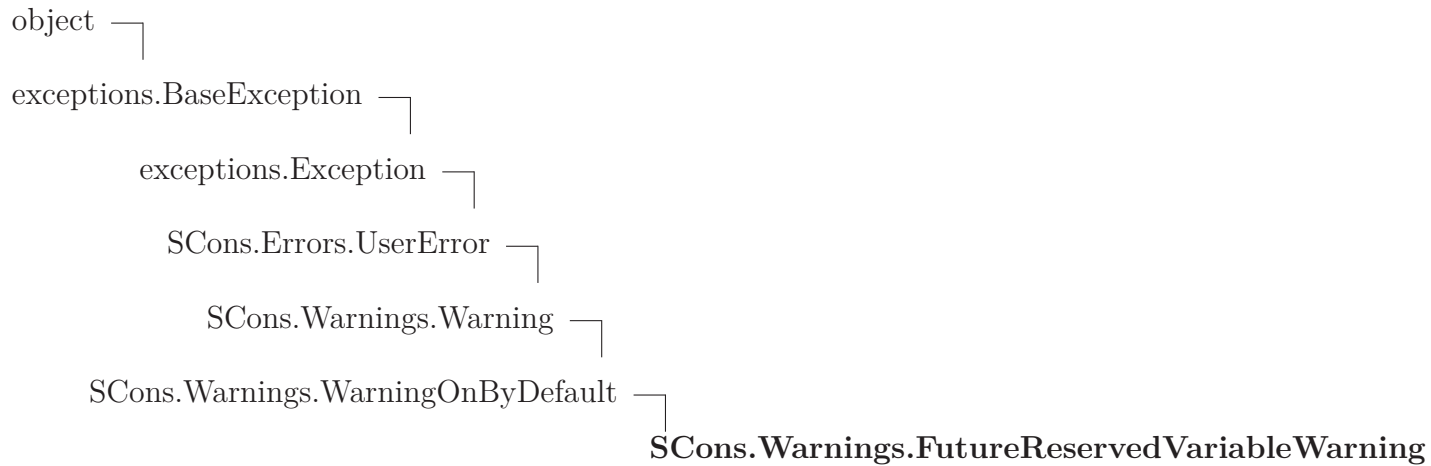
#### *Inherited from object*

\_\_format\_\_(), \_\_hash\_\_(), \_\_reduce\_ex\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

### 43.10.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

### 43.11 Class FutureReservedVariableWarning



#### 43.11.1 Methods

##### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

##### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

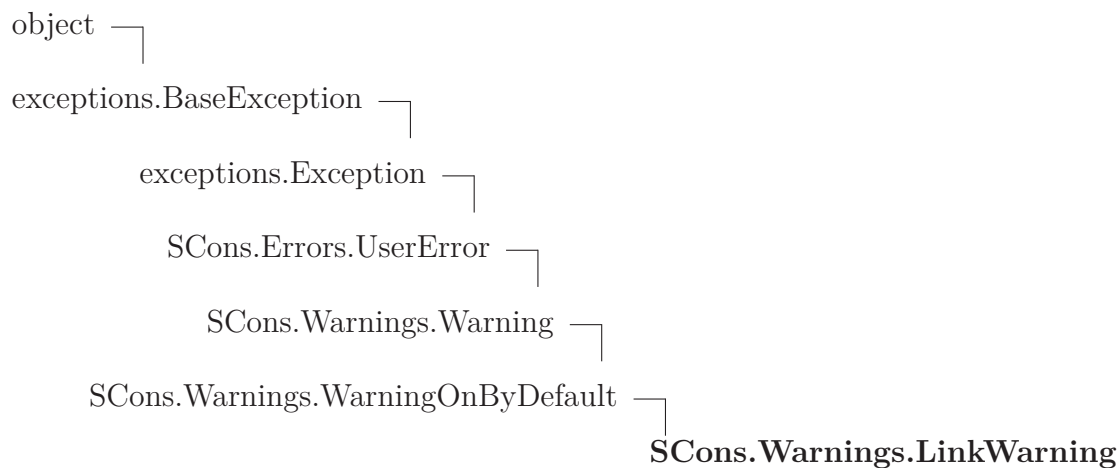
##### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 43.11.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

## 43.12 Class LinkWarning



**Known Subclasses:** SCons.Warnings.FortranCxxMixWarning

### 43.12.1 Methods

*Inherited from exceptions.Exception*

`__init__()`, `__new__()`

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

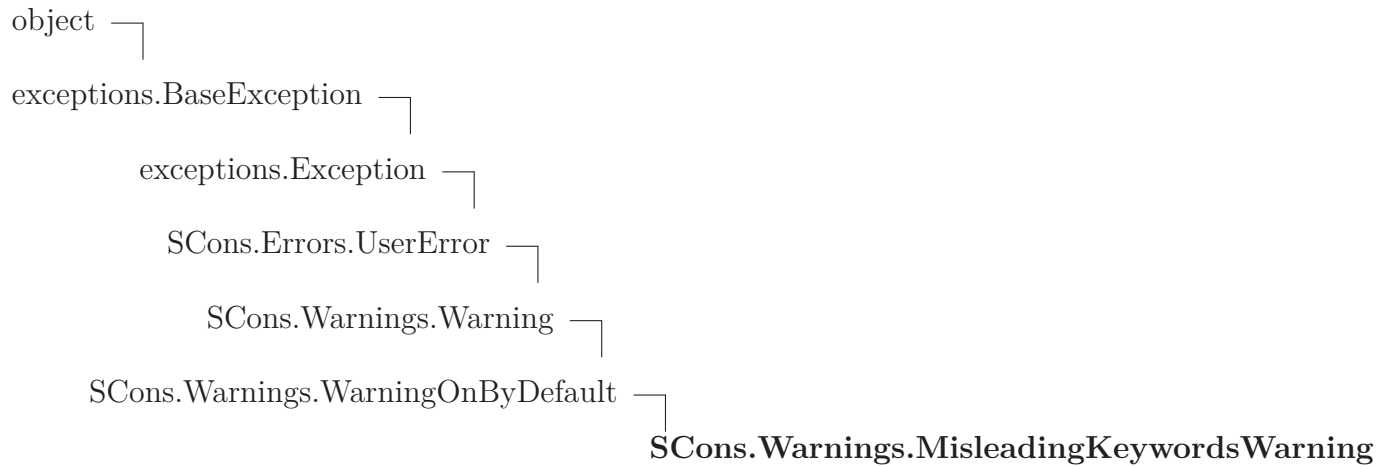
*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.12.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
	args, message
<i>Inherited from object</i>	
<code>__class__</code>	

### 43.13 Class `MisleadingKeywordsWarning`



#### 43.13.1 Methods

##### *Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

##### *Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

##### *Inherited from `object`*

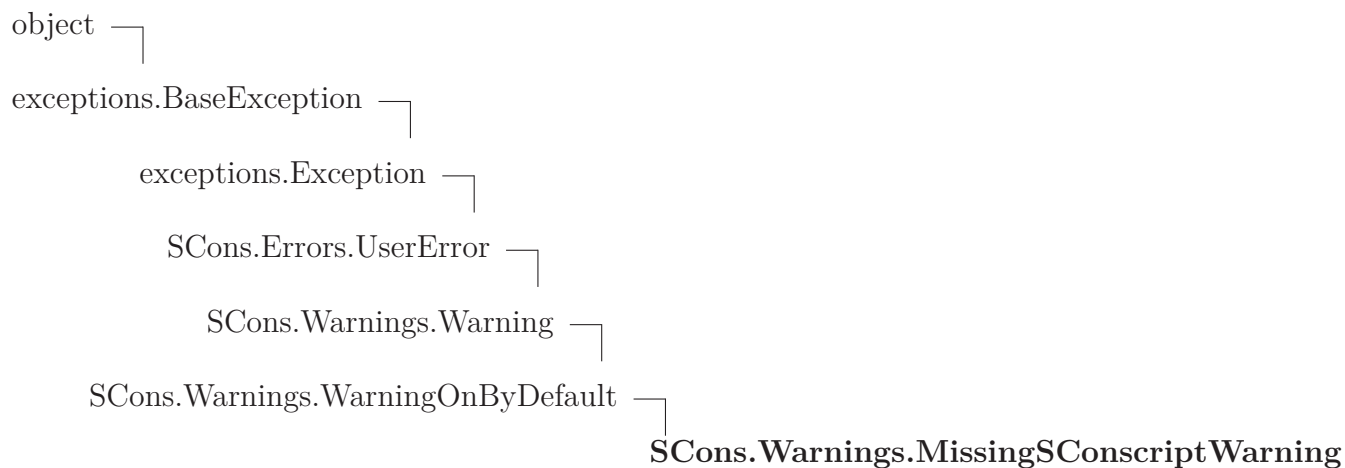
`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 43.13.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
args, message	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	



### 43.14 Class `MissingSConscriptWarning`



#### 43.14.1 Methods

##### *Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

##### *Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

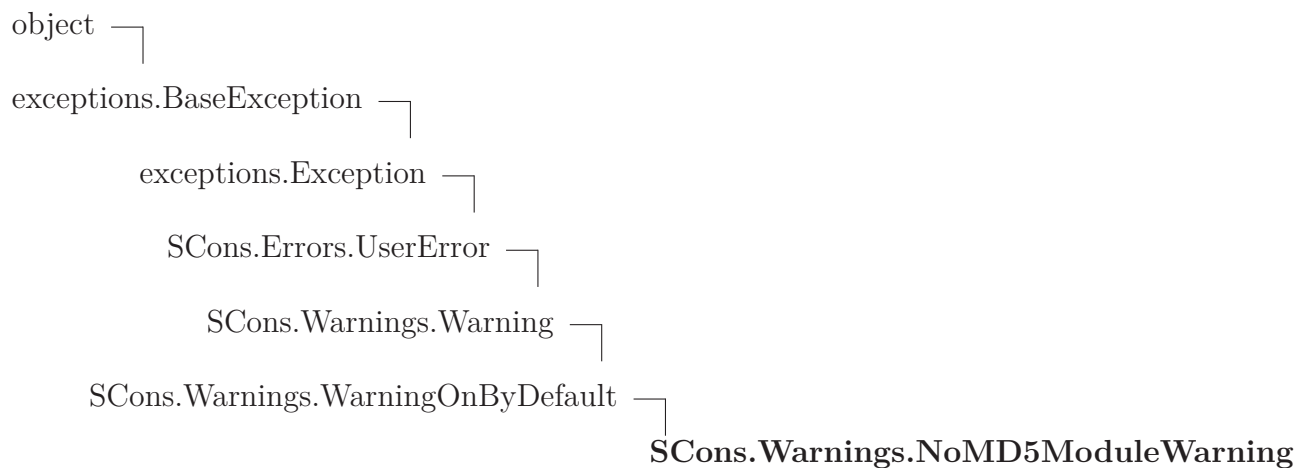
##### *Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 43.14.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
args, message	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

### 43.15 Class NoMD5ModuleWarning



#### 43.15.1 Methods

##### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

##### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

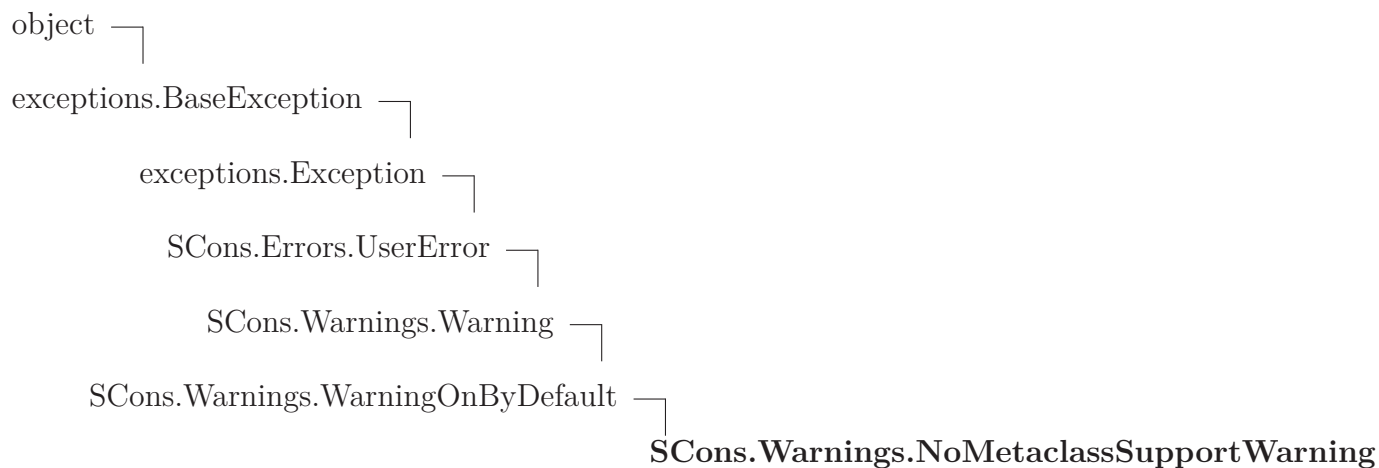
##### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 43.15.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

## 43.16 Class NoMetaclassSupportWarning



### 43.16.1 Methods

#### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

#### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

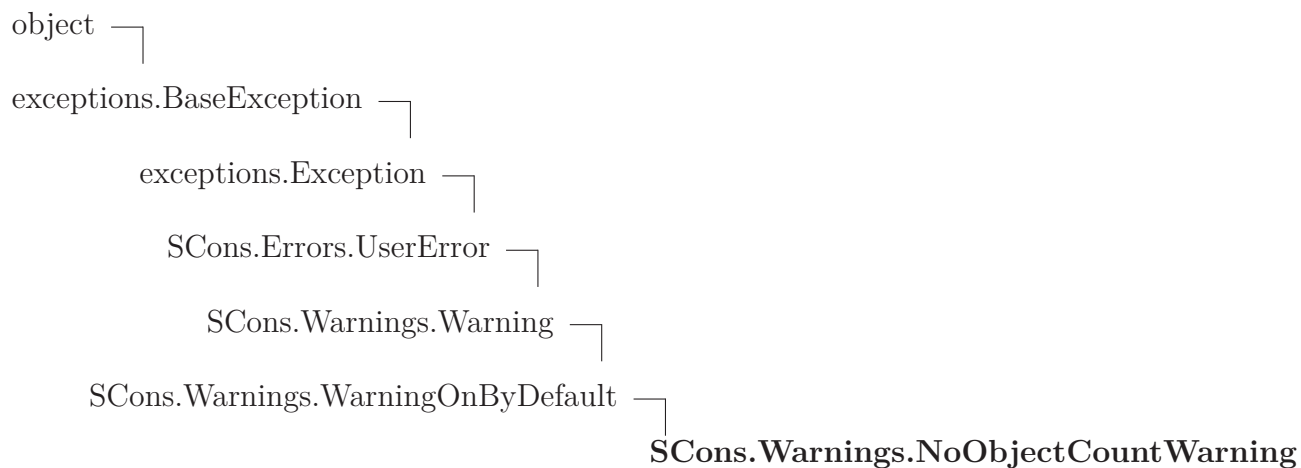
#### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.16.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

### 43.17 Class NoObjectCountWarning



#### 43.17.1 Methods

##### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

##### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

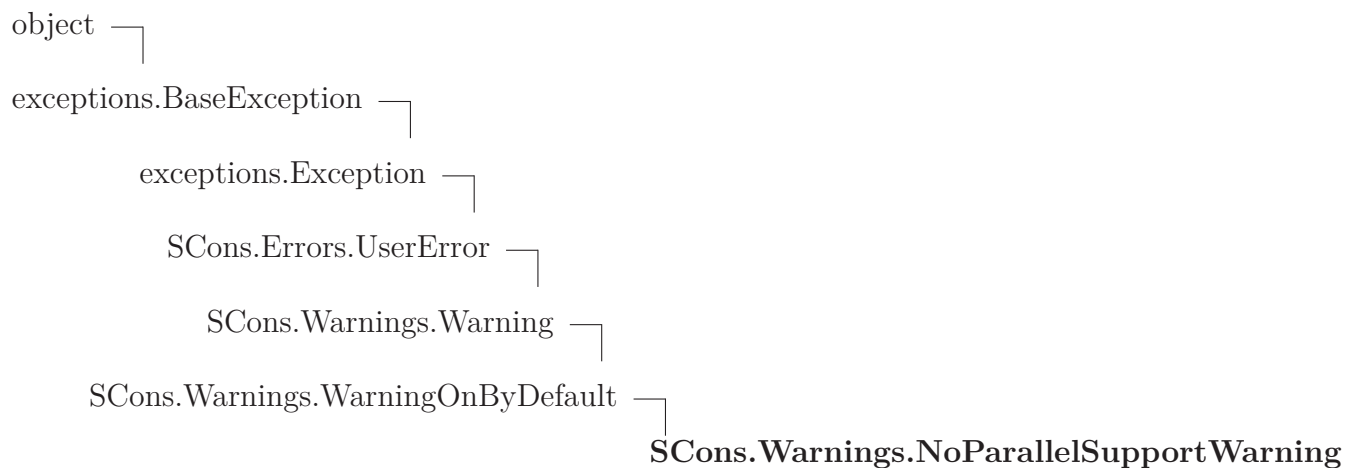
##### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 43.17.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

### 43.18 Class NoParallelSupportWarning



#### 43.18.1 Methods

##### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

##### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

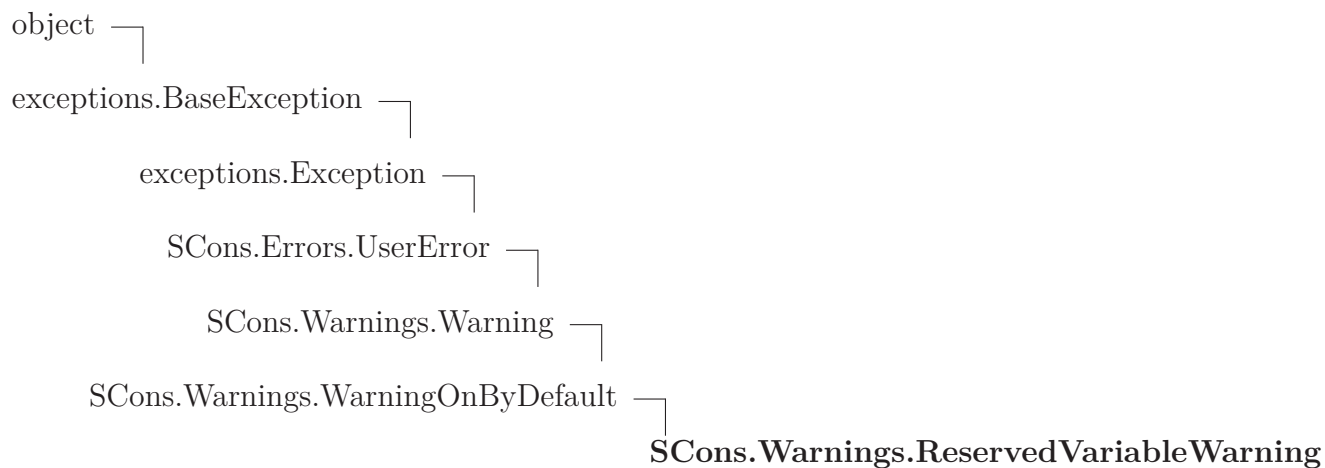
##### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 43.18.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
	args, message
<i>Inherited from object</i>	
	<code>__class__</code>

## 43.19 Class ReservedVariableWarning



### 43.19.1 Methods

#### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

#### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

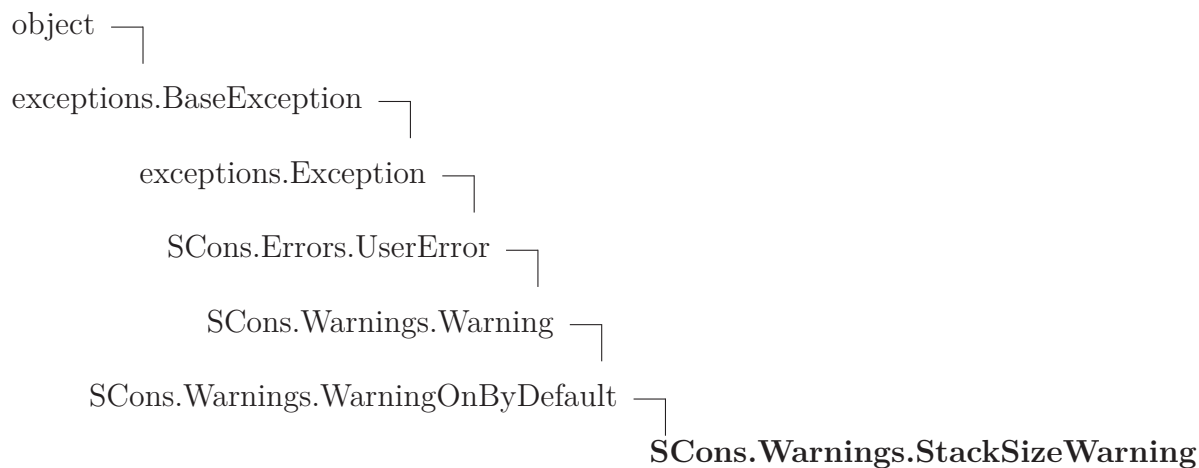
#### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.19.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

## 43.20 Class StackSizeWarning



### 43.20.1 Methods

#### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

#### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

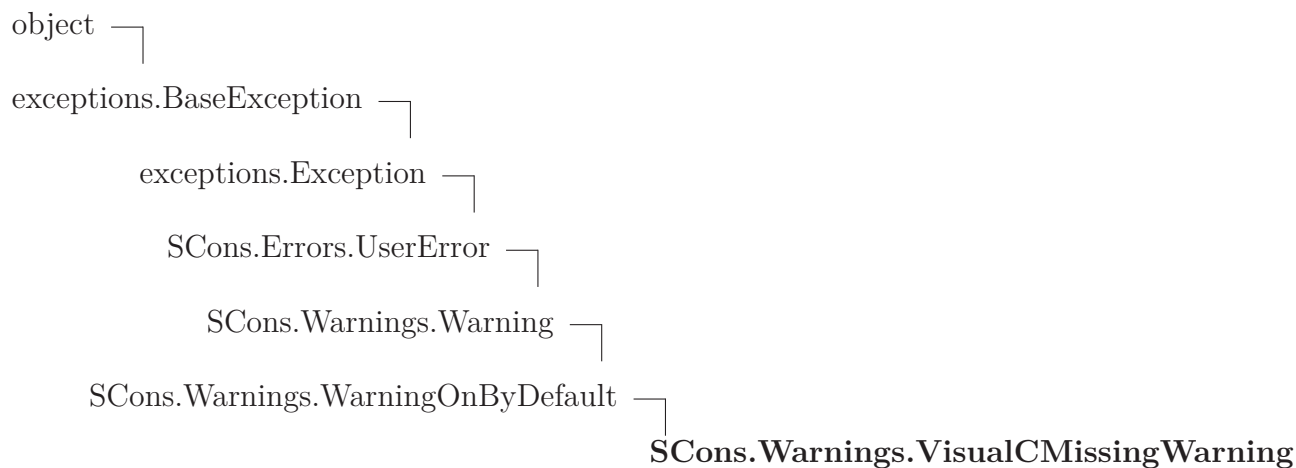
#### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.20.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

## 43.21 Class VisualCMissingWarning



### 43.21.1 Methods

#### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

#### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

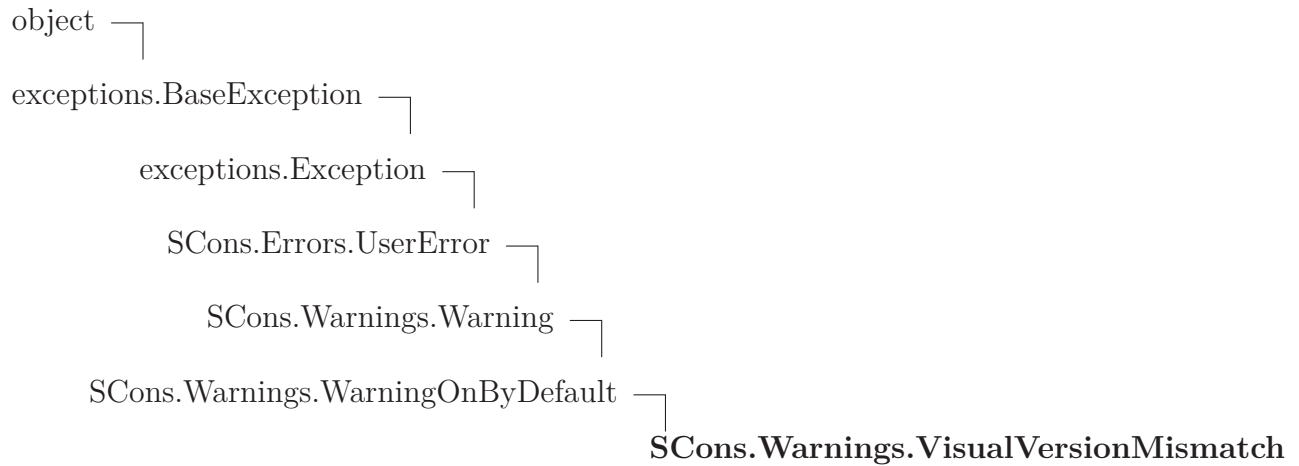
#### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.21.2 Properties

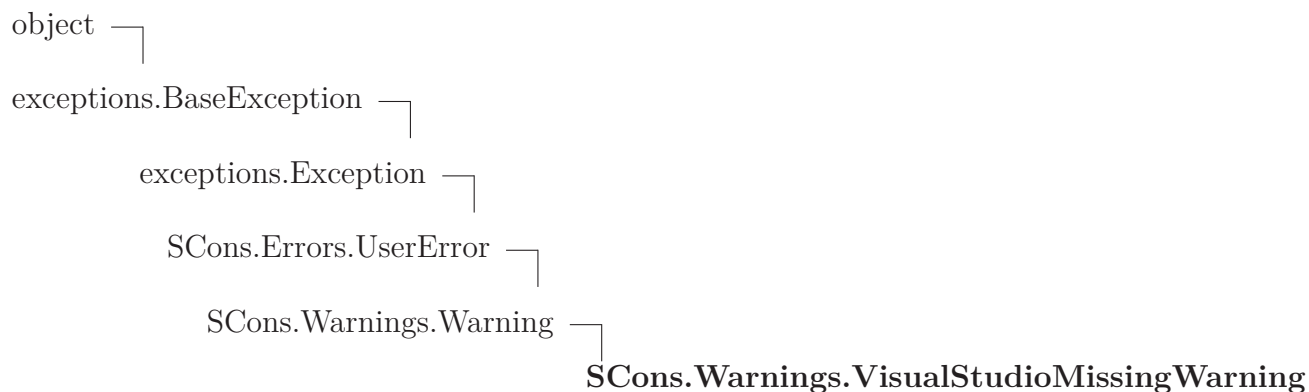
Name	Description
<i>Inherited from exceptions.BaseException</i>	
	args, message
<i>Inherited from object</i>	
<code>__class__</code>	



**43.22 Class `VisualVersionMismatch`****43.22.1 Methods*****Inherited from `exceptions.Exception`***`__init__()`, `__new__()`***Inherited from `exceptions.BaseException`***`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`***Inherited from `object`***`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`**43.22.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

### 43.23 Class VisualStudioMissingWarning



#### 43.23.1 Methods

##### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

##### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

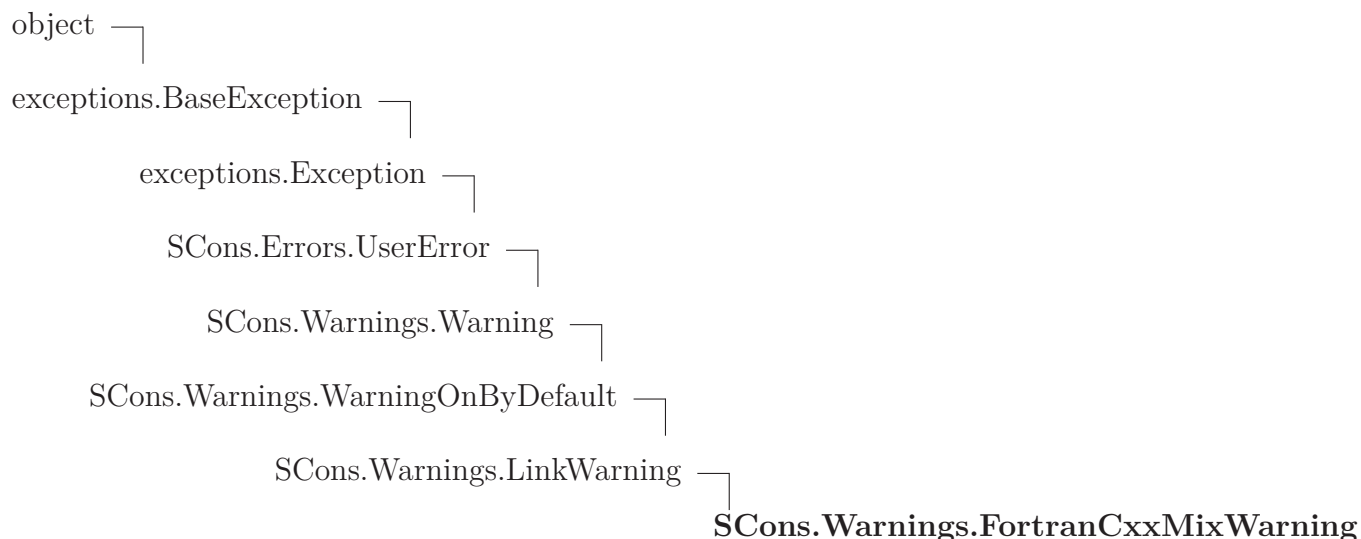
##### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 43.23.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 43.24 Class FortranCxxMixWarning



### 43.24.1 Methods

#### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

#### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

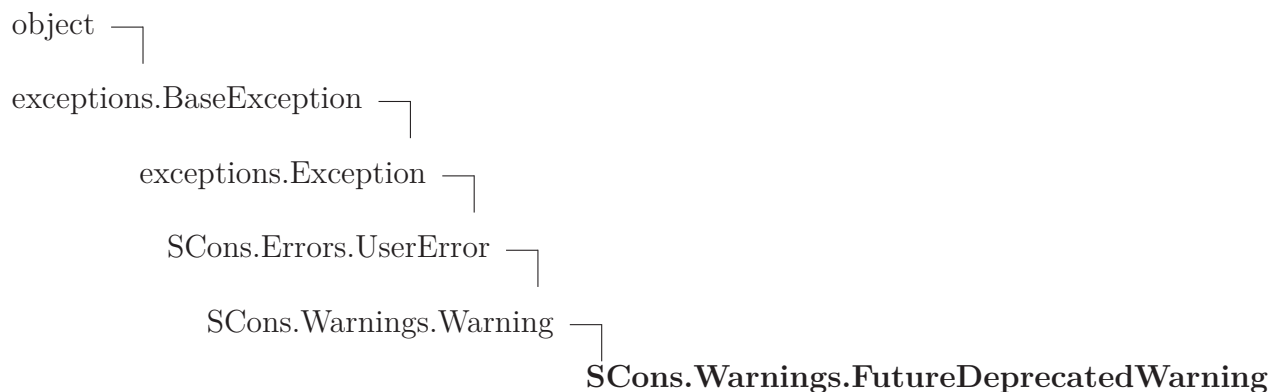
#### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.24.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
	args, message
<i>Inherited from object</i>	
<code>__class__</code>	

## 43.25 Class FutureDeprecatedWarning



**Known Subclasses:** `SCons.Warnings.DeprecatedSourceCodeWarning`

### 43.25.1 Methods

*Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

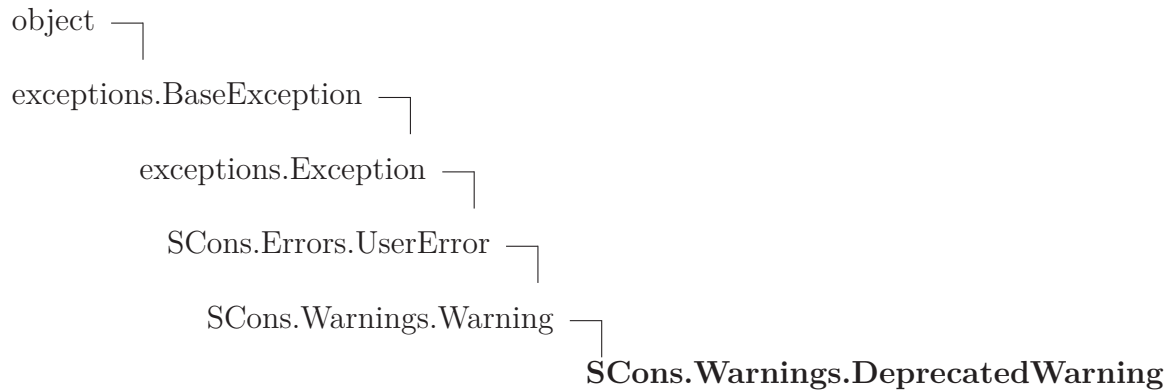
*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.25.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 43.26 Class `DeprecatedWarning`



**Known Subclasses:** `SCons.Warnings.DeprecatedBuildDirWarning`, `SCons.Warnings.MandatoryDeprecatedWarning`, `SCons.Warnings.PythonVersionWarning`, `SCons.Warnings.TaskmasterNeedsExecuteWarning`

### 43.26.1 Methods

#### *Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

#### *Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

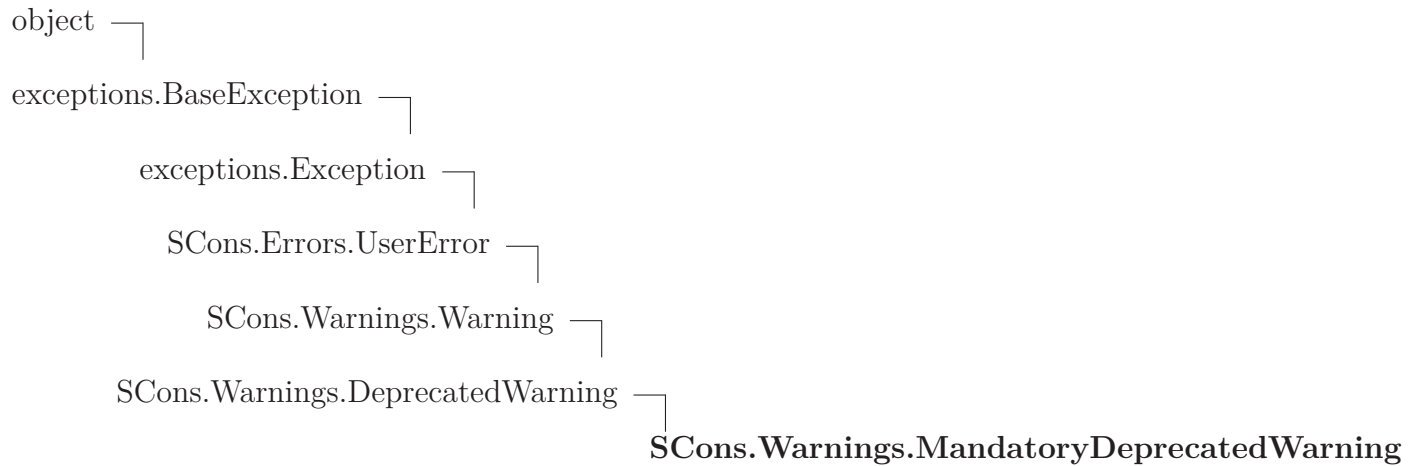
#### *Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.26.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

### 43.27 Class MandatoryDeprecatedWarning



**Known Subclasses:** SCons.Warnings.DeprecatedBuilderKeywordsWarning, SCons.Warnings.DeprecatedSCons.Warnings.DeprecatedDebugOptionsWarning, SCons.Warnings.DeprecatedOptionsWarning, SCons.Warnings.DeprecatedSigModuleWarning, SCons.Warnings.DeprecatedSourceSignaturesWarning, SCons.Warnings.DeprecatedTargetSignaturesWarning

#### 43.27.1 Methods

*Inherited from exceptions.Exception*

`__init__()`, `__new__()`

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

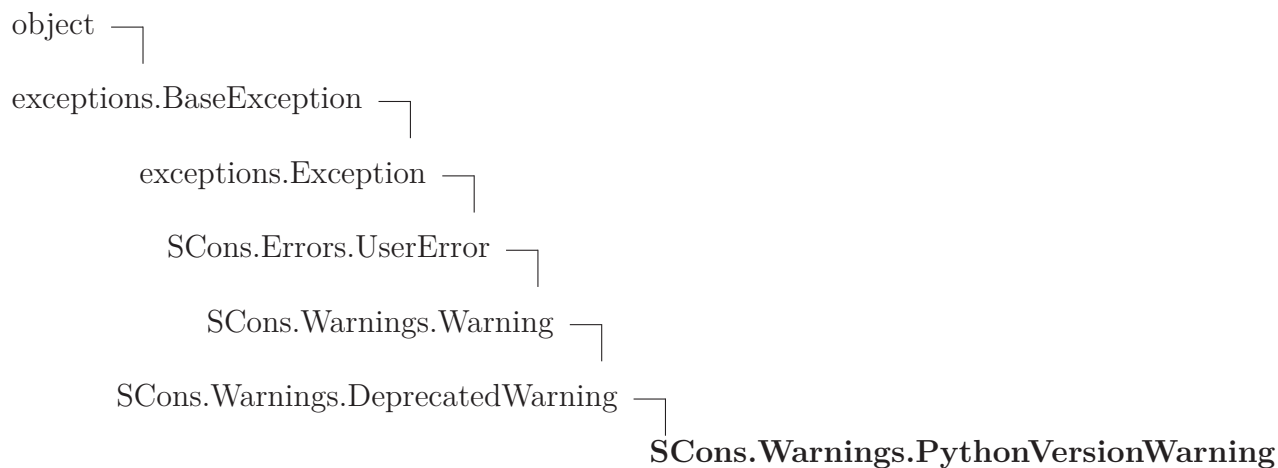
*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 43.27.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
	args, message
<i>Inherited from object</i>	
<code>__class__</code>	

## 43.28 Class `PythonVersionWarning`



### 43.28.1 Methods

#### *Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

#### *Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

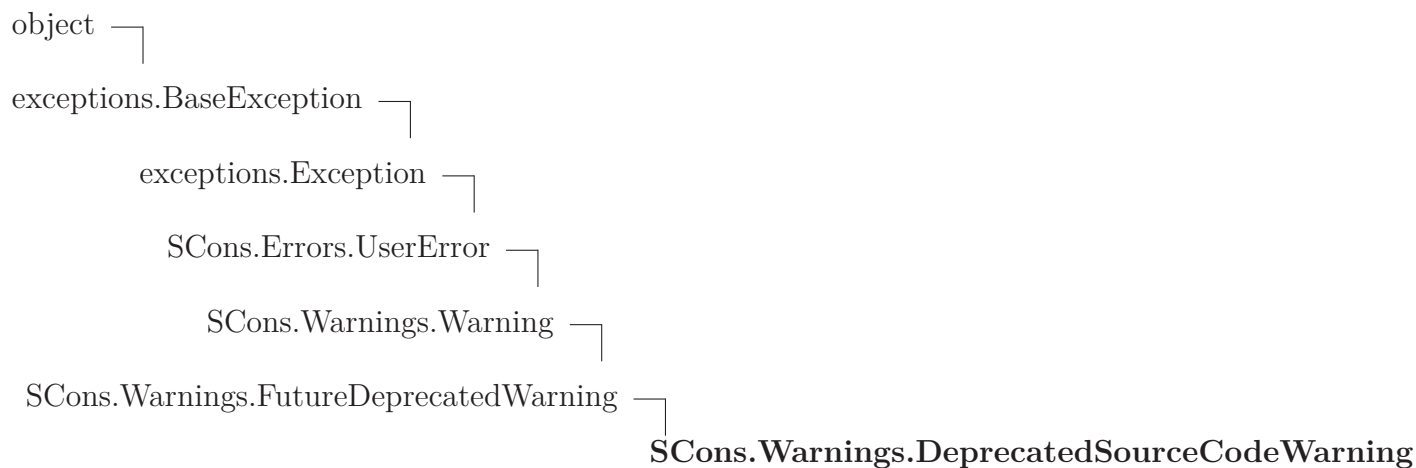
#### *Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.28.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
	args, message
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 43.29 Class `DeprecatedSourceCodeWarning`



### 43.29.1 Methods

#### *Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

#### *Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

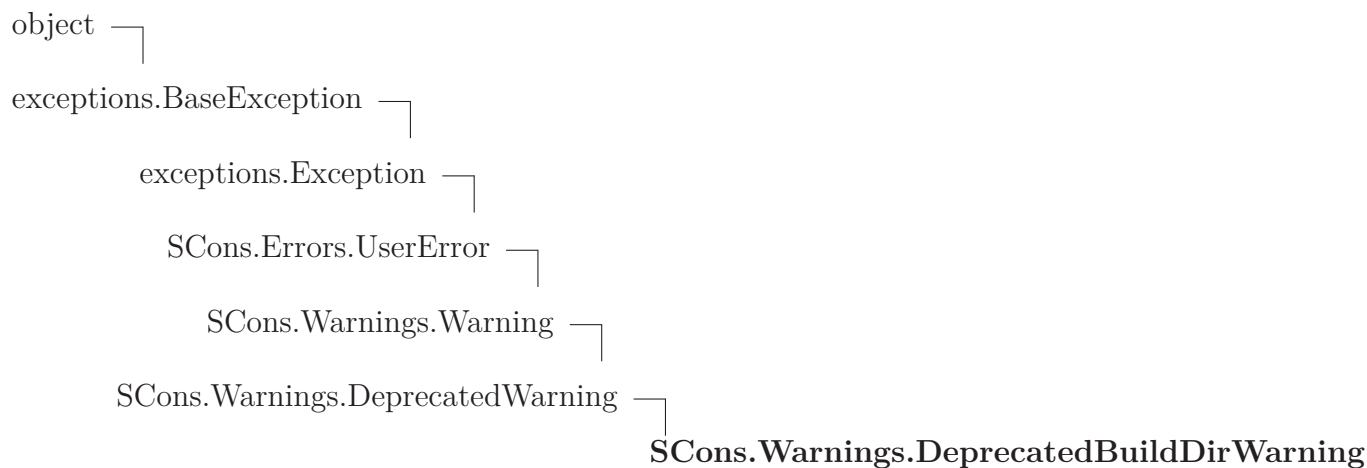
#### *Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 43.29.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
args, message	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	



**43.30 Class *DeprecatedBuildDirWarning*****43.30.1 Methods*****Inherited from exceptions.Exception***

`__init__()`, `__new__()`

***Inherited from exceptions.BaseException***

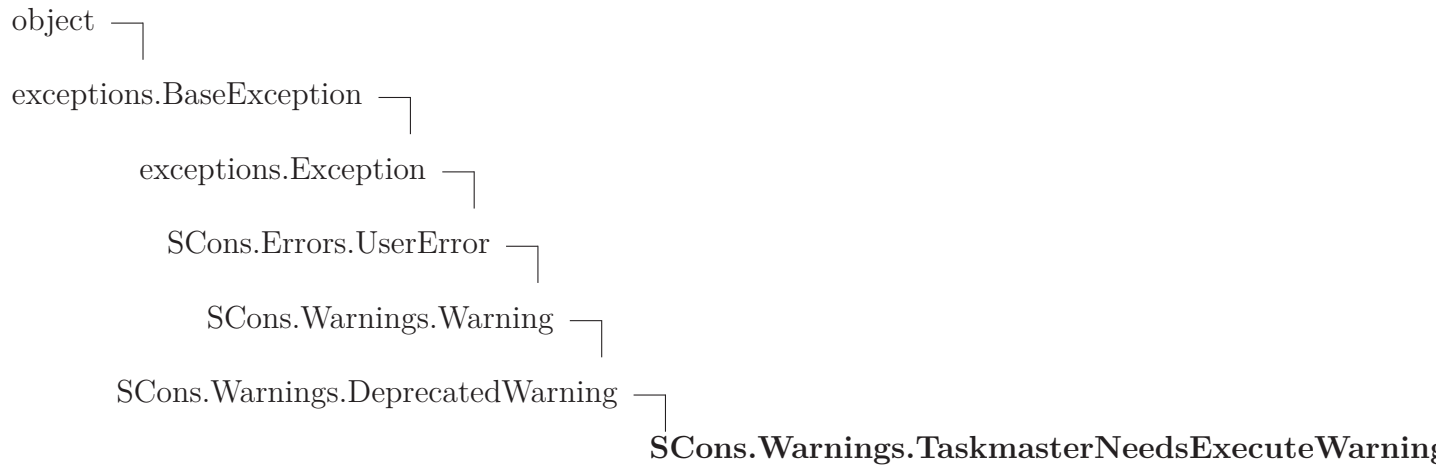
`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**43.30.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

**43.31 Class TaskmasterNeedsExecuteWarning****43.31.1 Methods*****Inherited from exceptions.Exception***

`__init__()`, `__new__()`

***Inherited from exceptions.BaseException***

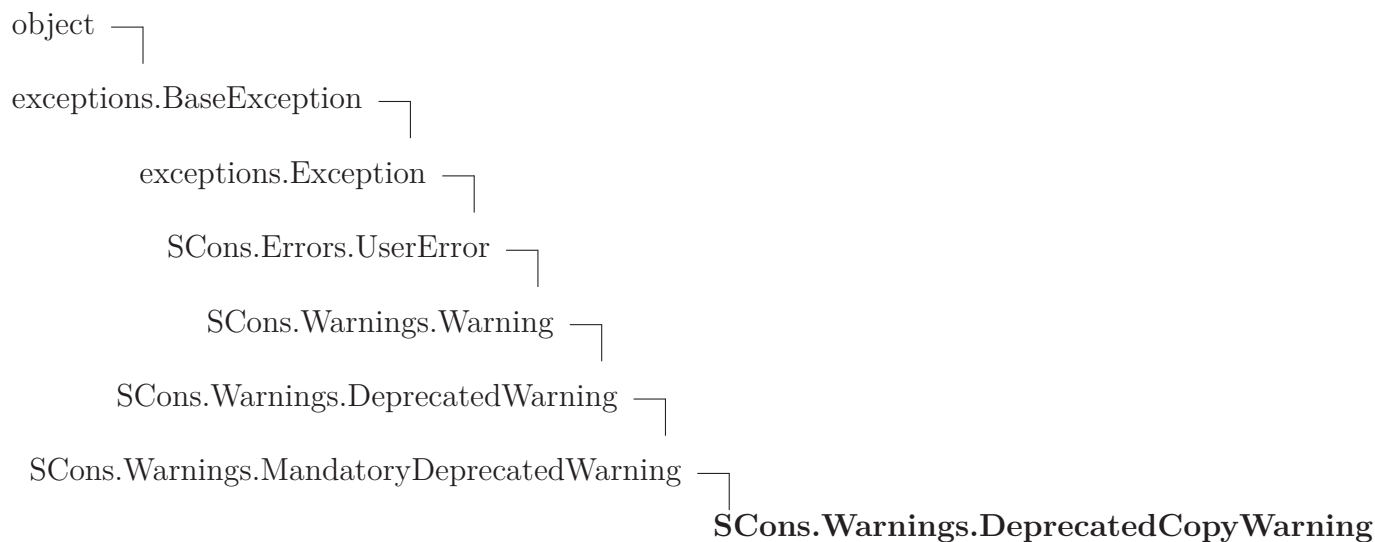
`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**43.31.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

**43.32 Class *DeprecatedCopyWarning*****43.32.1 Methods*****Inherited from exceptions.Exception***

`__init__()`, `__new__()`

***Inherited from exceptions.BaseException***

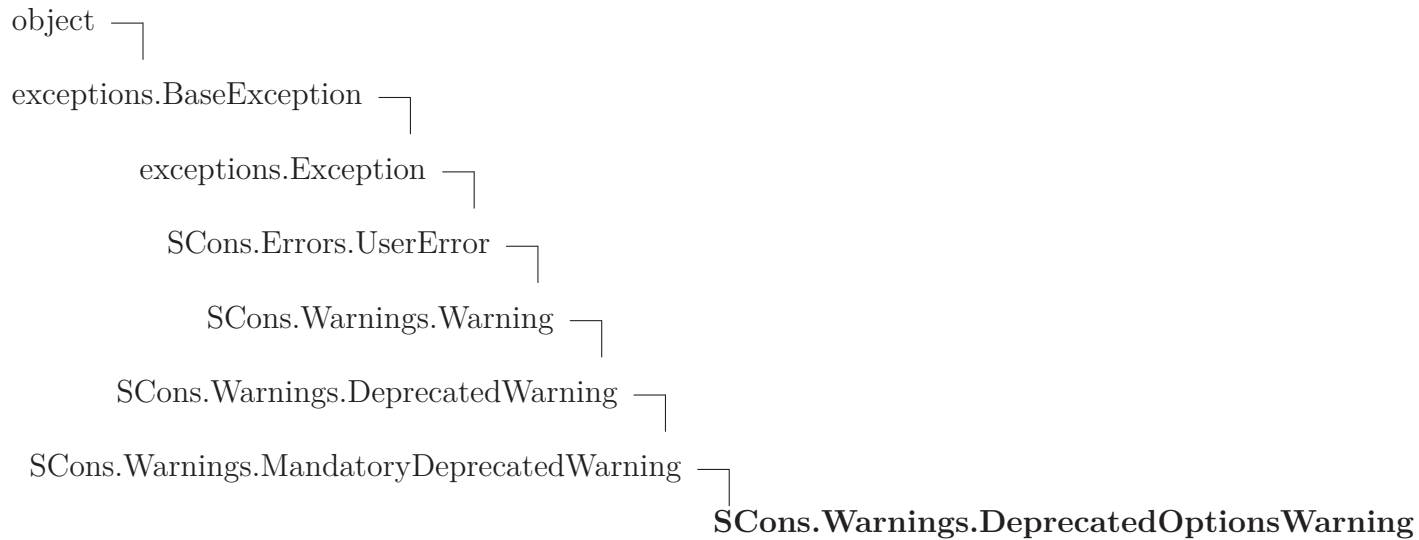
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

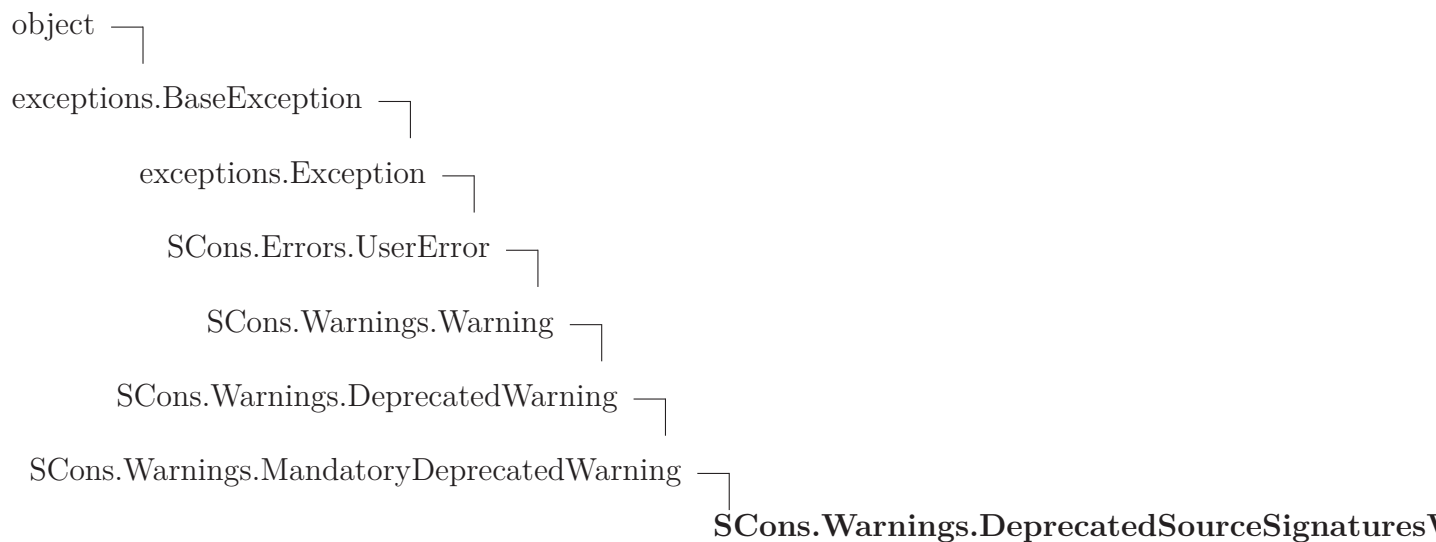
`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**43.32.2 Properties**

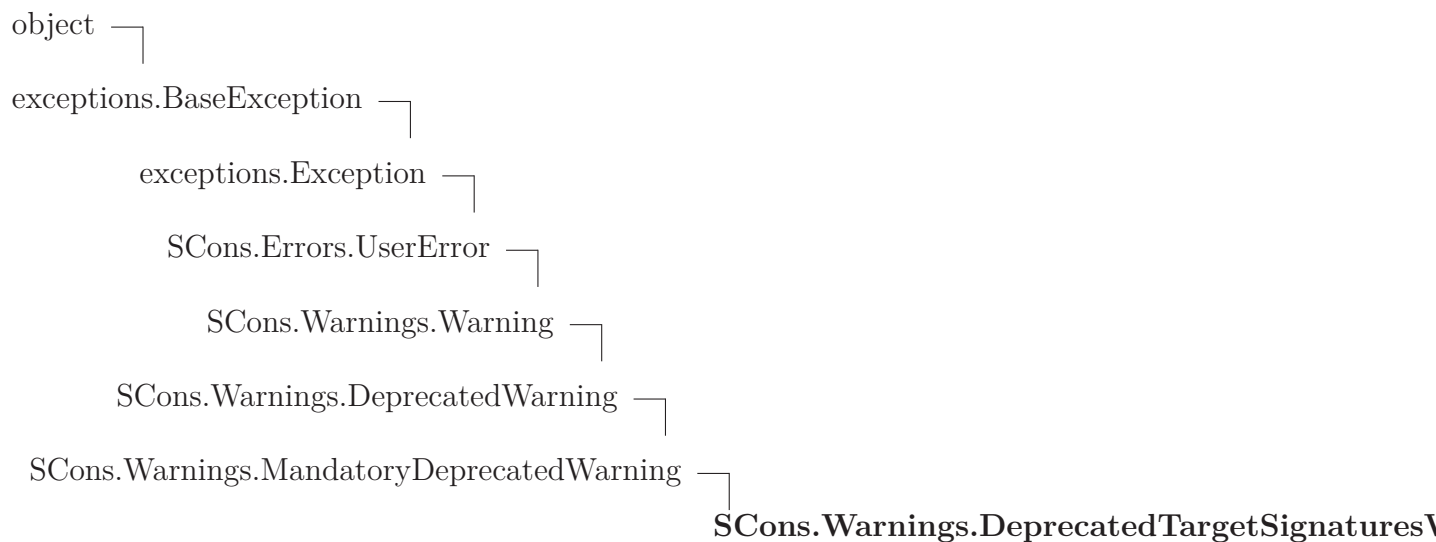
Name	Description
<i>Inherited from exceptions.BaseException</i>	
	args, message
<i>Inherited from object</i>	
<code>__class__</code>	

**43.33 Class `DeprecatedOptionsWarning`****43.33.1 Methods*****Inherited from `exceptions.Exception`***`__init__()`, `__new__()`***Inherited from `exceptions.BaseException`***`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`***Inherited from `object`***`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`**43.33.2 Properties**

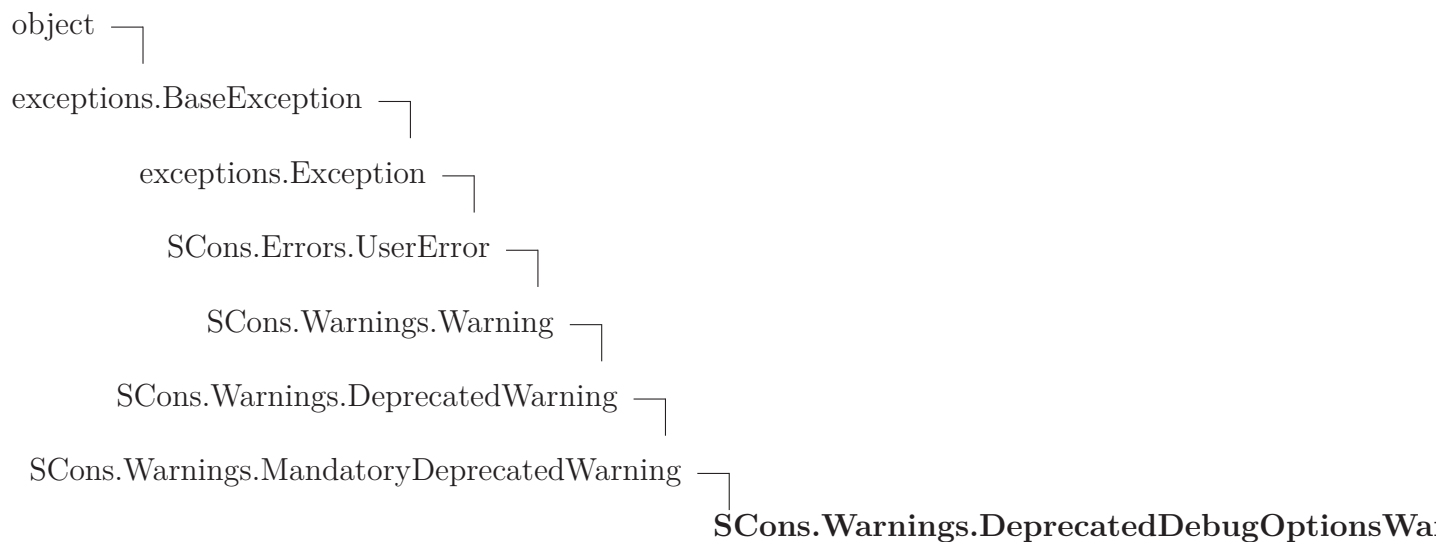
Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
	args, message
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**43.34 Class `DeprecatedSourceSignaturesWarning`****43.34.1 Methods*****Inherited from `exceptions.Exception`***`__init__()`, `__new__()`***Inherited from `exceptions.BaseException`***`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`***Inherited from `object`***`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`**43.34.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
	args, message
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**43.35 Class `DeprecatedTargetSignaturesWarning`****43.35.1 Methods***Inherited from `exceptions.Exception`*`__init__()`, `__new__()`*Inherited from `exceptions.BaseException`*`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`*Inherited from `object`*`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`**43.35.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
	args, message
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**43.36 Class *DeprecatedDebugOptionsWarning*****43.36.1 Methods*****Inherited from exceptions.Exception***

`__init__()`, `__new__()`

***Inherited from exceptions.BaseException***

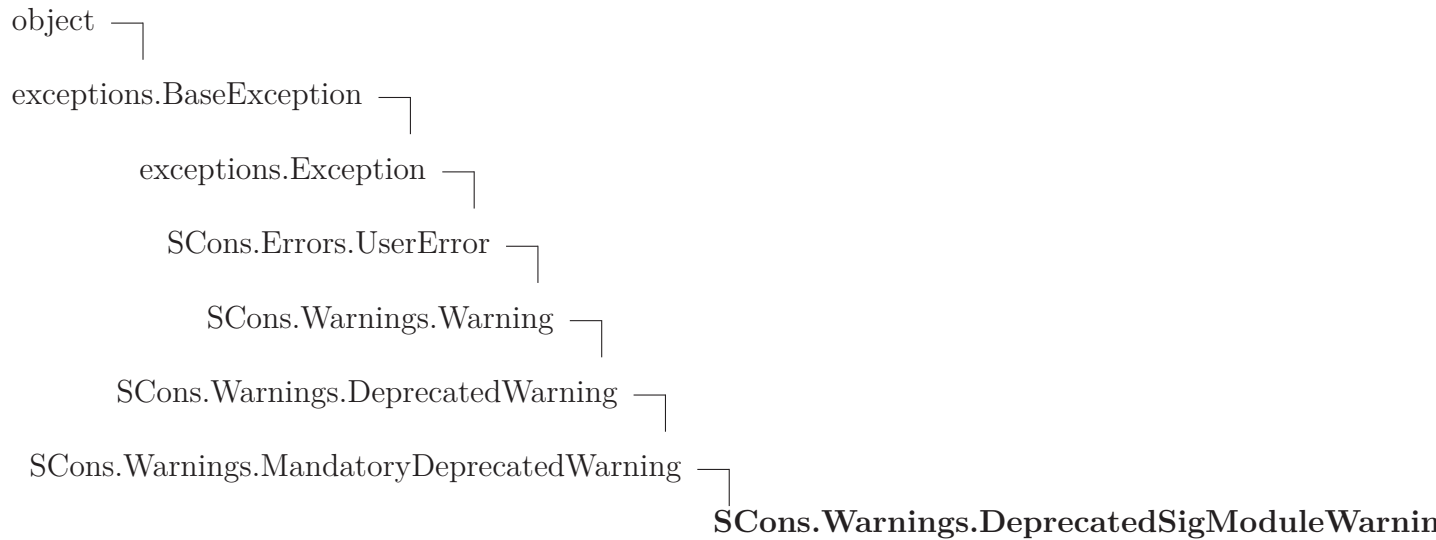
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

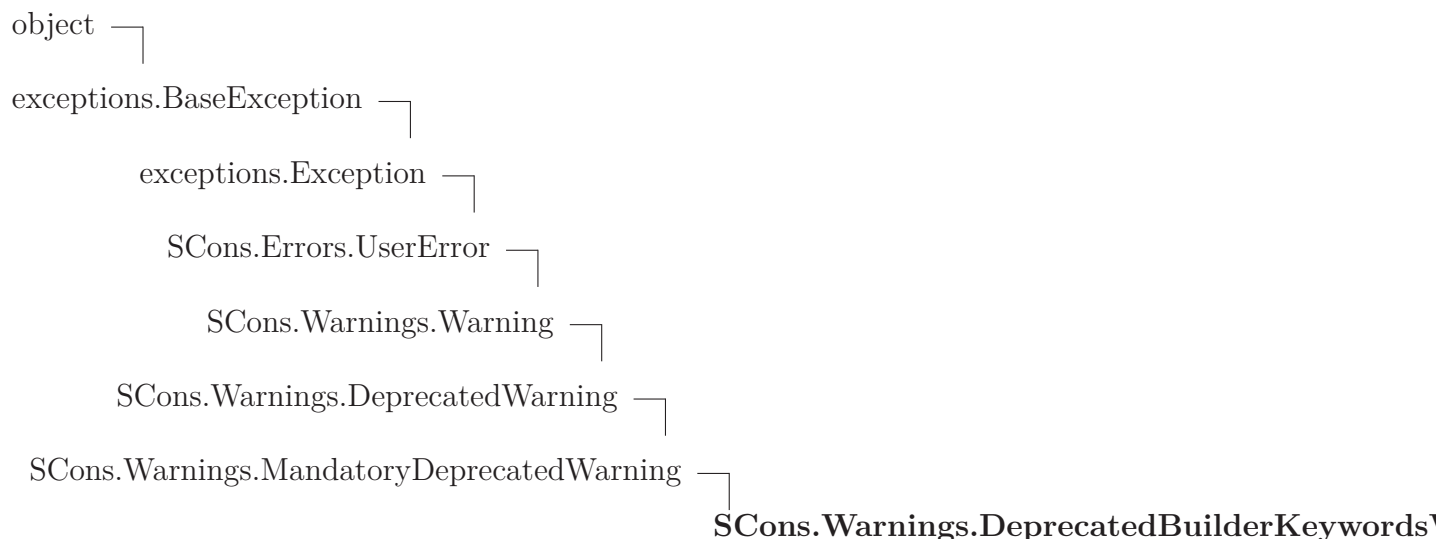
**43.36.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
	args, message
<i>Inherited from object</i>	
<code>__class__</code>	

**43.37 Class *DeprecatedSigModuleWarning*****43.37.1 Methods*****Inherited from exceptions.Exception***`__init__()`, `__new__()`***Inherited from exceptions.BaseException***`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`***Inherited from object***`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`**43.37.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
	args, message
<i>Inherited from object</i>	
<code>__class__</code>	



**43.38 Class `DeprecatedBuilderKeywordsWarning`****43.38.1 Methods***Inherited from `exceptions.Exception`*`__init__()`, `__new__()`*Inherited from `exceptions.BaseException`*`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`*Inherited from `object`*`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`**43.38.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
args, message	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 44 Module SCons.cpp

SCons C Pre-Processor module

### 44.1 Functions

<b>CPP_to_Python_Ops_Sub</b> ( <i>m</i> )
---

<b>CPP_to_Python</b> ( <i>s</i> )
-----------------------------------

Converts a C pre-processor expression into an equivalent Python expression that can be evaluated.
---

### 44.2 Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> ...
<code>cpp_lines_dict</code>	<b>Value:</b> {'define': '\\s+([_A-Za-z][_A-Za-z0-9_]*)\\(([^)]*\\))'?...
Table	<b>Value:</b> {'define': re.compile(r'\\s+([_A-Za-z][_A-Za-z0-9_]*)\\(([^...
e	<b>Value:</b> '^\\s*#\\s*(elif undef include_next endif else include if.
CPP_Expression	<b>Value:</b> re.compile(r'(?m)^\\s*#\\s*(elif undef include_next endif e.
CPP_to_Python_Ops_Dict	<b>Value:</b> {'\\r': ' ', '!': ' not ', '!=': ' != ', '&&': ' and ', ':'...
CPP_to_Python_Ops_Expression	<b>Value:</b> re.compile(r'\\ \\ && != ! \\r :\\ ?')
CPP_to_Python_Eval_List	<b>Value:</b> [[re.compile(r'defined\\s+(\\w+)'), '"\\1" in __dict__'], [...
line_continuations	<b>Value:</b> re.compile(r'\\\\r?\\n')
function_name	<b>Value:</b> re.compile(r'\\S+\\(((\\^\\))*)\\)')
function_arg_separator	<b>Value:</b> re.compile(r',\\s*')
<code>__package__</code>	<b>Value:</b> 'SCons'
x	<b>Value:</b> 'if'

### 44.3 Class FunctionEvaluator

object —  
**SCons.cpp.FunctionEvaluator**

Handles delayed evaluation of a #define function call.

#### 44.3.1 Methods

<b>__init__</b> ( <i>self, name, args, expansion</i> )
Squirrels away the arguments and expansion value of a #define macro function for later evaluation when we must actually expand a value that uses it. Overrides: object.__init__
<b>__call__</b> ( <i>self, *values</i> )
Evaluates the expansion of a #define macro function called with the specified values.

#### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(),  
 \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(),  
 \_\_str\_\_(), \_\_subclasshook\_\_()

#### 44.3.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

### 44.4 Class PreProcessor

object —  
**SCons.cpp.PreProcessor**

**Known Subclasses:** SCons.cpp.DumbPreProcessor, SCons.Scanner.C.SConsCPPScanner

The main workhorse class for handling C pre-processing.

#### 44.4.1 Methods

**\_\_call\_\_**(*self*, *file*)

Pre-processes a file.

This is the main public entry point.

**\_\_init\_\_**(*self*, *current*='.', *cpppath*=(), *dict*={}, *all*=0)

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature Overrides: object.**\_\_init\_\_** **\_\_exit\_\_**(inherited documentation)

**all\_include**(*self*, *t*)

**do\_define**(*self*, *t*)

Default handling of a `#define` line.

**do\_elif**(*self*, *t*)

Default handling of a `#elif` line.

**do\_else**(*self*, *t*)

Default handling of a `#else` line.

**do\_endif**(*self*, *t*)

Default handling of a `#endif` line.

**do\_if**(*self*, *t*)

Default handling of a `#if` line.

**do\_ifdef**(*self*, *t*)

Default handling of a `#ifdef` line.

**do\_ifndef**(*self*, *t*)

Default handling of a `#ifndef` line.

**do\_import**(*self*, *t*)

Default handling of a `#import` line.

**do\_include**(*self*, *t*)

Default handling of a `#include` line.

**do\_include\_next**(*self*, *t*)

Default handling of a `#include` line.

**do\_nothing**(*self*, *t*)

Null method for when we explicitly want the action for a specific preprocessor directive to do nothing.

**do\_undef**(*self*, *t*)

Default handling of a `#undef` line.

**eval\_expression**(*self*, *t*)

Evaluates a C preprocessor expression.

This is done by converting it to a Python equivalent and eval()ing it in the C preprocessor namespace we use to track #define values.

**finalize\_result**(*self*, *fname*)**find\_include\_file**(*self*, *t*)

Finds the #include file for a given preprocessor tuple.

**initialize\_result**(*self*, *fname*)**process\_contents**(*self*, *contents*, *fname*=None)

Pre-processes a file contents.

This is the main internal entry point.

**read\_file**(*self*, *file*)**resolve\_include**(*self*, *t*)

Resolve a tuple-ized #include line.

This handles recursive expansion of values without "" or <> surrounding the name until an initial " or < is found, to handle

#include FILE

where FILE is a #define somewhere else.

**restore**(*self*)

Pops the previous dispatch table off the stack and makes it the current one.

**save**(*self*)

Pushes the current dispatch table on the stack and re-initializes the current dispatch table to the default.

**scons\_current\_file**(*self*, *t*)

**start\_handling\_includes**(*self*, *t*=None)

Causes the PreProcessor object to start processing `#import`, `#include` and `#include_next` lines.

This method will be called when a `#if`, `#ifdef`, `#ifndef` or `#elif` evaluates True, or when we reach the `#else` in a `#if`, `#ifdef`, `#ifndef` or `#elif` block where a condition already evaluated False.

**stop\_handling\_includes**(*self*, *t*=None)

Causes the PreProcessor object to stop processing `#import`, `#include` and `#include_next` lines.

This method will be called when a `#if`, `#ifdef`, `#ifndef` or `#elif` evaluates False, or when we reach the `#else` in a `#if`, `#ifdef`, `#ifndef` or `#elif` block where a condition already evaluated True.

**tupleize**(*self*, *contents*)

Turns the contents of a file into a list of easily-processed tuples describing the CPP lines in the file.

The first element of each tuple is the line's preprocessor directive (`#if`, `#include`, `#define`, etc., minus the initial `'#'`). The remaining elements are specific to the type of directive, as pulled apart by the regular expression.

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

#### 44.4.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

### 44.5 Class DumbPreProcessor



A preprocessor that ignores all `#if/#elif/#else/#endif` directives and just reports back *all* of the `#include` files (like the classic SCons scanner did).

This is functionally equivalent to using a regular expression to find all of the `#include` lines, only slower. It exists mainly as an example of how the main PreProcessor class can be sub-classed to tailor its behavior.

#### 44.5.1 Methods

```
__init__(self, *args, **kw)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature Overrides:  
object.\_\_init\_\_ extit(inherited documentation)

#### *Inherited from SCons.cpp.PreProcessor(Section 44.4)*

```
__call__(), all_include(), do_define(), do_elif(), do_else(), do_endif(), do_if(),
do_ifdef(), do_ifndef(), do_import(), do_include(), do_include_next(), do_nothing(),
do_undef(), eval_expression(), finalize_result(), find_include_file(), initialize_result(),
process_contents(), read_file(), resolve_include(), restore(), save(), scons_current_file(),
start_handling_includes(), stop_handling_includes(), tupleize()
```

#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

#### 44.5.2 Properties



Name	Description
<i>Inherited from object</i> __class__	

## 45 Module *SCons.dblite*

### 45.1 Functions

`corruption_warning(filename)`

`is_string(s)`

`unicode(s)`

`open(file, flag=None, mode=438)`

### 45.2 Variables

Name	Description
<code>keep_all_files</code>	<b>Value:</b> 0
<code>ignore_corrupt_dbfiles</code>	<b>Value:</b> 0
<code>dblite_suffix</code>	<b>Value:</b> <code>' .dblite'</code>
<code>tmp_suffix</code>	<b>Value:</b> <code>' .tmp'</code>
<code>__package__</code>	<b>Value:</b> <code>'SCons'</code>

### 45.3 Class *dblite*

```

object └─ SCons.dblite.dblite

```

#### 45.3.1 Methods

`__init__(self, file_base_name, flag, mode)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature Overrides:  
`object.__init__` `exitit`(inherited documentation)

`close(self)`

`__del__(self)`

`sync(self)``__getitem__(self, key)``__setitem__(self, key, value)``keys(self)``has_key(self, key)``__contains__(self, key)``iterkeys(self)``__iter__(self)``__len__(self)`***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`,  
`__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`,  
`__str__()`, `__subclasshook__()`

**45.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 46 Module *SCons.exitfuncs*

*SCons.exitfuncs*

Register functions which are executed when *SCons* exits for any reason.

### 46.1 Functions

<b>register</b> ( <i>func</i> , * <i>targs</i> , ** <i>kargs</i> )
--

register a function to be executed upon normal program termination
--

func - function to be called at exit	targs - optional arguments to pass to func
--------------------------------------	--

kargs - optional keyword arguments to pass to func
--

### 46.2 Variables

Name	Description
__revision__	<b>Value:</b> 'src/engine/SCons/exitfuncs.py rel_2.4.1:3453:73fef3ea0b...
__package__	<b>Value:</b> 'SCons'

## Index

- SCons (*package*), 2–4
  - SCons.Action (*module*), 5–17
  - SCons.Builder (*module*), 18–29
    - SCons.Builder.Builder (*function*), 19
    - SCons.Builder.BuilderBase (*class*), 26–28
    - SCons.Builder.CallableSelector (*class*), 21–22
    - SCons.Builder.CompositeBuilder (*class*), 28–29
    - SCons.Builder.DictCmdGenerator (*class*), 20–21
    - SCons.Builder.DictEmitter (*class*), 22–23
    - SCons.Builder.EmitterProxy (*class*), 25–26
    - SCons.Builder.is\_a\_Builder (*function*), 19
    - SCons.Builder.ListEmitter (*class*), 23–24
    - SCons.Builder.match\_splitext (*function*), 19
    - SCons.Builder.OverrideWarner (*class*), 24–25
  - SCons.CacheDir (*module*), 30–32
  - SCons.Conftest (*module*), 33–37
  - SCons.cpp (*module*), 387–394
  - SCons.dblite (*module*), 395–396
    - SCons.dblite.corruption\_warning (*function*), 395
    - SCons.dblite.dblite (*class*), 395–396
    - SCons.dblite.is\_string (*function*), 395
    - SCons.dblite.open (*function*), 395
    - SCons.dblite.unicode (*function*), 395
  - SCons.Debug (*module*), 38–39
    - SCons.Debug.caller\_stack (*function*), 38
    - SCons.Debug.caller\_trace (*function*), 38
    - SCons.Debug.countLoggedInstances (*function*), 38
    - SCons.Debug.dump\_caller\_counts (*function*), 38
    - SCons.Debug.dumpLoggedInstances (*function*), 38
    - SCons.Debug.fetchLoggedInstances (*function*), 38
    - SCons.Debug.func\_shorten (*function*), 38
    - SCons.Debug.listLoggedInstances (*function*), 38
    - SCons.Debug.logInstanceCreation (*function*), 38
    - SCons.Debug.memory (*function*), 38
    - SCons.Debug.string\_to\_classes (*function*), 38
    - SCons.Debug.Trace (*function*), 38
  - SCons.Defaults (*module*), 40–44
  - SCons.Environment (*module*), 45–73
    - SCons.Environment.alias\_builder (*function*), 45
    - SCons.Environment.apply\_tools (*function*), 45
    - SCons.Environment.Base (*class*), 53–61, 64–73
    - SCons.Environment.BuilderDict (*class*), 48–49
    - SCons.Environment.BuilderWrapper (*class*), 47–48
    - SCons.Environment.copy\_non\_reserved\_keywords (*function*), 45
    - SCons.Environment.default\_copy\_from\_cache (*function*), 45
    - SCons.Environment.default\_decide\_source (*function*), 45
    - SCons.Environment.default\_decide\_target (*function*), 45
    - SCons.Environment.is\_valid\_construction\_var (*function*), 45
    - SCons.Environment.MethodWrapper (*class*), 46–47
    - SCons.Environment.NoSubstitutionProxy (*function*), 45
    - SCons.Environment.OverrideEnvironment (*class*), 61–64
    - SCons.Environment.SubstitutionEnvironment

- (*class*), 49–53
- SCons.Errors (*module*), 74–81
- SCons.Executor (*module*), 82–93
  - SCons.Executor.AddBatchExecutor (*function*), 82
  - SCons.Executor.Batch (*class*), 83
  - SCons.Executor.execute\_action\_list (*function*), 82
  - SCons.Executor.execute\_actions\_str (*function*), 82
  - SCons.Executor.execute\_nothing (*function*), 82
  - SCons.Executor.execute\_null\_str (*function*), 82
  - SCons.Executor.Executor (*class*), 86–90
  - SCons.Executor.get\_NullEnvironment (*function*), 82
  - SCons.Executor.GetBatchExecutor (*function*), 82
  - SCons.Executor.Null (*class*), 91–93
  - SCons.Executor.NullEnvironment (*class*), 90–91
  - SCons.Executor.rfile (*function*), 82
  - SCons.Executor.TSList (*class*), 83–85
  - SCons.Executor.TSObject (*class*), 85–86
- SCons.exitfuncs (*module*), 397
  - SCons.exitfuncs.register (*function*), 397
- SCons.Job (*module*), 94–101
  - SCons.Job.InterruptState (*class*), 94–95
  - SCons.Job.Jobs (*class*), 95–96
  - SCons.Job.Parallel (*class*), 100–101
  - SCons.Job.Serial (*class*), 96–97
  - SCons.Job.ThreadPool (*class*), 99–100
  - SCons.Job.Worker (*class*), 97–99
- SCons.Memoize (*module*), 102–107
  - SCons.Memoize.CountDict (*class*), 106–107
  - SCons.Memoize.CountDictCall (*function*), 104
  - SCons.Memoize.Counter (*class*), 104–105
  - SCons.Memoize.CountMethodCall (*function*), 103
  - SCons.Memoize.CountValue (*class*), 105–106
- SCons.Memoize.Dump (*function*), 103
- SCons.Memoize.EnableMemoization (*function*), 103
- SCons.Node (*package*), 108–130
  - SCons.Node.Alias (*module*), 131–136
  - SCons.Node.FS (*module*), 137–184
  - SCons.Node.Python (*module*), 185–190
- SCons.PathList (*module*), 191
  - SCons.PathList.node\_conv (*function*), 191
  - SCons.PathList.PathList (*function*), 191
- SCons.Scanner (*module*)
  - SCons.Scanner.Base (*class*), 216–219
  - SCons.Scanner.Classic (*class*), 226–229
  - SCons.Scanner.ClassicCPP (*class*), 229–230
  - SCons.Scanner.Current (*class*), 223–226
  - SCons.Scanner.FindPathDirs (*class*), 216
  - SCons.Scanner.Scanner (*function*), 215
  - SCons.Scanner.Selector (*class*), 219–223
- SCons.Scanner (*package*), 215–230
  - SCons.Scanner.C (*module*), 231–233
  - SCons.Scanner.D (*module*), 234–237
  - SCons.Scanner.Dir (*module*), 238–239
  - SCons.Scanner.Fortran (*module*), 240–244
  - SCons.Scanner.IDL (*module*), 245
  - SCons.Scanner.LaTeX (*module*), 246–252
  - SCons.Scanner.Prog (*module*), 253
  - SCons.Scanner.RC (*module*), 254
- SCons.SConf (*module*), 192–207
  - SCons.SConf.CheckCC (*function*), 193
  - SCons.SConf.CheckCHheader (*function*), 193
  - SCons.SConf.CheckContext (*class*), 205–207
  - SCons.SConf.CheckCXX (*function*), 193
  - SCons.SConf.CheckCXXHeader (*function*), 193
  - SCons.SConf.CheckDeclaration (*function*), 193
  - SCons.SConf.CheckFunc (*function*), 192
  - SCons.SConf.CheckHeader (*function*), 193

- SCons.SConf.CheckLib (*function*), 193
- SCons.SConf.CheckLibWithHeader (*function*), 193
- SCons.SConf.CheckProg (*function*), 194
- SCons.SConf.CheckSHCC (*function*), 193
- SCons.SConf.CheckSHCXX (*function*), 193
- SCons.SConf.CheckType (*function*), 192
- SCons.SConf.CheckTypeSize (*function*), 192
- SCons.SConf.ConfigureCacheError (*class*), 197–198
- SCons.SConf.ConfigureDryRunError (*class*), 196–197
- SCons.SConf.CreateConfigHBuilder (*function*), 192
- SCons.SConf.createIncludesFromHeaders (*function*), 193
- SCons.SConf.NeedConfigHBuilder (*function*), 192
- SCons.SConf.SConf (*function*), 192
- SCons.SConf.SConfBase (*class*), 202–205
- SCons.SConf.SConfBuildInfo (*class*), 198–200
- SCons.SConf.SConfBuildTask (*class*), 201–202
- SCons.SConf.SConfError (*class*), 195–196
- SCons.SConf.SConfWarning (*class*), 194–195
- SCons.SConf.SetBuildType (*function*), 192
- SCons.SConf.SetCacheMode (*function*), 192
- SCons.SConf.SetProgressDisplay (*function*), 192
- SCons.SConf.Streamer (*class*), 200–201
- SCons.SConsign (*module*), 208–214
  - SCons.SConsign.Base (*class*), 210–211
  - SCons.SConsign.corrupt\_\_dblite\_\_warning (*function*), 208
  - SCons.SConsign.DB (*class*), 211–212, 214
  - SCons.SConsign.Dir (*class*), 212
  - SCons.SConsign.DirFile (*class*), 212–214
  - SCons.SConsign.File (*function*), 208
  - SCons.SConsign.Get\_\_DataBase (*function*), 208
  - SCons.SConsign.Reset (*function*), 208
  - SCons.SConsign.SConsignEntry (*class*), 208–210
  - SCons.SConsign.write (*function*), 208
- SCons.Script (*module*)
  - SCons.Script.HelpFunction (*function*), 255
  - SCons.Script.Options (*function*), 255
  - SCons.Script.TargetList (*class*), 261–263
  - SCons.Script.Variables (*function*), 255
- SCons.Script (*package*), 255–263
  - SCons.Script.Interactive (*module*), 264–266
  - SCons.Script.Main (*module*), 267–280
  - SCons.Script.SConscript' (*module*), 281–286
- SCons.Sig (*module*), 287–289
  - SCons.Sig.MD5Null (*class*), 287–288
  - SCons.Sig.TimeStampNull (*class*), 288–289
- SCons.Subst (*module*), 290–300
- SCons.Taskmaster (*module*), 301–311
  - SCons.Taskmaster.AlwaysTask (*class*), 307–308
  - SCons.Taskmaster.dump\_stats (*function*), 301
  - SCons.Taskmaster.find\_cycle (*function*), 301
  - SCons.Taskmaster.OutOfDateTask (*class*), 308–309
  - SCons.Taskmaster.Stats (*class*), 302
  - SCons.Taskmaster.Task (*class*), 302–307
  - SCons.Taskmaster.Taskmaster (*class*), 309–311
- SCons.Util (*module*), 312–338
- SCons.Variables (*package*), 339–342
  - SCons.Variables.BoolVariable' (*module*), 343
  - SCons.Variables.EnumVariable' (*module*), 344–345
  - SCons.Variables.ListVariable' (*module*), 346

- SCons.Variables.PackageVariable' (*module*), 347
- SCons.Variables.PathVariable' (*module*), 348–349
- SCons.Variables.Variables (*class*), 339–342
- SCons.Warnings (*module*), 350–386
- SCons.Action.Action (*function*), 6
- SCons.Action.ActionBase (*class*), 6–7
  - SCons.Action.ActionBase.\_\_add\_\_ (*method*), 7
  - SCons.Action.ActionBase.\_\_cmp\_\_ (*method*), 7
  - SCons.Action.ActionBase.\_\_radd\_\_ (*method*), 7
  - SCons.Action.ActionBase.genstring (*method*), 7
  - SCons.Action.ActionBase.get\_contents (*method*), 7
  - SCons.Action.ActionBase.get\_targets (*method*), 7
  - SCons.Action.ActionBase.get\_varlist (*method*), 7
  - SCons.Action.ActionBase.no\_batch\_key (*method*), 7
  - SCons.Action.ActionBase.presub\_lines (*method*), 7
- SCons.Action.ActionCaller (*class*), 15–16
  - SCons.Action.ActionCaller.\_\_call\_\_ (*method*), 16
  - SCons.Action.ActionCaller.get\_contents (*method*), 16
  - SCons.Action.ActionCaller.strfunction (*method*), 16
  - SCons.Action.ActionCaller.subst (*method*), 16
  - SCons.Action.ActionCaller.subst\_args (*method*), 16
  - SCons.Action.ActionCaller.subst\_kw (*method*), 16
- SCons.Action.ActionFactory (*class*), 16–17
  - SCons.Action.ActionFactory.\_\_call\_\_ (*method*), 17
- SCons.Action.CommandAction (*class*), 7–9
  - SCons.Action.CommandAction.execute (*method*), 8
  - SCons.Action.CommandAction.get\_implicit\_deps (*method*), 9
  - SCons.Action.CommandAction.get\_presig (*method*), 8
  - SCons.Action.CommandAction.process (*method*), 8
  - SCons.Action.CommandAction.strfunction (*method*), 8
- SCons.Action.CommandGeneratorAction (*class*), 9–11
  - SCons.Action.CommandGeneratorAction.\_\_call\_\_ (*method*), 10
  - SCons.Action.CommandGeneratorAction.get\_implicit\_deps (*method*), 10
  - SCons.Action.CommandGeneratorAction.get\_presig (*method*), 10
  - SCons.Action.default\_exitstatfunc (*function*), 6
- SCons.Action.FunctionAction (*class*), 12–14
  - SCons.Action.FunctionAction.execute (*method*), 13
  - SCons.Action.FunctionAction.function\_name (*method*), 13
  - SCons.Action.FunctionAction.get\_implicit\_deps (*method*), 13
  - SCons.Action.FunctionAction.get\_presig (*method*), 13
  - SCons.Action.FunctionAction.strfunction (*method*), 13
  - SCons.Action.get\_default\_ENV (*function*), 6
- SCons.Action.LazyAction (*class*), 11–12
  - SCons.Action.LazyAction.get\_parent\_class (*method*), 11
- SCons.Action.ListAction (*class*), 14–15
  - SCons.Action.ListAction.\_\_call\_\_ (*method*), 15
  - SCons.Action.ListAction.get\_implicit\_deps (*method*), 15
  - SCons.Action.ListAction.get\_presig (*method*), 15
  - SCons.Action.remove\_set\_lineno\_codes (*function*), 6



- SCons.Action.rfile (*function*), 6  
 SCons.CacheDir.CacheDir (*class*), 30–32  
     SCons.CacheDir.CacheDir.CacheDebug (*method*), 31  
     SCons.CacheDir.CacheDir.cachepath (*method*), 31  
     SCons.CacheDir.CacheDir.is\_enabled (*method*), 31  
     SCons.CacheDir.CacheDir.is\_readonly (*method*), 31  
     SCons.CacheDir.CacheDir.push (*method*), 31  
     SCons.CacheDir.CacheDir.push\_if\_forced (*method*), 31  
     SCons.CacheDir.CacheDir.retrieve (*method*), 31  
 SCons.CacheDir.CachePushFunc (*function*), 30  
 SCons.CacheDir.CacheRetrieveFunc (*function*), 30  
 SCons.CacheDir.CacheRetrieveString (*function*), 30  
 SCons.Conftest.CheckBuilder (*function*), 33  
 SCons.Conftest.CheckCC (*function*), 33  
 SCons.Conftest.CheckCXX (*function*), 33  
 SCons.Conftest.CheckDeclaration (*function*), 35  
 SCons.Conftest.CheckFunc (*function*), 34  
 SCons.Conftest.CheckHeader (*function*), 34  
 SCons.Conftest.CheckLib (*function*), 36  
 SCons.Conftest.CheckProg (*function*), 36  
 SCons.Conftest.CheckSHCC (*function*), 33  
 SCons.Conftest.CheckSHCXX (*function*), 34  
 SCons.Conftest.CheckType (*function*), 34  
 SCons.Conftest.CheckTypeSize (*function*), 35  
 SCons.cpp.CPP\_to\_Python (*function*), 387  
 SCons.cpp.CPP\_to\_Python\_Ops\_Sub (*function*), 387  
 SCons.cpp.DumbPreProcessor (*class*), 393–394  
 SCons.cpp.FunctionEvaluator (*class*), 387–388  
     SCons.cpp.FunctionEvaluator.\_\_call\_\_ (*method*), 388  
 SCons.cpp.PreProcessor (*class*), 388–393  
     SCons.cpp.PreProcessor.\_\_call\_\_ (*method*), 389  
     SCons.cpp.PreProcessor.all\_include (*method*), 389  
     SCons.cpp.PreProcessor.do\_define (*method*), 389  
     SCons.cpp.PreProcessor.do\_elif (*method*), 389  
     SCons.cpp.PreProcessor.do\_else (*method*), 389  
     SCons.cpp.PreProcessor.do\_endif (*method*), 389  
     SCons.cpp.PreProcessor.do\_if (*method*), 389  
     SCons.cpp.PreProcessor.do\_ifdef (*method*), 390  
     SCons.cpp.PreProcessor.do\_ifndef (*method*), 390  
     SCons.cpp.PreProcessor.do\_import (*method*), 390  
     SCons.cpp.PreProcessor.do\_include (*method*), 390  
     SCons.cpp.PreProcessor.do\_nothing (*method*), 390  
     SCons.cpp.PreProcessor.do\_undef (*method*), 390  
     SCons.cpp.PreProcessor.eval\_expression (*method*), 390  
     SCons.cpp.PreProcessor.finalize\_result (*method*), 391  
     SCons.cpp.PreProcessor.find\_include\_file (*method*), 391  
     SCons.cpp.PreProcessor.initialize\_result (*method*), 391  
     SCons.cpp.PreProcessor.process\_contents (*method*), 391  
     SCons.cpp.PreProcessor.read\_file (*method*), 391  
     SCons.cpp.PreProcessor.resolve\_include (*method*), 391  
     SCons.cpp.PreProcessor.restore (*method*), 391  
     SCons.cpp.PreProcessor.save (*method*), 391  
     SCons.cpp.PreProcessor.scons\_current\_file (*method*), 392  
     SCons.cpp.PreProcessor.start\_handling\_includes

- (*method*), 392
- SCons.cpp.PreProcessor.stop\_handling\_includes (*method*), 113
- (*method*), 392
- SCons.cpp.PreProcessor.tupleize (*method*), 392
- SCons.Defaults.chmod\_func (*function*), 40
- SCons.Defaults.chmod\_strfunc (*function*), 40
- SCons.Defaults.copy\_func (*function*), 40
- SCons.Defaults.DefaultEnvironment (*function*), 40
- SCons.Defaults.delete\_func (*function*), 41
- SCons.Defaults.delete\_strfunc (*function*), 41
- SCons.Defaults.get\_paths\_str (*function*), 40
- SCons.Defaults.mkdir\_func (*function*), 41
- SCons.Defaults.move\_func (*function*), 41
- SCons.Defaults.NullCmdGenerator (*class*), 42–43
- SCons.Defaults.NullCmdGenerator.\_\_call (*method*), 43
- SCons.Defaults.processDefines (*function*), 41
- SCons.Defaults.SharedFlagChecker (*function*), 40
- SCons.Defaults.SharedObjectEmitter (*function*), 40
- SCons.Defaults.StaticObjectEmitter (*function*), 40
- SCons.Defaults.touch\_func (*function*), 41
- SCons.Defaults.Variable\_Method\_Caller (*class*), 43–44
- SCons.Defaults.Variable\_Method\_Caller.\_\_call (*method*), 44
- SCons.Errors.BuildError (*class*), 74–76
- SCons.Errors.convert\_to\_BuildError (*function*), 74
- SCons.Errors.EnvironmentError (*class*), 79
- SCons.Errors.ExplicitExit (*class*), 80–81
- SCons.Errors.InternalError (*class*), 76–77
- SCons.Errors.MSVCError (*class*), 79–80
- SCons.Errors.StopError (*class*), 78–79
- SCons.Errors.UserError (*class*), 77–78
- SCons.Node.Annotate (*function*), 108
- SCons.Node.BuildInfoBase (*class*), 113–114
- SCons.Node.BuildInfoBase.\_\_getstate\_\_ (*method*), 113
- SCons.Node.BuildInfoBase.\_\_setstate\_\_ (*method*), 113
- SCons.Node.BuildInfoBase.merge (*method*), 113
- SCons.Node.changed\_since\_last\_build\_alias (*function*), 110
- SCons.Node.changed\_since\_last\_build\_entry (*function*), 110
- SCons.Node.changed\_since\_last\_build\_node (*function*), 109
- SCons.Node.changed\_since\_last\_build\_python (*function*), 110
- SCons.Node.changed\_since\_last\_build\_state\_changed (*function*), 110
- SCons.Node.classname (*function*), 108
- SCons.Node.decide\_source (*function*), 110
- SCons.Node.decide\_target (*function*), 110
- SCons.Node.do\_nothing (*function*), 110
- SCons.Node.exists\_always (*function*), 108
- SCons.Node.exists\_base (*function*), 109
- SCons.Node.exists\_entry (*function*), 109
- SCons.Node.exists\_file (*function*), 109
- SCons.Node.exists\_none (*function*), 108
- SCons.Node.get\_children (*function*), 110
- SCons.Node.get\_contents\_dir (*function*), 109
- SCons.Node.get\_contents\_entry (*function*), 109
- SCons.Node.get\_contents\_file (*function*), 109
- SCons.Node.get\_contents\_none (*function*), 109
- SCons.Node.ignore\_cycle (*function*), 110
- SCons.Node.is\_derived\_node (*function*), 108
- SCons.Node.is\_derived\_none (*function*), 108
- SCons.Node.Node (*class*), 114–128
- SCons.Node.Node.add\_dependency (*method*), 115
- SCons.Node.Node.add\_ignore (*method*), 115
- SCons.Node.Node.add\_prerequisite (*method*), 115
- SCons.Node.Node.add\_source (*method*), 115
- SCons.Node.Node.add\_to\_implicit (*method*), 115
- SCons.Node.Node.add\_to\_waiting\_parents (*method*), 115

- SCons.Node.Node.add\_to\_waiting\_set (*method*), 116
- SCons.Node.Node.add\_wkid (*method*), 116
- SCons.Node.Node.all\_children (*method*), 116
- SCons.Node.Node.alter\_targets (*method*), 116
- SCons.Node.Node.build (*method*), 116
- SCons.Node.Node.builder\_set (*method*), 116
- SCons.Node.Node.built (*method*), 116
- SCons.Node.Node.changed (*method*), 117
- SCons.Node.Node.children (*method*), 117
- SCons.Node.Node.children\_are\_up\_to\_date (*method*), 117
- SCons.Node.Node.clear (*method*), 117
- SCons.Node.Node.clear\_memoized\_values (*method*), 118
- SCons.Node.Node.Decider (*method*), 115
- SCons.Node.Node.del\_binfo (*method*), 118
- SCons.Node.Node.disambiguate (*method*), 118
- SCons.Node.Node.env\_set (*method*), 118
- SCons.Node.Node.executor\_cleanup (*method*), 118
- SCons.Node.Node.exists (*method*), 118
- SCons.Node.Node.explain (*method*), 118
- SCons.Node.Node.for\_signature (*method*), 118
- SCons.Node.Node.get\_abspath (*method*), 118
- SCons.Node.Node.get\_binfo (*method*), 119
- SCons.Node.Node.get\_build\_env (*method*), 119
- SCons.Node.Node.get\_build\_scanner\_path (*method*), 119
- SCons.Node.Node.get\_builder (*method*), 119
- SCons.Node.Node.get\_cachedir\_csig (*method*), 119
- SCons.Node.Node.get\_contents (*method*), 119
- SCons.Node.Node.get\_csig (*method*), 119
- SCons.Node.Node.get\_env (*method*), 119
- SCons.Node.Node.get\_env\_scanner (*method*), 120
- SCons.Node.Node.get\_executor (*method*), 120
- SCons.Node.Node.get\_found\_includes (*method*), 120
- SCons.Node.Node.get\_implicit\_deps (*method*), 120
- SCons.Node.Node.get\_ninfo (*method*), 120
- SCons.Node.Node.get\_source\_scanner (*method*), 120
- SCons.Node.Node.get\_state (*method*), 120
- SCons.Node.Node.get\_stored\_implicit (*method*), 121
- SCons.Node.Node.get\_stored\_info (*method*), 121
- SCons.Node.Node.get\_string (*method*), 121
- SCons.Node.Node.get\_subst\_proxy (*method*), 121
- SCons.Node.Node.get\_suffix (*method*), 121
- SCons.Node.Node.get\_target\_scanner (*method*), 121
- SCons.Node.Node.GetTag (*method*), 115
- SCons.Node.Node.has\_builder (*method*), 121, 123
- SCons.Node.Node.has\_explicit\_builder (*method*), 122
- SCons.Node.Node.is\_derived (*method*), 122
- SCons.Node.Node.is\_literal (*method*), 122
- SCons.Node.Node.is\_up\_to\_date (*method*), 122
- SCons.Node.Node.make\_ready (*method*), 122
- SCons.Node.Node.missing (*method*), 123
- SCons.Node.Node.new\_binfo (*method*), 123
- SCons.Node.Node.new\_ninfo (*method*), 123
- SCons.Node.Node.postprocess (*method*), 123
- SCons.Node.Node.prepare (*method*), 123
- SCons.Node.Node.push\_to\_cache (*method*), 124
- SCons.Node.Node.release\_target\_info (*method*), 124
- SCons.Node.Node.remove (*method*), 124
- SCons.Node.Node.render\_include\_tree (*method*), 124

- 124
- SCons.Node.Node.reset\_executor (*method*), 125
- 125
- SCons.Node.Node.retrieve\_from\_cache (*method*), 125
- 125
- SCons.Node.Node.rexists (*method*), 125
- SCons.Node.Node.scan (*method*), 125
- SCons.Node.Node.scanner\_key (*method*), 125
- 125
- SCons.Node.Node.select\_scanner (*method*), 125
- 125
- SCons.Node.Node.set\_always\_build (*method*), 125
- 125
- SCons.Node.Node.set\_executor (*method*), 126
- 126
- SCons.Node.Node.set\_explicit (*method*), 126
- 126
- SCons.Node.Node.set\_nocache (*method*), 126
- 126
- SCons.Node.Node.set\_noclean (*method*), 126
- 126
- SCons.Node.Node.set\_precious (*method*), 126
- 126
- SCons.Node.Node.set\_pseudo (*method*), 126
- SCons.Node.Node.set\_specific\_source (*method*), 126
- 126
- SCons.Node.Node.set\_state (*method*), 126
- SCons.Node.Node.Tag (*method*), 115
- SCons.Node.Node.visited (*method*), 126
- SCons.Node.NodeInfoBase (*class*), 111–113
- SCons.Node.NodeInfoBase.\_\_getstate\_\_ (*method*), 112
- SCons.Node.NodeInfoBase.\_\_setstate\_\_ (*method*), 112
- SCons.Node.NodeInfoBase.convert (*method*), 112
- 112
- SCons.Node.NodeInfoBase.format (*method*), 112
- 112
- SCons.Node.NodeInfoBase.merge (*method*), 112
- 112
- SCons.Node.NodeInfoBase.update (*method*), 112
- 112
- SCons.Node.NodeList (*class*), 128–129
- SCons.Node.rexists\_base (*function*), 109
- SCons.Node.rexists\_node (*function*), 109
- SCons.Node.rexists\_none (*function*), 109
- SCons.Node.store\_info\_file (*function*), 110
- SCons.Node.store\_info\_pass (*function*), 110
- SCons.Node.target\_from\_source\_base (*function*), 109
- 109
- SCons.Node.target\_from\_source\_none (*function*), 109
- 109
- SCons.Node.Walker (*class*), 129–130
- SCons.Node.Walker.get\_next (*method*), 130
- SCons.Node.Walker.is\_done (*method*), 130
- SCons.Scanner.Dir.DirEntryScanner (*function*), 238
- 238
- SCons.Scanner.Dir.DirScanner (*function*), 238
- SCons.Scanner.Dir.do\_not\_scan (*function*), 238
- SCons.Scanner.Dir.only\_dirs (*function*), 238
- SCons.Scanner.Dir.scan\_in\_memory (*function*), 238
- 238
- SCons.Scanner.Dir.scan\_on\_disk (*function*), 238
- 238
- SCons.Script.Interactive.interact (*function*), 264
- SCons.Script.Interactive.SConsInteractiveCmd (*class*), 264–266
- 264–266
- SCons.Script.Interactive.SConsInteractiveCmd.do\_build (*method*), 265
- SCons.Script.Interactive.SConsInteractiveCmd.do\_clean (*method*), 265
- SCons.Script.Interactive.SConsInteractiveCmd.do\_EOF (*method*), 265
- SCons.Script.Interactive.SConsInteractiveCmd.do\_exit (*method*), 265
- SCons.Script.Interactive.SConsInteractiveCmd.do\_shell (*method*), 266
- SCons.Script.Interactive.SConsInteractiveCmd.do\_version (*method*), 266
- 266
- SCons.Subst.CmdStringHolder (*class*), 294–295
- 295
- SCons.Subst.CmdStringHolder.escape (*method*), 294
- 294
- SCons.Subst.CmdStringHolder.is\_literal (*method*), 294
- 294
- SCons.Subst.escape\_list (*function*), 290
- SCons.Subst.Literal (*class*), 292–293
- SCons.Subst.Literal.\_\_eq\_\_ (*method*), 292

- SCons.Subst.Literal.\_\_neq\_\_ (method), 292
- SCons.Subst.Literal.escape (method), 292
- SCons.Subst.Literal.for\_signature (method), 292
- SCons.Subst.Literal.is\_literal (method), 292
- SCons.Subst.NLWrapper (class), 295–296
- SCons.Subst.NullNodeList (class), 299–300
- SCons.Subst.quote\_spaces (function), 290
- SCons.Subst.raise\_exception (function), 290
- SCons.Subst.scons\_subst (function), 290
- SCons.Subst.scons\_subst\_list (function), 291
- SCons.Subst.scons\_subst\_once (function), 291
- SCons.Subst.SetAllowableExceptions (function), 290
- SCons.Subst.SpecialAttrWrapper (class), 293–294
- SCons.Subst.SpecialAttrWrapper.escape (method), 293
- SCons.Subst.SpecialAttrWrapper.for\_signature (method), 293
- SCons.Subst.SpecialAttrWrapper.is\_literal (method), 293
- SCons.Subst.subst\_dict (function), 290
- SCons.Subst.Target\_or\_Source (class), 298–299
- SCons.Subst.Target\_or\_Source.\_\_getattr\_\_ (method), 299
- SCons.Subst.Targets\_or\_Sources (class), 296–298
- SCons.Subst.Targets\_or\_Sources.\_\_getattr\_\_ (method), 297
- SCons.Util.\_NoError (class), 324–325
- SCons.Util.AddMethod (function), 317
- SCons.Util.adjustixes (function), 317
- SCons.Util.AppendPath (function), 316
- SCons.Util.case\_sensitive\_suffixes (function), 317
- SCons.Util.CLVar (class), 326–328
- SCons.Util.CLVar.\_\_coerce\_\_ (method), 327
- SCons.Util.containsAll (function), 312
- SCons.Util.containsAny (function), 312
- SCons.Util.containsOnly (function), 312
- SCons.Util.Delegate (class), 324
- SCons.Util.Delegate.\_\_get\_\_ (method), 324
- SCons.Util.dictify (function), 312
- SCons.Util.DisplayEngine (class), 321–322
- SCons.Util.DisplayEngine.\_\_call\_\_ (method), 322
- SCons.Util.DisplayEngine.set\_mode (method), 322
- SCons.Util.do\_flatten (function), 314
- SCons.Util.flatten (function), 314
- SCons.Util.flatten\_sequence (function), 314
- SCons.Util.get\_environment\_var (function), 312
- SCons.Util.get\_native\_path (function), 316
- SCons.Util.IDX (function), 313
- SCons.Util.is\_Dict (function), 313
- SCons.Util.is\_List (function), 313
- SCons.Util.is\_Scalar (function), 314
- SCons.Util.is\_Sequence (function), 313
- SCons.Util.is\_String (function), 314
- SCons.Util.is\_Tuple (function), 314
- SCons.Util.LogicalLines (class), 330–331
- SCons.Util.LogicalLines.readline (method), 331
- SCons.Util.LogicalLines.readlines (method), 331
- SCons.Util.make\_path\_relative (function), 317
- SCons.Util.MD5collect (function), 318
- SCons.Util.MD5filesignature (function), 318
- SCons.Util.MD5signature (function), 318
- SCons.Util.NodeList (class), 319–321
- SCons.Util.NodeList.\_\_call\_\_ (method), 320
- SCons.Util.NodeList.\_\_getattr\_\_ (method), 320
- SCons.Util.NodeList.\_\_nonzero\_\_ (method), 320
- SCons.Util.Null (class), 336–337
- SCons.Util.Null.\_\_call\_\_ (method), 337
- SCons.Util.Null.\_\_getattr\_\_ (method), 337
- SCons.Util.Null.\_\_nonzero\_\_ (method), 337
- SCons.Util.NullSeq (class), 337–338



- SCons.Util.NullSeq.\_\_delitem\_\_ (method), 338
- SCons.Util.NullSeq.\_\_getitem\_\_ (method), 338
- SCons.Util.NullSeq.\_\_iter\_\_ (method), 338
- SCons.Util.NullSeq.\_\_len\_\_ (method), 338
- SCons.Util.NullSeq.\_\_setitem\_\_ (method), 338
- SCons.Util.OrderedDict (class), 328–330
- SCons.Util.PrependPath (function), 315
- SCons.Util.print\_tree (function), 313
- SCons.Util.Proxy (class), 322–324
  - SCons.Util.Proxy.\_\_cmp\_\_ (method), 323
  - SCons.Util.Proxy.\_\_getattr\_\_ (method), 323
  - SCons.Util.Proxy.get (method), 323
- SCons.Util.RegGetValue (function), 315
- SCons.Util.RegOpenKeyEx (function), 315
- SCons.Util.RenameFunction (function), 318
- SCons.Util.render\_tree (function), 313
- SCons.Util.rightmost\_separator (function), 312
- SCons.Util.Selector (class), 330
  - SCons.Util.Selector.\_\_call\_\_ (method), 330
- SCons.Util.semi\_deepcopy (function), 315
- SCons.Util.semi\_deepcopy\_dict (function), 315
- SCons.Util.silent\_intern (function), 318
- SCons.Util.Split (function), 316
- SCons.Util.splitext (function), 312
- SCons.Util.to\_String (function), 315
- SCons.Util.to\_String\_for\_signature (function), 315
- SCons.Util.to\_String\_for\_subst (function), 315
- SCons.Util.Unbuffered (class), 335–336
  - SCons.Util.Unbuffered.\_\_getattr\_\_ (method), 336
  - SCons.Util.Unbuffered.write (method), 336
- SCons.Util.unique (function), 317
- SCons.Util.UniqueList (class), 331–335
- SCons.Util.uniquer (function), 317
- SCons.Util.uniquer\_hashables (function), 317
- SCons.Util.updrive (function), 312
- SCons.Util.WhereIs (function), 315
- SCons.Util.WindowsError (class), 325–326
- SCons.Warnings.CacheWriteErrorWarning (class), 354–355
- SCons.Warnings.CorruptSConsignWarning (class), 355–356
- SCons.Warnings.DependencyWarning (class), 356–357
- SCons.Warnings.DeprecatedBuildDirWarning (class), 377–378
- SCons.Warnings.DeprecatedBuilderKeywordsWarning (class), 385–386
- SCons.Warnings.DeprecatedCopyWarning (class), 379–380
- SCons.Warnings.DeprecatedDebugOptionsWarning (class), 383–384
- SCons.Warnings.DeprecatedOptionsWarning (class), 380–381
- SCons.Warnings.DeprecatedSigModuleWarning (class), 384–385
- SCons.Warnings.DeprecatedSourceCodeWarning (class), 376–377
- SCons.Warnings.DeprecatedSourceSignaturesWarning (class), 381–382
- SCons.Warnings.DeprecatedTargetSignaturesWarning (class), 382–383
- SCons.Warnings.DeprecatedWarning (class), 373–374
- SCons.Warnings.DevelopmentVersionWarning (class), 357–358
- SCons.Warnings.DuplicateEnvironmentWarning (class), 358–359
- SCons.Warnings.enableWarningClass (function), 350
- SCons.Warnings.FortranCxxMixWarning (class), 371–372
- SCons.Warnings.FutureDeprecatedWarning (class), 372–373
- SCons.Warnings.FutureReservedVariableWarning (class), 359
- SCons.Warnings.LinkWarning (class), 359–360
- SCons.Warnings.MandatoryDeprecatedWarning (class), 374–375
- SCons.Warnings.MisleadingKeywordsWarning (class), 360–361

SCons.Warnings.MissingSConscriptWarning (*class*),  
361–362

SCons.Warnings.NoMD5ModuleWarning (*class*),  
362–363

SCons.Warnings.NoMetaclassSupportWarning  
(*class*), 363–364

SCons.Warnings.NoObjectCountWarning (*class*),  
364–365

SCons.Warnings.NoParallelSupportWarning (*class*),  
365–366

SCons.Warnings.process\_warn\_strings (*func-*  
*tion*), 350

SCons.Warnings.PythonVersionWarning (*class*),  
375–376

SCons.Warnings.ReservedVariableWarning (*class*),  
366–367

SCons.Warnings.StackSizeWarning (*class*), 367–  
368

SCons.Warnings.suppressWarningClass (*func-*  
*tion*), 350

SCons.Warnings.TargetNotBuiltWarning (*class*),  
353–354

SCons.Warnings.TaskmasterNeedsExecuteWarning  
(*class*), 378–379

SCons.Warnings.VisualCMissingWarning (*class*),  
368–369

SCons.Warnings.VisualStudioMissingWarning  
(*class*), 370–371

SCons.Warnings.VisualStudioVersionMismatch (*class*),  
369–370

SCons.Warnings.warn (*function*), 350

SCons.Warnings.Warning (*class*), 351–352

SCons.Warnings.warningAsException (*func-*  
*tion*), 350

SCons.Warnings.WarningOnByDefault (*class*),  
352–353