go
Source code build wrapper.

'go' is a generic script to build a trivial test package which is unusual in that it can be built using either the autotools or cmake. Use this to check the autotools and cmake (both must already exist in $PATH).

'go.gcc' is commented example of how to compile a generic gcc. It contains a fudge function to download gcc source. Ordinarily you have the correct package to hand already.

'go.db' is an example of using cmake.

Installation:
Anywhere you like but consider making it globally accessible.

The author has it at ~/usr/src/git/go/ and symlinks to the files residing at /usr/local/sd/syschk/lib/. Pick somewhere and stick to it across machines.

Once the decision is made, edit the..
GOSUB="/usr/local/sd/syschk/lib"
..line at the top of each "go" file to point to where it got installed. You might..
$ sudo mkdir /usr/local/go
$ sudo chown `id -un`:`id -gn` /usr/local/go/
$ cd /usr/local/
$ git clone https://github.com/swampdawg/go
..then set..
GOSUB="/usr/local/go"
..in each script, for instance.

Usage:
Typically each script is simply called 'go' and resides in a folder associated with the package. eg:
$ find ~/usr/src/ -maxdepth 2 -type f -name 'go'
/home/foo/usr/src/virtmanager/go
/home/foo/usr/src/e2fsprogs/go
[snip]
/home/foo/usr/src/junk/go
/home/foo/usr/src/icu/go
A simple script can then iterate all the folders invoking "./go all" within each to get all the sources built and installed. It's for this reason one is best off ensuring that nothing needs to be downloaded. In the 'go.gcc' example the expected state of affairs is the user will have already performed the job of "fcp_init".

~~~
2do: document shell commands used (eg: 'bc').
2do: write a non-proprietory sdtimetool (only eye candy here though).
2do: auto patching example.
2do: f_tmp example.
~~~