.

Cook
A File Construction Tool

Reference Manual

Peter Miller
pmiller@opensource.org.au

.

This document describes Cook version 2.34
and was prepared 27 July 2017.

 cook − a file construction tool The *cook* program is a tool for constructing files, and maintaining referential integrity between files. It is given a set of files to create, and recipes of how to create and maintain them. In any non-trivial program there will be prerequisites to performing the actions necessary to creating any file, such as include files. The *cook* program provides a mechanism to define these. When a program is being developed or maintained, the programmer will typically change one file of several which comprise the program. The *cook* program examines the last-modified times of the files to see when the prerequisites of a file have changed, implying that the file needs to be recreated as it is logically out of date. The *cook* program also provides a facility for implicit recipes, allowing users to specify how to form a file with a given suffix from a file with a different suffix. For example, to create *filename*.o from *filename*.c

- Cook is a replacement for the traditional *make*(1) tool.
- Cook is more powerful than the traditional *make* tool.
- There is a *make2cook* utility included in the distribution to help convert makefiles into cookbooks.
- Cook has true variables, not simple macros.
- Cook has user defined functions.
- Cook has a simple but powerful string-based description language with many built-in functions. This allows sophisticated filename specification and manipulation without loss of readability or performance.
- Cook can build in parallel.
- Cook can distribute builds across your LAN.
- Cook is able to build your project with multiple parallel threads, with support for rules which must be single threaded. It is possible to distribute parallel builds over your LAN, allowing you to turn your network into a virtual parallel build engine.
- Cook is able to use fingerprints to supplement file modification times. This allows build optimization without contorted rules.
- In addition to walking the dependency graph, Cook can turn the input rules into a shell script, or a web page.
- Cook can be configured with an explicit list of primary source files. This allow the dependency graph to be constructed faster by not going down dead ends, and also allows better error messages when the graph can't be constructed. This requires an accurate source file manifest.
- Cook runs on almost any flavor of UNIX. The source distribution is self configuring using a GNU Autoconf generated configure script.
- Cook has special *cascade* dependencies, allowing powerful include dependency specification, amongst other things.

If you are putting together a source-code distribution and planning to write a makefile, consider writing a cookbook instead. Although Cook takes a day or two to learn, it is much more powerful and a bit more intuitive than the traditional *make*(1) tool. And Cook doesn't interpret tab differently to 8 space characters!

The latest version of *cook* is available on the Web from:

REFERENCES