

Program Maintenance

SYNOPSIS

This chapter will cover the use of the "*make*" facility as it is implemented on many of the UNIX systems at Cal Poly. It is assumed that the reader has a working knowledge of the task to be performed and a UNIX text editor such as vi.

INTRODUCTION

The "*make*" command allows you to recompile portions of a program without having to recompile the full package. The make facility will check to see which sources have been modified and recompile them only as long as an object file "*filename.o*" exists for the other sources.

COMMON USES OF A MAKEFILE

Makefiles often are used with distributed software to provide an easy method for the receiving user to build or "*make*" the package. Make uses targets with the format of the command usually being

```
% make -f make_command_file target<CR>
```

where "*make_command_file*" is the name of the file containing the directives for *make* (*Makefile* and *makefile* are, in order, the defaults looked for by *make*), and *target* is the name of the target (the author of the makefile must specify a default target which is usually "*all*"). Some common targets are

<i>all</i>	Usually causes all components to be compiled and linked.
<i>install</i>	Usually causes all components to be moved to a predetermined location on the system. If the components are not compiled, it should also trigger the compiles as well.
<i>clean</i>	Usually deletes any files with an extension of ".o" as well as the filename "core".
<i>dist</i>	Usually deletes the same files as " <i>clean</i> " would have, and the actual executable versions as well.

For additional information on parameters for the "*make*" command, please refer to the system man pages.

ELEMENTS OF A MAKEFILE

The following is a sample "*Makefile*" that contains many of the targets listed above. To the left of the "*Makefile*" lines are note references to the explanation for each element. (The *Makefile* text is in the mono spaced font.)

```
1) # This is a comment line
   # Program: program name

2) OBJECTS1 = main.o sub1.o sub2.o sub3.o sub4.o sub5.o
   OBJECTS2 = sub6.o sub7.o sub8.o sub9.o sub10.o sub11.o
   OBJECTS3 = sub12.o sub13.o sub14.o sub15.o

   OBJECTS = $(OBJECTS1) $(OBJECTS2) $(OBJECTS3)
```

```

LOCALHEADERS2 = header1.h header2.h
LOCALHEADERS1 = header3.h header4.h header5.h
MAINHEAD = mheader1.h mheader2.h mheader3.h mheader4.h
ALLHEADERS = $(MAINHEAD) $(LOCALHEADERS1) $(LOCALHEADERS2)

3) EXENAME = progname

4) CC = cc
5) CFLAGS = -DCC=cc -O -DBSD_CURSES -D_BSD -Hgcc
6) .c.o:
    $(CC) $(CFLAGS) -c $*.c

7) all      :
8) default : $(OBJECTS)
    $(CC) -o$(EXENAME) $(OBJECTS) -lcurses -lterm lib -lbsd

9) main.o : main.c $(MAINHEAD) extraheader1.h
sub1.o : sub1.c $(MAINHEAD)
sub2.o : sub2.c $(MAINHEAD)
sub3.o : sub3.c $(MAINHEAD) extraheader1.h
sub4.o : sub4.c $(MAINHEAD) extraheader2.h
sub5.o : sub5.c $(MAINHEAD)
sub6.o : sub6.c $(MAINHEAD) extraheader1.h
sub7.o : sub7.c $(MAINHEAD)
sub8.o : sub8.c $(MAINHEAD)
sub9.o : sub9.c $(MAINHEAD)
sub10.o : sub10.c $(MAINHEAD)
sub11.o : sub11.c $(MAINHEAD)
sub12.o : sub12.c $(MAINHEAD) extraheader2.h
sub13.o : sub13.c $(ALLHEADERS)
sub14.o : sub14.c $(MAINHEAD) extraheader3.h
sub15.o : sub15.c $(ALLHEADERS)

10) clean :
    rm *.o core

11) dist :
    rm *.o core $(EXENAME)

12) install :
    mv $(EXENAME) $HOME/bin

```

The definitions of each of these sections are as follows:

- 1) An example of a commented section of lines that would be used to document the `Makefile`.
- 2) Define some of the variables that we will want to use throughout the code to save typing and make the code a little more readable.
- 3) A variable for the name of the executable is useful in allowing you to change the name in one spot in the code.
- 4) Specify the compiler in a variable for the same reason.
- 5) Specify all of the desired compiler flags for the same reason.

- 6) Specifies that for each file that does not have an *. o file, or the *. c file is newer than the *. o file, perform the next sequence of statements up to a blank line (in this case, compile any changed or yet uncompiled . c file when using inference rules (see note 10).
- 7) Specify that if the user uses a target of "all", use the target of "*default*".
- 8) Specify the default rule for use when no target is specified. In this case the C compile is actually being used to link the components together.
- 9) Specify the rules for the dependencies of each of the subprograms. When a routine needs to be recompiled, these rules are consulted to determine what part is affected by which other parts.
- 10) Specify that if the user uses a target of "clean", *make* will delete any file ending with ". o" and any file named "core".
- 11) Specify that if the user uses a target of "clean", *make* will delete any file ending with ". o" and any file named "core", just like "clean"; in addition, it will also delete the executable defined by the variable "EXENAME".
- 12) Specify that the executable defined by the variable "EXENAME" be moved to a directory which is located below the directory defined by the variable "HOME", which is usually set by the shell to the user's home directory.

DOCUMENT CODE: UNIX-60501B

DATE REVISED: September 7, 1995

NOTES