C Programming Tools

Utilities

This section introduces the following utilities, listed in alphabetical order:

- ar
- gcc
- gdb
- gprof
- make
- strip
- touch

We have discussed some of these tools already:

Utility: gcc -cv [ -o fileName ] [ -pg ] { fileName }*

The gcc utility compiles C program code in one or more files and produces object modules or an executable file. Files specified should have a “.c” extension. Use the -c option to produce object modules suitable for linking later. Use the -o option to specify a filename other than the default “a.out” for the executable. Use the -pg option to produce profiling data for the GNU profiler gprof. Use the -v option to produce verbose commentary during the compilation and/or linking process.

Figure 11–1 Description of the gcc utility.
Utility: **ar** *key* *archiveName* { *fileName* }*

**ar** allows you to create and manipulate archives. *archiveName* is the name of the archive file that you wish to access, and it should end with a “.a” suffix. *key* may be one of the following:

- **d** - deletes a file from an archive
- **q** - appends a file onto the end of an archive, even if it’s already present
- **r** - adds a file to an archive if it isn’t already there, or replaces the current version if it is
- **s** - builds an index (table of contents) of the library for faster access
- **t** - displays an archive’s table of contents to standard output
- **x** - copies a list of files from an archive into the current directory
- **v** - generates verbose output

**Figure 11-2** Description of the **ar** command.

Utility: **touch** -c { *fileName* }+

**touch** updates the last modification and access times of the named files to the current time. By default, if a specified file doesn’t exist, it is created with zero size. To prevent this, use the **-c** option.

**Figure 11-13** Description of the **touch** command.
Utility: `gprof` -b [ `executableFile` [ `profileFile` ] ]

`gprof` is the GNU profiler. It generates a table of time and repetitions of each function in the executable `executableFile` based on the performance trace stored in the file `profileFile`. If `profileFile` is omitted, “gmon.out” is assumed. If `executableFile` is omitted, “a.out” is assumed. The executable file must have been compiled using the `-pg` option of `gcc`, which instructs the compiler to generate special code that writes a “gmon.out” file when the program runs. The `gprof` utility then looks at this output file after the program has terminated and displays the information contained therein. The output of `gprof` is verbose (but helpful); to instruct `gprof` to be brief, use the `-b` option.

Figure 11–15  Description of the `gprof` command.

Utility: `gdb` `executableFilename`

`gdb` is a standard GNU/Linux debugger. The named executable file is loaded into the debugger and a user prompt is displayed. To obtain information on the various `gdb` commands, enter `help` at the prompt.

Figure 11–16  Description of the `gdb` command.