Change SCons code to be runnable on py2.7 and py3.0

The current source code of SCons is based on Python 2.4 as floor version.
Recent requests on the mailing lists showed an increasing interest of users to
run it under Python 3.x. That's why the project has decided to raise
the floor to 2.7, in order to finally allow an easy switch to the more
recent Python versions. The current release of SCons deprecates python versions older than 2.7, so we are on the road.

After getting familiar with SCons as a program, and gaining an overview
of its basic folder structure, the work on this task should probably
start by running the 2to3.py script over the source tree.
An analysis of the arising problems would follow.
The gap towards Python 3.x should then be tried to narrow, by writing
little fixer scripts that can be applied on top of the current main
repository and would be run before 2to3.py.

While collecting and applying these helper scripts, the student should
try to ensure that SCons' basic functionality doesn't break. An extensive
testing framework is already setup and used for CI (buildbot.scons.org).
It could be further wrapped and used during development, in order to
protect against accidental mistakes.
This is also important for the automatically created example output texts,
that get included to the documentation (UserGuide).

Over all, the changes to the test setups and in the example outputs should be kept at an absolute minimum.

Since SCons gets output from subprocesses, stores and processes it, and outputs it to users, some Unicode conversion will probably be necessary since Python 3 strings are always Unicode.

The final goal, which doesn't have to be reached of course, would be:

- A set of fixer scripts exist, that can be applied on top of the current source code for SCons. They rewrite the code, such that the whole source tree can be run through 2to3.py afterwards, without any errors. The SCons version derived by this, is able to run successfully under a Python 3.x version. (Testing this, could involve a simple project where the filenames require Unicode support.)

- A simple wrapper around SCons' testing framework was programmed,

which can be used to demonstrate that no core functionality gets
broken by the various rewrites. All tests of the core features
and the supporting Tools pass without any errors.

- A list of problems that can't be solved by the approach described
above, or that couldn't be finished in the given time.
In the first case, giving a short description about the nature of
each problem would be considered a big plus.

Keywords: Python, programming, code parsing, code rewrite, testing framework,
continuous integration, CI

Get SCons to work better/at all on cygwin

Currently SCons doesn't work that well in a cygwin environment. This mainly has to do with the use of posix paths inside the cygwin environment and with it's tools, but not with tools which are normal win32 applications. So this would require the student to handle translating paths and recognizing which tools require which paths.

Keywords: Python, cygwin, win32 environment knowledge.

Change code to use more modern constructs (slots, generators/iterators)

SCons recently moved to a floor of Python 2.7; many modern python idioms are now available to us to increase performance, readability and maintainability as well as reducing memory usage.

Keywords: Python, software engineering skills

Improve packaging (get it into pypy and make it work with python setup.py and also in virtualenvs)

Work on improving the way that SCons is packaged to enable easier distribution, and development.

Keywords: Python, setuptools, virtualenv, win32 installers.

Migrate some functionality from Parts project (http://parts.tigris.org/)

The Parts project currently sits on top of SCons and provides some macro functionality as well as some patches to internals which improve upon SCons's baseline.
The Parts project creator has expressed interest for some time to migrate some (or all) of this logic into the core of SCons.

Possible items that can be migrated:
1) move over parts.api.output code to allow an API independent of the python print logic to deal with messages of different types (such as messages, warnings, verbose, tracing). so the text can be routed to log files, and colorized on the screen
2) move the code to allow python to correctly open files on windows, so that symlinks and hardlinks can be correctly supported. With this move over the Part CCopy and CCopyAs builder and --ccopy logic
3) GIven 2) or as a stretch goal, add the Part SymbolicFileNode object and api to allow SCons to deal with Symlinks as first class objects
4) move over the Version and VersionRange object.
5) move over the Namespace and bindable logic and objects
6) Move over the SystemPlatform logic and api. This support the idea of stuff like win32-x86, or android-arm so the system can better support the idea of host system and target system for cross build, etc..

Keyword: Python

Refactor the Node object

The Node is the central core object in SCons. It represents a node in the dependency graph; a file, directory, alias, or abstract dependency or target node. Over the years it has grown large and unwieldy and needs to be refactored. This is a significant project, but not out of the realm of a summer student.

This project has several goals:

- Reduce memory usage
- Improve performance
- Improve code readability and maintainability
- Lay the groundwork for future work

The work itself takes several phases:

- Understand all the members and uses of the current Node object
- Design an improved refactoring
- Review with the dev team
- Implement the refactoring in stages
- Update tests

Migrate bug list from Tigris to better bug tracker