Program development

October 13, 2016

Make

- My description of the make program is that it
  - takes a set of rules describing dependencies that
  - describes creation of new files in order to satisfy the requirements for the “creation” of some target.

Make

- Another description from Chapter 1 of the Gnu Make manual:

```
The make utility automatically determines
which pieces of a large program need to be
recompiled, and issues commands to recompile
them.
```

Invoking make

- There are several options that are generally useful with make:
  - -f MAKEFILE # specify an alternative makefile (defaults are Makefile, makefile, and GNUmakefile)
  - -k # continue for other targets even after an errror
  - -i # completely ignore errors
  - -d # print debugging information
  - -j [N] # create children to handle tasks. N specifies the maximum number of children to create
  - -C dir # change directory to DIR before starting the make process
  - -s # silent mode, don't echo commands

Makefiles

- Makefiles use rules to determine their actions. The rules look like:

```
target: [prerequisites]
 - TAB - action
 - TAB - action
 - TAB - ...
```

Targets

- Targets usually either specify a file that is to be made via this rule or just identify the rules for execution (often called a "phony" target.)
- Targets may also be implicit.

Prerequisites

- These generally define the files that the target depends on, and the general idea is that if any of those have a modified (or creation) time later than the target, then actions for the rule will be executed to create a new version of the target (which you should try to make sure has a new modified or created time.)

Actions

- These generally define the actions that are needed to create the target from the prerequisites. These actions are largely executions of discrete programs such as gcc, make (yes, recursion is quite common), ld, bison, flex, and so on.

Actions

Rules must consist of consecutive lines that start with a TAB character.

Actions

Since these are usually interpreted as shell commands, you can do things such as multi-lines (but use the backslash to make sure that the "single-linedness" of your construction is clear):

```
for name in dir1 dir2 dir3 \
do ; \
  ${MAKE} $name ; \
done
```

Actions

- There are also actual make conditionals which are interpreted by make and not by the shell; these look like

```
ifeq (ARG1,ARG2)
...
endif
ifdef (ARG1)
...
endif
```