

SCons

API Documentation

July 31, 2015

Contents

Contents	1
1 Package SCons	2
1.1 Modules	2
1.2 Variables	4
2 Module SCons.Action	5
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
3 Module SCons.Builder	18
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
3.9.3	Class Variables	28
3.10	Class CompositeBuilder	29
3.10.1	Methods	29
3.10.2	Properties	29
4	Module SCons.CacheDir	30
4.1	Functions	30
4.2	Variables	30
4.3	Class CacheDir	30
4.3.1	Methods	31
4.3.2	Properties	32
5	Module SCons.Conftest	33
5.1	Functions	33
5.2	Variables	37
6	Module SCons.Debug	38
6.1	Functions	38
6.2	Variables	38
7	Module SCons.Defaults	40
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

8	Module SCons.Environment	45
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
8.6.3	Class Variables	53
8.7	Class Base	53
8.7.1	Methods	53
8.7.2	Properties	61
8.7.3	Class Variables	61
8.8	Class OverrideEnvironment	61
8.8.1	Methods	62
8.8.2	Properties	64
8.8.3	Class Variables	64
8.9	Class Base	65
8.9.1	Methods	65
8.9.2	Properties	73
8.9.3	Class Variables	73
9	Module SCons.Errors	74
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 MSVCErrror	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
10	Module SCons.Executor	82
10.1	Functions	82
10.2	Variables	82
10.3	Class Batch	82
10.3.1	Methods	83
10.3.2	Properties	83
10.4	Class TList	83
10.4.1	Methods	84
10.4.2	Properties	85
10.4.3	Class Variables	85
10.5	Class TObject	85
10.5.1	Methods	85
10.5.2	Properties	86
10.6	Class Executor	86
10.6.1	Methods	86
10.6.2	Properties	90
10.6.3	Class Variables	90
10.6.4	Instance Variables	90
10.7	Class NullEnvironment	90
10.7.1	Methods	90
10.7.2	Properties	91
10.8	Class Null	91
10.8.1	Methods	91
10.8.2	Properties	92
11	Module SCons.Job	93
11.1	Variables	93
11.2	Class InterruptState	93
11.2.1	Methods	93
11.2.2	Properties	94
11.3	Class Jobs	94
11.3.1	Methods	94
11.3.2	Properties	95
11.4	Class Serial	95
11.4.1	Methods	95
11.4.2	Properties	96
11.5	Class Worker	96
11.5.1	Methods	97
11.5.2	Properties	98
11.6	Class ThreadPool	98
11.6.1	Methods	98
11.6.2	Properties	99
11.7	Class Parallel	99
11.7.1	Methods	100
11.7.2	Properties	100
12	Module SCons.Memoize	101
12.1	Functions	103
12.2	Variables	103
12.3	Class Counter	103

12.3.1	Methods	104
12.3.2	Properties	104
12.4	Class CountValue	104
12.4.1	Methods	105
12.4.2	Properties	105
12.5	Class CountDict	105
12.5.1	Methods	106
12.5.2	Properties	106
12.6	Class Memoizer	106
12.6.1	Methods	106
12.6.2	Properties	107
12.7	Class Memoized_Metaclass	107
12.7.1	Methods	107
12.7.2	Properties	107
13	Package SCons.Node	109
13.1	Modules	109
13.2	Functions	109
13.3	Variables	109
13.4	Class NodeInfoBase	110
13.4.1	Methods	110
13.4.2	Properties	111
13.4.3	Class Variables	111
13.5	Class BuildInfoBase	111
13.5.1	Methods	112
13.5.2	Properties	112
13.5.3	Class Variables	112
13.6	Class Node	112
13.6.1	Methods	112
13.6.2	Properties	126
13.6.3	Class Variables	126
13.7	Class NodeList	126
13.7.1	Methods	126
13.7.2	Properties	127
13.7.3	Class Variables	127
13.8	Class Walker	127
13.8.1	Methods	128
13.8.2	Properties	128
14	Module SCons.Node.Alias	129
14.1	Variables	129
14.2	Class AliasNameSpace	129
14.2.1	Methods	129
14.2.2	Class Variables	129
14.3	Class AliasNodeInfo	130
14.3.1	Methods	130
14.3.2	Properties	130
14.3.3	Class Variables	130
14.4	Class AliasBuildInfo	131
14.4.1	Methods	131
14.4.2	Properties	131
14.4.3	Class Variables	131

14.5	Class Alias	132
14.5.1	Methods	132
14.5.2	Properties	134
14.5.3	Class Variables	134
15	Module SCons.Node.FS	136
15.1	Functions	136
15.2	Variables	138
15.3	Class EntryProxyAttributeError	139
15.3.1	Methods	139
15.3.2	Properties	140
15.4	Class DiskChecker	140
15.4.1	Methods	140
15.4.2	Properties	141
15.5	Class EntryProxy	141
15.5.1	Methods	141
15.5.2	Properties	141
15.5.3	Class Variables	142
15.6	Class Base	142
15.6.1	Methods	142
15.6.2	Properties	146
15.6.3	Class Variables	146
15.6.4	Instance Variables	147
15.7	Class Entry	147
15.7.1	Methods	147
15.7.2	Properties	150
15.7.3	Class Variables	150
15.7.4	Instance Variables	150
15.8	Class LocalFS	150
15.8.1	Methods	151
15.8.2	Properties	152
15.8.3	Class Variables	152
15.9	Class FS	152
15.9.1	Methods	152
15.9.2	Properties	155
15.9.3	Class Variables	155
15.10	Class DirNodeInfo	155
15.10.1	Methods	155
15.10.2	Properties	156
15.10.3	Class Variables	156
15.11	Class DirBuildInfo	156
15.11.1	Methods	156
15.11.2	Properties	157
15.11.3	Class Variables	157
15.12	Class Dir	157
15.12.1	Methods	157
15.12.2	Properties	165
15.12.3	Class Variables	165
15.12.4	Instance Variables	165
15.13	Class RootDir	166
15.13.1	Methods	166
15.13.2	Properties	168

15.13.3 Class Variables	168
15.13.4 Instance Variables	168
15.14 Class FileNodeInfo	169
15.14.1 Methods	169
15.14.2 Properties	169
15.14.3 Class Variables	169
15.15 Class FileBuildInfo	170
15.15.1 Methods	170
15.15.2 Properties	171
15.15.3 Class Variables	171
15.16 Class File	171
15.16.1 Methods	172
15.16.2 Properties	179
15.16.3 Class Variables	179
15.16.4 Instance Variables	179
15.17 Class FileFinder	180
15.17.1 Methods	180
15.17.2 Properties	181
15.17.3 Class Variables	181
16 Module SCons.Node.Python	182
16.1 Variables	182
16.2 Class ValueNodeInfo	182
16.2.1 Methods	182
16.2.2 Properties	182
16.2.3 Class Variables	183
16.3 Class ValueBuildInfo	183
16.3.1 Methods	183
16.3.2 Properties	183
16.3.3 Class Variables	184
16.4 Class Value	184
16.4.1 Methods	184
16.4.2 Properties	187
16.4.3 Class Variables	187
17 Module SCons.PathList	188
17.1 Functions	188
17.2 Variables	188
18 Module SCons.SConf	189
18.1 Functions	189
18.2 Variables	191
18.3 Class SConfWarning	191
18.3.1 Methods	192
18.3.2 Properties	192
18.4 Class SConfError	192
18.4.1 Methods	193
18.4.2 Properties	193
18.5 Class ConfigureDryRunError	193
18.5.1 Methods	194
18.5.2 Properties	194
18.6 Class ConfigureCacheError	195

18.6.1	Methods	195
18.6.2	Properties	195
18.7	Class SConfBuildInfo	196
18.7.1	Methods	196
18.7.2	Properties	196
18.7.3	Class Variables	196
18.8	Class Streamer	197
18.8.1	Methods	197
18.8.2	Properties	197
18.9	Class SConfBuildTask	198
18.9.1	Methods	198
18.9.2	Properties	199
18.10	Class SConfBase	199
18.10.1	Methods	200
18.10.2	Properties	202
18.11	Class CheckContext	202
18.11.1	Methods	203
18.11.2	Properties	204
19	Module SCons.SConsign	205
19.1	Functions	205
19.2	Variables	205
19.3	Class SConsignEntry	206
19.3.1	Methods	206
19.3.2	Properties	206
19.3.3	Class Variables	206
19.4	Class Base	207
19.4.1	Methods	207
19.4.2	Properties	208
19.5	Class DB	208
19.5.1	Methods	208
19.5.2	Properties	208
19.6	Class Dir	209
19.6.1	Methods	209
19.6.2	Properties	209
19.7	Class DirFile	209
19.7.1	Methods	210
19.7.2	Properties	210
19.8	Class DB	211
19.8.1	Methods	211
19.8.2	Properties	211
20	Package SCons.Scanner	212
20.1	Modules	212
20.2	Functions	212
20.3	Variables	212
20.4	Class FindPathDirs	213
20.4.1	Methods	213
20.4.2	Properties	213
20.5	Class Base	213
20.5.1	Methods	214
20.5.2	Properties	216

20.6	Class Selector	216
20.6.1	Methods	219
20.6.2	Properties	220
20.7	Class Current	220
20.7.1	Methods	222
20.7.2	Properties	223
20.8	Class Classic	223
20.8.1	Methods	225
20.8.2	Properties	226
20.9	Class ClassicCPP	226
20.9.1	Methods	227
20.9.2	Properties	227
21	Module SCons.Scanner.C	228
21.1	Functions	228
21.2	Variables	228
21.3	Class SConsCPPScanner	228
21.3.1	Methods	229
21.3.2	Properties	229
21.4	Class SConsCPPScannerWrapper	230
21.4.1	Methods	230
21.4.2	Properties	230
22	Module SCons.Scanner.D	231
22.1	Functions	231
22.2	Variables	231
22.3	Class D	231
22.3.1	Methods	233
22.3.2	Properties	234
23	Module SCons.Scanner.Dir	235
23.1	Functions	235
23.2	Variables	235
24	Module SCons.Scanner.Fortran	237
24.1	Functions	237
24.2	Variables	237
24.3	Class F90Scanner	237
24.3.1	Methods	240
24.3.2	Properties	241
25	Module SCons.Scanner.IDL	242
25.1	Functions	242
25.2	Variables	242
26	Module SCons.Scanner.LaTeX	243
26.1	Functions	243
26.2	Variables	243
26.3	Class FindENVPPathDirs	243
26.3.1	Methods	244
26.3.2	Properties	244
26.4	Class LaTeX	244

26.4.1	Methods	247
26.4.2	Properties	248
26.4.3	Class Variables	248
27	Module SCons.Scanner.Prog	250
27.1	Functions	250
27.2	Variables	250
28	Module SCons.Scanner.RC	251
28.1	Functions	251
28.2	Variables	251
29	Package SCons.Script	252
29.1	Modules	252
29.2	Functions	252
29.3	Variables	252
29.4	Class TargetList	259
29.4.1	Methods	259
29.4.2	Properties	260
29.4.3	Class Variables	260
30	Module SCons.Script.Interactive	261
30.1	Functions	261
30.2	Variables	261
30.3	Class SConsInteractiveCmd	261
30.3.1	Methods	262
30.3.2	Class Variables	263
31	Module SCons.Script.Main	264
31.1	Functions	264
31.2	Variables	265
31.3	Class SConsPrintHelpException	266
31.3.1	Methods	266
31.3.2	Properties	266
31.4	Class Progressor	267
31.4.1	Methods	267
31.4.2	Properties	267
31.4.3	Class Variables	267
31.5	Class BuildTask	268
31.5.1	Methods	268
31.5.2	Properties	270
31.5.3	Class Variables	270
31.6	Class CleanTask	270
31.6.1	Methods	271
31.6.2	Properties	272
31.7	Class QuestionTask	272
31.7.1	Methods	273
31.7.2	Properties	274
31.8	Class TreePrinter	274
31.8.1	Methods	274
31.8.2	Properties	274
31.9	Class FakeOptionParser	275

31.9.1	Methods	275
31.9.2	Properties	275
31.9.3	Class Variables	275
31.10	Class Stats	275
31.10.1	Methods	276
31.10.2	Properties	276
31.11	Class CountStats	276
31.11.1	Methods	276
31.11.2	Properties	277
31.12	Class MemStats	277
31.12.1	Methods	277
31.12.2	Properties	277
32	Module SCons.Script.SConscript'	278
32.1	Functions	278
32.2	Variables	279
32.3	Class SConscriptReturn	279
32.3.1	Methods	279
32.3.2	Properties	280
32.4	Class Frame	280
32.4.1	Methods	280
32.4.2	Properties	280
32.5	Class SConsEnvironment	281
32.5.1	Methods	281
32.5.2	Properties	282
32.5.3	Class Variables	283
32.6	Class DefaultEnvironmentCall	283
32.6.1	Methods	283
32.6.2	Properties	283
33	Module SCons.Sig	284
33.1	Variables	284
33.2	Class MD5Null	284
33.2.1	Methods	284
33.2.2	Properties	285
33.3	Class TimeStampNull	285
33.3.1	Methods	285
33.3.2	Properties	285
34	Module SCons.Subst	287
34.1	Functions	287
34.2	Variables	288
34.3	Class Literal	289
34.3.1	Methods	289
34.3.2	Properties	290
34.4	Class SpecialAttrWrapper	290
34.4.1	Methods	290
34.4.2	Properties	291
34.5	Class CmdStringHolder	291
34.5.1	Methods	291
34.5.2	Properties	292
34.5.3	Class Variables	292

34.6	Class NLWrapper	293
34.6.1	Methods	293
34.6.2	Properties	293
34.7	Class Targets_or_Sources	294
34.7.1	Methods	294
34.7.2	Properties	295
34.7.3	Class Variables	295
34.8	Class Target_or_Source	296
34.8.1	Methods	296
34.8.2	Properties	296
34.9	Class NullNodeList	297
34.9.1	Methods	297
34.9.2	Properties	297
35	Module SCons.Taskmaster	298
35.1	Functions	298
35.2	Variables	298
35.3	Class Stats	299
35.3.1	Methods	299
35.3.2	Properties	299
35.4	Class Task	300
35.4.1	Methods	300
35.4.2	Properties	304
35.5	Class AlwaysTask	304
35.5.1	Methods	305
35.5.2	Properties	305
35.6	Class OutOfDateTask	305
35.6.1	Methods	306
35.6.2	Properties	306
35.7	Class Taskmaster	306
35.7.1	Methods	306
35.7.2	Properties	308
36	Module SCons.Util	309
36.1	Functions	309
36.2	Variables	315
36.3	Class NodeList	317
36.3.1	Methods	317
36.3.2	Properties	318
36.3.3	Class Variables	318
36.4	Class DisplayEngine	318
36.4.1	Methods	319
36.4.2	Properties	319
36.4.3	Class Variables	319
36.5	Class Proxy	319
36.5.1	Methods	320
36.5.2	Properties	320
36.6	Class Delegate	321
36.6.1	Methods	321
36.6.2	Properties	321
36.7	Class _NoError	321
36.7.1	Methods	322

36.7.2 Properties	322
36.8 Class WindowsError	322
36.8.1 Methods	322
36.8.2 Properties	323
36.9 Class CLVar	324
36.9.1 Methods	324
36.9.2 Properties	325
36.9.3 Class Variables	325
36.10 Class OrderedDict	325
36.10.1 Methods	326
36.10.2 Class Variables	327
36.11 Class Selector	327
36.11.1 Methods	327
36.11.2 Class Variables	327
36.12 Class LogicalLines	328
36.12.1 Methods	328
36.12.2 Properties	328
36.13 Class UniqueList	329
36.13.1 Methods	329
36.13.2 Properties	332
36.13.3 Class Variables	332
36.14 Class Unbuffered	332
36.14.1 Methods	333
36.14.2 Properties	333
36.15 Class Null	333
36.15.1 Methods	333
36.15.2 Properties	334
36.16 Class NullSeq	335
36.16.1 Methods	335
36.16.2 Properties	335
37 Package SCons.Variables	336
37.1 Modules	336
37.2 Variables	336
37.3 Class Variables	336
37.3.1 Methods	337
37.3.2 Properties	339
37.3.3 Class Variables	339
38 Module SCons.Variables.BoolVariable'	340
38.1 Functions	340
39 Module SCons.Variables.EnumVariable'	341
39.1 Functions	342
40 Module SCons.Variables.ListVariable'	343
40.1 Functions	343
41 Module SCons.Variables.PackageVariable'	344
41.1 Functions	344
42 Module SCons.Variables.PathVariable'	345

42.1	Variables	346
43	Module SCons.Warnings	347
43.1	Functions	347
43.2	Variables	348
43.3	Class Warning	348
43.3.1	Methods	349
43.3.2	Properties	349
43.4	Class WarningOnByDefault	349
43.4.1	Methods	350
43.4.2	Properties	350
43.5	Class TargetNotBuiltWarning	350
43.5.1	Methods	350
43.5.2	Properties	351
43.6	Class CacheWriteErrorWarning	351
43.6.1	Methods	351
43.6.2	Properties	352
43.7	Class CorruptSConsignWarning	352
43.7.1	Methods	352
43.7.2	Properties	353
43.8	Class DependencyWarning	353
43.8.1	Methods	353
43.8.2	Properties	353
43.9	Class DevelopmentVersionWarning	354
43.9.1	Methods	354
43.9.2	Properties	354
43.10	Class DuplicateEnvironmentWarning	355
43.10.1	Methods	355
43.10.2	Properties	355
43.11	Class FutureReservedVariableWarning	356
43.11.1	Methods	356
43.11.2	Properties	356
43.12	Class LinkWarning	357
43.12.1	Methods	357
43.12.2	Properties	357
43.13	Class MisleadingKeywordsWarning	358
43.13.1	Methods	358
43.13.2	Properties	358
43.14	Class MissingSConscriptWarning	359
43.14.1	Methods	359
43.14.2	Properties	359
43.15	Class NoMD5ModuleWarning	360
43.15.1	Methods	360
43.15.2	Properties	360
43.16	Class NoMetaclassSupportWarning	361
43.16.1	Methods	361
43.16.2	Properties	361
43.17	Class NoObjectCountWarning	362
43.17.1	Methods	362
43.17.2	Properties	362
43.18	Class NoParallelSupportWarning	363
43.18.1	Methods	363

43.18.2 Properties	363
43.19 Class ReservedVariableWarning	364
43.19.1 Methods	364
43.19.2 Properties	364
43.20 Class StackSizeWarning	365
43.20.1 Methods	365
43.20.2 Properties	365
43.21 Class VisualCMissingWarning	366
43.21.1 Methods	366
43.21.2 Properties	366
43.22 Class VisualVersionMismatch	367
43.22.1 Methods	367
43.22.2 Properties	367
43.23 Class VisualStudioMissingWarning	368
43.23.1 Methods	368
43.23.2 Properties	368
43.24 Class FortranCxxMixWarning	369
43.24.1 Methods	369
43.24.2 Properties	369
43.25 Class FutureDeprecatedWarning	370
43.25.1 Methods	370
43.25.2 Properties	370
43.26 Class DeprecatedWarning	371
43.26.1 Methods	371
43.26.2 Properties	371
43.27 Class MandatoryDeprecatedWarning	372
43.27.1 Methods	372
43.27.2 Properties	372
43.28 Class PythonVersionWarning	373
43.28.1 Methods	373
43.28.2 Properties	373
43.29 Class DeprecatedSourceCodeWarning	374
43.29.1 Methods	374
43.29.2 Properties	374
43.30 Class DeprecatedBuildDirWarning	375
43.30.1 Methods	375
43.30.2 Properties	375
43.31 Class TaskmasterNeedsExecuteWarning	376
43.31.1 Methods	376
43.31.2 Properties	376
43.32 Class DeprecatedCopyWarning	377
43.32.1 Methods	377
43.32.2 Properties	377
43.33 Class DeprecatedOptionsWarning	378
43.33.1 Methods	378
43.33.2 Properties	378
43.34 Class DeprecatedSourceSignaturesWarning	379
43.34.1 Methods	379
43.34.2 Properties	379
43.35 Class DeprecatedTargetSignaturesWarning	380
43.35.1 Methods	380

43.35.2 Properties	380
43.36 Class DeprecatedDebugOptionsWarning	381
43.36.1 Methods	381
43.36.2 Properties	381
43.37 Class DeprecatedSigModuleWarning	382
43.37.1 Methods	382
43.37.2 Properties	382
43.38 Class DeprecatedBuilderKeywordsWarning	383
43.38.1 Methods	383
43.38.2 Properties	383
44 Module SCons.cpp	384
44.1 Functions	384
44.2 Variables	384
44.3 Class FunctionEvaluator	385
44.3.1 Methods	385
44.3.2 Properties	385
44.4 Class PreProcessor	385
44.4.1 Methods	386
44.4.2 Properties	390
44.5 Class DumbPreProcessor	390
44.5.1 Methods	390
44.5.2 Properties	390
45 Module SCons.dblite	392
45.1 Functions	392
45.2 Variables	392
45.3 Class dblite	392
45.3.1 Methods	392
45.3.2 Properties	393
46 Module SCons.exitfuncs	394
46.1 Functions	394
46.2 Variables	394

1 Package SCons

SCons

The main package for the SCons software construction utility. **Version:** 2.3.6

Date: 2015/07/31 14:36:10

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. 93)
- **Memoize:** Memoizer
(Section 12, p. 101)
- **Node:** SCons.Node
(Section 13, p. 109)
 - **Alias:** scons.Node.Alias
(Section 14, p. 129)
 - **FS:** scons.Node.FS
(Section 15, p. 136)
 - **Python:** scons.Node.Python
(Section 16, p. 182)
- **PathList:** SCons.PathList
(Section 17, p. 188)
- **SConf:** SCons.SConf
(Section 18, p. 189)
- **SConsign:** SCons.SConsign
(Section 19, p. 205)
- **Scanner:** SCons.Scanner
(Section 20, p. 212)
 - **C:** SCons.Scanner.C
(Section 21, p. 228)
 - **D:** SCons.Scanner.D

- (Section 22, p. 231)
 - **Dir** (Section 23, p. 235)
 - **Fortran**: SCons.Scanner.Fortran
(Section 24, p. 237)
 - **IDL**: SCons.Scanner.IDL
(Section 25, p. 242)
 - **LaTeX**: SCons.Scanner.LaTeX
(Section 26, p. 243)
 - **Prog** (Section 27, p. 250)
 - **RC**: SCons.Scanner.RC
(Section 28, p. 251)
- **Script**: SCons.Script
(Section 29, p. 252)
 - **Interactive**: SCons interactive mode
(Section 30, p. 261)
 - **Main**: SCons.Script
(Section 31, p. 264)
 - **SConscript'**: SCons.Script.SConscript
(Section 32, p. 278)
- **Sig**: Place-holder for the old SCons.Sig module hierarchy
(Section 33, p. 284)
- **Subst**: SCons.Subst
(Section 34, p. 287)
- **Taskmaster**: Generic Taskmaster module for the SCons build engine.
(Section 35, p. 298)
- **Util**: SCons.Util
(Section 36, p. 309)
- **Variables**: engine.SCons.Variables
(Section 37, p. 336)
 - **BoolVariable** (Section ??, p. ??)
 - **BoolVariable'**: engine.SCons.Variables.BoolVariable
(Section 38, p. 340)
 - **EnumVariable** (Section ??, p. ??)
 - **EnumVariable'**: engine.SCons.Variables.EnumVariable
(Section 39, p. 341)
 - **ListVariable** (Section ??, p. ??)
 - **ListVariable'**: engine.SCons.Variables.ListVariable
(Section 40, p. 343)
 - **PackageVariable** (Section ??, p. ??)
 - **PackageVariable'**: engine.SCons.Variables.PackageVariable
(Section 41, p. 344)
 - **PathVariable** (Section ??, p. ??)
 - **PathVariable'**: SCons.Variables.PathVariable
(Section 42, p. 345)
- **Warnings**: SCons.Warnings
(Section 43, p. 347)
- **cpp**: SCons C Pre-Processor module
(Section 44, p. 384)
- **dblite** (Section 45, p. 392)
- **exitfuncs**: SCons.exitfuncs
(Section 46, p. 394)

1.2 Variables

Name	Description
__build__	Value: 'rel_2.3.5:3347:d31d5a4e74b6 [MODIFIED]'
__buildsys__	Value: 'hpmicrodog'
__developer__	Value: 'bdbaddog'
__package__	Value: 'SCons'
__revision__	Value: 'src/engine/SCons/__init__.py rel_2.3.5:3347:d31d5a4e74b6...'

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

<code>rfile(<i>n</i>)</code>

<code>default_exitstatfunc(<i>s</i>)</code>

<code>remove_set_lineno_codes(<i>x</i>)</code>
--

<code>Action(<i>act</i>, *<i>args</i>, **<i>kw</i>)</code>
--

A factory for action objects.

<code>get_default_ENV(<i>env</i>)</code>
--

2.2 Variables

Name	Description
<code>__revision__</code>	Value: 'src/engine/SCons/Action.py rel_2.3.5:3347:d31d5a4e74b6 2...
<code>print_actions</code>	Value: 1
<code>execute_actions</code>	Value: 1
<code>print_actions_presub</code>	Value: 0
<code>SET_LINENO</code>	Value: <code>dis.SET_LINENO</code>
<code>HAVE_ARGUMENT</code>	Value: <code>dis.HAVE_ARGUMENT</code>
<code>strip_quotes</code>	Value: <code>re.compile(r'^[\'](.*)[\']\$')</code>
<code>default_ENV</code>	Value: None
<code>__package__</code>	Value: '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>
<hr/> Returns the type of targets (\$TARGETS, \$CHANGED_TARGETS) used by this action.

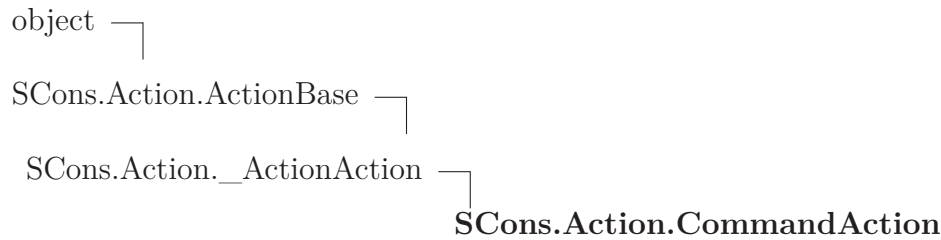
Inherited from object

`__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>	
<code>__class__</code>	

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)

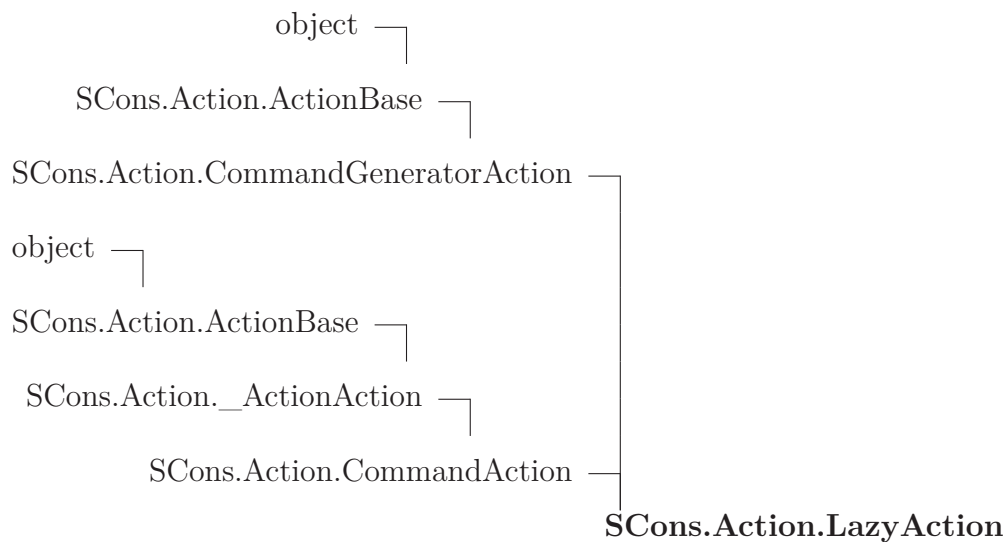
`__add__()`, `__cmp__()`, `__radd__()`, `get_contents()`, `no_batch_key()`, `pre_sub_lines()`

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**

<p><code>__init__(self, var, kw)</code></p> <p><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)</p>
--

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

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

Overrides: SCons.Action._ActionAction.__call__
--

<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: SCons.Action.CommandAction.get_presig
 exitit(inherited documentation)

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

Overrides: SCons.Action.ActionBase.get_varlist
--

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()`

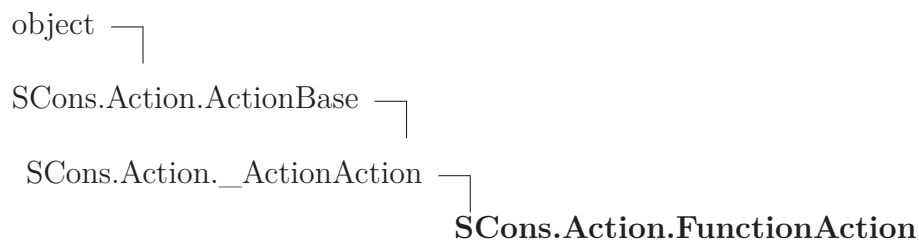
Inherited from object

`__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__(), print_cmd_line()
```

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

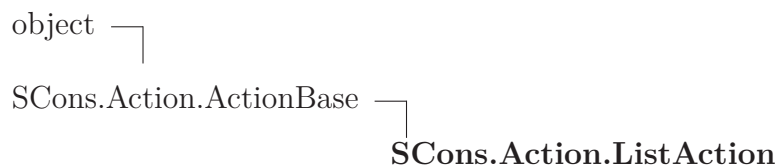
```
__add__(), __cmp__(), __radd__(), batch_key(), genstring(), get_contents(),  
get_targets(), get_varlist(), no_batch_key(), presub_lines()
```

Inherited from object

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

2.7.2 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

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

<code>genstring</code> (<i>self</i> , <i>target</i> , <i>source</i> , <i>env</i>) Overrides: <code>SCons.Action.ActionBase.genstring</code>

<code>__str__</code> (<i>self</i>) <code>str(x)</code> Overrides: <code>object.__str__</code> extit(inherited documentation)
--

<code>presub_lines</code> (<i>self</i> , <i>env</i>) Overrides: <code>SCons.Action.ActionBase.presub_lines</code>

```
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

<code>__init__(self, parent, args, kw)</code>

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

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

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

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

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

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

<code>strfunction(self, target, source, env)</code>

<code>__str__(self)</code>

<code>str(x)</code> Overrides: <code>object.__str__</code> <code>__exit__(inherited documentation)</code>

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
0x7f056d4e9de8>)

```

```

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>	
<code>__class__</code>	

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>
--

Builder (** <i>kw</i>)
<hr/>
A factory for builder objects.

is_a_Builder(*obj*)

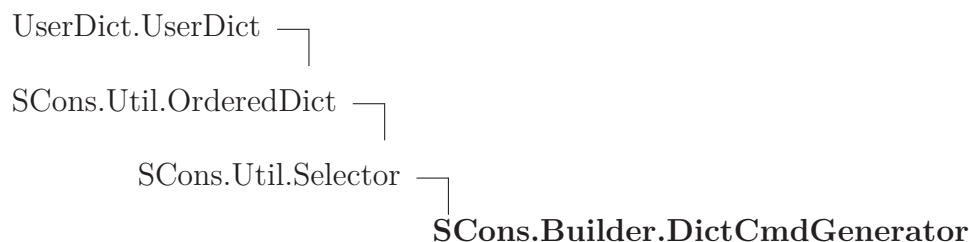
"Returns True if the specified obj is one of our Builder classes.

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>	Value: 'src/engine/SCons/Builder.py rel_2.3.5:3347:d31d5a4e74b6 ...
<code>misleading_keywords</code>	Value: {'sources': 'source', 'targets': 'target'}
<code>__package__</code>	Value: 'SCons'

3.3 Class 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*)

add_action (<i>self</i> , <i>suffix</i> , <i>action</i>)
Add a suffix-action pair to the mapping.

__call__ (<i>self</i> , <i>target</i> , <i>source</i> , <i>env</i> , <i>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()`

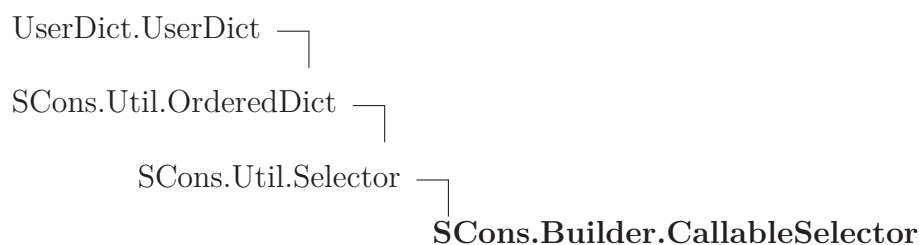
Inherited from UserDict.UserDict

`__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>	
<code>__hash__</code>	

3.4 Class CallableSelector



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

3.4.1 Methods

__call__ (<i>self</i> , <i>env</i> , <i>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()`

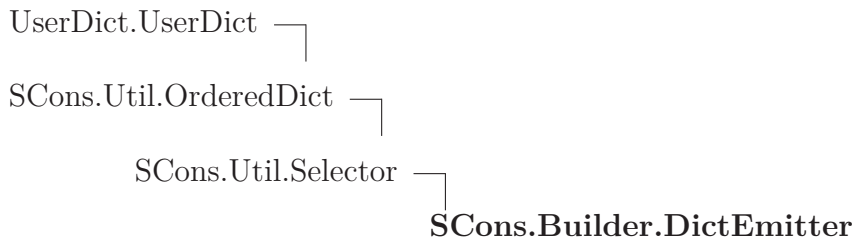
Inherited from UserDict.UserDict

`__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()`

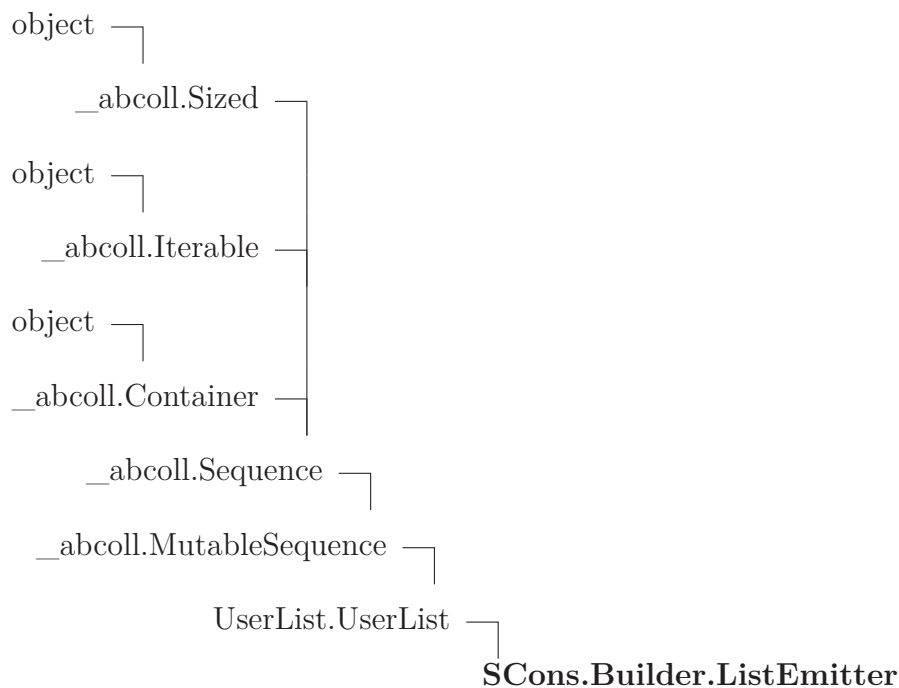
Inherited from UserDict.UserDict

`__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__()`, `ap-`
`pend()`, `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 UserList.UserList</i> <code>__abstractmethods__</code> , <code>__hash__</code>	

3.7 Class 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: <code>UserDict.UserDict.__init__</code>
<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

object —
 SCons.Builder.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__ (<i>self</i> , <i>var</i>) x. __init__ (...) initializes x; see help(type(x)) for signature Overrides: object. __init__ exitit(inherited documentation)

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

__cmp__ (<i>self</i> , <i>other</i>)

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__ exitit(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={})
```

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

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.

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

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.

<code>src_suffixes(self, env)</code>

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>	
<code>__class__</code>	

3.9.3 Class Variables

Name	Description
<code>__metaclass__</code>	Value: <code>SCons.Memoize.Memoized_Metaclass</code>
<code>memoizer_counters</code>	Value: <code>[]</code>

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__` `exit`(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

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

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

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

4.2 Variables

Name	Description
<code>__revision__</code>	Value: 'src/engine/SCons/CacheDir.py rel_2.3.5:3347:d31d5a4e74b6...
<code>__doc__</code>	Value: ...
<code>cache_enabled</code>	Value: True
<code>cache_debug</code>	Value: False
<code>cache_force</code>	Value: False
<code>cache_show</code>	Value: False
<code>cache_readonly</code>	Value: False
<code>CacheRetrieve</code>	Value: SCons.Action.Action(CacheRetrieveFunc, CacheRetrieveString)
<code>CacheRetrieveSilent</code>	Value: SCons.Action.Action(CacheRetrieveFunc, None)
<code>CachePush</code>	Value: SCons.Action.Action(CachePushFunc, None)
<code>__package__</code>	Value: '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.

5.2 Variables

Name	Description
LogInputFiles	Value: 1
LogErrorMessage	Value: 1
__package__	Value: '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
__revision__	Value: 'src/engine/SCons/Debug.py rel_2.3.5:3347:d31d5a4e74b6 20...
track_instances	Value: False
tracked_classes	Value: {}
caller_bases	Value: {}
caller_dicts	Value: {}
shorten_list	Value: [('/scons/SCons/', 1), ('/src/engine/SCons/', 1), ('usr/...
TraceFP	Value: {}
TraceDefault	Value: '/dev/tty'
TimeStampDefault	Value: None
StartTime	Value: 1438378854.13
PreviousTime	Value: 1438378854.13
__package__	Value: '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
<code>__revision__</code>	Value: 'src/engine/SCons/Defaults.py rel_2.3.5:3347:d31d5a4e74b6...
SharedCheck	Value: SCons.Action.Action(SharedFlagChecker, None)
CScan	Value: SCons.Defaults.CScan
DScan	Value: SCons.Tool.DScanner
LaTeXScan	Value: SCons.Tool.LaTeXScanner
ObjSourceScan	Value: SCons.Tool.SourceFileScanner
ProgScan	Value: SCons.Tool.ProgramScanner
DirScanner	Value: SCons.Defaults.DirScanner
DirEntryScanner	Value: SCons.Scanner.Dir.DirEntryScanner()
CAction	Value: SCons.Action.Action("\$CCCOM", "\$CCCOMSTR")

continued on next page

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

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>	
<code>__class__</code>	

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>

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>__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>	Value: 'src/engine/SCons/Environment.py rel_2.3.5:3347:d31d5a4e7...
<code>CleanTargets</code>	Value: {}
<code>CalculatorArgs</code>	Value: {}
<code>AliasBuilder</code>	Value: <code>SCons.Builder.Builder(action= alias_builder, target_facto...</code>

continued on next page

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

8.3 Class 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

```

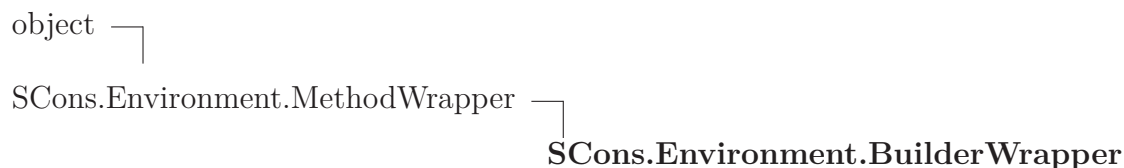
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

```

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

```

__call__(self, target=None, source=<class
'SCons.Environment._Null'>, *args, **kw)
Overrides: SCons.Environment.MethodWrapper.__call__

```

```

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

```

__str__ (<i>self</i>) str(x) Overrides: object.__str__ extit(inherited documentation)

__getattr__ (<i>self, name</i>)
--

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

Inherited from SCons.Environment.MethodWrapper(Section 8.3)

__init__(*self*), clone()

Inherited from object

__delattr__(*self, name*), **__format__**(*self, format_spec*), **__getattr__**(*self, name*), **__hash__**(*self*), **__new__**(*cls, *args, **kwargs*), **__reduce__**(*self*), **__reduce_ex__**(*self, protocol*), **__sizeof__**(*self*), **__subclasshook__**(*self, subclass, bases*)

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__ (<i>self, dict, env</i>) Overrides: UserDict.UserDict.__init__

__semi_deepcopy__ (<i>self</i>)
--

<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 construction 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

<code>__init__(self, **kw)</code>
Initialization of an underlying SubstitutionEnvironment class. Overrides: object.__init__
<code>__cmp__(self, other)</code>
<code>__delitem__(self, key)</code>
<code>__getitem__(self, key)</code>
<code>__setitem__(self, key, value)</code>
<code>get(self, key, default=None)</code>
Emulates the get() method of dictionaries.
<code>has_key(self, key)</code>
<code>__contains__(self, key)</code>
<code>items(self)</code>
<code>arg2nodes(self, args, node_factory=<class 'SCons.Environment._Null'>, lookup_list=<class 'SCons.Environment._Null'>, **kw)</code>
<code>gvars(self)</code>
<code>lvars(self)</code>

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>	

8.6.3 Class Variables

Name	Description
<code>__metaclass__</code>	Value: <code>SCons.Memoize.Memoized_MetaClass</code>

8.7 Class Base



Known Subclasses: `SCons.Environment.OverrideEnvironment`, `SCons.Script.SConsScript!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 (<i>self</i> , *args, **kw)
--

AddPostAction (<i>self</i> , files, action)

AddPreAction (<i>self</i> , files, action)
--

Alias (<i>self</i> , 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 scon's 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*)

<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__()`

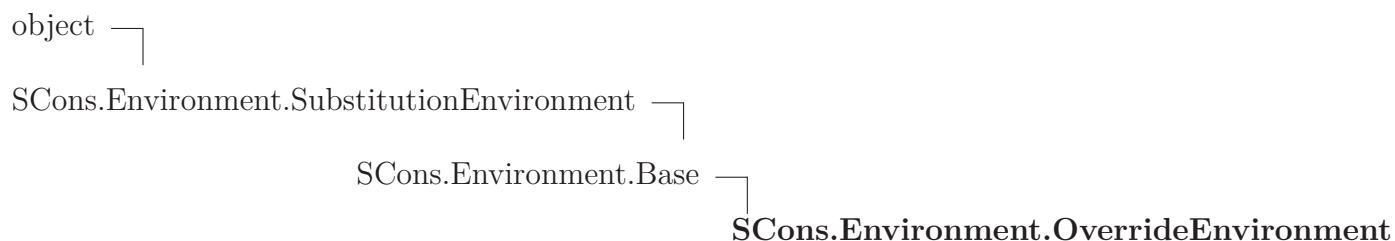
8.7.2 Properties

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

8.7.3 Class Variables

Name	Description
<code>memoizer_counters</code>	Value: []
<i>Inherited from <code>SCons.Environment.SubstitutionEnvironment</code> (Section 8.6)</i>	
<code>__metaclass__</code>	

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 exit(inherited documentation)

Inherited from SCons.Environment.Base(Section 8.9)

Action(), AddPostAction(), AddPreAction(), Alias(), AlwaysBuild(), Append(), AppendENVPath(), AppendUnique(), BuildDir(), Builder(), CacheDir(), Clean(), Clone(), Command(), Configure(), Copy(), Decider(), Depends(), Detect(), Dir(), Dump(), Entry(), Environment(), Execute(), File(), FindFile(), FindInstalledFiles(), FindIxex(), FindSourceFiles(), Flatten(), GetBuildPath(), Glob(), Ignore(), Literal(), Local(), NoCache(), NoClean(), ParseConfig(), ParseDepends(), Platform(), Precious(), Prepend(), PrependENVPath(), PrependUnique(), Pseudo(), ReplaceIxex(), 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__(), __subclasshook__()

8.8.2 Properties

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

8.8.3 Class Variables

Name	Description
<i>Inherited from SCons.Environment.Base (Section 8.9)</i>	
memoizer_counters	
<i>Inherited from SCons.Environment.SubstitutionEnvironment (Section 8.6)</i>	
__metaclass__	

8.9 Class 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 scon's 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*)

<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>	
<code>__class__</code>	

8.9.3 Class Variables

Name	Description
memoizer_counters	Value: []
<i>Inherited from <i>SCons.Environment.SubstitutionEnvironment</i> (Section 8.6)</i>	
<code>__metaclass__</code>	

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)
```

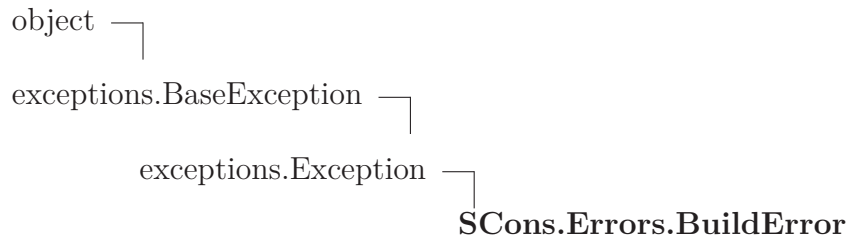
Convert any return code a BuildError Exception.

'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__	Value: 'src/engine/SCons/Errors.py rel_2.3.5:3347:d31d5a4e74b6 2...
__package__	Value: '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__` `exit`(inherited documentation)

```
__str__(self)
```

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

Inherited from `exceptions.Exception`

```
__new__()
```

Inherited from `exceptions.BaseException`

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

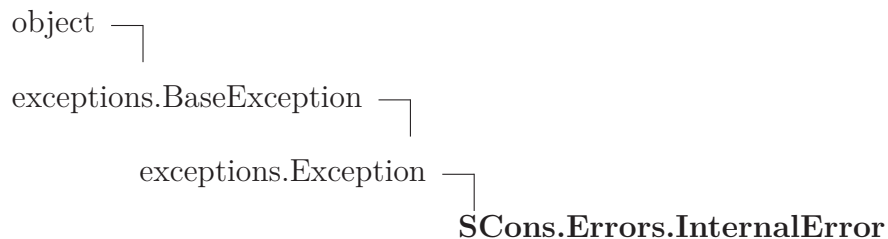
Inherited from `object`

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

9.3.2 Properties

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

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__()`

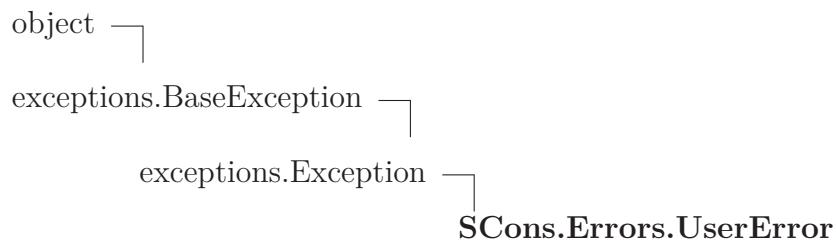
Inherited from `object`

`__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__()`

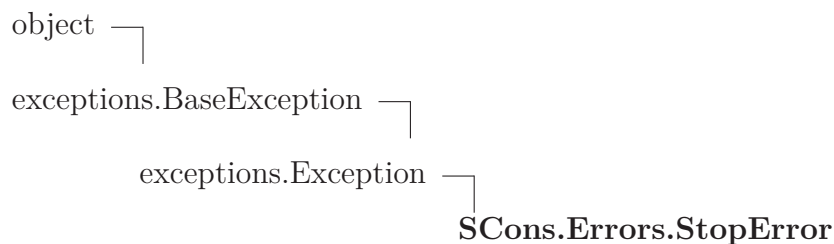
Inherited from object

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

9.5.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</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__()`

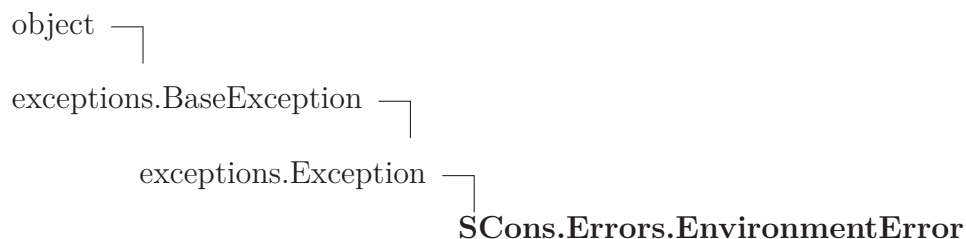
Inherited from object

`__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__()

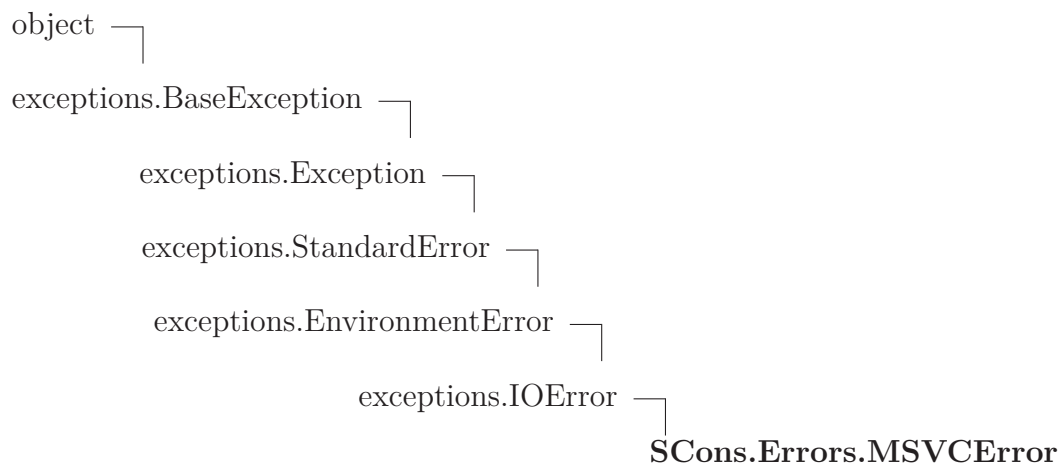
Inherited from object

__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 MSVCErr



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__()`

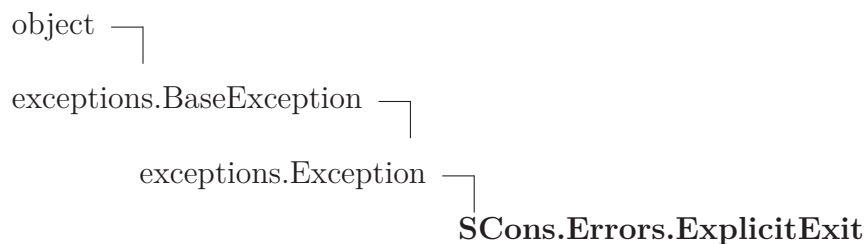
Inherited from object

`__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__` `exit`(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 <code>exceptions.BaseException</code></i>	<code>args</code> , <code>message</code>
<i>Inherited from <code>object</code></i>	<code>__class__</code>

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.).

GetBatchExecutor(*key*)

AddBatchExecutor(*key*, *executor*)

get_NullEnvironment()

Use singleton pattern for Null Environments.

10.2 Variables

Name	Description
__revision__	Value: 'src/engine/SCons/Executor.py rel_2.3.5:3347:d31d5a4e74b6...
nullenv	Value: None
__package__	Value: 'SCons'

10.3 Class Batch

object —
SCons.Executor.Batch

Remembers exact association between targets and sources of executor.

10.3.1 Methods

```
__init__(self, targets=[], sources=[])
```

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

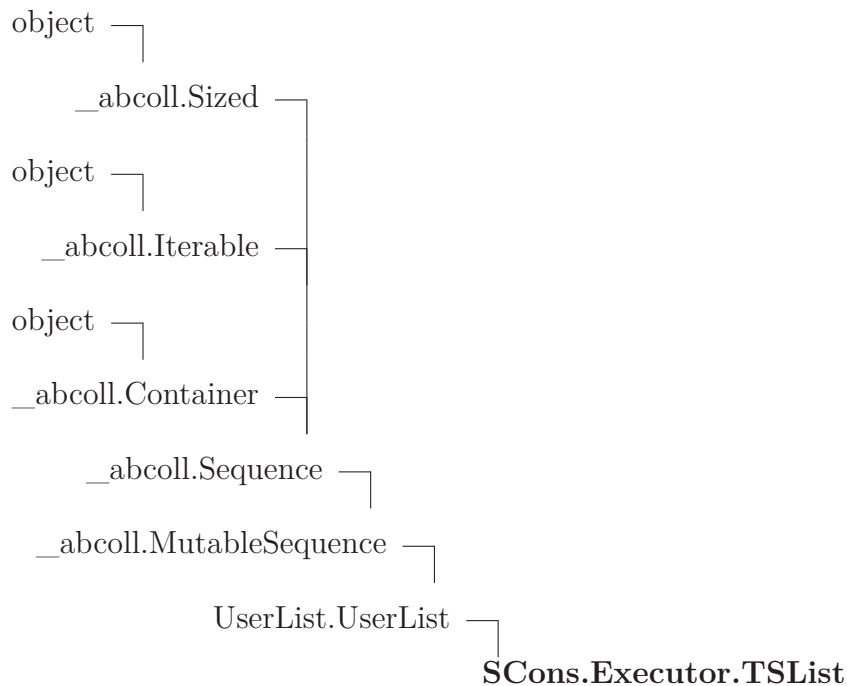
Inherited from object

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

10.3.2 Properties

Name	Description
<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__` `exitit`(inherited documentation)

`__getattr__(self, attr)`

`__getitem__(self, i)`

Overrides: `_abcoll.Sequence.__getitem__`

`__getslice__(self, i, j)`

Overrides: `UserList.UserList.__getslice__`

`__str__(self)`

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

`__repr__(self)`

`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

object —
SCons.Executor.TSOBJECT

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

10.5.1 Methods

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

<p><code>__getattr__(self, attr)</code></p>

<p><code>__str__(self)</code></p> <p>str(x) Overrides: object.<code>__str__</code> extit(inherited documentation)</p>

```
__repr__(self)
```

```
repr(x) Overrides: object.__repr__ extit(inherited documentation)
```

Inherited from object

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

10.5.2 Properties

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

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__ extit(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={})
```

```
do_nothing(self, target, kw)
```

```
do_execute(self, target, kw)
```

Actually execute the action list.

```
__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
<i>Inherited from object</i>	
<code>__class__</code>	

10.6.3 Class Variables

Name	Description
<code>memoizer_counters</code>	Value: []

10.6.4 Instance Variables

Name	Description
<code>my_str</code>	Value: <function my_str at 0x7f056d45a1b8>

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__()`, `__getattr__()`, `__hash__()`, `__reduce__()`, `__reduce_ex__()`,
`__sizeof__()`, `__str__()`, `__subclasshook__()`

10.7.2 Properties

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

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_targets(self)
```

```
get_action_list(self)
```

```
get_all_targets(self)
```

<code>get_all_sources(<i>self</i>)</code>

<code>get_all_children(<i>self</i>)</code>
--

<code>get_all_prerequisites(<i>self</i>)</code>

<code>get_action_side_effects(<i>self</i>)</code>

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

<code>get_contents(<i>self</i>)</code>
--

<code>add_pre_action(<i>self</i>, <i>action</i>)</code>

<code>add_post_action(<i>self</i>, <i>action</i>)</code>
--

<code>set_action_list(<i>self</i>, <i>action</i>)</code>
--

Inherited from object

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

10.8.2 Properties

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

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>	Value: 'src/engine/SCons/Job.py rel_2.3.5:3347:d31d5a4e74b6 2015...
<code>explicit_stack_size</code>	Value: None
<code>default_stack_size</code>	Value: 256
<code>interrupt_msg</code>	Value: 'Build interrupted.'
<code>__package__</code>	Value: 'SCons'

11.2 Class InterruptState

object —
 SCons.Job.InterruptState

11.2.1 Methods

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

<code>set</code> (<i>self</i>)

<code>__call__</code> (<i>self</i>)
--

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

<p>__init__(<i>self</i>, <i>num</i>, <i>taskmaster</i>)</p> <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>
--

<p>run(<i>self</i>, <i>postfunc</i>=<function <lambda> at 0x7f056d2728c0>)</p> <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>

were_interrupted (<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>	
__class__	

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

__init__ (<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.

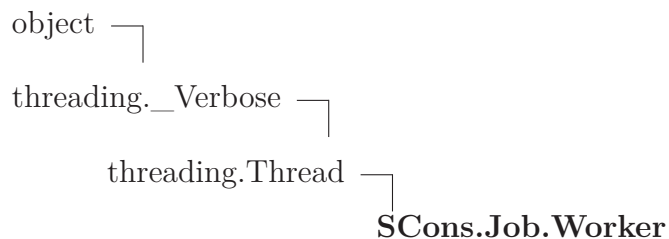
Inherited from object

`__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

<code>__init__(self, num, stack_size, interrupted)</code>
<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>

<code>put(self, task)</code>
<p>Put task into request queue.</p>

<code>get(self)</code>
<p>Remove and return a result tuple from the results queue.</p>

<code>preparation_failed(self, task)</code>

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> <code>__class__</code>	

11.7 Class 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

<code>__init__(self, taskmaster, num, stack_size)</code>
<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>

<code>start(self)</code>
<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 metaclass 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. Here is an example of wrapping a method that returns a computed value, with no input parameters:

```
memoizer_counters = [] # Memoization

memoizer_counters.append(SCons.Memoize.CountValue('foo')) # Memoization

def foo(self):

    try: # Memoization
        return self._memo['foo'] # Memoization
    except KeyError: # Memoization
        pass # Memoization

    result = self.compute_foo_value()

    self._memo['foo'] = result # Memoization

    return result
```

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): # Memoization
    return argument # Memoization

memoizer_counters.append(SCons.Memoize.CountDict('bar', _bar_key)) # Memoization

def bar(self, argument):

    memo_key = argument # Memoization
    try: # Memoization
        memo_dict = self._memo['bar'] # Memoization
    except KeyError: # Memoization
        memo_dict = {} # Memoization
    self._memo['dict'] = memo_dict # Memoization
```

```

else:
    try:
        return memo_dict[memo_key]
    except KeyError:
        pass

result = self.compute_bar_value(argument)

memo_dict[memo_key] = result

return result

```

At one point we avoided replicating this sort of logic in all the methods by putting it right into this module, but we've moved away from that at present (see the "Historical Note," below.).

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.

Historical Note: The initial Memoizer implementation actually handled the caching of values for the wrapped methods, based on a set of generic algorithms for computing hashable values based on the method's arguments. This collected caching logic nicely, but had two drawbacks:

Running arguments through a generic key-conversion mechanism is slower (and less flexible) than just coding these things directly. Since the

methods that need memoized values are generally performance-critical, slowing them down in order to collect the logic isn't the right tradeoff.

Use of the memoizer really obscured what was being called, because all the memoized methods were wrapped with re-used generic methods. This made it more difficult, for example, to use the Python profiler to figure out how to optimize the underlying methods.

12.1 Functions

<code>Dump(title=None)</code>

<code>EnableMemoization()</code>

12.2 Variables

Name	Description
<code>__revision__</code>	Value: 'src/engine/SCons/Memoize.py rel_2.3.5:3347:d31d5a4e74b6 ...
<code>__doc__</code>	Value: ""Memoi...
<code>use_memoizer</code>	Value: None
<code>CounterList</code>	Value: []
<code>__package__</code>	Value: 'SCons'

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 metaclass initialization will have filled in the `.name` attribute that represents the name of the function being counted.

12.3.1 Methods

```
__init__(self, method_name)
```

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

```
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>	
__class__	

12.4 Class CountValue



A counter class for simple, atomic memoized values.

A CountValue object should be instantiated in a class for each of the class's methods that memoizes its return value by simply storing the return value in its `__memo` dictionary.

We expect that the metaclass initialization will fill in the `.underlying_method` attribute with the method that we're wrapping. We then call the `underlying_method` method after counting whether its memoized value has already been set (a hit) or not (a miss).

12.4.1 Methods

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

Inherited from SCons.Memoize.Counter(Section 12.3)

`__cmp__()`, `__init__()`, `display()`

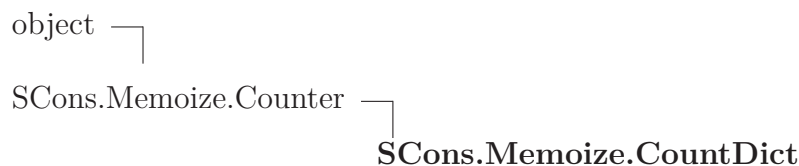
Inherited from object

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

12.4.2 Properties

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

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 class 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.

We expect that the metaclass initialization will fill in the `.underlying_method` attribute with the method that we're wrapping. We then call the `underlying_method` method after counting whether the computed key value is already present in the memoization dictionary (a hit) or not (a miss).

12.5.1 Methods

<code>__init__(self, method_name, keymaker)</code>
--

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

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

Inherited from SCons.Memoize.Counter(Section 12.3)

`__cmp__()`, `display()`

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>	

12.6 Class Memoizer

```

object ┌
      │
      └─ SCons.Memoize.Memoizer
  
```

Object which performs caching of method calls for its 'primary' instance.

12.6.1 Methods

<code>__init__(self)</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__()`

12.6.2 Properties

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

12.7 Class Memoized_Metaclass



12.7.1 Methods

<code>__init__(cls, name, bases, cls_dict)</code>
x. <code>__init__</code> (...) initializes x; see <code>help(type(x))</code> for signature Return Value the object's type
Overrides: <code>object.__init__</code> <code>__init__</code> <code>__exit__</code> (inherited documentation)

Inherited from type

`__call__()`, `__delattr__()`, `__eq__()`, `__ge__()`, `__getattr__()`, `__gt__()`,
`__hash__()`, `__instancecheck__()`, `__le__()`, `__lt__()`, `__ne__()`, `__new__()`,
`__repr__()`, `__setattr__()`, `__subclasscheck__()`, `__subclasses__()`, `mro()`

Inherited from object

`__format__()`, `__reduce__()`, `__reduce_ex__()`, `__sizeof__()`, `__str__()`,
`__subclasshook__()`

12.7.2 Properties

Name	Description
<i>Inherited from type</i>	

continued on next page

Name	Description
__abstractmethods__, __base__, __bases__, __basicsize__, __dictoffset__, __flags__, __itemsized__, __mro__, __name__, __weakrefoffset__	
<i>Inherited from object</i>	
__class__	

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. 129)
- **FS:** `scons.Node.FS`
(Section 15, p. 136)
- **Python:** `scons.Node.Python`
(Section 16, p. 182)

13.2 Functions

<code>classname(obj)</code>

<code>Annotate(node)</code>

<code>get_children(node, parent)</code>

<code>ignore_cycle(node, stack)</code>
--

<code>do_nothing(node, parent)</code>

13.3 Variables

Name	Description
<code>__revision__</code>	Value: 'src/engine/SCons/Node/__init__.py rel_2.3.5:3347:d31d5a4...'
<code>do_store_info</code>	Value: True
<code>no_state</code>	Value: 0
<code>pending</code>	Value: 1
<code>executing</code>	Value: 2
<code>up_to_date</code>	Value: 3
<code>executed</code>	Value: 4
<code>failed</code>	Value: 5
<code>StateString</code>	Value: {0: 'no_state', 1: 'pending', 2: 'executing', 3: 'up_to_d...'
<code>implicit_cache</code>	Value: 0
<code>implicit_deps_unchanged</code>	Value: 0
<code>implicit_deps_changed</code>	Value: 0
<code>interactive</code>	Value: False
<code>arg2nodes_lookups</code>	Value: [<bound method AliasNameSpace.lookup of {}>]
<code>__package__</code>	Value: '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

```
__init__(self, node=None)
```

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

<code>convert(self, node, val)</code>

<code>format(self, field_list=None, names=0)</code>

<code>merge(self, other)</code>

<code>update(self, node)</code>

Inherited from object

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

13.4.2 Properties

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

13.4.3 Class Variables

Name	Description
<code>current_version_id</code>	Value: 1

13.5 Class BuildInfoBase

```

object
└── SCons.Node.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

```
__init__(self, node=None)
```

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

```
merge(self, other)
```

Inherited from object

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

13.5.2 Properties

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

13.5.3 Class Variables

Name	Description
current_version_id	Value: 1

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)
```

__init__(*self*)

x.**__init__**(...) initializes x; see help(type(x)) for signature Overrides:
object.**__init__** **__exit__**(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()

changed_since_last_build(*self*, *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.

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*)**do_not_store_info**(*self*)**env_set**(*self*, *env*, *safe=0*)**executor_cleanup**(*self*)

Let the executor clean up any cached information.

exists(*self*)

Does this node exist?

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 cache
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_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)
```

```
state_has_changed(self, target, prev_ni)
```

```
store_info(self)
```

Make the build signature permanent (that is, store it in the .sconsign file or equivalent).

```
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__()
```

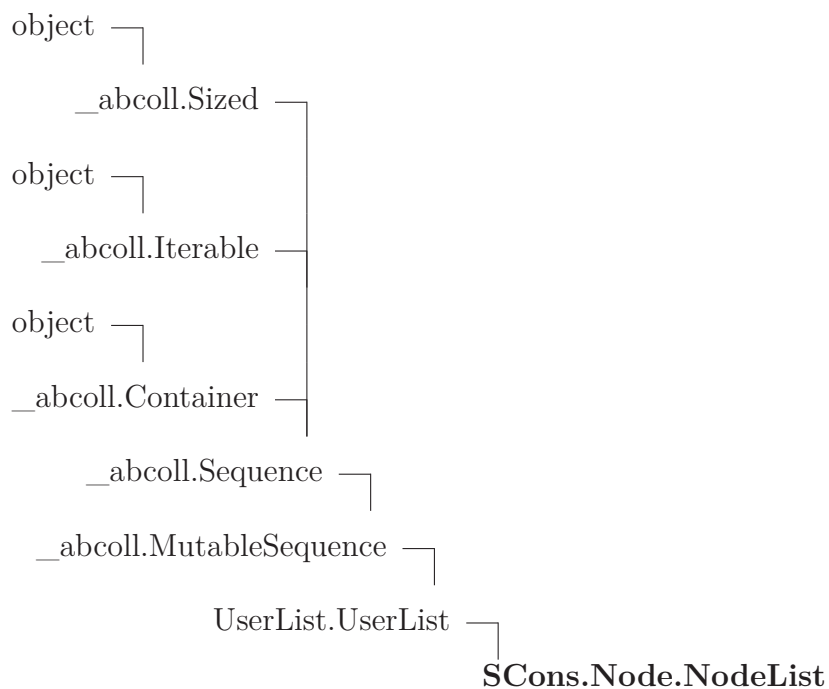
13.6.2 Properties

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

13.6.3 Class Variables

Name	Description
__metaclass__	Value: SCons.Memoize.Memoized_Metaclass
memoizer_counters	Value: []

13.7 Class NodeList



13.7.1 Methods

__str__ (<i>self</i>) str(x) Overrides: object.__str__ extit(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__()`, `ap-`
`pend()`, `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 <code>object</code></i>	
<code>__class__</code>	

13.7.3 Class Variables

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

13.8 Class 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
0x7f056d2d9758>, cycle_func=<function ignore_cycle at
0x7f056d2d97d0>, eval_func=<function do_nothing at 0x7f056d2d9848>)
x.__init__(...) initializes x; see help(type(x)) for signature Overrides:
object.__init__ exitit(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>	
<code>__class__</code>	

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>	Value: <code>'src/engine/SCons/Node/Alias.py rel_2.3.5:3347:d31d5a4e74...</code>
<code>default_ans</code>	Value: <code>{}</code>
<code>__package__</code>	Value: <code>'SCons.Node'</code>

14.2 Class *AliasNameSpace*

`UserDict.UserDict` —
 `SCons.Node.Alias.AliasNameSpace`

14.2.1 Methods

<code>Alias(self, name, **kw)</code>

<code>lookup(self, name, **kw)</code>

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

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

<code>str_to_node(self, s)</code>

Inherited from SCons.Node.NodeInfoBase(Section 13.4)

`__init__()`, `convert()`, `format()`, `merge()`, `update()`

Inherited from object

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

14.3.2 Properties

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

14.3.3 Class Variables

Name	Description
<code>current_version_id</code>	Value: 1
<code>field_list</code>	Value: ['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)

`__init__()`, `merge()`

Inherited from object

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

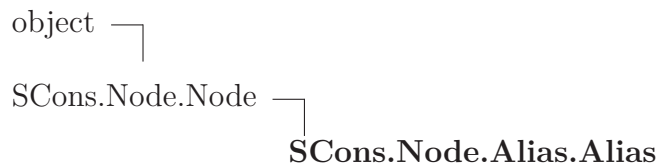
14.4.2 Properties

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

14.4.3 Class Variables

Name	Description
<code>current_version_id</code>	Value: 1

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.

sconsign(*self*)

An Alias is not recorded in .sconsign files

changed_since_last_build(*self*, *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. Overrides:
SCons.Node.Node.changed_since_last_build extit(inherited documentation)

build(*self*)

A "builder" for aliases. Overrides: SCons.Node.Node.build

convert (<i>self</i>)

get_csig (<i>self</i>)

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(), 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(), chil-
dren_are_up_to_date(), clear(), clear_memoized_values(), del_binfo(), disam-
biguate(), do_not_store_info(), 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(), ren-
der_include_tree(), reset_executor(), retrieve_from_cache(), reexists(), scan(), scan-
ner_key(), select_scanner(), set_always_build(), set_executor(), set_explicit(),
set_nocache(), set_noclean(), set_precious(), set_pseudo(), set_specific_source(),
set_state(), state_has_changed(), store_info(), 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 object</i>	
__class__	

14.5.3 Class Variables

Name	Description
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	__metaclass__, memoizer_counters

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

`save_strings(val)`

`initialize_do_splitdrive()`

`needs_normpath_match(...)`

match(string[, pos[, endpos]]) --> match object or None. Matches zero or more characters at the beginning of the string

`set_duplicate(duplicate)`

`LinkFunc(target, source, env)`

`LocalString(target, source, env)`

`UnlinkFunc(target, source, env)`

`MkdirFunc(target, source, env)`

`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 (<i>targets</i>)
<p>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.</p>

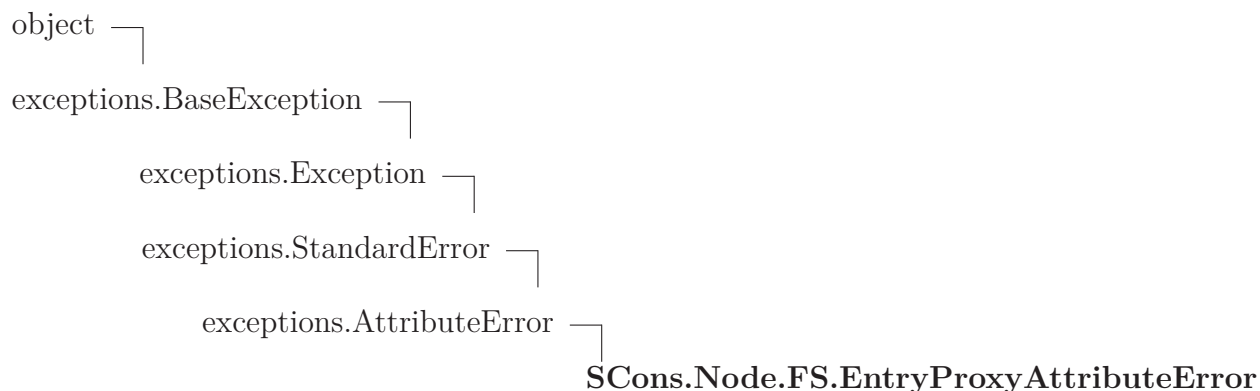
15.2 Variables

Name	Description
<code>__revision__</code>	Value: 'src/engine/SCons/Node/FS.py rel_2.3.5:3347:d31d5a4e74b6 ...
<code>do_store_info</code>	Value: True
<code>print_duplicate</code>	Value: 0
<code>default_max_drift</code>	Value: 172800
<code>Save_Strings</code>	Value: None
<code>do_splitdrive</code>	Value: False
<code>needs_normpath_check</code>	Value: re.compile(r'(?x).*// (.*/)?\.\.(?:/ \$) \./ .*\.(?:/ \$)')
<code>Valid_Duplicates</code>	Value: ['hard-soft-copy', 'soft-hard-copy', 'hard-copy', 'soft-c...']
<code>Link_Funcs</code>	Value: []
<code>Link</code>	Value: SCons.Action.Action(LinkFunc, None)
<code>LocalCopy</code>	Value: SCons.Action.Action(LinkFunc, LocalString)
<code>Unlink</code>	Value: SCons.Action.Action(UnlinkFunc, None)
<code>Mkdir</code>	Value: SCons.Action.Action(MkdirFunc, None, presub= None)
<code>MkdirBuilder</code>	Value: None
<code>DefaultSCCSBuilder</code>	Value: None
<code>DefaultRCSBuilder</code>	Value: None
<code>diskcheck_match</code>	Value: DiskChecker('match', do_diskcheck_match, ignore_diskcheck...)
<code>diskcheck_rcs</code>	Value: DiskChecker('rcs', do_diskcheck_rcs, ignore_diskcheck_rcs)

continued on next page

Name	Description
<code>diskcheck_sccs</code>	Value: <code>DiskChecker('sccs', do_diskcheck_sccs, ignore_diskcheck_s...</code>
<code>diskcheckers</code>	Value: <code>[diskcheck_match, diskcheck_rcs, diskcheck_sccs,]</code>
<code>glob_magic_check</code>	Value: <code>re.compile(r'[*\?\[\]]')</code>
<code>default_fs</code>	Value: <code>None</code>
<code>OS_SEP</code>	Value: <code>'/'</code>
<code>UNC_PREFIX</code>	Value: <code>'//'</code>
<code>__package__</code>	Value: <code>'SCons.Node'</code>
<code>has_unc</code>	Value: <code>False</code>
<code>os_sep_is_slash</code>	Value: <code>True</code>

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__  extit(inherited documentation)

```

```

__str__(self)

str(x)  Overrides: object.__str__  extit(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 exceptions.BaseException</i> args, message	
<i>Inherited from object</i> __class__	

15.4 Class DiskChecker**15.4.1 Methods**

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

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

<code>set(self, list)</code>

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> <code>__class__</code>	

15.5 Class `EntryProxy`



15.5.1 Methods

<code>__str__</code> (...)
<p>A Python Descriptor class that delegates attribute fetches to an underlying wrapped subject of a Proxy. Typical use:</p> <pre>class Foo(Proxy): __str__ = Delegate('__str__')</pre> <p>Overrides: <code>object.__str__</code></p>

<code>__getattr__</code> (<i>self</i> , <i>name</i>)
<p>Retrieve an attribute from the wrapped object. If the named attribute doesn't exist, <code>AttributeError</code> is raised. Overrides: <code>SCons.Util.Proxy.__getattr__</code> extit(inherited documentation)</p>

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

`__cmp__`(), `__init__`(), `get`()

Inherited from object

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

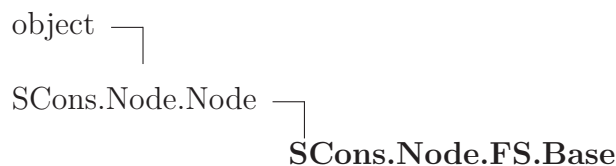
15.5.2 Properties

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

15.5.3 Class Variables

Name	Description
dictSpecialAttrs	Value: {"base": __get_base_path, "posix": __get_posix_path, "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

<p>__init__(<i>self</i>, <i>name</i>, <i>directory</i>, <i>fs</i>)</p> <hr/> <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: <code>object.__init__</code></p>
<p>str_for_display(<i>self</i>)</p>

must_be_same(*self*, *klass*)

This node, which already existed, is being looked up as the specified class. Raise an exception if it isn't.

get_dir(*self*)

get_suffix(*self*)

Overrides: SCons.Node.Node.get_suffix

rfile(*self*)

__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 exist? 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 ourself.

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_abspath(*self*)

Get the absolute path of the file. Overrides: SCons.Node.Node.get_abspath

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` `exitit`(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` `exitit`(inherited documentation)

target_from_source(*self*, *prefix*, *suffix*, *splitext*=<function splitext at 0x7f056d4cb140>)

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 (<i>self</i> , <i>pathlist</i>)
--

Search for a list of directories in the Repository list.
--

reentry (<i>self</i>)

Inherited from SCons.Node.Node(Section 13.6)

Decider(), 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(), changed_since_last_build(), children(), children_are_up_to_date(), clear(), clear_memoized_values(), del_binfo(), disambiguate(), do_not_store_info(), 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_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(), state_has_changed(), store_info(), visited()

Inherited from object

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

15.6.2 Properties

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

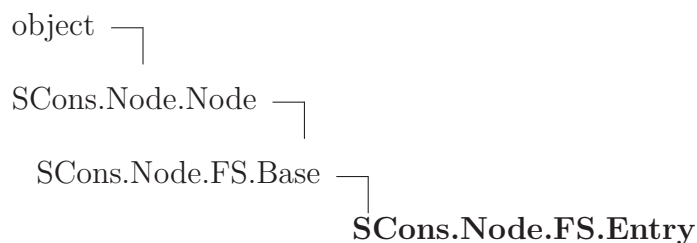
15.6.3 Class Variables

Name	Description
memoizer_counters	Value: []
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	
__metaclass__	

15.6.4 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
suffix	Cached filename extension
fs	Reference to parent Node.FS object

15.7 Class 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

diskcheck_match (<i>self</i>)
--

disambiguate (<i>self</i> , <i>must_exist=None</i>)
--

Overrides: SCons.Node.Node.disambiguate

rfile (<i>self</i>)

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 (<i>self</i>)

Overrides: SCons.Node.Node.scanner_key
--

get_contents(*self*)

Fetch the contents of the entry. Returns the exact binary contents of the file.

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*)

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. Overrides: SCons.Node.Node.exists

rel_path(*self*, *other*)**new_ninfo**(*self*)

Overrides: SCons.Node.Node.new_ninfo

changed_since_last_build(*self*, *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. Overrides: SCons.Node.Node.changed_since_last_build 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)

Inherited from SCons.Node.FS.Base(Section 15.6)

RDirs(), Rfindalldirs(), `__init__()`, `__str__()`, for_signature(), get_abspath(), get_dir(), get_path(), get_suffix(), getmtime(), getsize(), is_under(), isdir(), is_file(), islink(), reentry(), reexists(), 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(), 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(), do_not_store_info(), 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_bininfo(), 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(), state_has_changed(), store_info(), visited()

Inherited from object

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

15.7.2 Properties

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

15.7.3 Class Variables

Name	Description
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i>	
memoizer_counters	
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	
__metaclass__	

15.7.4 Instance Variables

Name	Description
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i>	
fs, name, suffix	

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)`

<code>readlink(self, file)</code>

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>	
<code>__class__</code>	

15.8.3 Class Variables

Name	Description
<code>__metaclass__</code>	Value: <code>SCons.Memoize.Memoized_Metaclass</code>

15.9 Class FS



15.9.1 Methods

<code>__init__(self, path=None)</code>
--

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__`

<code>set_SConstruct_dir(self, dir)</code>
--

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__()

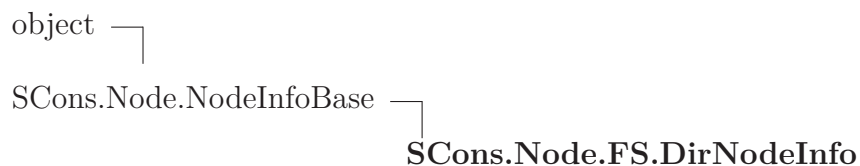
15.9.2 Properties

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

15.9.3 Class Variables

Name	Description
memoizer_counters	Value: []
<i>Inherited from SCons.Node.FS.LocalFS (Section 15.8)</i>	
__metaclass__	

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

<code>str_to_node(self, s)</code>

Inherited from SCons.Node.NodeInfoBase(Section 13.4)

__init__(), convert(), format(), merge(), update()

Inherited from object

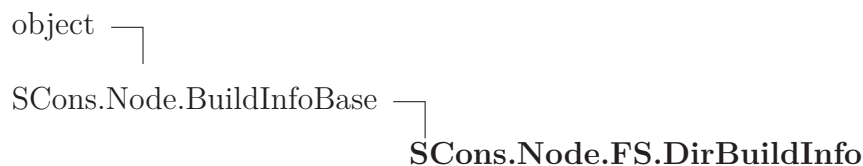
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
 __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
 __str__(), __subclasshook__()

15.10.2 Properties

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

15.10.3 Class Variables

Name	Description
current_version_id	Value: 1
fs	Value: 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)***

__init__(), merge()

Inherited from object

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

`__str__()`, `__subclasshook__()`

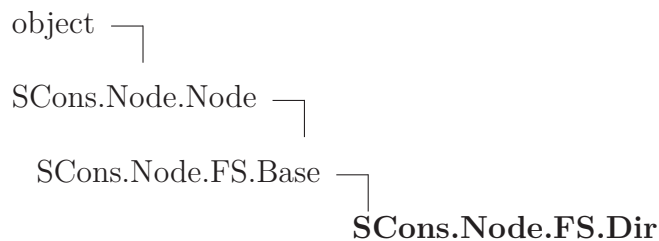
15.11.2 Properties

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

15.11.3 Class Variables

Name	Description
<code>current_version_id</code>	Value: 1

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__` `extit`(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 files 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` `exitit`(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.

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*)

changed_since_last_build(*self*, *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. Overrides: SCons.Node.Node.changed_since_last_build exit(inherited documentation)

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, creating it first if necessary.

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

entry_abstractmethod(*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(), __str__(), exists(), for_signature(), get_abspath(), 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(), 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(), do_not_store_info(),
```

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(), state_has_changed(), store_info(), visited()

Inherited from object

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

15.12.2 Properties

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

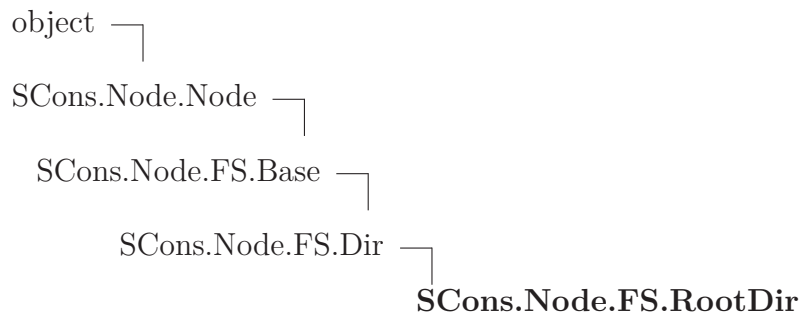
15.12.3 Class Variables

Name	Description
memoizer_counters	Value: []
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	
__metaclass__	

15.12.4 Instance Variables

Name	Description
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i>	
fs, name, suffix	

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

`__init__`(*self*, *drive*, *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)

`must_be_same`(*self*, *klass*)

This node, which already existed, is being looked up as the specified class.

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_abstractmethod`(*self*, *name*)

Overrides: *SCons.Node.FS.Dir.entry_abstractmethod*

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*
 extit(inherited documentation)

Inherited from SCons.Node.FS.Dir (Section 15.12)

Dir(), *Entry()*, *File()*, *addRepository()*, *alter_targets()*, *build()*, *changed_since_last_build()*, *dir_on_disk()*, *diskcheck_match()*, *do_duplicate()*, *entry_exists_on_disk()*, *file_on_disk()*, *getRepositories()*, *get_all_rdirs()*, *get_contents()*, *get_csig()*, *get_env_scanner()*, *get_found_includes()*, *get_target_scanner()*, *get_text_contents()*, *get_timestamp()*, *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()*, *exists()*, *for_signature()*, *get_abspath()*, *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(), add_dependency(), add_ignore(), add_prerequisite(), add_source(),
 add_to_implicit(), add_to_waiting_parents(), add_to_waiting_se(), add_wkid(),
 all_children(), builder_set(), built(), changed(), children(), children_are_up_to_date(),
 clear(), clear_memoized_values(), del_binfo(), disambiguate(), do_not_store_info(),
 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(),
 state_has_changed(), store_info(), 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 object</i> __class__	

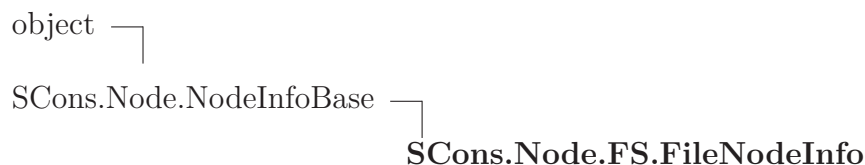
15.13.3 Class Variables

Name	Description
<i>Inherited from SCons.Node.FS.Dir (Section 15.12)</i> memoizer_counters	
<i>Inherited from SCons.Node.Node (Section 13.6)</i> __metaclass__	

15.13.4 Instance Variables

Name	Description
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i> fs, name, suffix	

15.14 Class FileNodeInfo



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>

Inherited from SCons.Node.NodeInfoBase(Section 13.4)

`__init__()`, `convert()`, `format()`, `merge()`, `update()`

Inherited from object

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

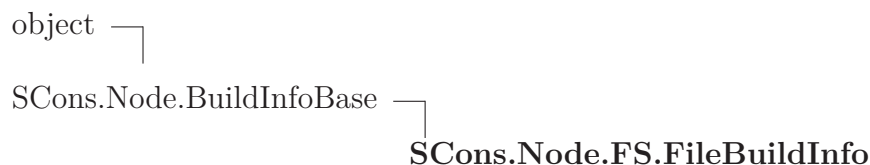
15.14.2 Properties

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

15.14.3 Class Variables

Name	Description
<code>current_version_id</code>	Value: 1
<code>field_list</code>	Value: ['csig', 'timestamp', 'size']
<code>fs</code>	Value: 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)

__init__(), merge()

Inherited from object

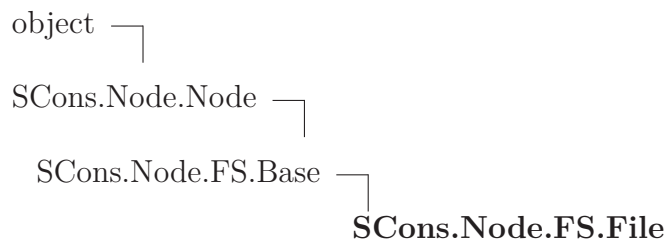
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
 __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
 __str__(), __subclasshook__()

15.15.2 Properties

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

15.15.3 Class Variables

Name	Description
current_version_id	Value: 1

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*)

get_text_contents(*self*)

get_content_hash(*self*)

Compute and return the MD5 hash for this file.

get_size(*self*)**get_timestamp**(*self*)**store_info**(*self*)

Make the build signature permanent (that is, store it in the .sconsign file or equivalent). Overrides: SCons.Node.Node.store_info extit(inherited documentation)

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)
```

```
decide_source(self, 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.

```
decide_target(self, target, prev_ni)
```

```
changed_since_last_build(self, 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. Overrides: SCons.Node.Node.changed_since_last_build extit(inherited documentation)

```
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 extit(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 extit(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(), __str__(), for_signature(), get_abspath(), get_dir(), get_path(), get_subst_proxy(), get_suffix(), getmtime(), getsize(), is_under(), isdir(), isfile(), islink(), must_be_same(), reentry(), reexists(), set_local(), set_src_builder(), src_builder(), srcnode(), stat(), str_for_display(), target_from_source()

Inherited from SCons.Node.Node(Section 13.6)

Decider(), add_dependency(), add_ignore(), add_prerequisite(), add_source(),
 add_to_implicit(), add_to_waiting_parents(), add_to_waiting_set(), add_wkid(),
 all_children(), build(), children(), children_are_up_to_date(), clear(), clear_memoized_values(),
 del_binfo(), disambiguate(), do_not_store_info(), 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(),
 state_has_changed()

Inherited from object

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

15.16.2 Properties

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

15.16.3 Class Variables

Name	Description
memoizer_counters	Value: []
md5_chunksize	Value: 64
convert_copy_attrs	Value: ['bsources', 'bimplicit', 'bdepends', 'bact', 'bactsig', ...]
convert_sig_attrs	Value: ['bsourcesigs', 'bimplicitsigs', 'bdependSIGs']
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	
__metaclass__	

15.16.4 Instance Variables

Name	Description
<i>Inherited from SCons.Node.FS.Base (Section 15.6)</i>	

continued on next page

Name	Description
fs, name, suffix	

15.17 Class 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>	
__class__	

15.17.3 Class Variables

Name	Description
__metaclass__	Value: SCons.Memoize.Memoized_Metaclass
memoizer_counters	Value: []

16 Module SCons.Node.Python

scons.Node.Python

Python nodes.

16.1 Variables

Name	Description
<code>__revision__</code>	Value: 'src/engine/SCons/Node/Python.py rel_2.3.5:3347:d31d5a4e7...
<code>__package__</code>	Value: 'SCons.Node'

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

<code>str_to_node(self, s)</code>

Inherited from SCons.Node.NodeInfoBase(Section 13.4)

`__init__()`, `convert()`, `format()`, `merge()`, `update()`

Inherited from object

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

16.2.2 Properties

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

16.2.3 Class Variables

Name	Description
current_version_id	Value: 1
field_list	Value: ['csig']

16.3 Class 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)

__init__(), merge()

Inherited from object

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

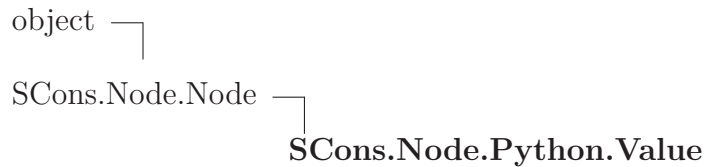
16.3.2 Properties

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

16.3.3 Class Variables

Name	Description
current_version_id	Value: 1

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 exitit(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.

changed_since_last_build(self, 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. Overrides: SCons.Node.Node.changed_since_last_build extit(inherited documentation)

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(), 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(), do_not_store_info(), 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(), state_has_changed(), store_info(), 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 object</i>	
__class__	

16.4.3 Class Variables

Name	Description
<i>Inherited from SCons.Node.Node (Section 13.6)</i>	
__metaclass__, memoizer_counters	

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
<code>__revision__</code>	Value: 'src/engine/SCons/PathList.py rel_2.3.5:3347:d31d5a4e74b6...
<code>__doc__</code>	Value: ""SCons.PathL...
<code>TYPE_STRING_NO_SUBST</code>	Value: 0
<code>TYPE_STRING_SUBST</code>	Value: 1
<code>TYPE_OBJECT</code>	Value: 2
<code>__package__</code>	Value: '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

SetBuildType(*type*)

SetCacheMode(*mode*)

Set the Configure cache mode. mode must be one of "auto", "force", or "cache".

SetProgressDisplay(*display*)

Set the progress display to use (called from SCons.Script)

NeedConfigHBuilder()

CreateConfigHBuilder(*env*)

Called if necessary just before the building targets phase begins.

SConf(**args*, ***kw*)

CheckFunc(*context*, *function_name*, *header=*None, *language=*None)

CheckType(*context*, *type_name*, *includes=*'', *language=*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.

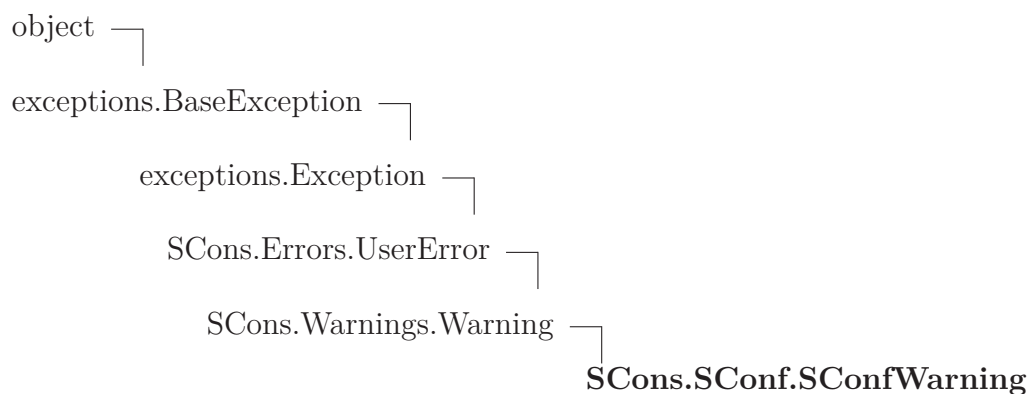
CheckLibWithHeader(*context, libs, header, language, call=None, autoadd=1*)

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.

18.2 Variables

Name	Description
<code>__revision__</code>	Value: 'src/engine/SCons/SConf.py rel_2.3.5:3347:d31d5a4e74b6 20...
<code>build_type</code>	Value: None
<code>build_types</code>	Value: ['clean', 'help']
<code>dryrun</code>	Value: 0
<code>AUTO</code>	Value: 0
<code>FORCE</code>	Value: 1
<code>CACHE</code>	Value: 2
<code>cache_mode</code>	Value: 0
<code>progress_display</code>	Value: DisplayEngine()
<code>SConfFS</code>	Value: None
<code>sconf_global</code>	Value: None
<code>__package__</code>	Value: 'SCons'

18.3 Class SConfWarning



18.3.1 Methods

Inherited from exceptions.Exception

`__init__()`, `__new__()`

Inherited from exceptions.BaseException

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

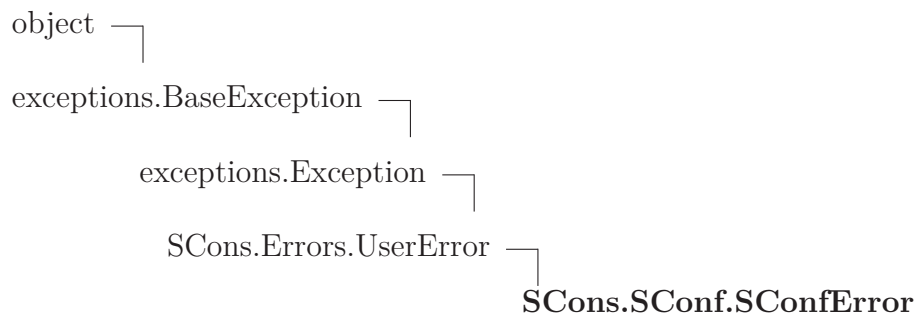
Inherited from object

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

18.3.2 Properties

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

18.4 Class SConfError



Known Subclasses: SCons.SConf.ConfigureCacheError, SCons.SConf.ConfigureDryRunError

18.4.1 Methods

<code>__init__(self, msg)</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)

Inherited from `exceptions.Exception`

`__new__()`

Inherited from `exceptions.BaseException`

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

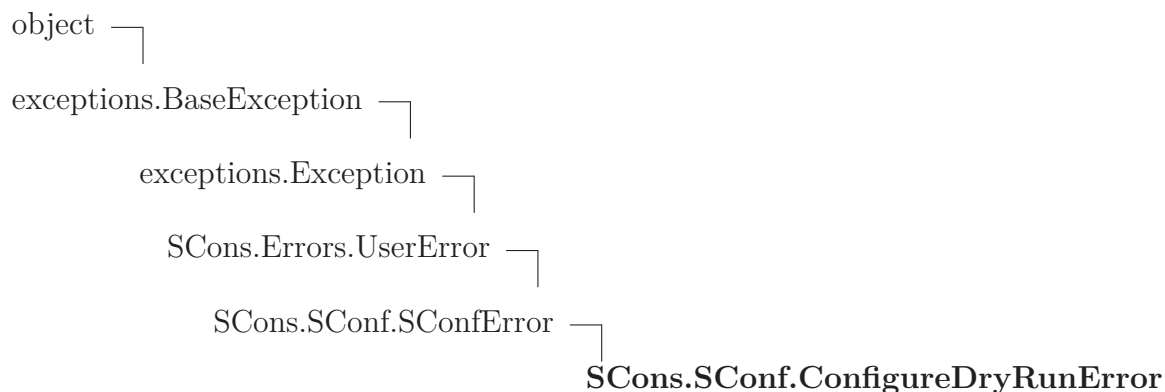
Inherited from `object`

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

18.4.2 Properties

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

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

<code>__init__(self, target)</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)
--

Inherited from `exceptions.Exception`

`__new__()`

Inherited from `exceptions.BaseException`

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

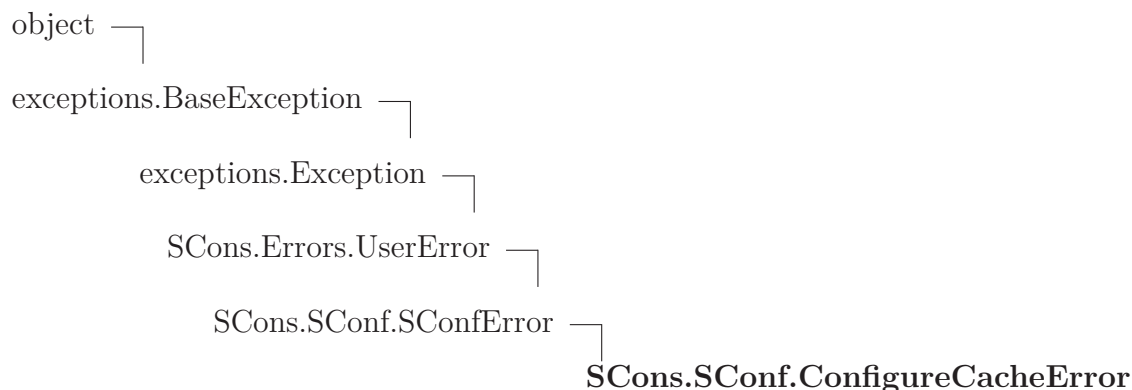
Inherited from `object`

`__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__()
```

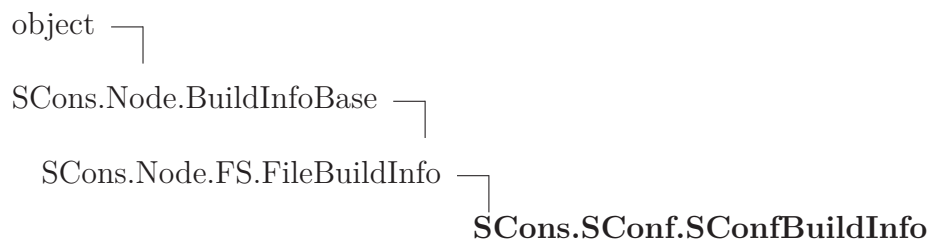
Inherited from `object`

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

18.6.2 Properties

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

<code>set_build_result(self, result, string)</code>

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)

`__init__()`, `merge()`

Inherited from object

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

18.7.2 Properties

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

18.7.3 Class Variables

Name	Description
result	Value: None
string	Value: None
<i>Inherited from SCons.Node.FS.FileBuildInfo (Section 15.15)</i>	
current_version_id	

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__** **exitit**(inherited documentation)

```
write(self, str)
```

```
writelines(self, lines)
```

```
getvalue(self)
```

Return everything written to orig since the Streamer was created.

```
flush(self)
```

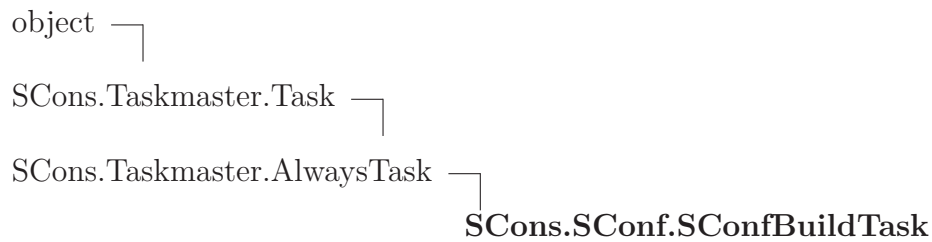
Inherited from object

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

18.8.2 Properties

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

18.9 Class *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* extit(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

<p><code>__init__</code>(<i>self</i>, <i>sconf</i>)</p> <hr/> <p>Constructor. Pass the corresponding <code>SConf</code> instance. Overrides: <code>object.__init__</code></p>
<p><code>Message</code>(<i>self</i>, <i>text</i>)</p> <hr/> <p>Inform about what we are doing right now, e.g. 'Checking for SOMETHING ...'</p>
<p><code>Result</code>(<i>self</i>, <i>res</i>)</p> <hr/> <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 <code>self.did_show_result</code> is not set.</p>
<p><code>TryBuild</code>(<i>self</i>, *<i>args</i>, **<i>kw</i>)</p>
<p><code>TryAction</code>(<i>self</i>, *<i>args</i>, **<i>kw</i>)</p>
<p><code>TryCompile</code>(<i>self</i>, *<i>args</i>, **<i>kw</i>)</p>
<p><code>TryLink</code>(<i>self</i>, *<i>args</i>, **<i>kw</i>)</p>
<p><code>TryRun</code>(<i>self</i>, *<i>args</i>, **<i>kw</i>)</p>
<p><code>__getattr__</code>(<i>self</i>, <i>attr</i>)</p>
<p><code>BuildProg</code>(<i>self</i>, <i>text</i>, <i>ext</i>)</p>
<p><code>CompileProg</code>(<i>self</i>, <i>text</i>, <i>ext</i>)</p>

CompileSharedObject (<i>self</i> , <i>text</i> , <i>ext</i>)

RunProg (<i>self</i> , <i>text</i> , <i>ext</i>)

AppendLIBS (<i>self</i> , <i>lib_name_list</i>)
--

PrependLIBS (<i>self</i> , <i>lib_name_list</i>)

SetLIBS (<i>self</i> , <i>val</i>)

Display (<i>self</i> , <i>msg</i>)

Log (<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__	Value: 'src/engine/SCons/SConsign.py rel_2.3.5:3347:d31d5a4e74b6...
sig_files	Value: []
DataBase	Value: {}
DB_Name	Value: '.sconsign'
DB_sync_list	Value: []
__package__	Value: '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)
```

Inherited from object

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

19.3.2 Properties

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

19.3.3 Class Variables

Name	Description
current_version_id	Value: 1

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*)

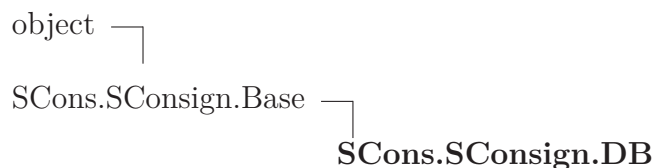
Inherited from object

__delattr__(*self*, *name*), **__format__**(*self*, *format_spec*), **__getattr__**(*self*, *name*), **__hash__**(*self*), **__new__**(*cls*, *args*, *kwargs*), **__reduce__**(*self*), **__reduce_ex__**(*self*, *proto*), **__repr__**(*self*), **__setattr__**(*self*, *name*, *value*), **__sizeof__**(*self*), **__str__**(*self*), **__subclasshook__**(*self*, *subclass*)

19.4.2 Properties

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

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

__init__ (<i>self</i> , <i>dir</i>)
x. __init__ (...) initializes x; see help(type(x)) for signature Overrides: object. __init__ extit(inherited documentation)

write (<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__()`

19.5.2 Properties

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

19.6 Class Dir



Known Subclasses: SCons.SConsign.DirFile

19.6.1 Methods

<code>__init__(self, fp=None, dir=None)</code>
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()

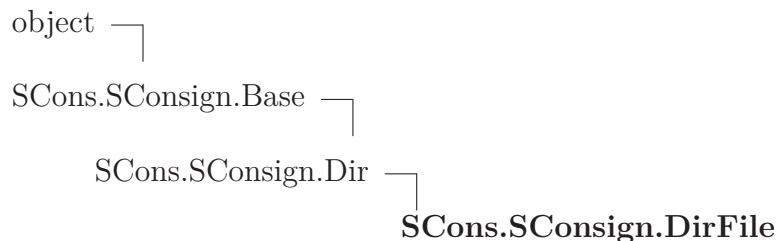
Inherited from object

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

19.6.2 Properties

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

19.7 Class DirFile



Encapsulates reading and writing a per-directory `.sconsign` file.

19.7.1 Methods

<code>__init__(self, dir)</code> <hr/> dir - the directory for the file Overrides: <code>object.__init__</code>

<code>write(self, sync=1)</code> <hr/> 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()`

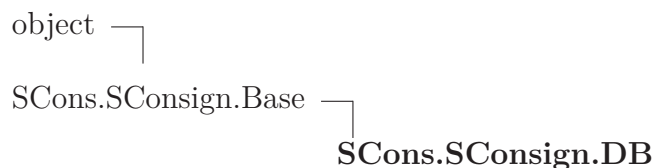
Inherited from `object`

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

19.7.2 Properties

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

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

```
__init__(self, dir)
```

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

```
write(self, sync=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>	
<code>__class__</code>	

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. 228)
- **D**: SCons.Scanner.D
(Section 22, p. 231)
- **Dir** (Section 23, p. 235)
- **Fortran**: SCons.Scanner.Fortran
(Section 24, p. 237)
- **IDL**: SCons.Scanner.IDL
(Section 25, p. 242)
- **LaTeX**: SCons.Scanner.LaTeX
(Section 26, p. 243)
- **Prog** (Section 27, p. 250)
- **RC**: SCons.Scanner.RC
(Section 28, p. 251)

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__	Value: 'src/engine/SCons/Scanner/__init__.py rel_2.3.5:3347:d31d...

continued on next page

Name	Description
<code>__package__</code>	Value: <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 <code>help(type(x))</code> for signature Overrides: object. <code>__init__</code> <code>__exit__</code> (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
<code>__class__</code>	Inherited from object

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)
```

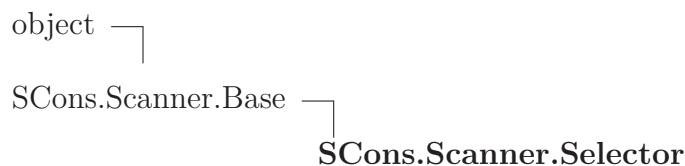
<code>__str__(self)</code> str(x) Overrides: object.__str__ extit(inherited documentation)
<code>add_scanner(self, key, scanner)</code>
<code>add_key(self, key)</code> <hr/> Add a key to the list of skeys
<code>get_skeys(self, env=None)</code>
<code>path(self, env, dir=None, target=None, source=None)</code>
<code>recurse_nodes(self, nodes)</code>
<code>select(self, node)</code>

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>	
<code>__class__</code>	

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`.

'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) 234
```

```
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___exit(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()
```

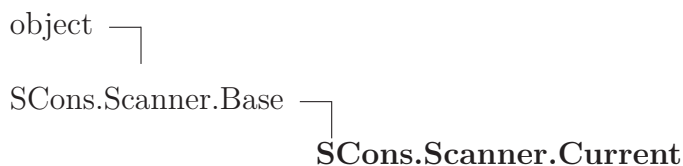
Inherited from object

```
__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`.

'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) 237
```

```
s = Scanner(function = my_scanner_function)
```

```
s = Scanner(function = my_scanner_function, argument = 'foo') Overrides:
```

Inherited from SCons.Scanner.Base(Section 20.5)

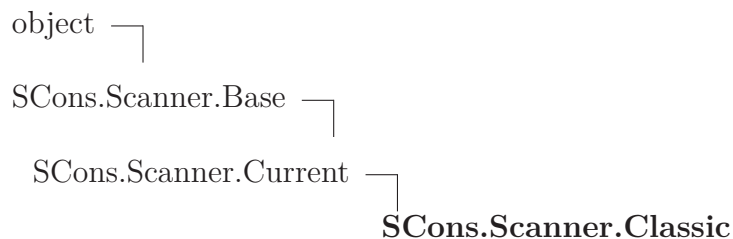
`__call__()`, `__cmp__()`, `__hash__()`, `__str__()`, `add_scanner()`, `add_skey()`,
`get_skeys()`, `path()`, `recurse_nodes()`, `select()`

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`.

'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) 240
```

```
s = Scanner(function = my_scanner_function)
```

```
s = Scanner(function = my_scanner_function, argument = 'foo') Overrides:
```


<code>find_include(self, include, source_dir, path)</code>
--

<code>find_include_names(self, node)</code>

<code>scan(self, node, path=())</code>
--

<code>sort_key(self, include)</code>

Inherited from SCons.Scanner.Base(Section 20.5)

`__call__()`, `__cmp__()`, `__hash__()`, `__str__()`, `add_scanner()`, `add_skey()`, `get_skeys()`, `path()`, `recurse_nodes()`, `select()`

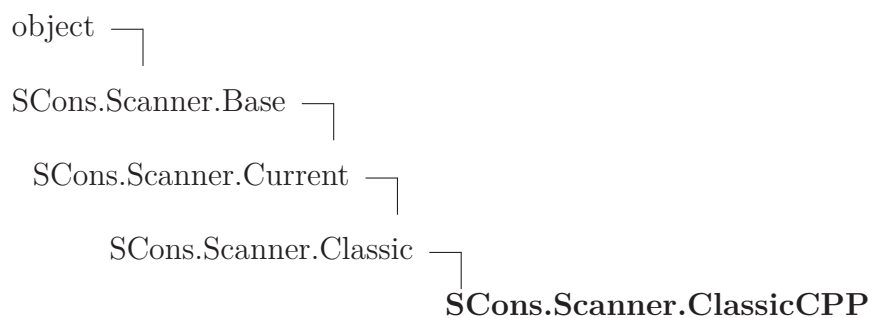
Inherited from object

`__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

find_include (<i>self</i> , <i>include</i> , <i>source_dir</i> , <i>path</i>)
--

Overrides: <i>SCons.Scanner.Classic.find_include</i>
--

sort_key (<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

<code>dictify_CPPDEFINES(<i>env</i>)</code>

<code>CScanner()</code>

Return a prototype Scanner instance for scanning source files that use the C pre-processor
--

21.2 Variables

Name	Description
<code>__revision__</code>	Value: 'src/engine/SCons/Scanner/C.py rel_2.3.5:3347:d31d5a4e74b...
<code>__package__</code>	Value: 'SCons.Scanner'

21.3 Class 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

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

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

<code>initialize_result(self, fname)</code>

Overrides: SCons.cpp.PreProcessor.initialize_result

<code>finalize_result(self, fname)</code>

Overrides: SCons.cpp.PreProcessor.finalize_result

<code>find_include_file(self, t)</code>

Finds the #include file for a given preprocessor tuple. Overrides: SCons.cpp.PreProcessor.find_include_file exitit(inherited documentation)

<code>read_file(self, file)</code>

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()`, `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__()`

21.3.2 Properties

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

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

```
__init__(self, name, variable)
```

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

```
__call__(self, node, env, path=())
```

```
recurse_nodes(self, nodes)
```

```
select(self, node)
```

Inherited from object

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),  

__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),  

__str__(), __subclasshook__()
```

21.4.2 Properties

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

22 Module SCons.Scanner.D

SCons.Scanner.D

Scanner for the Digital Mars "D" programming language.

Coded by Andy Friesen 17 Nov 2003

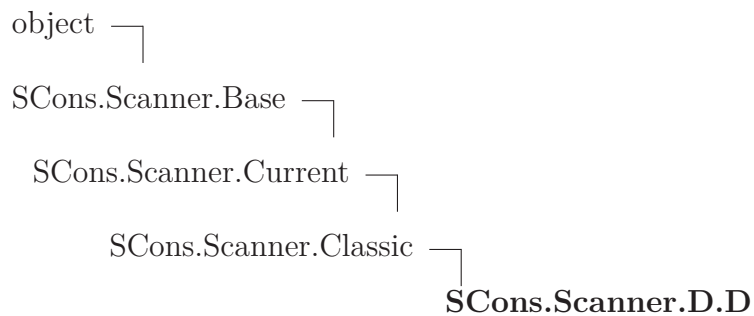
22.1 Functions

DScanner()
Return a prototype Scanner instance for scanning D source files

22.2 Variables

Name	Description
__revision__	Value: 'src/engine/SCons/Scanner/D.py rel_2.3.5:3347:d31d5a4e74b...
__package__	Value: '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`.

'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) 248
```

```
s = Scanner(function = my_scanner_function)
```

```
s = Scanner(function = my_scanner_function, argument = 'foo') Overrides:
```


<code>find_include(self, include, source_dir, path)</code>
--

Overrides: SCons.Scanner.Classic.find_include

<code>find_include_names(self, node)</code>

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>	Value: 'src/engine/SCons/Scanner/Dir.py rel_2.3.5:3347:d31d5a4e7...
<code>skip_entry</code>	Value: {'.': 1, '..': 1, '.sconsign': 1, '.sconsign.bak': 1, '.s...

continued on next page

Name	Description
skip_entry_list	Value: ['.', '..', '.sconsign', .sconsign.dblite', '.sconsign.d...
__package__	Value: 'SCons.Scanner'
skip	Value: '.sconsign.db'

24 Module SCons.Scanner.Fortran

SCons.Scanner.Fortran

This module implements the dependency scanner for Fortran code.

24.1 Functions

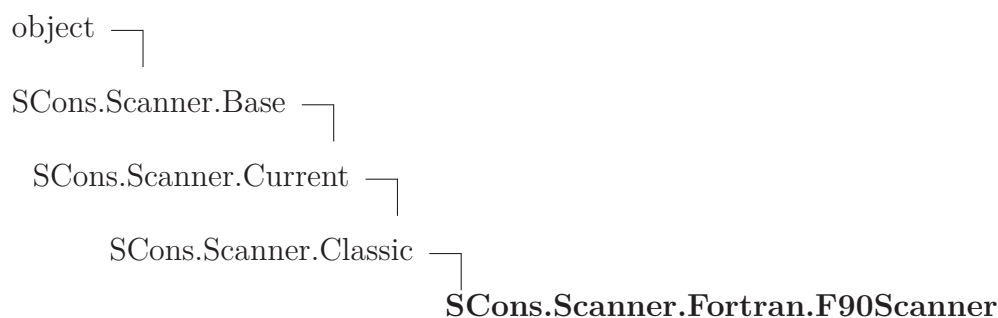
FortranScan(*path_variable*= 'FORTRANPATH')

Return a prototype Scanner instance for scanning source files for Fortran USE & INCLUDE statements

24.2 Variables

Name	Description
__revision__	Value: 'src/engine/SCons/Scanner/Fortran.py rel_2.3.5:3347:d31d5...
__package__	Value: '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:

255

```
s = Scanner(my_scanner_function)
```

```
s = Scanner(function = my_scanner_function)
```

<code>scan(self, node, env, path=())</code>

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

IDLScan()
Return a prototype Scanner instance for scanning IDL source files

25.2 Variables

Name	Description
<code>__revision__</code>	Value: <code>'src/engine/SCons/Scanner/IDL.py rel_2.3.5:3347:d31d5a4e7...</code>
<code>__package__</code>	Value: <code>'SCons.Scanner'</code>

26 Module SCons.Scanner.LaTeX

SCons.Scanner.LaTeX

This module implements the dependency scanner for LaTeX code.

26.1 Functions

modify__env__var (<i>env</i> , <i>var</i> , <i>abspath</i>)
--

LaTeXScanner ()

Return a prototype Scanner instance for scanning LaTeX source files when built with latex.
--

PDFLaTeXScanner ()

Return a prototype Scanner instance for scanning LaTeX source files when built with pdflatex.

26.2 Variables

Name	Description
__revision__	Value: 'src/engine/SCons/Scanner/LaTeX.py rel_2.3.5:3347:d31d5a4...'
TexGraphics	Value: ['.eps', '.ps']
LatexGraphics	Value: ['.pdf', '.png', '.jpg', '.gif', '.tif']
__package__	Value: '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__` `__exit__`(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



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`.

'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) 262
```

```
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>	Value: {'addbibresource': 'BIBINPUTS', 'addglobalbib': 'BIBINPUT...'}.

continued on next page

Name	Description
env_variables	Value: ['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__	Value: 'src/engine/SCons/Scanner/Prog.py rel_2.3.5:3347:d31d5a4e...
print_find_libs	Value: None
__package__	Value: '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

RCScan()
Return a prototype Scanner instance for scanning RC source files

28.2 Variables

Name	Description
<code>__revision__</code>	Value: 'src/engine/SCons/Scanner/RC.py rel_2.3.5:3347:d31d5a4e74...'
<code>__package__</code>	Value: 'SCons.Scanner'

29 Package SCons.Script

SCons.Script

This file implements the main() function used by the sconscript.

Architecturally, this *is* the sconscript, 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. 261)
- **Main:** SCons.Script
(Section 31, p. 264)
- **SConscript':** SCons.Script.SConscript
(Section 32, p. 278)

29.2 Functions

HelpFunction (<i>text</i>)

Variables (<i>files</i> =[], <i>args</i> ={})

Options (<i>files</i> =[], <i>args</i> ={})

29.3 Variables

Name	Description
__revision__	Value: 'src/engine/SCons/Script/__init__.py rel_2.3.5:3347:d31d5...
start_time	Value: 1438378856.37
call_stack	Value: []
PathVariable	Value: SCons.Variables.PathVariable
PathOption	Value: SCons.Options.PathOption
Chmod	Value: SCons.Defaults.Chmod
Copy	Value: SCons.Defaults.Copy
Delete	Value: SCons.Defaults.Delete

continued on next page

Name	Description
Mkdir	Value: SCons.Defaults.Mkdir
Move	Value: SCons.Defaults.Move
Touch	Value: SCons.Defaults.Touch
CScanner	Value: SCons.Defaults.CScan
DScanner	Value: SCons.Tool.DScanner
DirScanner	Value: SCons.Defaults.DirScanner
ProgramScanner	Value: SCons.Tool.ProgramScanner
SourceFileScanner	Value: SCons.Tool.SourceFileScanner
CScan	Value: SCons.Defaults.CScan
ARGUMENTS	Value: {}
ARGLIST	Value: []
BUILD_TARGETS	Value: []
COMMAND_LINE_TARGETS	Value: []
DEFAULT_TARGETS	Value: []
help_text	Value: None
sconscrip_reading	Value: 0
GlobalDefaultEnvironmentFunctions	Value: ['Default', 'EnsurePythonVersion', 'EnsureSConsVersion', ...]
GlobalDefaultBuilders	Value: ['CFile', 'CXXFile', 'DVI', 'Jar', 'Java', 'JavaH', 'Libr...]
SConscript	Value: <code>_SConscript.DefaultEnvironmentCall('SConscript')</code>
Command	Value: <code>_SConscript.DefaultEnvironmentCall('Command', subst= 1)</code>
AddPostAction	Value: <code><SCons.Script.SConscript.DefaultEnvironmentCall object at...></code>
AddPreAction	Value: <code><SCons.Script.SConscript.DefaultEnvironmentCall object at...></code>
Alias	Value: <code><SCons.Script.SConscript.DefaultEnvironmentCall object at...></code>
AlwaysBuild	Value: <code><SCons.Script.SConscript.DefaultEnvironmentCall object at...></code>
BuildDir	Value: <code><SCons.Script.SConscript.DefaultEnvironmentCall object at...></code>

continued on next page

Name	Description
CFile	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
CXXFile	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
CacheDir	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Clean	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
DVI	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Decider	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Default	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Depends	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Dir	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
EnsurePythonVersion	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
EnsureSConsVersion	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Entry	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Execute	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Exit	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...

continued on next page

Name	Description
Export	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
File	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
FindFile	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
FindInstalledFiles	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
FindSourceFiles	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Flatten	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
GetBuildPath	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
GetLaunchDir	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Glob	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Help	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Ignore	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Import	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Install	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
InstallAs	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...

continued on next page

Name	Description
Jar	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Java	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
JavaH	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Library	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Literal	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Local	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
M4	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
MSVSProject	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
NoCache	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
NoClean	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Object	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
PCH	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
PDF	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Package	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...

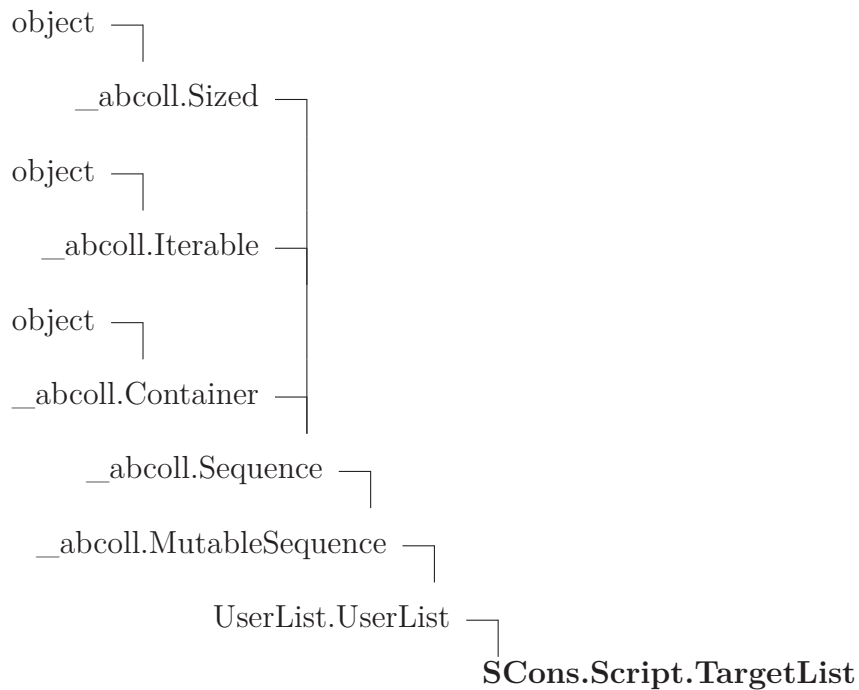
continued on next page

Name	Description
ParseDepends	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
PostScript	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Precious	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Program	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
RES	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
RMIC	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Repository	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Requires	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SConscriptChdir	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SConsignFile	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SharedLibrary	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SharedObject	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SideEffect	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
SourceCode	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...

continued on next page

Name	Description
SourceSignatures	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Split	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
StaticLibrary	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
StaticObject	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Tag	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Tar	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
TargetSignatures	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
TypeLibrary	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Value	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
VariantDir	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
Zip	Value: <SCons.Script.SConscript.DefaultEnvironmentCall object at...
__package__	Value: '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__()`, `ap-`
`pend()`, `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>	Value: 'src/engine/SCons/Script/Interactive.py rel_2.3.5:3347:d3...
<code>__doc__</code>	Value: ...
<code>__package__</code>	Value: '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 (<i>self</i> , <i>argv</i>)
exit Exit SCons interactive mode.

do_help (<i>self</i> , <i>argv</i>)
help [COMMAND] Prints help for the specified COMMAND. 'h' and '?' are synonyms. Overrides: cmd.Cmd.do_help

do_shell (<i>self</i> , <i>argv</i>)
shell [COMMANDLINE] Execute COMMANDLINE in a subshell. 'sh' and '!' are synonyms.

do_version (<i>self</i> , <i>argv</i>)
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	Value: {'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)
```

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>	Value: (2, 3, 0)
<code>deprecated_python_version</code>	Value: (2, 7, 0)
<code>__revision__</code>	Value: 'src/engine/SCons/Script/Main.py rel_2.3.5:3347:d31d5a4e7...
<code>display</code>	Value: <code>DisplayEngine()</code>
<code>progress_display</code>	Value: <code>SCons.Util.DisplayEngine()</code>
<code>first_command_start</code>	Value: <code>None</code>
<code>last_command_end</code>	Value: <code>None</code>
<code>ProgressObject</code>	Value: <code>Null(0x7F056CDF64D0)</code>
<code>print_objects</code>	Value: 0
<code>print_memoizer</code>	Value: 0
<code>print_stacktrace</code>	Value: 0
<code>print_time</code>	Value: 0
<code>sconscrip_time</code>	Value: 0
<code>cumulative_command_time</code>	Value: 0
<code>exit_status</code>	Value: 0
<code>this_build_status</code>	Value: 0
<code>num_jobs</code>	Value: <code>None</code>
<code>delayed_warnings</code>	Value: []
<code>OptionsParser</code>	Value: <code>FakeOptionParser()</code>

continued on next page

Name	Description
<code>count_stats</code>	Value: <code>CountStats()</code>
<code>memory_stats</code>	Value: <code>MemStats()</code>
<code>__package__</code>	Value: <code>'SCons.Script'</code>

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

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__** extit(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>	
<code>__class__</code>	

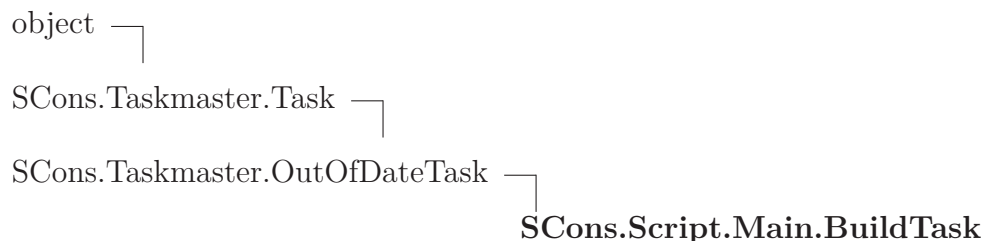
31.4.3 Class Variables

Name	Description
prev	Value: ''

continued on next page

Name	Description
count	Value: 0
target_string	Value: '\$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 actual 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` `exitit`(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` `exitit`(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` `exitit`(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` `exitit`(inherited documentation)

make_ready (<i>self</i>)
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__()`

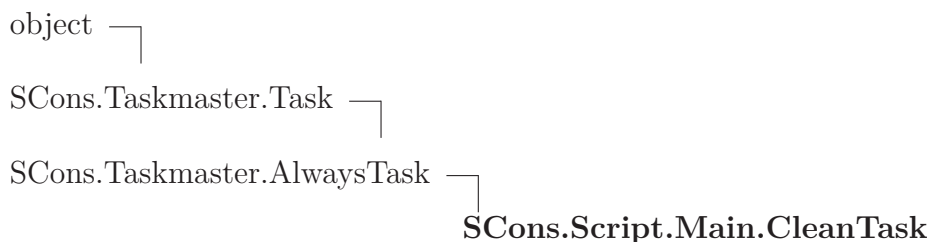
31.5.2 Properties

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

31.5.3 Class Variables

Name	Description
<code>progress</code>	Value: Null(0x7F056CDF64D0)

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` `exit`(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()

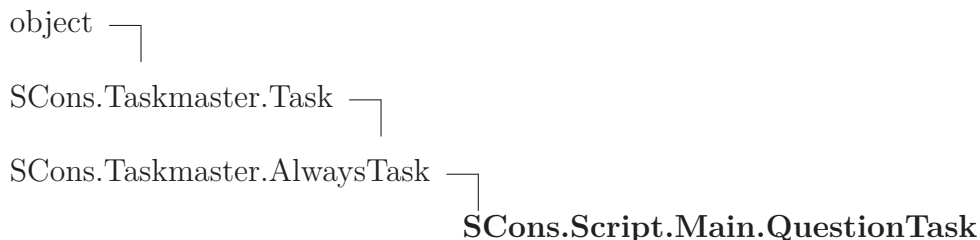
Inherited from object

__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`



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	Value: 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__ extit(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>	
__class__	

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()
```

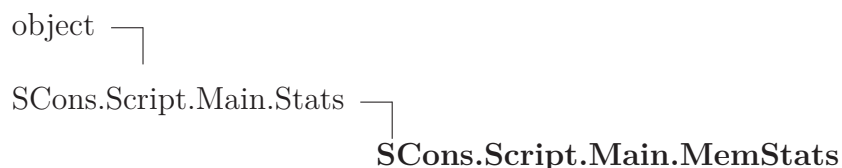
Inherited from object

__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>	Value: 'src/engine/SCons/Script/SConscript.py rel_2.3.5:3347:d31...'
<code>launch_dir</code>	Value: '/home/bdbaddog/scons/as_scons'
<code>GlobalDict</code>	Value: None
<code>global_exports</code>	Value: {}
<code>sconscript_chdir</code>	Value: 1
<code>call_stack</code>	Value: []
<code>stack_bottom</code>	Value: '% Stack boTtom %'
<code>__package__</code>	Value: '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> args, message
	<i>Inherited from object</i> <code>__class__</code>

32.4 Class Frame

A frame on the SConstruct/SConscript call stack

32.4.1 Methods

<code>__init__(self, fs, exports, sconscript)</code>
x. <code>__init__</code> (...) initializes x; see <code>help(type(x))</code> for signature Overrides: object. <code>__init__</code> extit(inherited documentation)

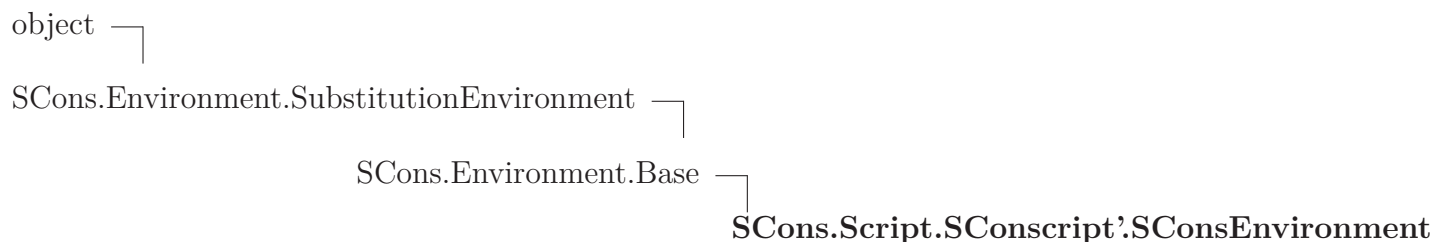
Inherited from object

`__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)

Help (<i>self</i> , <i>text</i>)

Import (<i>self</i> , * <i>vars</i>)

SConscript (<i>self</i> , * <i>ls</i> , ** <i>kw</i>)
--

SConscriptChdir (<i>self</i> , <i>flag</i>)
--

SetOption (<i>self</i> , <i>name</i> , <i>value</i>)

Inherited from SCons.Environment.Base(Section 8.9)

Action(), AddPostAction(), AddPreAction(), Alias(), AlwaysBuild(), Append(), AppendENVPPath(), 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(), PrependENVPPath(), 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__	

32.5.3 Class Variables

Name	Description
	<i>Inherited from <code>SCons.Environment.Base</code> (Section 8.9)</i>
<code>memoizer_counters</code>	
	<i>Inherited from <code>SCons.Environment.SubstitutionEnvironment</code> (Section 8.6)</i>
<code>__metaclass__</code>	

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>
x. <code>__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>	Value: 'src/engine/SCons/Sig.py rel_2.3.5:3347:d31d5a4e74b6 2015...
<code>__doc__</code>	Value: ""Place-holder for the old SCons.Sig module hierar...
<code>msg</code>	Value: 'The SCons.Sig module no longer exists.\n Remove the f...
<code>default_calc</code>	Value: None
<code>default_module</code>	Value: None
<code>MD5</code>	Value: MD5Null()
<code>TimeStamp</code>	Value: TimeStampNull()
<code>__package__</code>	Value: 'SCons'

33.2 Class MD5Null



33.2.1 Methods

<code>__repr__(self)</code>
<code>repr(x)</code> Overrides: <code>object.__repr__</code> extit(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>exit</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

SetAllowableExceptions(**excepts*)

raise_exception(*exception, target, s*)

quote_spaces(*arg*)

Generic function for putting double quotes around any string that has white space in it.

escape_list(*mylist, escape_func*)

Escape a list of arguments by running the specified `escape_func` on every object in the list that has an `escape()` method.

subst_dict(*target, source*)

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={},
lvvars={}, 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={}, lvvars={}, 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>	Value: 'src/engine/SCons/Subst.py rel_2.3.5:3347:d31d5a4e74b6 20...
<code>AllowableExceptions</code>	Value: (<type 'exceptions.IndexError'>, <type 'exceptions.NameEr...
<code>NullNodesList</code>	Value: Null(0x7F056D4E40D0)

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__** **__exit__**(inherited documentation)

__str__(*self*)

str(x) Overrides: object.**__str__** **__exit__**(inherited documentation)

escape(*self*, *escape_func*)

for_signature(*self*)

is_literal(*self*)

__eq__(*self*, *other*)

__neq__(*self*, *other*)

Inherited from object

__delattr__(*self*), **__format__**(*self*), **__getattr__**(*self*), **__hash__**(*self*), **__new__**(*cls*, *args, **kwargs), **__reduce__**(*self*), **__reduce_ex__**(*self*, protocol), **__repr__**(*self*), **__setattr__**(*self*, name, value), **__sizeof__**(*self*), **__subclasshook__**(*self*)

34.3.2 Properties

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

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

<code>__init__(self, lstr, for_signature=None)</code>
The <code>for_signature</code> parameter, if supplied, will be the canonical string we return from <code>for_signature()</code> . Else we will simply return <code>lstr</code> . Overrides: <code>object.__init__</code>

<code>__str__(self)</code>
<code>str(x)</code> Overrides: <code>object.__str__</code> <code>exit</code> (inherited documentation)

<code>escape(self, escape_func)</code>
--

<code>for_signature(self)</code>

<code>is_literal(self)</code>

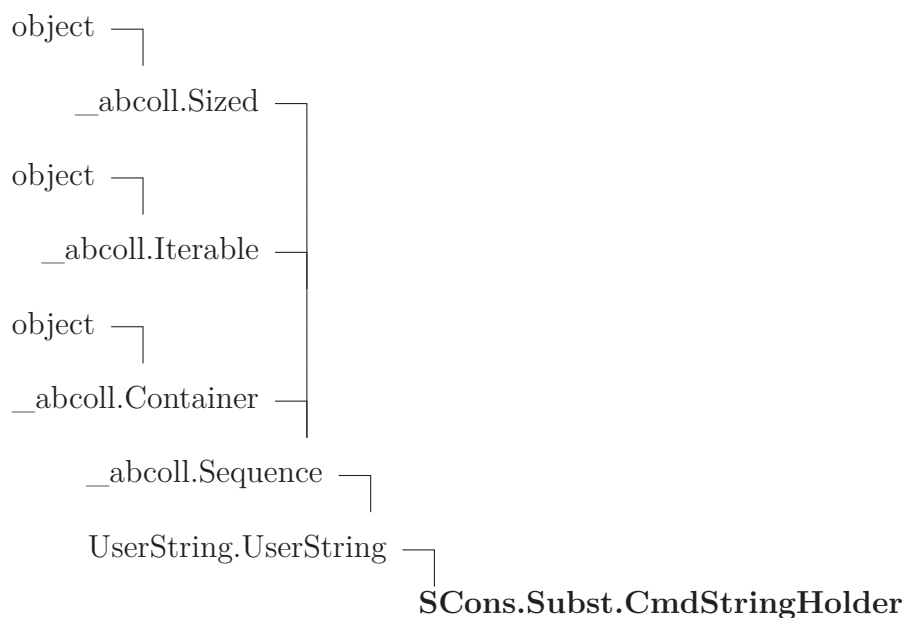
Inherited from object

`__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__</code> (<i>self</i> , <i>cmd</i> , <i>literal=None</i>) <i>x</i> . <code>__init__</code> (...) initializes <i>x</i> ; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)
<code>is_literal</code> (<i>self</i>)

```
escape(self, escape_func, quote_func=<function quote_spaces at
0x7f056d4ca500>)
```

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(),
istitle(), 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)

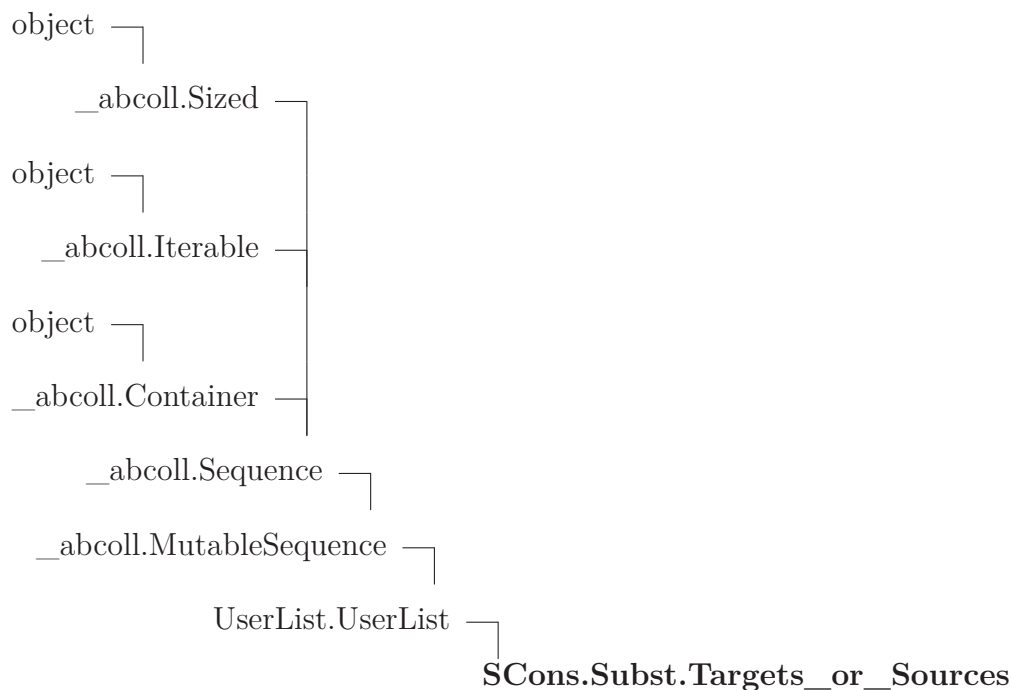
Inherited from object

```
__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__` `exitit` (inherited documentation)

`__getattr__(self, attr)`

`__getitem__(self, i)`

Overrides: `_abcoll.Sequence.__getitem__`

<code>__getitem__(self, i, j)</code>

Overrides: UserList.UserList.__getitem__
--

<code>__str__(self)</code>

str(x) Overrides: object.__str__ exit(herited documentation)
--

<code>__repr__(self)</code>

repr(x) Overrides: object.__repr__ exit(herited 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>exit</code> (inherited documentation)
--

<code>__getattr__(self, attr)</code>

<code>__str__(self)</code>

<code>str(x)</code> Overrides: <code>object.__str__</code> <code>exit</code> (inherited documentation)
--

<code>__repr__(self)</code>

<code>repr(x)</code> Overrides: <code>object.__repr__</code> <code>exit</code> (inherited documentation)
--

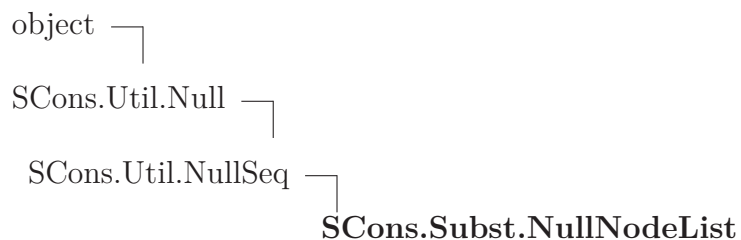
Inherited from object

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

34.8.2 Properties

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

34.9 Class `NullNodeList`



34.9.1 Methods

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

Overrides: <code>SCons.Util.Null.__call__</code>
--

<code>__str__(self)</code>

<code>str(x)</code> Overrides: <code>object.__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__()`, `__getattr__()`, `__hash__()`, `__reduce__()`, `__reduce_ex__()`,
`__sizeof__()`, `__subclasshook__()`

34.9.2 Properties

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

<code>dump_stats()</code>

<code>find_cycle(stack, visited)</code>

35.2 Variables

Name	Description
<code>__doc__</code>	Value: ...
<code>__revision__</code>	Value: 'src/engine/SCons/Taskmaster.py rel_2.3.5:3347:d31d5a4e74...
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	Value: 0
CollectStats	Value: None
StatsNodes	Value: []
fmt	Value: '%(considered)3d %(already_handled)3d %(problem)3d %(chil...
__package__	Value: 'SCons'

35.3 Class 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

<code>__init__(self)</code>
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



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>	
__class__	

35.6 Class OutOfDateTask



Known Subclasses: SCons.Script.Main.BuildTask

35.6.1 Methods

needs_execute (<i>self</i>)

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>	
<code>__class__</code>	

35.7 Class Taskmaster



The Taskmaster for walking the dependency DAG.

35.7.1 Methods

__init__ (<i>self</i> , <i>targets</i> =[], <i>tasker</i> =None, <i>order</i> =None, <i>trace</i> =None)
--

x. <code>__init__</code> (...) initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>__extit__</code> (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 0x7f056d20d140>)

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 0x7f056d4cbc08>)
```

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 0x7f056d4cbc08>)
```

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 0x7f056d4cbd70>,
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 0x7f056d4cbde8>, 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 $O(N*\log_2(N))$ 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

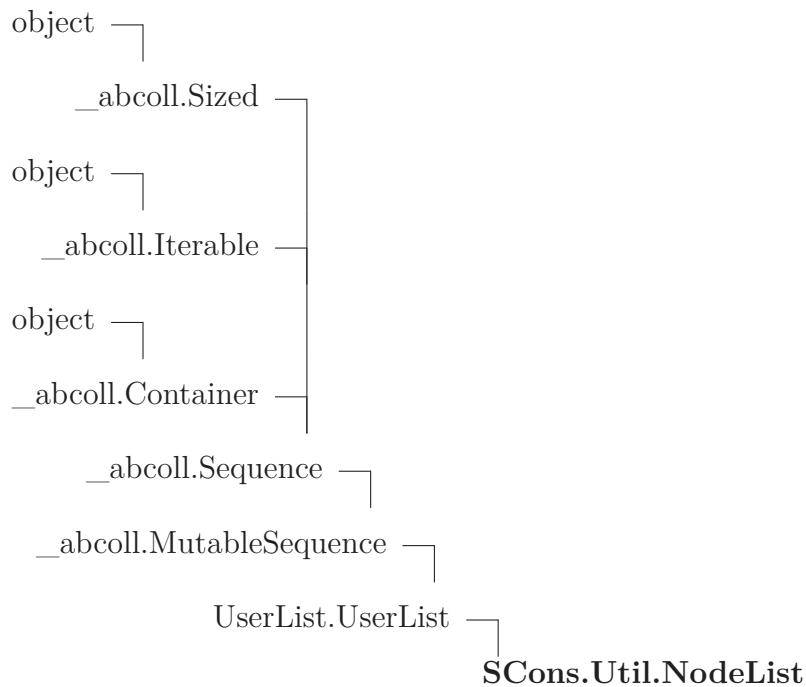
silent_intern(*x*)

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	Value: dict, UserDict
ListTypes	Value: (<type 'list'>, <class 'UserList.UserList'>)
SequenceTypes	Value: (<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>)
StringTypes	Value: (<type 'str'>, <type 'unicode'>, <class 'UserString.UserS...>)
BaseStringTypes	Value: (<type 'str'>, <type 'unicode'>)
d	Value: {}
can_read_reg	Value: 0
hkey_mod	Value: win32con
RegEnumKey	Value: win32api.RegEnumKey
RegEnumValue	Value: win32api.RegEnumValue
RegQueryValueEx	Value: win32api.RegQueryValueEx
HKEY_CLASSES_ROOT	Value: None
HKEY_LOCAL_MACHINE	Value: None
HKEY_CURRENT_USER	Value: None
HKEY_USERS	Value: None
display	Value: DisplayEngine()
md5	Value: True
__package__	Value: '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__()`, `__getattribute__()`, `__new__()`, `__reduce__()`,
`__reduce_ex__()`, `__setattr__()`, `__sizeof__()`

36.3.2 Properties

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

36.3.3 Class Variables

Name	Description
<i>Inherited from <code>UserList.UserList</code></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>	Value: 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__` `__exit__` (inherited documentation)

```
__get__(self, obj, cls)
```

Inherited from object

```

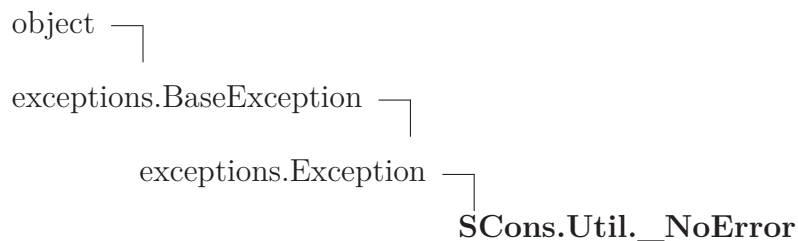
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

```

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__()`

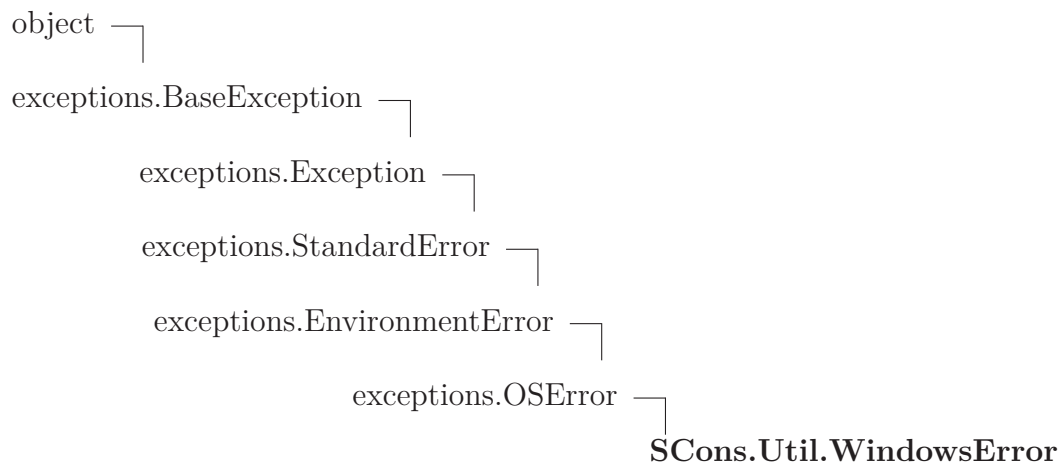
Inherited from `object`

`__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__()`

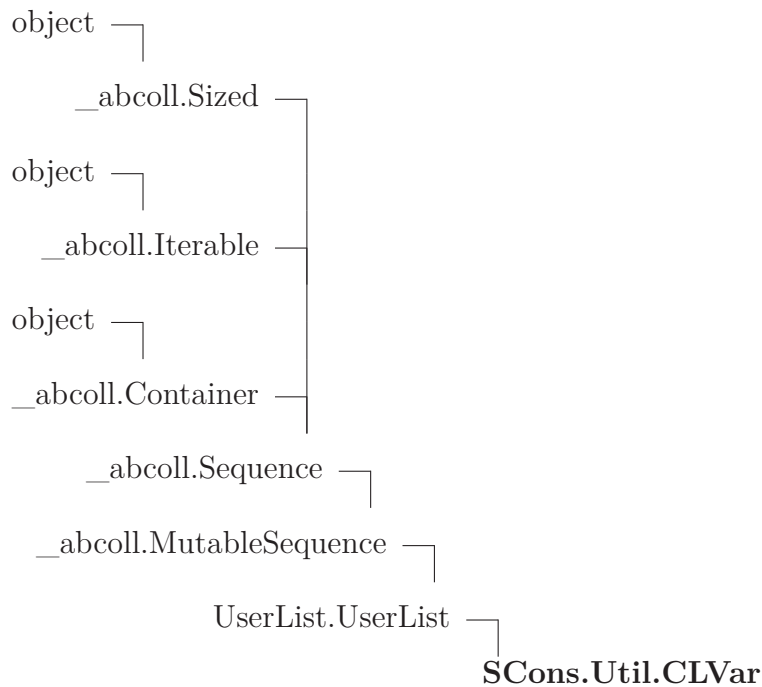
Inherited from `object`

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

36.8.2 Properties

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

36.9 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)
```

<code>__str__(self)</code>

<code>str(x)</code> Overrides: <code>object.__str__</code> <code>exit</code> (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 <code>object</code></i>	
<code>__class__</code>	

36.9.3 Class Variables

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

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`

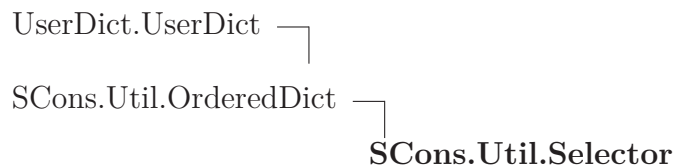
Inherited from UserDict.UserDict

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

36.10.2 Class Variables

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

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>	
<code>__hash__</code>	

36.12 Class LogicalLines



36.12.1 Methods

__init__ (<i>self</i> , <i>fileobj</i>) <i>x</i> . __init__ (...) initializes <i>x</i> ; see help(type(<i>x</i>)) for signature Overrides: object. __init__ extit(inherited documentation)

readline (<i>self</i>)

readlines (<i>self</i>)

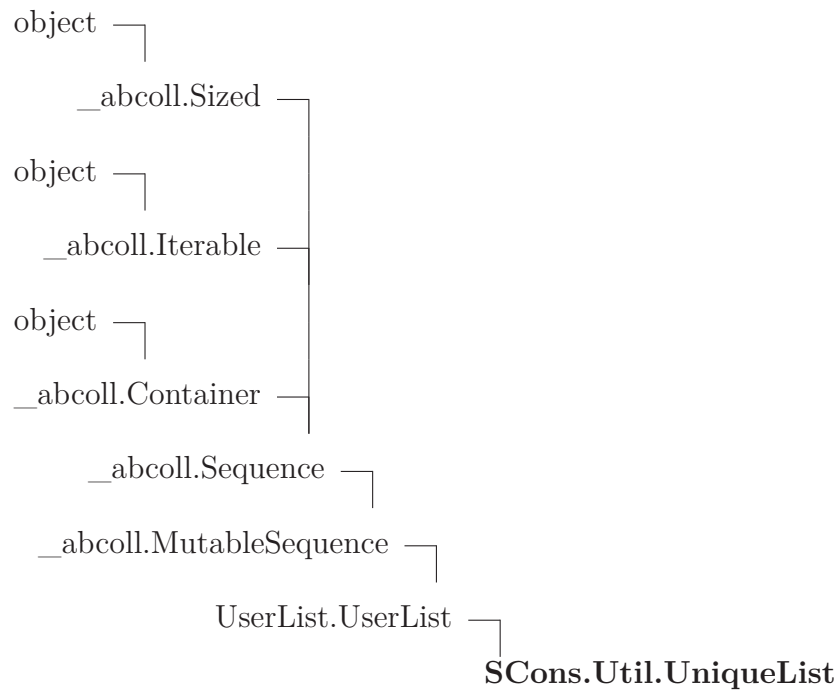
Inherited from object

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

36.12.2 Properties

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

36.13 Class UniqueList



36.13.1 Methods

__init__(*self*, *seq*=[])

x.**__init__**(...) initializes x; see help(type(x)) for signature Overrides: object.**__init__** extit(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

<code>__init__(self, file)</code>

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

<code>write(self, arg)</code>

<code>__getattr__(self, attr)</code>

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

<code>__new__(cls, *args, **kwargs)</code>
--

Return Value

a new object with type S, a subtype of T

Overrides: <code>object.__new__</code> <code>exitit</code> (inherited documentation)
--

```
__init__(self, *args, **kwargs)
```

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

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

```
__repr__(self)
```

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

```
__nonzero__(self)
```

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

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

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

```
__delattr__(self, name)
```

x.__delattr__('name') <==> del x.name Overrides: object.__delattr__
exitit(inherited documentation)

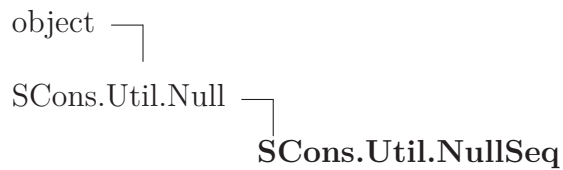
Inherited from object

```
__format__(), __getattr__(), __hash__(), __reduce__(), __reduce_ex__(),  
__sizeof__(), __str__(), __subclasshook__()
```

36.15.2 Properties

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

36.16 Class NullSeq



Known Subclasses: SCons.Subst.NullNodeList

36.16.1 Methods

<code>__len__(self)</code>

<code>__iter__(self)</code>

<code>__getitem__(self, i)</code>

<code>__delitem__(self, i)</code>

<code>__setitem__(self, i, v)</code>

Inherited from SCons.Util.Null(Section 36.15)

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

Inherited from object

`__format__()`, `__getattr__()`, `__hash__()`, `__reduce__()`, `__reduce_ex__()`,
`__sizeof__()`, `__str__()`, `__subclasshook__()`

36.16.2 Properties

Name	Description
<i>Inherited from object</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. 340*)
- **EnumVariable** (*Section ??, p. ??*)
- **EnumVariable'**: engine.SCons.Variables.EnumVariable
(*Section 39, p. 341*)
- **ListVariable** (*Section ??, p. ??*)
- **ListVariable'**: engine.SCons.Variables.ListVariable
(*Section 40, p. 343*)
- **PackageVariable** (*Section ??, p. ??*)
- **PackageVariable'**: engine.SCons.Variables.PackageVariable
(*Section 41, p. 344*)
- **PathVariable** (*Section ??, p. ??*)
- **PathVariable'**: SCons.Variables.PathVariable
(*Section 42, p. 345*)

37.2 Variables

Name	Description
__revision__	Value: 'src/engine/SCons/Variables/__init__.py rel_2.3.5:3347:d3...
__package__	Value: '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

GenerateHelpText (<i>self</i> , <i>env</i> , <i>sort</i> =None)

Generate the help text for the options.

env - an environment that is used to get the current values of the options.

FormatVariableHelpText (<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. Value: None
format	Value: '\n%s: %s\n default: %s\n actual: %s\n'
format_	Value: '\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

BoolVariable (<i>key</i> , <i>help</i> , <i>default</i>)
<hr/>
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

<p>ListVariable(<i>key, help, default, names, map={}</i>)</p> <hr/> <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 (separated by space).</p>
--

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

`PackageVariable`(*key*, *help*, *default*, *searchfunc*=None)

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 (seperated by space).

42 Module `SCons.Variables.PathVariable`

`SCons.Variables.PathVariable`

This file defines an option type for SCons implementing path settings.

To be used whenever a 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	Value: <code>SCons.Variables.PathVariable</code>

43 Module `SCons.Warnings`

`SCons.Warnings`

This file implements the warnings framework for `SCons`.

43.1 Functions

`suppressWarningClass(clazz)`

Suppresses all warnings that are of type *clazz* or derived from *clazz*.

`enableWarningClass(clazz)`

Enables all warnings that are of type *clazz* or derived from *clazz*.

`warningAsException(flag=1)`

Turn warnings into exceptions. Returns the old value of the flag.

`warn(clazz, *args)`

process_warn_strings(*arguments*)

Process string specifications of enabling/disabling warnings, as passed to the --warn option or the SetOption('warn') function.

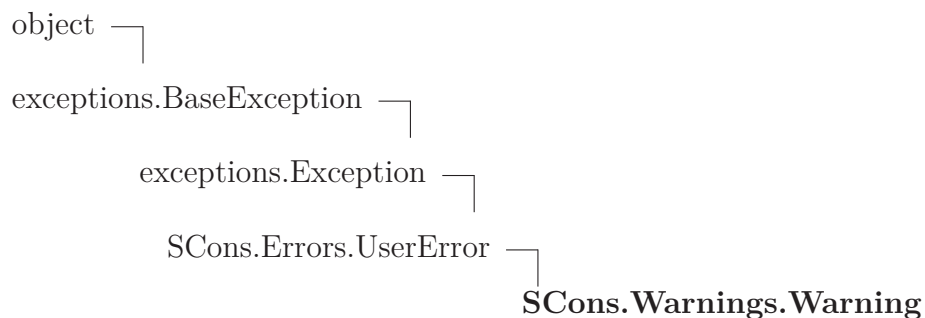
An argument to this option should be of the form <warning-class> or no-<warning-class>. The warning class is munged in order to get an actual class name from the classes above, which we need to pass to the {enable,disable}WarningClass() functions. The supplied <warning-class> is split on hyphens, each 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__	Value: 'src/engine/SCons/Warnings.py rel_2.3.5:3347:d31d5a4e74b6...
__package__	Value: '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__()`

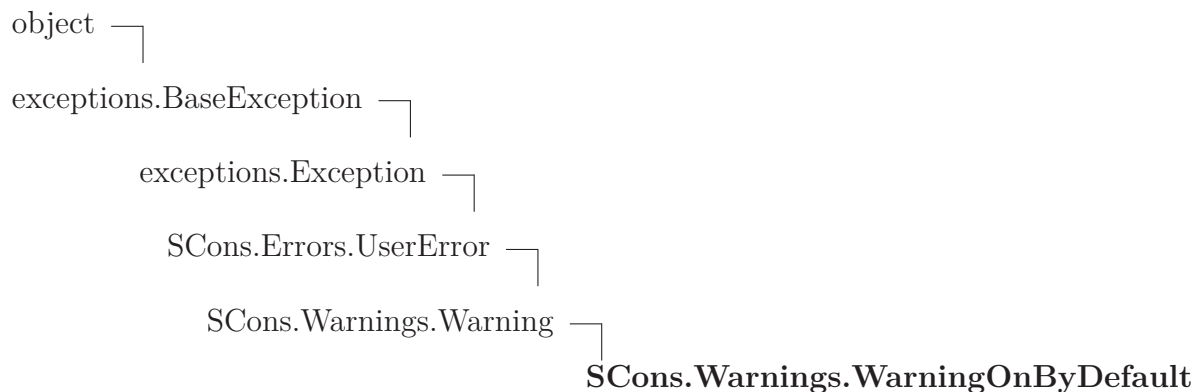
Inherited from object

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

43.3.2 Properties

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

43.4 Class `WarningOnByDefault`



Known Subclasses: `SCons.Warnings.CorruptSConsignWarning`, `SCons.Warnings.DevelopmentVersionWarning`, `SCons.Warnings.DuplicateEnvironmentWarning`, `SCons.Warnings.LinkWarning`, `SCons.Warnings.FutureRemovedWarning`, `SCons.Warnings.MisleadingKeywordsWarning`, `SCons.Warnings.MissingSConscriptWarning`, `SCons.Warnings.NoMD5ModuleWarning`, `SCons.Warnings.NoMetaclassSupportWarning`, `SCons.Warnings.NoParallelSupportWarning`, `SCons.Warnings.ReservedVariableWarning`, `SCons.Warnings.UnusedOptionWarning`, `SCons.Warnings.VisualCMissingWarning`, `SCons.Warnings.VisualVersionMismatch`

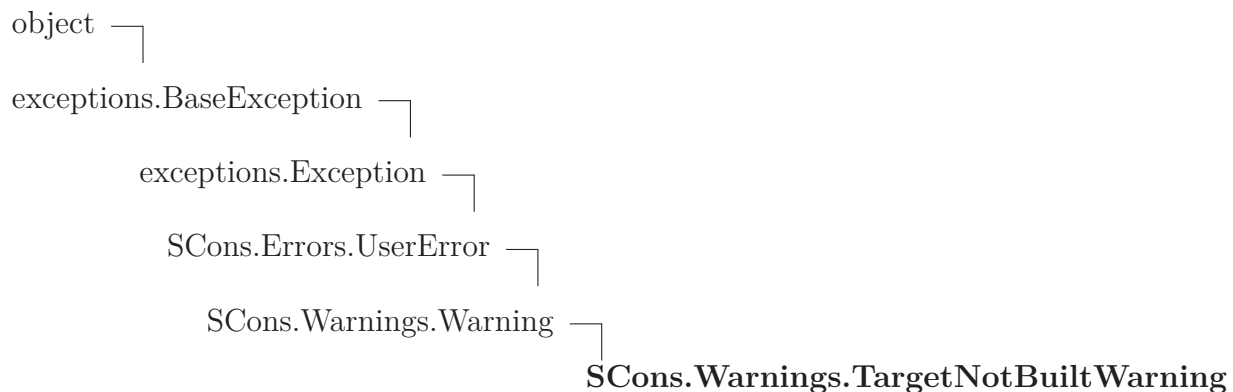
43.4.1 Methods

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

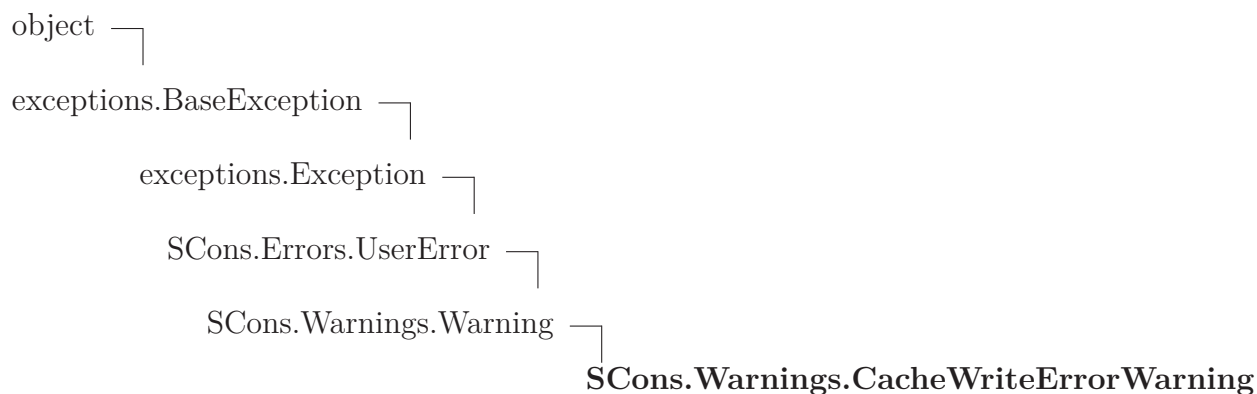
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

Inherited from object

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

43.5.2 Properties

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

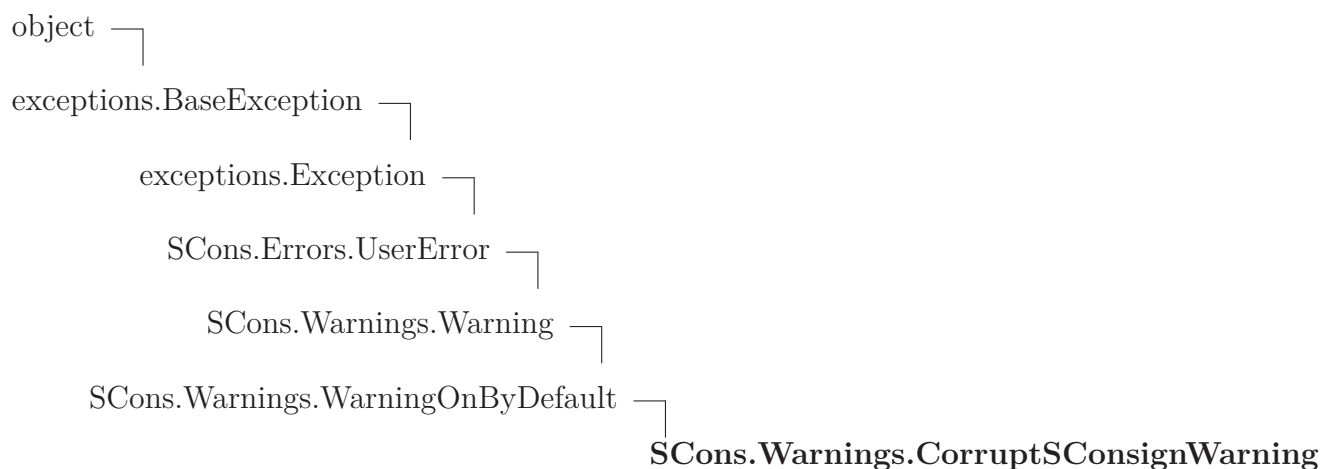
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

Inherited from object

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

43.6.2 Properties

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

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__()`

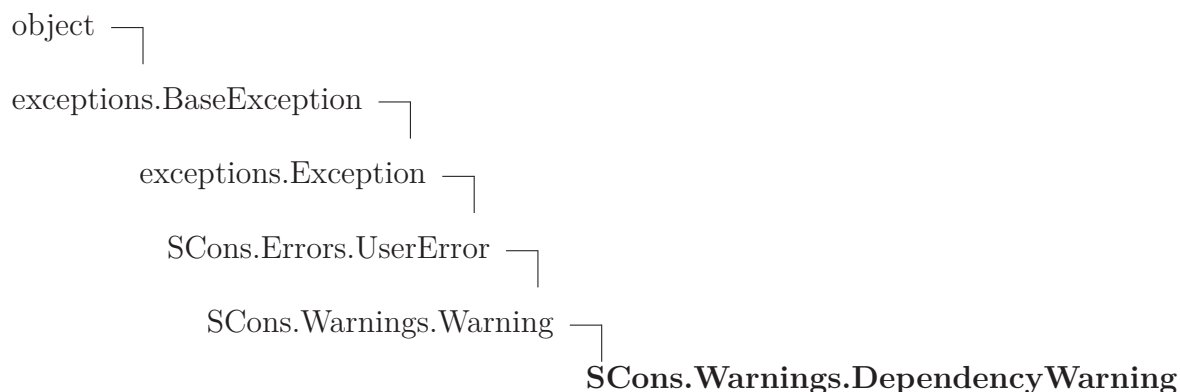
Inherited from object

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

43.7.2 Properties

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

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__()`

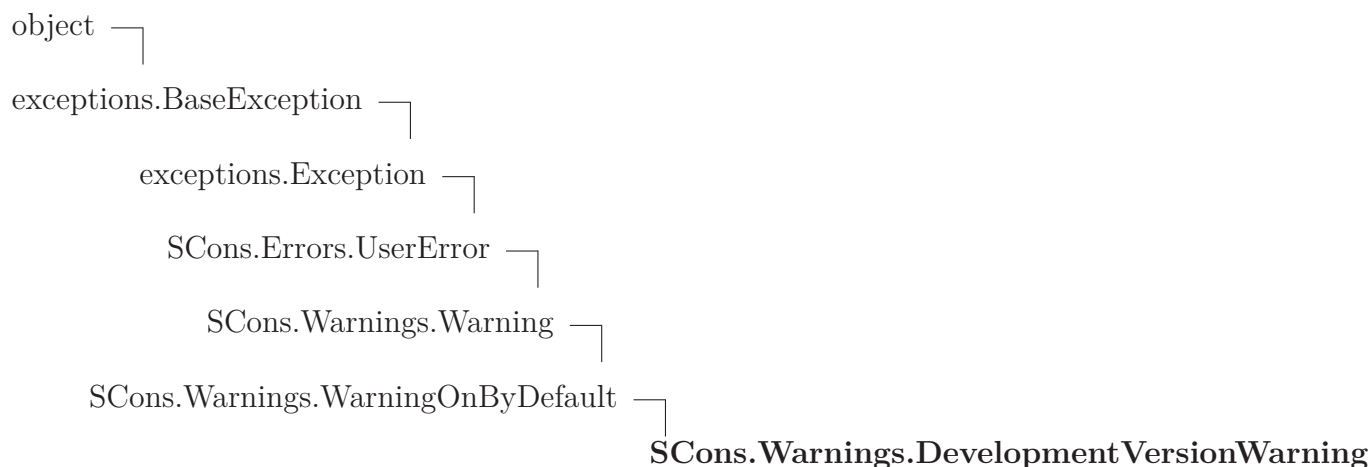
43.8.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></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__()

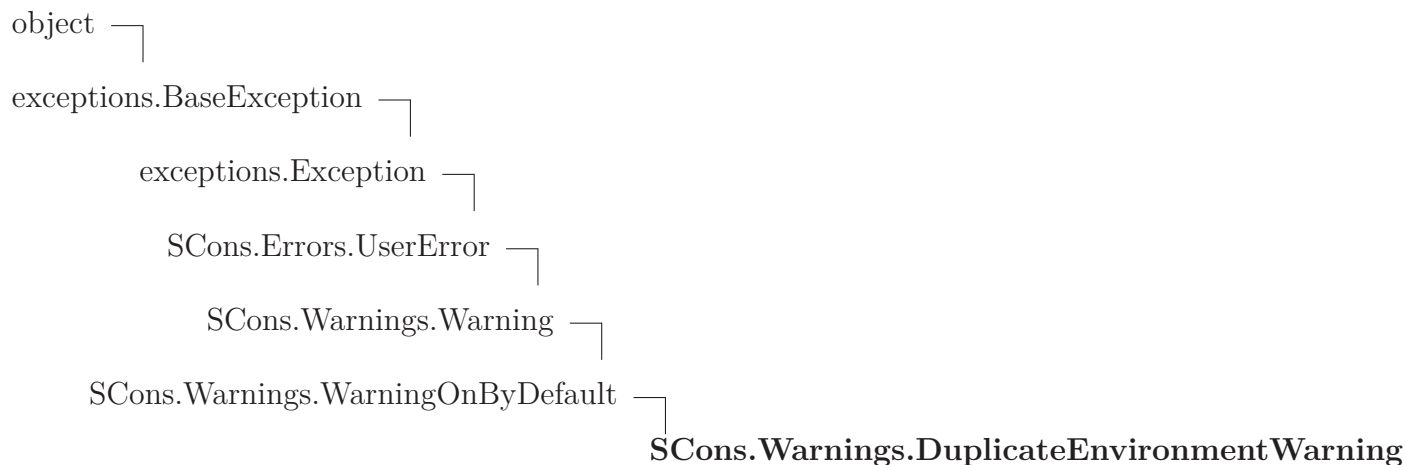
43.9.2 Properties

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__()

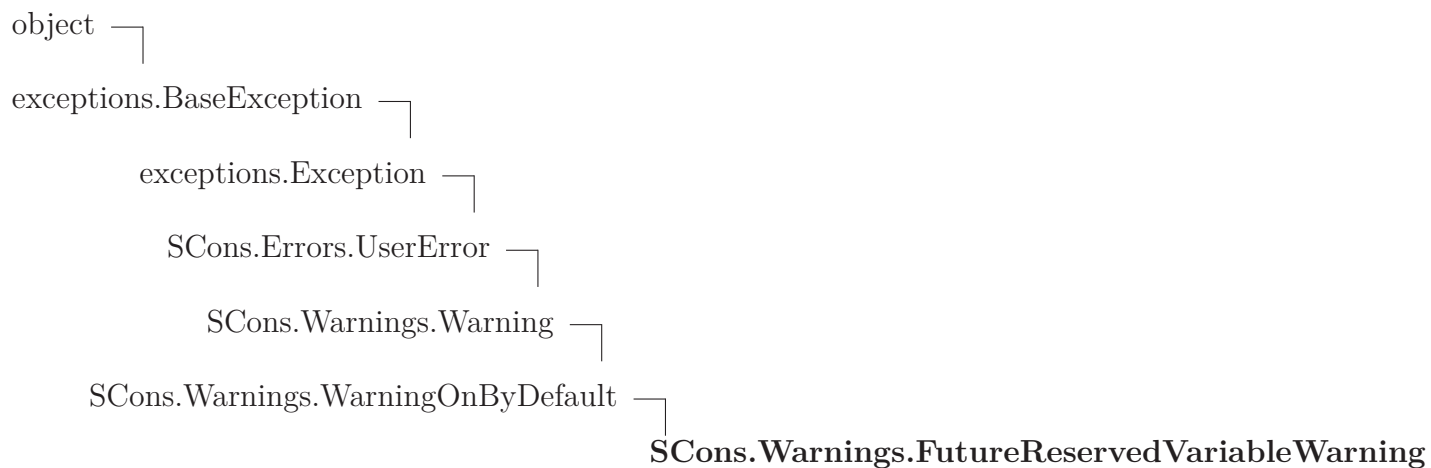
Inherited from object

__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__()`

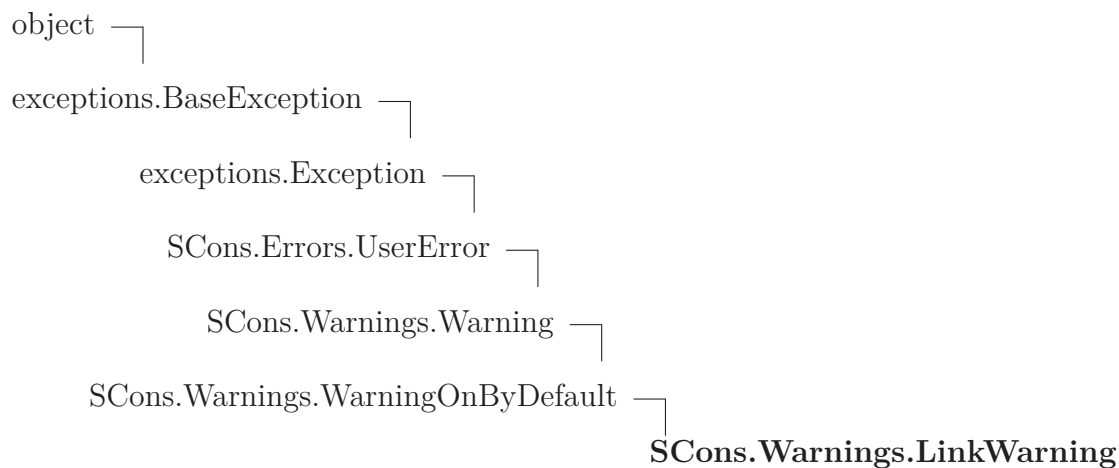
Inherited from object

`__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__()`

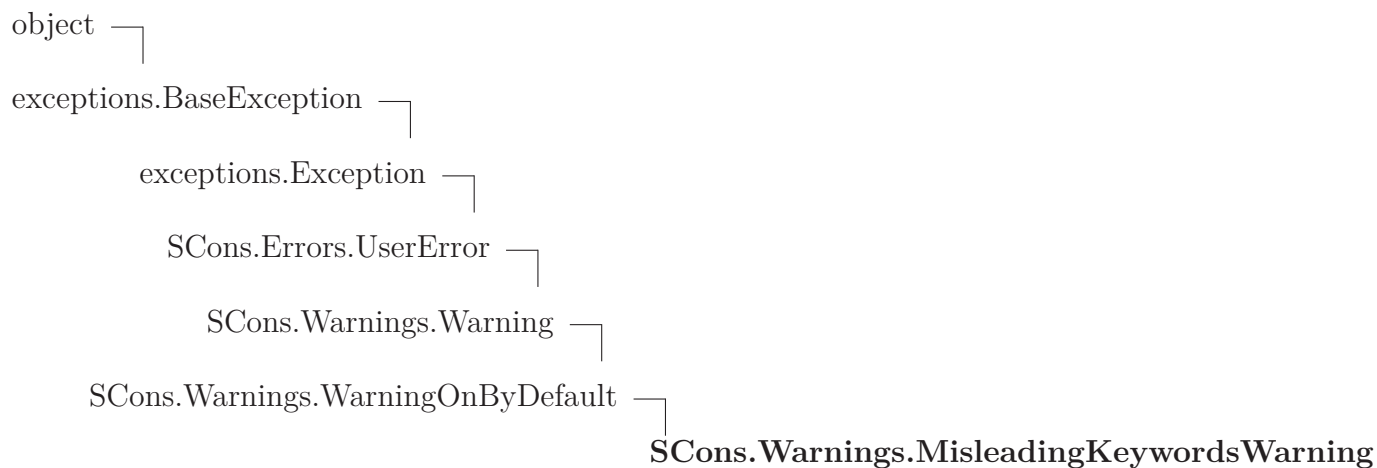
Inherited from object

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

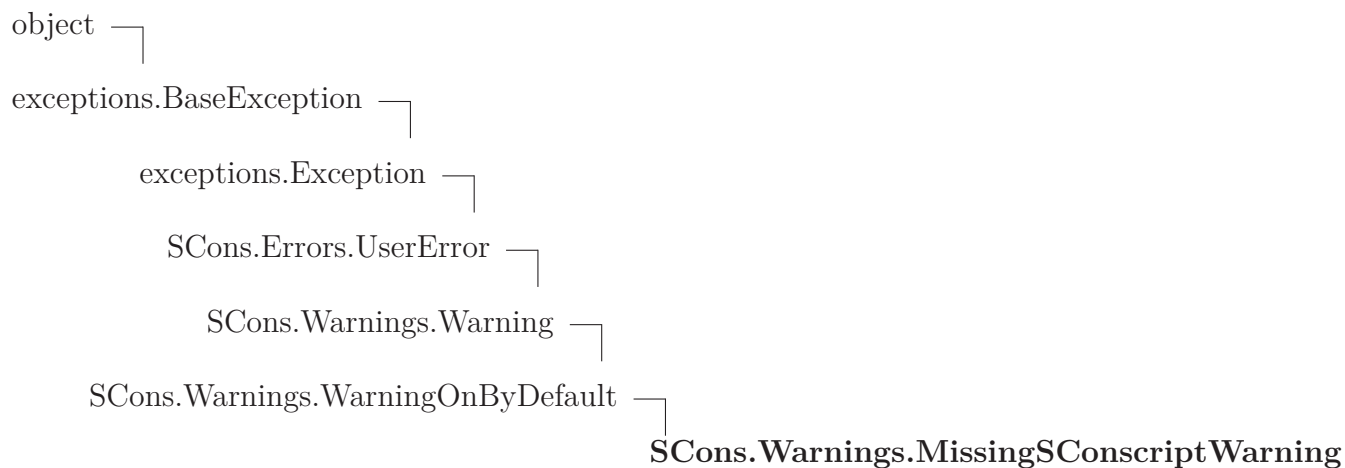
Inherited from `object`

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

43.13.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<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__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

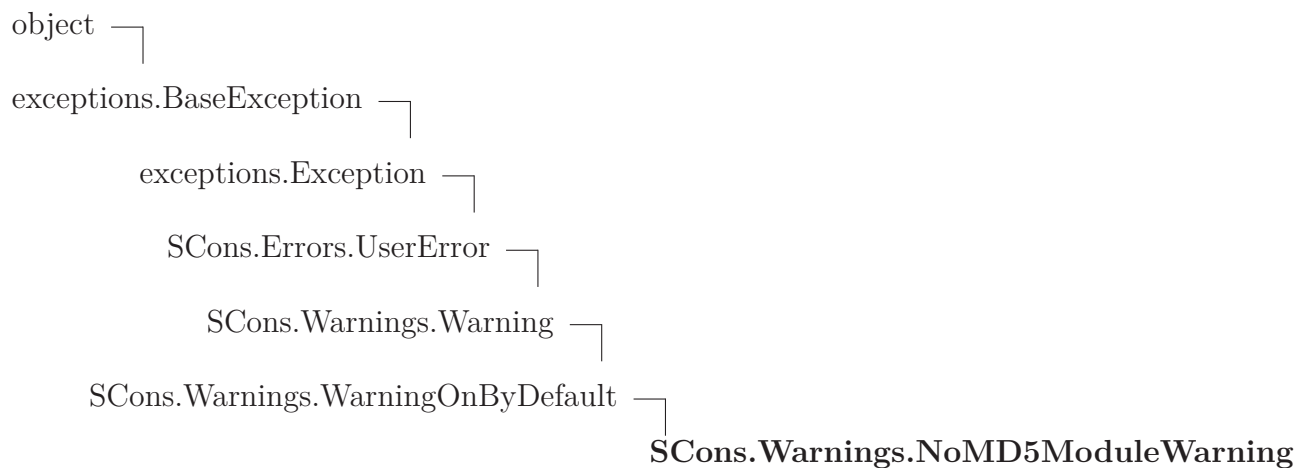
Inherited from `object`

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

43.14.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<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__()`

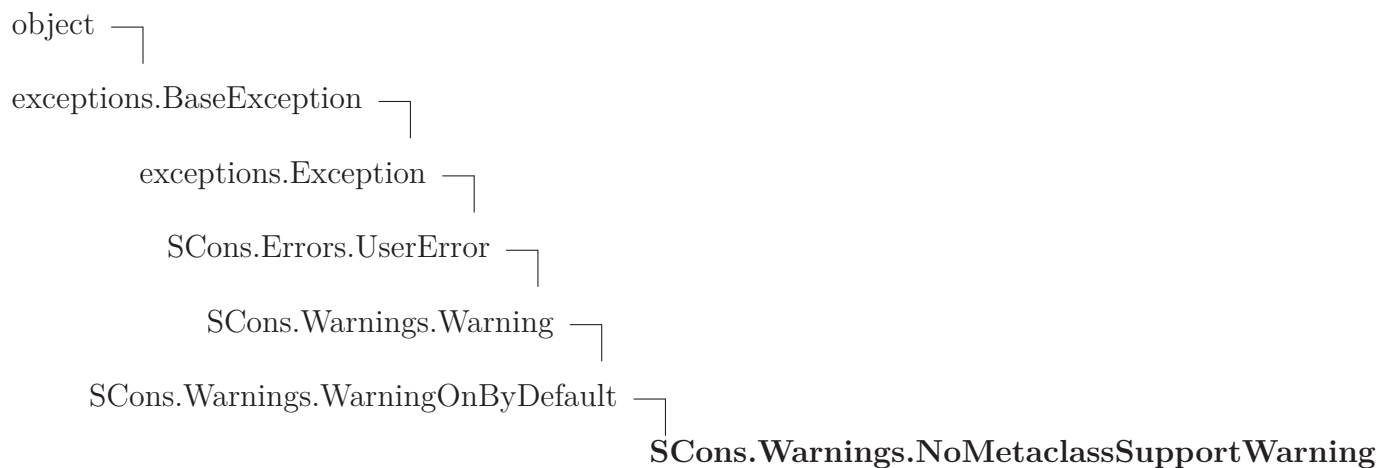
Inherited from object

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

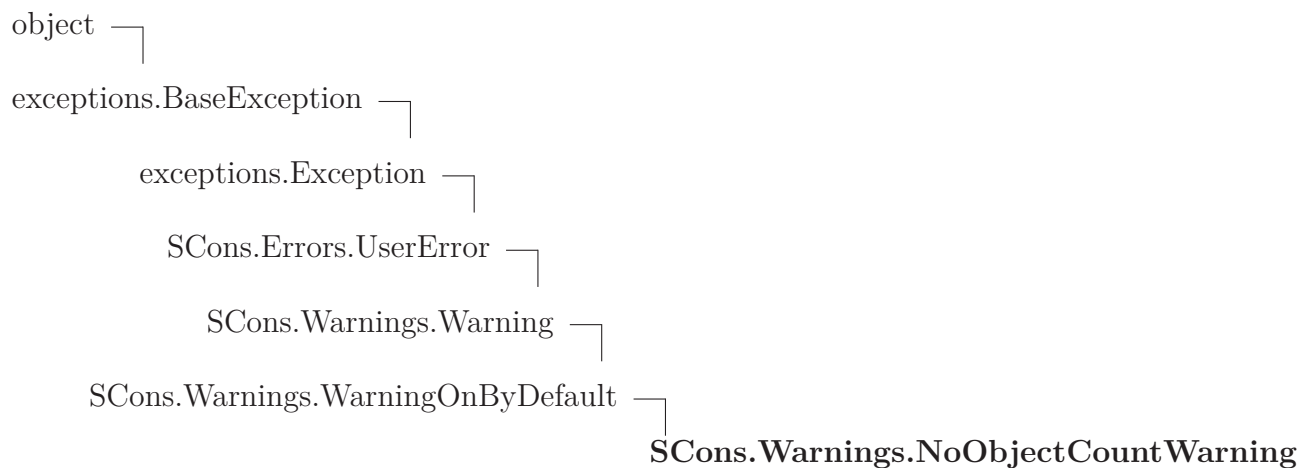
Inherited from object

`__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__()`

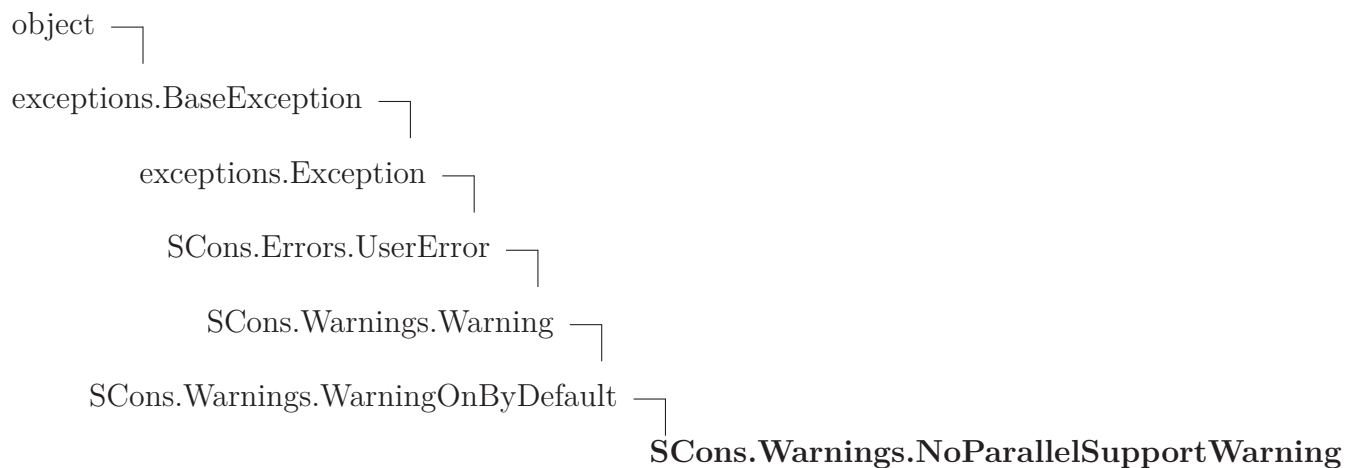
Inherited from object

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

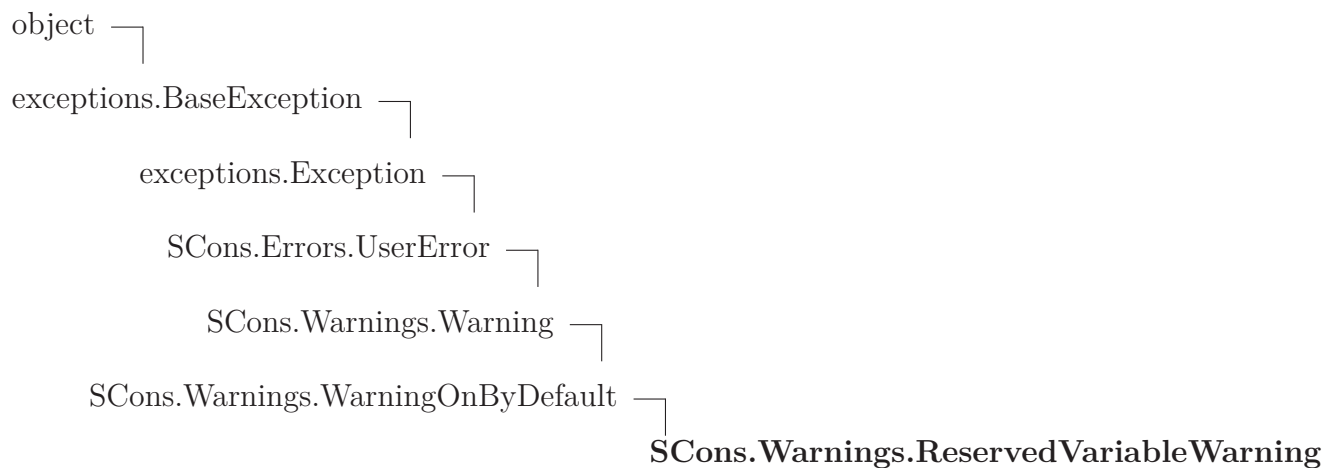
Inherited from object

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

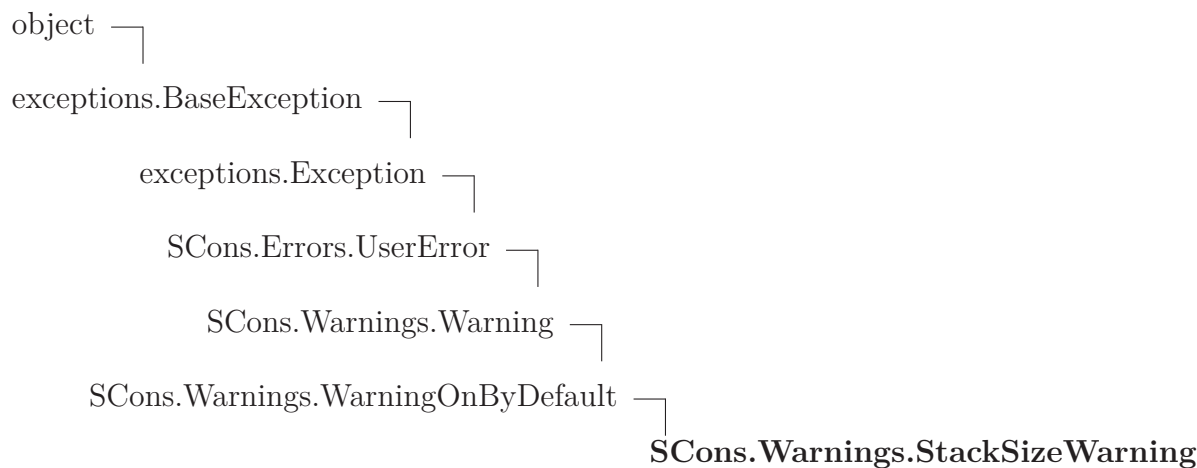
Inherited from `object`

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

43.19.2 Properties

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

43.20 Class StackSizeWarning



43.20.1 Methods

Inherited from exceptions.Exception

`__init__()`, `__new__()`

Inherited from exceptions.BaseException

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

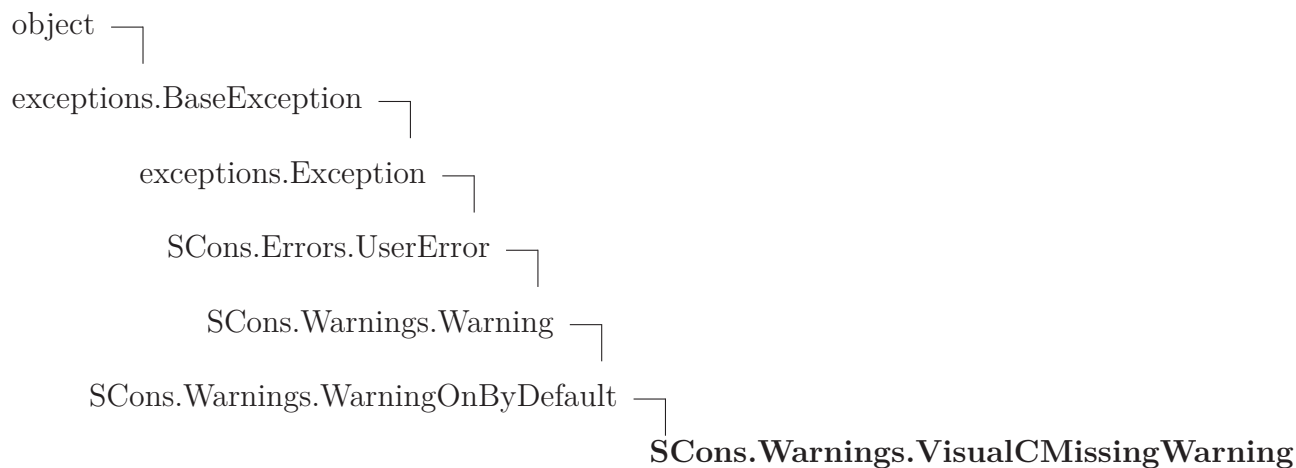
Inherited from object

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

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

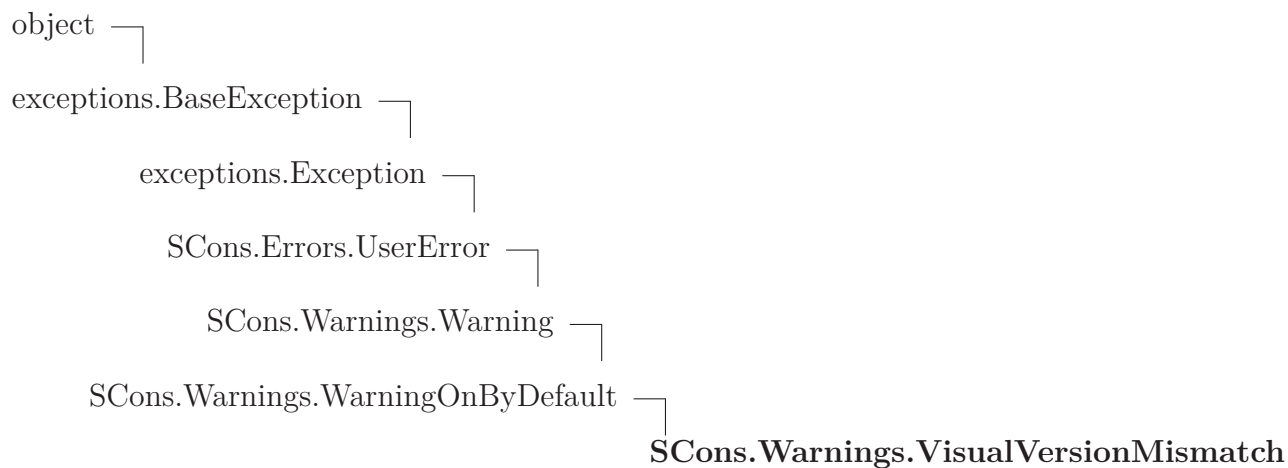
Inherited from object

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

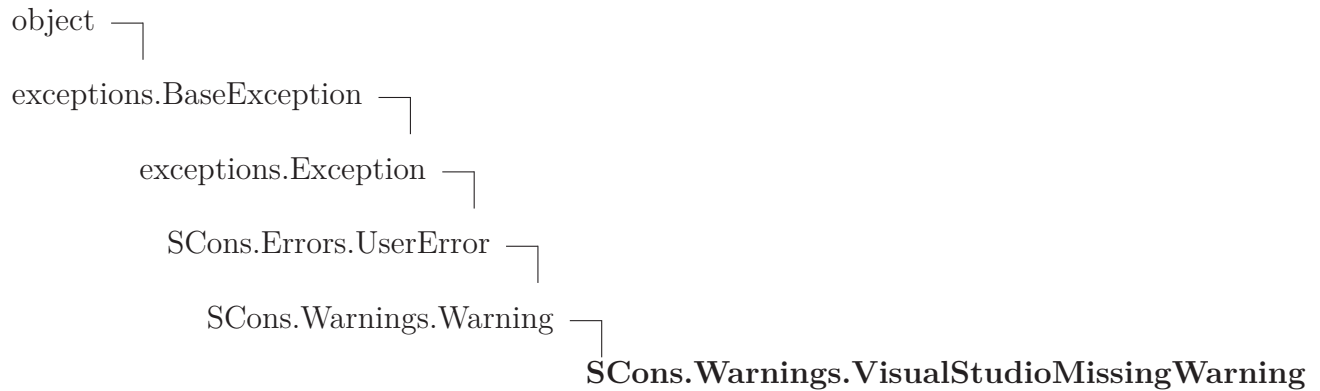
Inherited from `object`

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

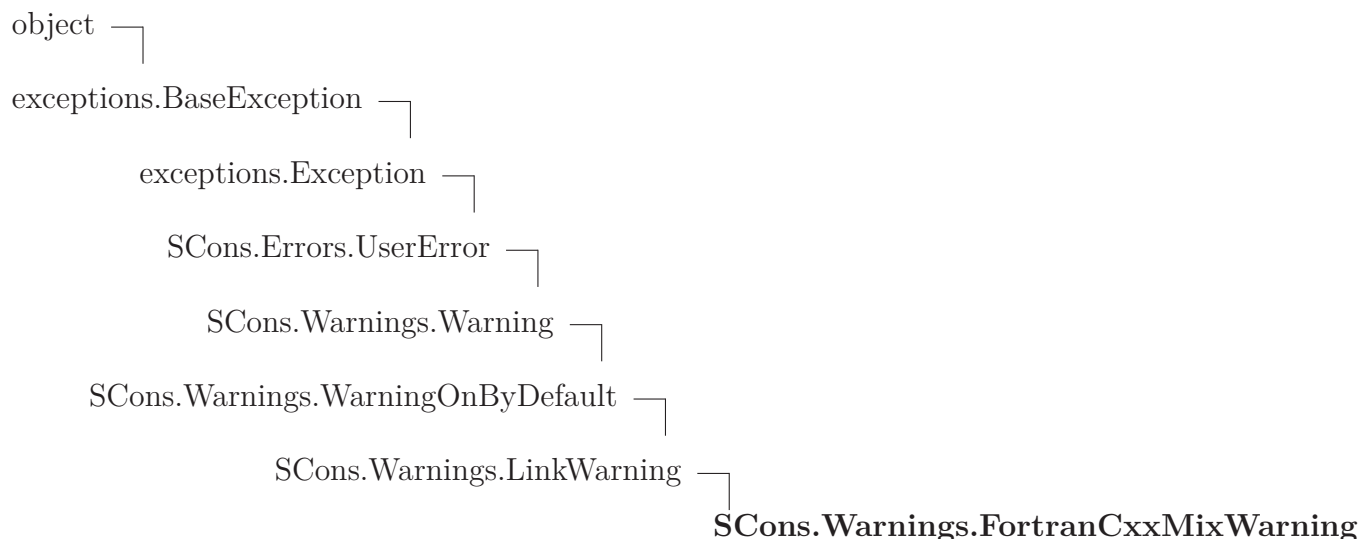
Inherited from `object`

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

43.23.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i> <code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></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__()`

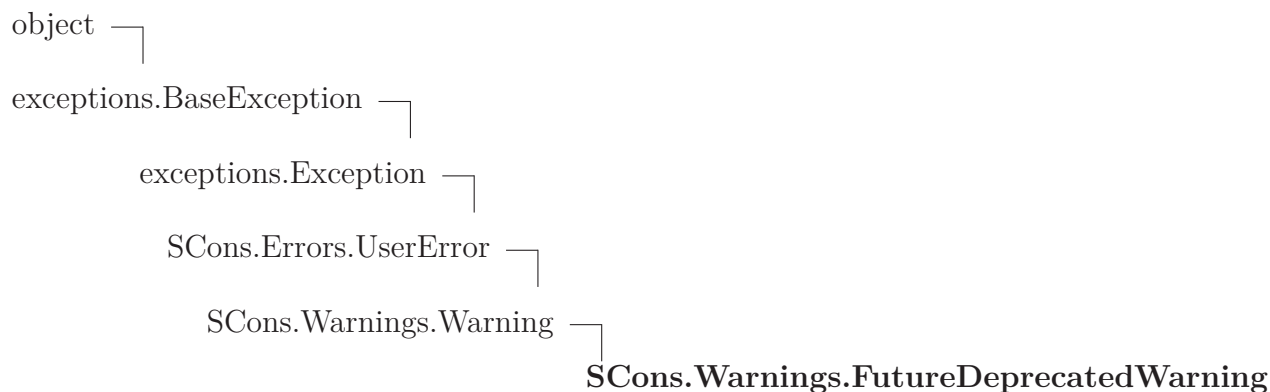
Inherited from object

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

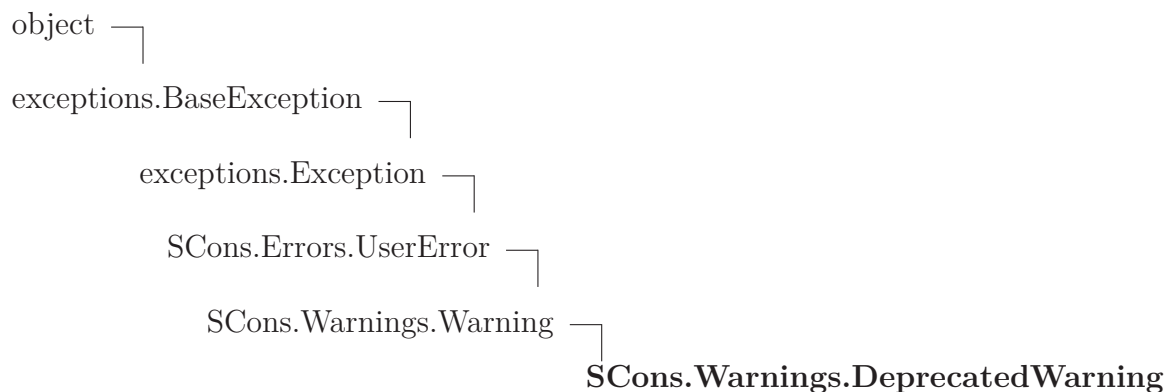
Inherited from object

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

43.25.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	args, message
<i>Inherited from object</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__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

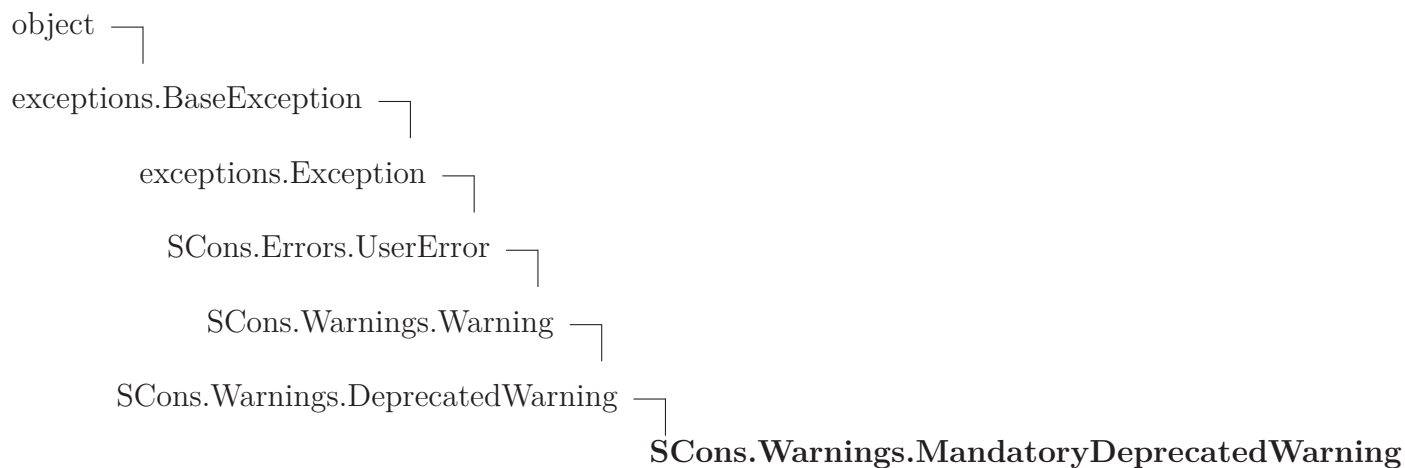
Inherited from `object`

`__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__()`

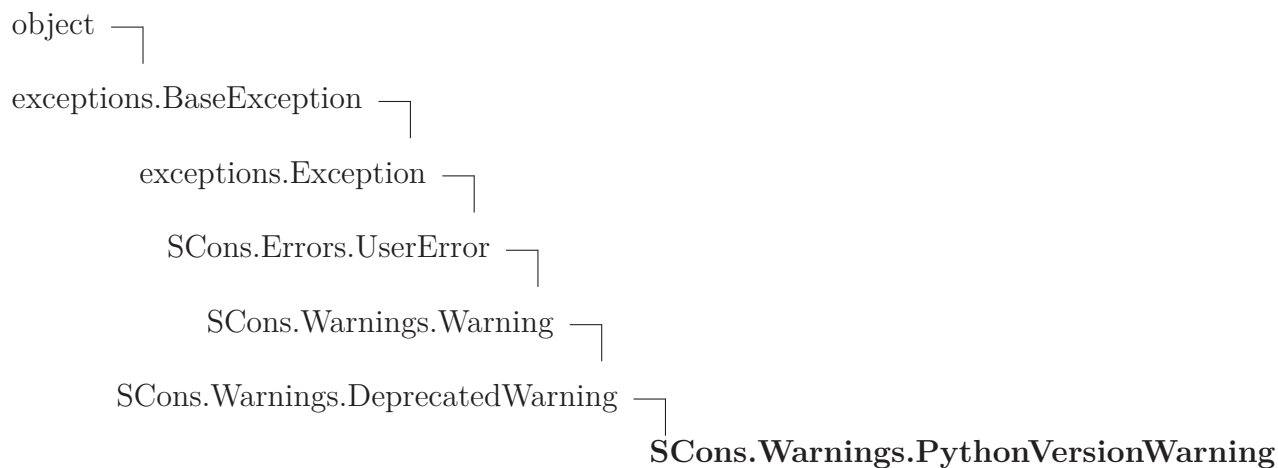
Inherited from object

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

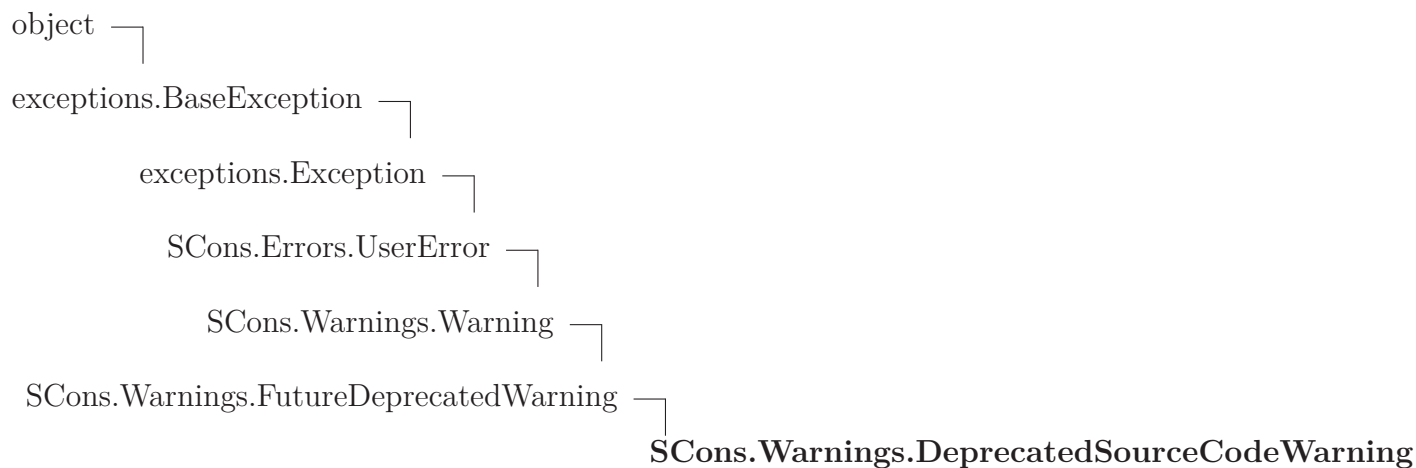
Inherited from object

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

43.28.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</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__()`

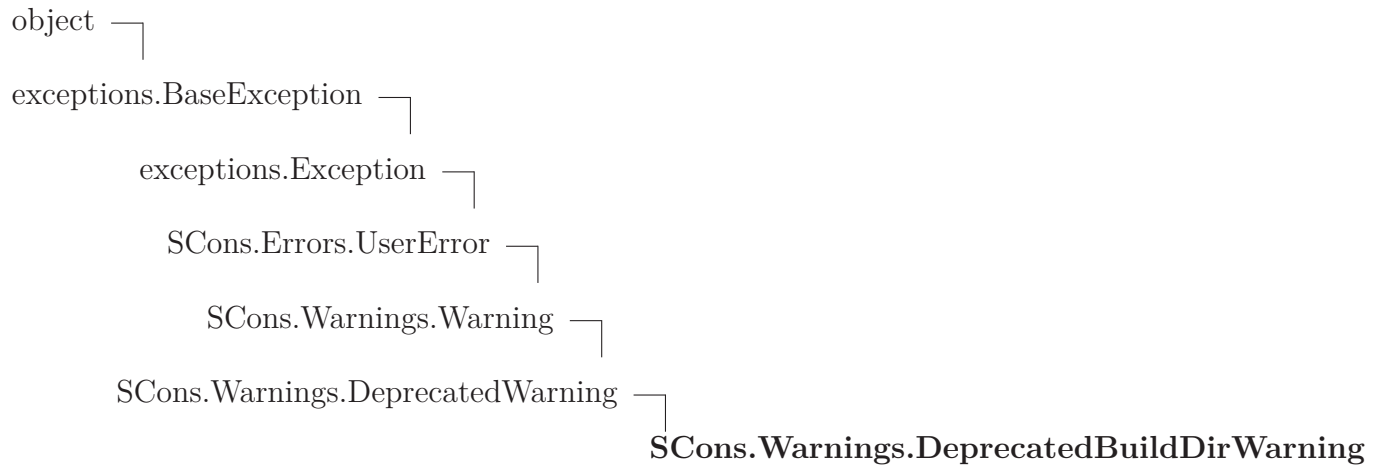
Inherited from `object`

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

43.29.2 Properties

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

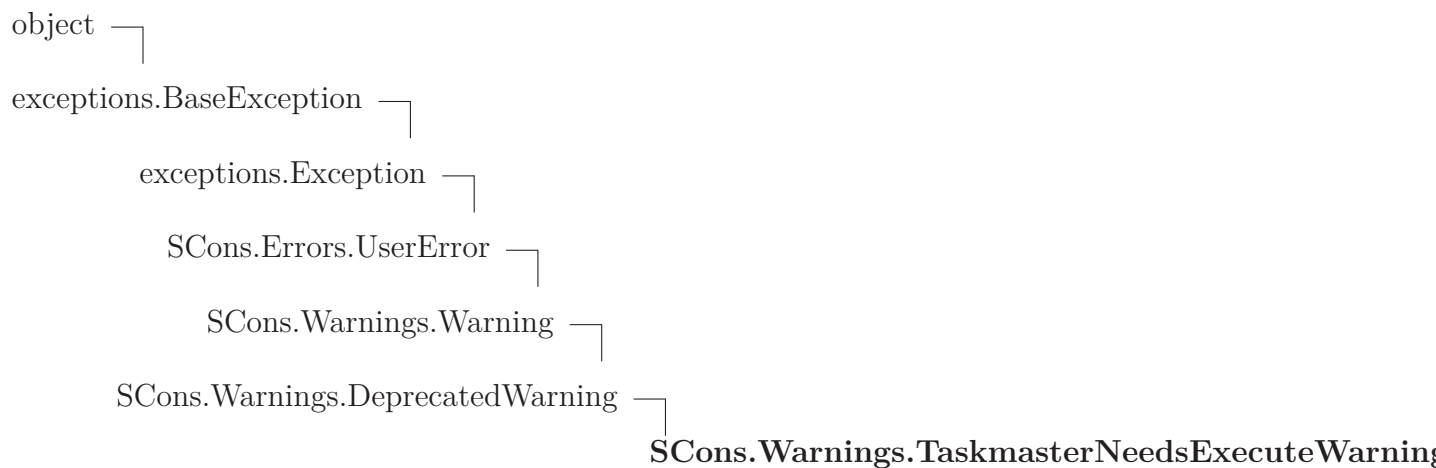
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

Inherited from `object`

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

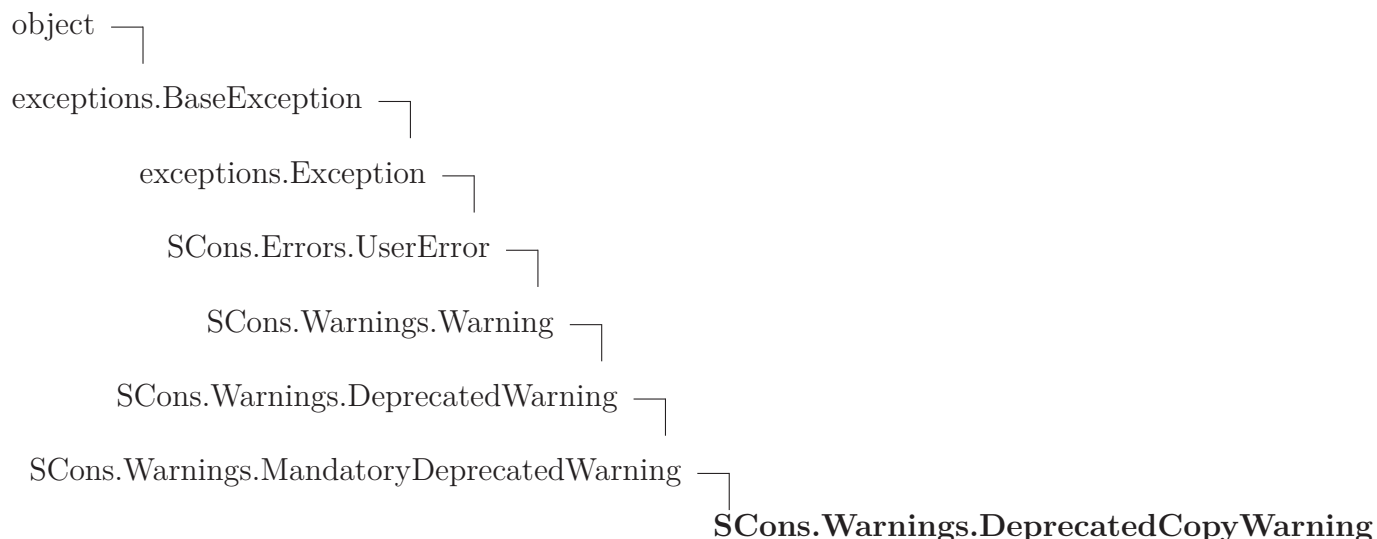
43.30.2 Properties

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

43.31 Class `TaskmasterNeedsExecuteWarning`**43.31.1 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__()`**43.31.2 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>	

43.32 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__()`

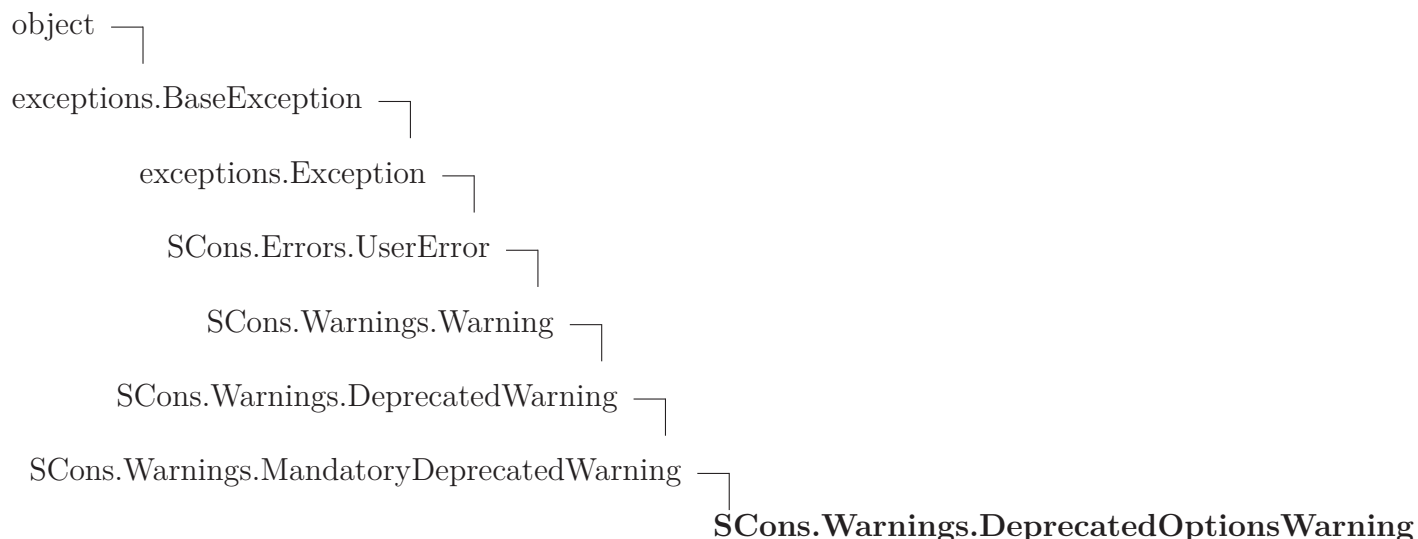
Inherited from `object`

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

43.32.2 Properties

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

43.33 Class `DeprecatedOptionsWarning`



43.33.1 Methods

Inherited from `exceptions.Exception`

`__init__()`, `__new__()`

Inherited from `exceptions.BaseException`

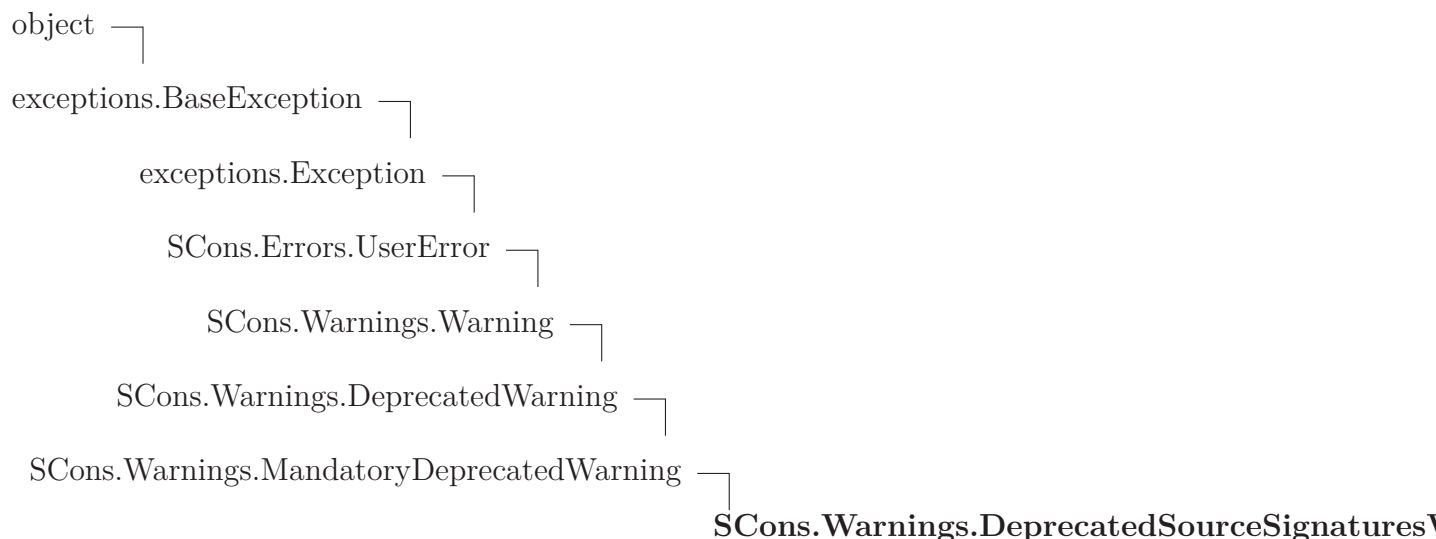
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

Inherited from `object`

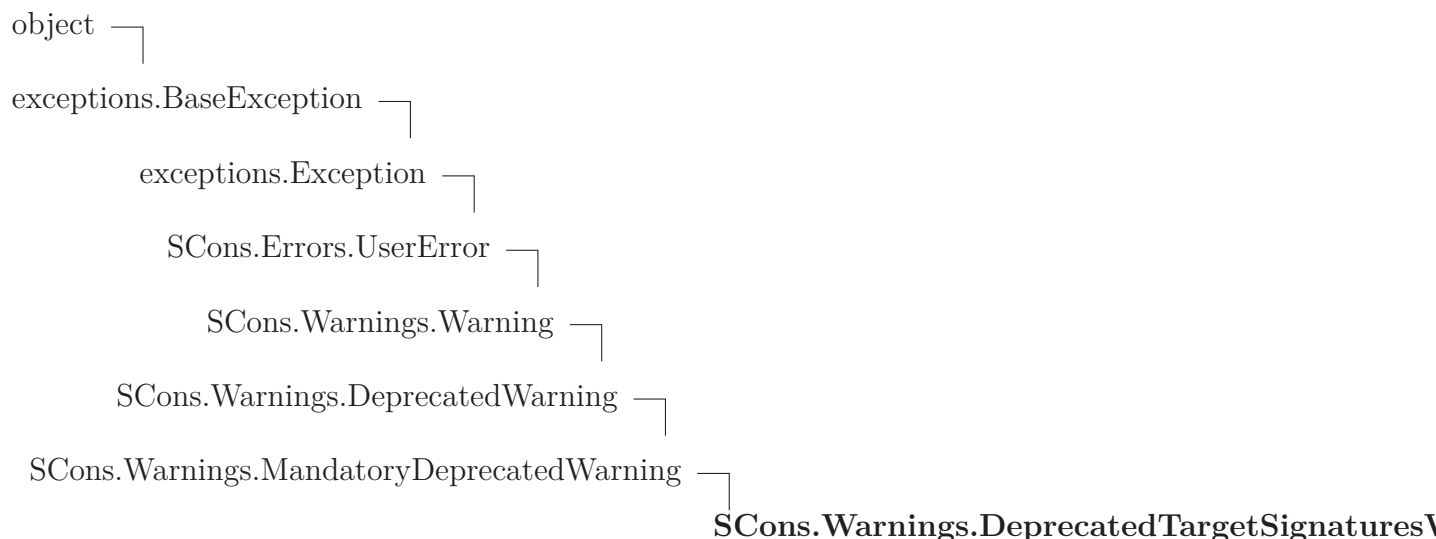
`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

43.33.2 Properties

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

43.34 Class `DeprecatedSourceSignaturesWarning`**43.34.1 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__()`**43.34.2 Properties**

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

43.35 Class `DeprecatedTargetSignaturesWarning`**43.35.1 Methods*****Inherited from `exceptions.Exception`***

`__init__()`, `__new__()`

Inherited from `exceptions.BaseException`

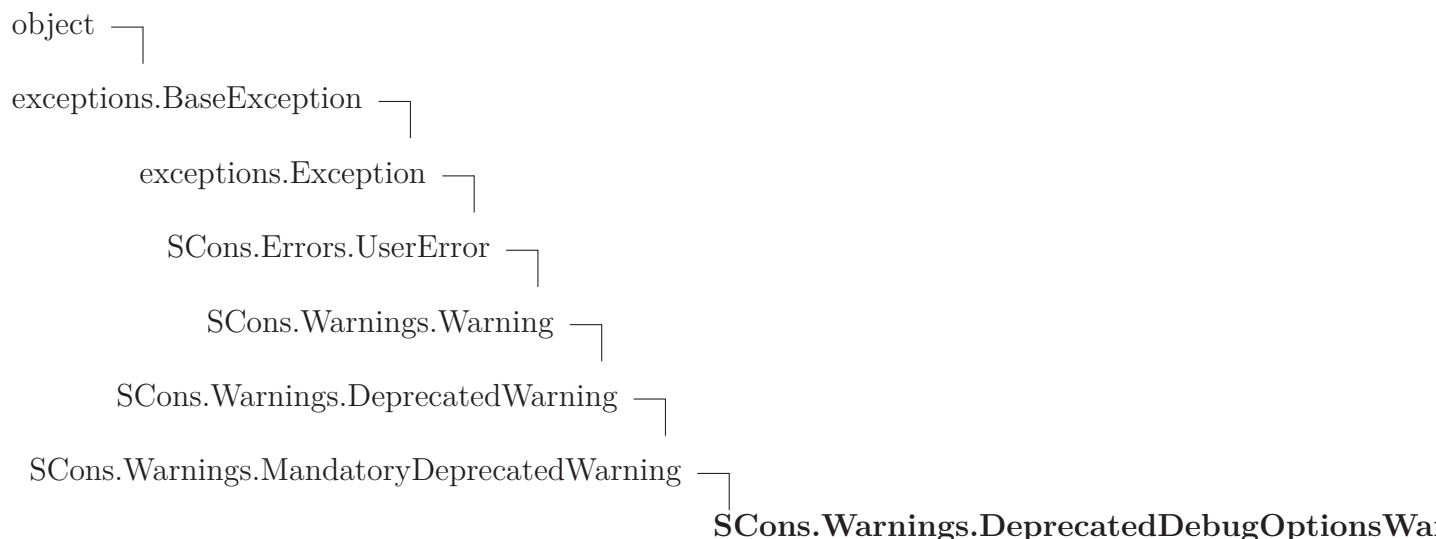
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

Inherited from `object`

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

43.35.2 Properties

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

43.36 Class `DeprecatedDebugOptionsWarning`**43.36.1 Methods*****Inherited from `exceptions.Exception`***

`__init__()`, `__new__()`

Inherited from `exceptions.BaseException`

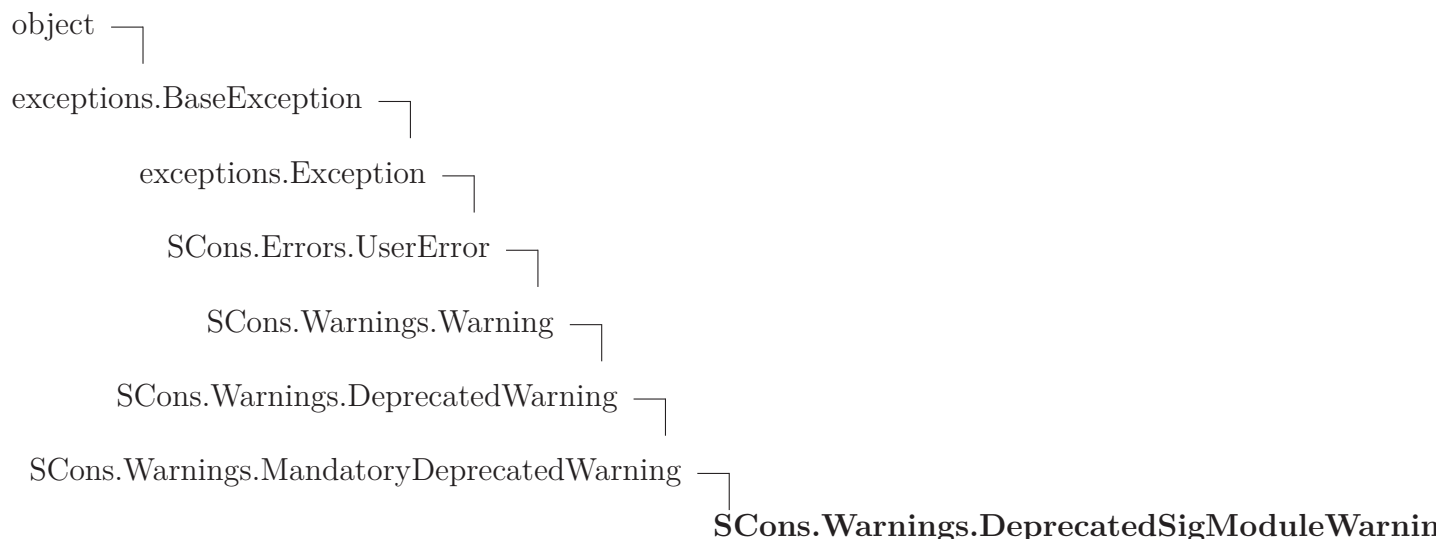
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

Inherited from `object`

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

43.36.2 Properties

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

43.37 Class `DeprecatedSigModuleWarning`**43.37.1 Methods*****Inherited from `exceptions.Exception`***

`__init__()`, `__new__()`

Inherited from `exceptions.BaseException`

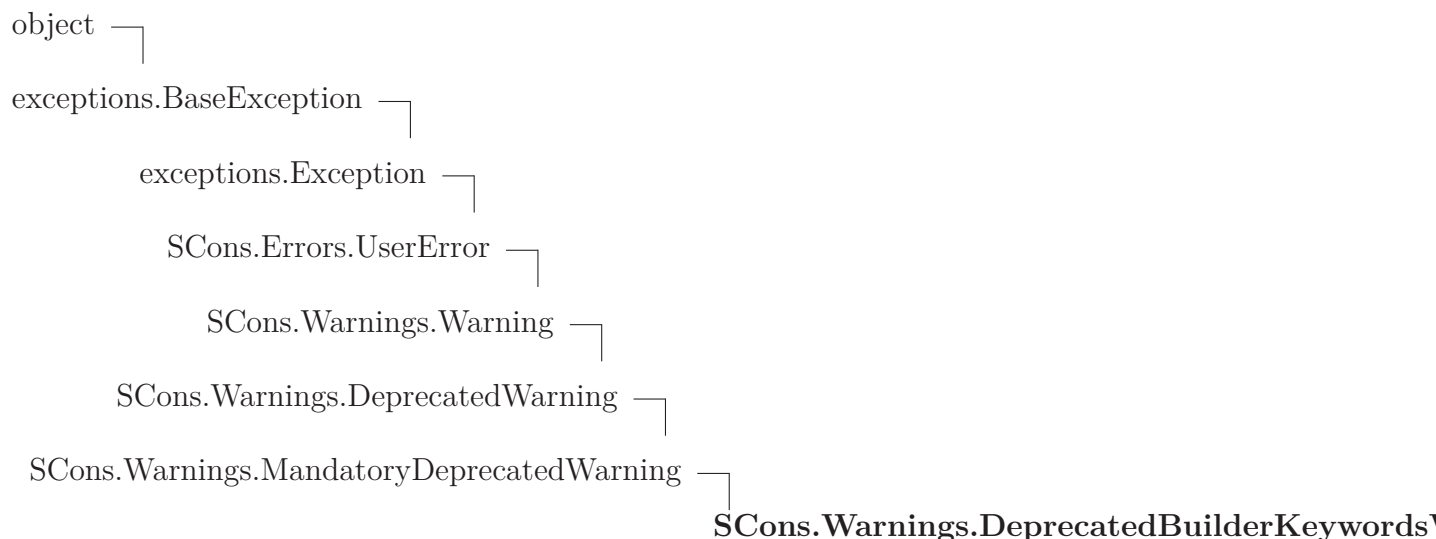
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

Inherited from `object`

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

43.37.2 Properties

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

43.38 Class `DeprecatedBuilderKeywordsWarning`**43.38.1 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__()`**43.38.2 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>

44 Module SCons.cpp

SCons C Pre-Processor module

44.1 Functions

CPP_to_Python_Ops_Sub(<i>m</i>)

CPP_to_Python(<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>	Value: ...
<code>cpp_lines_dict</code>	Value: {'define': '\\s+([_A-Za-z][_A-Za-z0-9]*)\\(([^)]*\\))'?...
Table	Value: {'define': re.compile(r'\\s+([_A-Za-z][_A-Za-z0-9]*)\\(([^)...
<code>e</code>	Value: '^\\s*#\\s*(elif undef include_next endif else include if.
CPP_Expression	Value: re.compile(r'(?m)^\\s*#\\s*(elif undef include_next endif e.
CPP_to_Python_Ops_Dict	Value: {'\r': ' ', '!': ' not ', '!=': ' != ', '&&': ' and ', ':'...
CPP_to_Python_Ops_Expression	Value: re.compile(r'\\ && != ! \\r :\\?')
CPP_to_Python_Eval_List	Value: [[re.compile(r'defined\\s+(\\w+)'), '"\\1" in __dict__'], [...
<code>line_continuations</code>	Value: re.compile(r'\\\\r?\\n')
<code>function_name</code>	Value: re.compile(r'\\S+\\(([^\\)]*\\)')
<code>function_arg_separator</code>	Value: re.compile(r',\\s*')
<code>__package__</code>	Value: 'SCons'
<code>x</code>	Value: 'if'

44.3 Class FunctionEvaluator

object —
SCons.cpp.FunctionEvaluator

Handles delayed evaluation of a #define function call.

44.3.1 Methods

__init__ (<i>self</i> , <i>name</i> , <i>args</i> , <i>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__

__call__ (<i>self</i> , * <i>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__ (<i>self</i> , *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>	Value: 0
<code>ignore_corrupt_dbfiles</code>	Value: 0
<code>dblite_suffix</code>	Value: <code>' .dblite'</code>
<code>tmp_suffix</code>	Value: <code>' .tmp'</code>
<code>__package__</code>	Value: <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

register(*func*, **targs*, ***kargs*)

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__	Value: 'src/engine/SCons/exitfuncs.py rel_2.3.5:3347:d31d5a4e74b...
__package__	Value: '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*), 384–391
 - SCons.dblite (*module*), 392–393
 - SCons.dblite.corruption_warning (*function*), 392
 - SCons.dblite.dblite (*class*), 392–393
 - SCons.dblite.is_string (*function*), 392
 - SCons.dblite.open (*function*), 392
 - SCons.dblite.unicode (*function*), 392
 - 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*), 82–92
 - SCons.Executor.AddBatchExecutor (*function*), 82
 - SCons.Executor.Batch (*class*), 82–83
 - SCons.Executor.Executor (*class*), 86–90
 - SCons.Executor.get_NullEnvironment (*function*), 82
 - SCons.Executor.GetBatchExecutor (*function*), 82
 - SCons.Executor.Null (*class*), 91–92
 - 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*), 394
 - SCons.exitfuncs.register (*function*), 394
- SCons.Job (*module*), 93–100
 - SCons.Job.InterruptState (*class*), 93–94
 - SCons.Job.Jobs (*class*), 94–95
 - SCons.Job.Parallel (*class*), 99–100
 - SCons.Job.Serial (*class*), 95–96
 - SCons.Job.ThreadPool (*class*), 98–99
 - SCons.Job.Worker (*class*), 96–98
- SCons.Memoize (*module*), 101–108
 - SCons.Memoize.CountDict (*class*), 105–106
 - SCons.Memoize.Counter (*class*), 103–104
 - SCons.Memoize.CountValue (*class*), 104–105
 - SCons.Memoize.Dump (*function*), 103
 - SCons.Memoize.EnableMemoization (*function*), 103
 - SCons.Memoize.Memoized_Metaclass (*class*), 107–108
 - SCons.Memoize.Memoizer (*class*), 106–107
- SCons.Node (*package*), 109–128
 - SCons.Node.Alias (*module*), 129–135
 - SCons.Node.FS (*module*), 136–181
 - SCons.Node.Python (*module*), 182–187
- SCons.PathList (*module*), 188
 - SCons.PathList.node_conv (*function*), 188
 - SCons.PathList.PathList (*function*), 188
- SCons.Scanner (*module*)
 - SCons.Scanner.Base (*class*), 213–216
 - SCons.Scanner.Classic (*class*), 223–226
 - SCons.Scanner.ClassicCPP (*class*), 226–227
 - SCons.Scanner.Current (*class*), 220–223
 - SCons.Scanner.FindPathDirs (*class*), 213
 - SCons.Scanner.Scanner (*function*), 212
 - SCons.Scanner.Selector (*class*), 216–220
- SCons.Scanner (*package*), 212–227
 - SCons.Scanner.C (*module*), 228–230
 - SCons.Scanner.D (*module*), 231–234
 - SCons.Scanner.Dir (*module*), 235–236
 - SCons.Scanner.Fortran (*module*), 237–241
 - SCons.Scanner.IDL (*module*), 242
 - SCons.Scanner.LaTeX (*module*), 243–249
 - SCons.Scanner.Prog (*module*), 250
 - SCons.Scanner.RC (*module*), 251
- SCons.SConf (*module*), 189–204
 - SCons.SConf.CheckCC (*function*), 190
 - SCons.SConf.CheckCHheader (*function*), 190
 - SCons.SConf.CheckContext (*class*), 202–204
 - SCons.SConf.CheckCXX (*function*), 190
 - SCons.SConf.CheckCXXHeader (*function*), 190
 - SCons.SConf.CheckDeclaration (*function*), 190
 - SCons.SConf.CheckFunc (*function*), 189
 - SCons.SConf.CheckHeader (*function*), 190
 - SCons.SConf.CheckLib (*function*), 190
 - SCons.SConf.CheckLibWithHeader (*function*), 190
 - SCons.SConf.CheckSHCC (*function*), 190
 - SCons.SConf.CheckSHCXX (*function*), 190
 - SCons.SConf.CheckType (*function*), 189
 - SCons.SConf.CheckTypeSize (*function*),

- 189
- SCons.SConf.ConfigureCacheError (*class*), 194–195
- SCons.SConf.ConfigureDryRunError (*class*), 193–194
- SCons.SConf.CreateConfigHBuilder (*function*), 189
- SCons.SConf.createIncludesFromHeaders (*function*), 190
- SCons.SConf.NeedConfigHBuilder (*function*), 189
- SCons.SConf.SConf (*function*), 189
- SCons.SConf.SConfBase (*class*), 199–202
- SCons.SConf.SConfBuildInfo (*class*), 195–196
- SCons.SConf.SConfBuildTask (*class*), 197–199
- SCons.SConf.SConfError (*class*), 192–193
- SCons.SConf.SConfWarning (*class*), 191–192
- SCons.SConf.SetBuildType (*function*), 189
- SCons.SConf.SetCacheMode (*function*), 189
- SCons.SConf.SetProgressDisplay (*function*), 189
- SCons.SConf.Streamer (*class*), 196–197
- SCons.SConsign (*module*), 205–211
 - SCons.SConsign.Base (*class*), 206–208
 - SCons.SConsign.corrupt_dblite_warning (*function*), 205
 - SCons.SConsign.DB (*class*), 208, 210–211
 - SCons.SConsign.Dir (*class*), 208–209
 - SCons.SConsign.DirFile (*class*), 209–210
 - SCons.SConsign.File (*function*), 205
 - SCons.SConsign.Get_DataBase (*function*), 205
 - SCons.SConsign.Reset (*function*), 205
 - SCons.SConsign.SConsignEntry (*class*), 205–206
 - SCons.SConsign.write (*function*), 205
- SCons.Script (*module*)
 - SCons.Script.HelpFunction (*function*), 252
 - SCons.Script.Options (*function*), 252
 - SCons.Script.TargetList (*class*), 258–260
 - SCons.Script.Variables (*function*), 252
- SCons.Script (*package*), 252–260
 - SCons.Script.Interactive (*module*), 261–263
 - SCons.Script.Main (*module*), 264–277
 - SCons.Script.SConscript' (*module*), 278–283
- SCons.Sig (*module*), 284–286
 - SCons.Sig.MD5Null (*class*), 284–285
 - SCons.Sig.TimeStampNull (*class*), 285–286
- SCons.Subst (*module*), 287–297
- SCons.Taskmaster (*module*), 298–308
 - SCons.Taskmaster.AlwaysTask (*class*), 304–305
 - SCons.Taskmaster.dump_stats (*function*), 298
 - SCons.Taskmaster.find_cycle (*function*), 298
 - SCons.Taskmaster.OutOfDateTask (*class*), 305–306
 - SCons.Taskmaster.Stats (*class*), 299
 - SCons.Taskmaster.Task (*class*), 299–304
 - SCons.Taskmaster.Taskmaster (*class*), 306–308
- SCons.Util (*module*), 309–335
- SCons.Variables (*package*), 336–339
 - SCons.Variables.BoolVariable' (*module*), 340
 - SCons.Variables.EnumVariable' (*module*), 341–342
 - SCons.Variables.ListVariable' (*module*), 343
 - SCons.Variables.PackageVariable' (*module*), 344
 - SCons.Variables.PathVariable' (*module*), 345–346
 - SCons.Variables.Variables (*class*), 336–339
- SCons.Warnings (*module*), 347–383
- 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*), 13
- 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*), 14
- 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*), 386
- 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.CheckSHCC (*function*), 33
- SCons.Conftest.CheckSHCXX (*function*), 34
- SCons.Conftest.CheckType (*function*), 34
- SCons.Conftest.CheckTypeSize (*function*), 35
- SCons.cpp.CPP_to_Python (*function*), 384
- SCons.cpp.CPP_to_Python_Ops_Sub (*function*), 384
- SCons.cpp.DumbPreProcessor (*class*), 390–391
- SCons.cpp.FunctionEvaluator (*class*), 384–385
 - SCons.cpp.FunctionEvaluator.__call__ (*method*), 385
- SCons.cpp.PreProcessor (*class*), 385–390
 - SCons.cpp.PreProcessor.__call__ (*method*), 386
 - SCons.cpp.PreProcessor.all_include (*method*), 386
 - SCons.cpp.PreProcessor.do_define (*method*), 386
 - SCons.cpp.PreProcessor.do_elif (*method*), 386
 - SCons.cpp.PreProcessor.do_else (*method*), 386
 - SCons.cpp.PreProcessor.do_endif (*method*), 386
 - SCons.cpp.PreProcessor.do_if (*method*), 386
 - SCons.cpp.PreProcessor.do_ifdef (*method*), 387
 - SCons.cpp.PreProcessor.do_ifndef (*method*), 387
 - SCons.cpp.PreProcessor.do_import (*method*), 387
 - SCons.cpp.PreProcessor.do_include (*method*), 387
 - SCons.cpp.PreProcessor.do_nothing (*method*), 387
 - SCons.cpp.PreProcessor.do_undef (*method*), 387
 - SCons.cpp.PreProcessor.eval_expression (*method*), 387
 - SCons.cpp.PreProcessor.finalize_result (*method*), 388
 - SCons.cpp.PreProcessor.find_include_file (*method*), 388
 - SCons.cpp.PreProcessor.initialize_result (*method*), 388
 - SCons.cpp.PreProcessor.process_contents (*method*), 388
 - SCons.cpp.PreProcessor.read_file (*method*), 388
 - SCons.cpp.PreProcessor.resolve_include (*method*), 388
 - SCons.cpp.PreProcessor.restore (*method*), 388
 - SCons.cpp.PreProcessor.save (*method*), 388
 - SCons.cpp.PreProcessor.scons_current_file (*method*), 389
 - SCons.cpp.PreProcessor.start_handling_includes (*method*), 389
 - SCons.cpp.PreProcessor.stop_handling_includes (*method*), 389
 - SCons.cpp.PreProcessor.tupleize (*method*), 389
- SCons.Defaults.chmod_func (*function*), 40
- SCons.Defaults.chmod_strfunc (*function*), 40
- SCons.Defaults.copy_func (*function*), 40
- SCons.Defaults.DefaultEnvironment (*function*), 40

- 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.MSVCErrors (*class*), 79–80
- SCons.Errors.StopError (*class*), 78–79
- SCons.Errors.UserError (*class*), 77–78
- SCons.Node.Annotate (*function*), 109
- SCons.Node.BuildInfoBase (*class*), 111–112
 - SCons.Node.BuildInfoBase.merge (*method*), 112
- SCons.Node.classname (*function*), 109
- SCons.Node.do_nothing (*function*), 109
- SCons.Node.get_children (*function*), 109
- SCons.Node.ignore_cycle (*function*), 109
- SCons.Node.Node (*class*), 112–126
 - SCons.Node.Node.add_dependency (*method*), 113
 - SCons.Node.Node.add_ignore (*method*), 113
 - SCons.Node.Node.add_prerequisite (*method*), 113
 - SCons.Node.Node.add_source (*method*), 113
 - SCons.Node.Node.add_to_implicit (*method*), 113
 - SCons.Node.Node.add_to_waiting_parents (*method*), 113
 - SCons.Node.Node.add_to_waiting_s_e (*method*), 113
 - SCons.Node.Node.add_wkid (*method*), 113
 - SCons.Node.Node.all_children (*method*), 113
 - SCons.Node.Node.alter_targets (*method*), 114
 - SCons.Node.Node.build (*method*), 114
 - SCons.Node.Node.builder_set (*method*), 114
 - SCons.Node.Node.built (*method*), 114
 - SCons.Node.Node.changed (*method*), 114
 - SCons.Node.Node.changed_since_last_build (*method*), 115
 - SCons.Node.Node.children (*method*), 115
 - SCons.Node.Node.children_are_up_to_date (*method*), 115
 - SCons.Node.Node.clear (*method*), 116
 - SCons.Node.Node.clear_memoized_values (*method*), 116
 - SCons.Node.Node.Decider (*method*), 112
 - SCons.Node.Node.del_binfo (*method*), 116
 - SCons.Node.Node.disambiguate (*method*), 116
 - SCons.Node.Node.do_not_store_info (*method*), 116
 - SCons.Node.Node.env_set (*method*), 116
 - SCons.Node.Node.executor_cleanup (*method*), 116
 - SCons.Node.Node.exists (*method*), 116
 - SCons.Node.Node.explain (*method*), 116
 - SCons.Node.Node.for_signature (*method*), 116
 - SCons.Node.Node.get_abspath (*method*), 117
 - SCons.Node.Node.get_binfo (*method*), 117

- SCons.Node.Node.get_build_env (*method*), 117
 SCons.Node.Node.get_build_scanner_path (*method*), 117
 SCons.Node.Node.get_builder (*method*), 117
 SCons.Node.Node.get_cachedir_csig (*method*), 118
 SCons.Node.Node.get_csig (*method*), 118
 SCons.Node.Node.get_env (*method*), 118
 SCons.Node.Node.get_env_scanner (*method*), 118
 SCons.Node.Node.get_executor (*method*), 118
 SCons.Node.Node.get_found_includes (*method*), 118
 SCons.Node.Node.get_implicit_deps (*method*), 118
 SCons.Node.Node.get_ninfo (*method*), 118
 SCons.Node.Node.get_source_scanner (*method*), 118
 SCons.Node.Node.get_state (*method*), 119
 SCons.Node.Node.get_stored_implicit (*method*), 119
 SCons.Node.Node.get_stored_info (*method*), 119
 SCons.Node.Node.get_string (*method*), 119
 SCons.Node.Node.get_subst_proxy (*method*), 119
 SCons.Node.Node.get_suffix (*method*), 120
 SCons.Node.Node.get_target_scanner (*method*), 120
 SCons.Node.Node.has_builder (*method*), 120, 121
 SCons.Node.Node.has_explicit_builder (*method*), 120
 SCons.Node.Node.is_derived (*method*), 120
 SCons.Node.Node.is_literal (*method*), 121
 SCons.Node.Node.is_up_to_date (*method*), 121
 SCons.Node.Node.make_ready (*method*), 121
 SCons.Node.Node.missing (*method*), 121
 SCons.Node.Node.new_binfo (*method*), 122
 SCons.Node.Node.new_ninfo (*method*), 122
 SCons.Node.Node.postprocess (*method*), 122
 SCons.Node.Node.prepare (*method*), 122
 SCons.Node.Node.push_to_cache (*method*), 122
 SCons.Node.Node.release_target_info (*method*), 123
 SCons.Node.Node.remove (*method*), 123
 SCons.Node.Node.render_include_tree (*method*), 123
 SCons.Node.Node.reset_executor (*method*), 123
 SCons.Node.Node.retrieve_from_cache (*method*), 123
 SCons.Node.Node.rexists (*method*), 124
 SCons.Node.Node.scan (*method*), 124
 SCons.Node.Node.scanner_key (*method*), 124
 SCons.Node.Node.select_scanner (*method*), 124
 SCons.Node.Node.set_always_build (*method*), 124
 SCons.Node.Node.set_executor (*method*), 124
 SCons.Node.Node.set_explicit (*method*), 124
 SCons.Node.Node.set_nocache (*method*), 124
 SCons.Node.Node.set_noclean (*method*), 125
 SCons.Node.Node.set_precious (*method*), 125
 SCons.Node.Node.set_pseudo (*method*), 125
 SCons.Node.Node.set_specific_source (*method*), 125
 SCons.Node.Node.set_state (*method*), 125
 SCons.Node.Node.state_has_changed (*method*), 125
 SCons.Node.Node.store_info (*method*), 125
 SCons.Node.Node.visited (*method*), 125
 SCons.Node.NodeInfoBase (*class*), 110–111
 SCons.Node.NodeInfoBase.convert (*method*), 110
 SCons.Node.NodeInfoBase.format (*method*),

- 111
 SCons.Node.NodeInfoBase.merge (*method*),
 111
 SCons.Node.NodeInfoBase.update (*method*),
 111
 SCons.Node.NodeList (*class*), 126–127
 SCons.Node.Walker (*class*), 127–128
 SCons.Node.Walker.get_next (*method*), 128
 SCons.Node.Walker.is_done (*method*), 128
 SCons.Scanner.Dir.DirEntryScanner (*function*),
 235
 SCons.Scanner.Dir.DirScanner (*function*), 235
 SCons.Scanner.Dir.do_not_scan (*function*), 235
 SCons.Scanner.Dir.only_dirs (*function*), 235
 SCons.Scanner.Dir.scan_in_memory (*function*),
 235
 SCons.Scanner.Dir.scan_on_disk (*function*),
 235
 SCons.Script.Interactive.interact (*function*), 261
 SCons.Script.Interactive.SConsInteractiveCmd
 (*class*), 261–263
 SCons.Script.Interactive.SConsInteractiveCmd.do
 (*method*), 262
 SCons.Script.Interactive.SConsInteractiveCmd.do
 (*method*), 262
 SCons.Script.Interactive.SConsInteractiveCmd.do
 (*method*), 262
 SCons.Script.Interactive.SConsInteractiveCmd.do
 (*method*), 263
 SCons.Script.Interactive.SConsInteractiveCmd.do
 (*method*), 263
 SCons.Subst.CmdStringHolder (*class*), 291–
 292
 SCons.Subst.CmdStringHolder.escape (*method*),
 291
 SCons.Subst.CmdStringHolder.is_literal (*method*),
 291
 SCons.Subst.escape_list (*function*), 287
 SCons.Subst.Literal (*class*), 289–290
 SCons.Subst.Literal.__eq__ (*method*), 289
 SCons.Subst.Literal.__neq__ (*method*),
 289
 SCons.Subst.Literal.escape (*method*), 289
 SCons.Subst.Literal.for_signature (*method*),
 289
 SCons.Subst.Literal.is_literal (*method*), 289
 SCons.Subst.NLWrapper (*class*), 292–293
 SCons.Subst.NullNodeList (*class*), 296–297
 SCons.Subst.quote_spaces (*function*), 287
 SCons.Subst.raise_exception (*function*), 287
 SCons.Subst.scons_subst (*function*), 287
 SCons.Subst.scons_subst_list (*function*), 288
 SCons.Subst.scons_subst_once (*function*), 288
 SCons.Subst.SetAllowableExceptions (*function*),
 287
 SCons.Subst.SpecialAttrWrapper (*class*), 290–
 291
 SCons.Subst.SpecialAttrWrapper.escape (*method*),
 290
 SCons.Subst.SpecialAttrWrapper.for_signature
 (*method*), 290
 SCons.Subst.SpecialAttrWrapper.is_literal
 (*method*), 290
 SCons.Subst.subst_dict (*function*), 287
 SCons.Subst.Target_or_Source (*class*), 295–
 296
 SCons.Subst.Target_or_Source.__getattr__
 (*method*), 296
 SCons.Subst.Targets_or_Sources (*class*), 293–
 295
 SCons.Subst.Targets_or_Sources.__getattr__
 (*method*), 294
 SCons.Util.NoError (*class*), 321–322
 SCons.Util.AddMethod (*function*), 314
 SCons.Util.adjustixes (*function*), 314
 SCons.Util.AppendPath (*function*), 313
 SCons.Util.case_sensitive_suffices (*function*),
 314
 SCons.Util.CLVar (*class*), 323–325
 SCons.Util.CLVar.__coerce__ (*method*),
 324
 SCons.Util.containsAll (*function*), 309
 SCons.Util.containsAny (*function*), 309
 SCons.Util.containsOnly (*function*), 309
 SCons.Util.Delegate (*class*), 321
 SCons.Util.Delegate.__get__ (*method*),

- 321
- SCons.Util.dictify (*function*), 309
- SCons.Util.DisplayEngine (*class*), 318–319
 - SCons.Util.DisplayEngine.__call__ (*method*), 319
 - SCons.Util.DisplayEngine.set_mode (*method*), 319
- SCons.Util.do_flatten (*function*), 311
- SCons.Util.flatten (*function*), 311
- SCons.Util.flatten_sequence (*function*), 311
- SCons.Util.get_environment_var (*function*), 309
- SCons.Util.get_native_path (*function*), 313
- SCons.Util.IDX (*function*), 310
- SCons.Util.is_Dict (*function*), 310
- SCons.Util.is_List (*function*), 310
- SCons.Util.is_Scalar (*function*), 311
- SCons.Util.is_Sequence (*function*), 310
- SCons.Util.is_String (*function*), 311
- SCons.Util.is_Tuple (*function*), 311
- SCons.Util.LogicalLines (*class*), 327–328
 - SCons.Util.LogicalLines.readline (*method*), 328
 - SCons.Util.LogicalLines.readlines (*method*), 328
- SCons.Util.make_path_relative (*function*), 314
- SCons.Util.MD5collect (*function*), 315
- SCons.Util.MD5filesignature (*function*), 315
- SCons.Util.MD5signature (*function*), 315
- SCons.Util.NodeList (*class*), 316–318
 - SCons.Util.NodeList.__call__ (*method*), 317
 - SCons.Util.NodeList.__getattr__ (*method*), 317
 - SCons.Util.NodeList.__nonzero__ (*method*), 317
- SCons.Util.Null (*class*), 333–334
 - SCons.Util.Null.__call__ (*method*), 334
 - SCons.Util.Null.__getattr__ (*method*), 334
 - SCons.Util.Null.__nonzero__ (*method*), 334
- SCons.Util.NullSeq (*class*), 334–335
 - SCons.Util.NullSeq.__delitem__ (*method*), 335
 - SCons.Util.NullSeq.__getitem__ (*method*), 335
 - SCons.Util.NullSeq.__iter__ (*method*), 335
 - SCons.Util.NullSeq.__len__ (*method*), 335
 - SCons.Util.NullSeq.__setitem__ (*method*), 335
- SCons.Util.OrderedDict (*class*), 325–327
- SCons.Util.PrependPath (*function*), 312
- SCons.Util.print_tree (*function*), 310
- SCons.Util.Proxy (*class*), 319–321
 - SCons.Util.Proxy.__cmp__ (*method*), 320
 - SCons.Util.Proxy.__getattr__ (*method*), 320
 - SCons.Util.Proxy.get (*method*), 320
- SCons.Util.RegGetValue (*function*), 312
- SCons.Util.RegOpenKeyEx (*function*), 312
- SCons.Util.RenameFunction (*function*), 315
- SCons.Util.render_tree (*function*), 310
- SCons.Util.rightmost_separator (*function*), 309
- SCons.Util.Selector (*class*), 327
 - SCons.Util.Selector.__call__ (*method*), 327
- SCons.Util.semi_deepcopy (*function*), 312
- SCons.Util.semi_deepcopy_dict (*function*), 312
- SCons.Util.silent_intern (*function*), 315
- SCons.Util.Split (*function*), 313
- SCons.Util.splittext (*function*), 309
- SCons.Util.to_String (*function*), 312
- SCons.Util.to_String_for_signature (*function*), 312
- SCons.Util.to_String_for_subst (*function*), 312
- SCons.Util.Unbuffered (*class*), 332–333
 - SCons.Util.Unbuffered.__getattr__ (*method*), 333
 - SCons.Util.Unbuffered.write (*method*), 333
- SCons.Util.unique (*function*), 314
- SCons.Util.UniqueList (*class*), 328–332
- SCons.Util.uniquer (*function*), 314
- SCons.Util.uniquer_hashables (*function*), 314
- SCons.Util.updrive (*function*), 309
- SCons.Util.WhereIs (*function*), 312
- SCons.Util.WindowsError (*class*), 322–323
- SCons.Warnings.CacheWriteErrorWarning (*class*),

- 351–352
- SCons.Warnings.CorruptSConsignWarning (*class*), 352–353
- SCons.Warnings.DependencyWarning (*class*), 353–354
- SCons.Warnings.DeprecatedBuildDirWarning (*class*), 374–375
- SCons.Warnings.DeprecatedBuilderKeywordsWarning (*class*), 382–383
- SCons.Warnings.DeprecatedCopyWarning (*class*), 376–377
- SCons.Warnings.DeprecatedDebugOptionsWarning (*class*), 380–381
- SCons.Warnings.DeprecatedOptionsWarning (*class*), 377–378
- SCons.Warnings.DeprecatedSigModuleWarning (*class*), 381–382
- SCons.Warnings.DeprecatedSourceCodeWarning (*class*), 373–374
- SCons.Warnings.DeprecatedSourceSignaturesWarning (*class*), 378–379
- SCons.Warnings.DeprecatedTargetSignaturesWarning (*class*), 379–380
- SCons.Warnings.DeprecatedWarning (*class*), 370–371
- SCons.Warnings.DevelopmentVersionWarning (*class*), 354–355
- SCons.Warnings.DuplicateEnvironmentWarning (*class*), 355–356
- SCons.Warnings.enableWarningClass (*function*), 347
- SCons.Warnings.FortranCxxMixWarning (*class*), 368–369
- SCons.Warnings.FutureDeprecatedWarning (*class*), 369–370
- SCons.Warnings.FutureReservedVariableWarning (*class*), 356
- SCons.Warnings.LinkWarning (*class*), 356–357
- SCons.Warnings.MandatoryDeprecatedWarning (*class*), 371–372
- SCons.Warnings.MisleadingKeywordsWarning (*class*), 357–358
- SCons.Warnings.MissingSConscriptWarning (*class*), 358–359
- SCons.Warnings.NoMD5ModuleWarning (*class*), 359–360
- SCons.Warnings.NoMetaclassSupportWarning (*class*), 360–361
- SCons.Warnings.NoObjectCountWarning (*class*), 361–362
- SCons.Warnings.NoParallelSupportWarning (*class*), 362–363
- SCons.Warnings.process_warn_strings (*function*), 347
- SCons.Warnings.PythonVersionWarning (*class*), 372–373
- SCons.Warnings.ReservedVariableWarning (*class*), 363–364
- SCons.Warnings.StackSizeWarning (*class*), 364–365
- SCons.Warnings.suppressWarningClass (*function*), 347
- SCons.Warnings.TargetNotBuiltWarning (*class*), 350–351
- SCons.Warnings.TaskmasterNeedsExecuteWarning (*class*), 375–376
- SCons.Warnings.VisualCMissingWarning (*class*), 365–366
- SCons.Warnings.VisualStudioMissingWarning (*class*), 367–368
- SCons.Warnings.VisualStudioVersionMismatch (*class*), 366–367
- SCons.Warnings.warn (*function*), 347
- SCons.Warnings.Warning (*class*), 348–349
- SCons.Warnings.warningAsException (*function*), 347
- SCons.Warnings.WarningOnByDefault (*class*), 349–350