12

Answers to Even-numbered Exercises

2. What command could you use to compile **prog.c** and **func.c** into an executable named **cprog**?

```
$ cc -o cprog prog.c func.c
```

4. How are the names of system libraries abbreviated on the gcc command line? Where does gcc search for libraries named in this manner? Describe how to specify your own library on the gcc command line.

   Libraries have names such as **lib*xxx*.a** or **lib*xxx*.dylib**. When you link on the command line with a library, strip away the **lib** prefix and the **.a** or **.dylib** suffix; then prefix the remaining name with **–l**, as in **–l*xxx***.

   By default gcc searches the **/usr/lib** directory for libraries. When you specify **–L*pathname*** on the command line, gcc searches the ***pathname*** directory for each instance of **–L**. To specify a library in **/Users/mine** named **libyyy.a**, you could use either of the following command lines:

```
$ gcc -o myprog myfile.c -L/Users/mine -lyyy
$ gcc -o myprog myfile.c /Users/mine/libyyy.a
```

6. If you retrieve version 4.1 of the file **answer** for editing and then attempt to retrieve the same version again, what will CVS do? Why is CVS set up this way?

   When you retrieve version 4.1 of the file **answer** for editing and then attempt to retrieve the same version again, CVS does nothing. If you do not change the file between the two retrievals, the file is still checked out and ready for editing. If you do change the file after you check it out and later attempt to check it out again, CVS generates a message that starts with **M** (modified), which indicates that the file has been modified but not checked in; the file is still checked out with the changes you made.

If you want to start over again with the original version of the file, you can remove the file (**rm answer**) and then run **cvs update** to check out a new version.

8. For the makefile

```
$ cat Makefile
leads: menu.o users.o resellers.o prospects.o
            gcc -o leads menu.o users.o resellers.o prospects.o

menu.o: menu.h dialog.h inquiry.h

users.o: menu.h dialog.h

prospects.o: dialog.h
```

identify:

a. Targets.

The targets are **leads**, **menu.o**, **users.o**, and **prospects.o** (tokens to the left of **:**).

b. Construction commands.

The construction command is the **gcc –o** ... command.

c. Prerequisites.

The prerequisites of **leads** are **menu.o**, **users.o**, **resellers.o**, and **prospects.o**.

The prerequisites of **menu.o** are **menu.h**, **dialog.h**, and **inquiry.h**.

The prerequisites of **users.o** are **menu.h** and **dialog.h**.

The prerequisite of **prospects.o** is **dialog.h**.

10. Review the make info page to answer the following questions:

a. What does the **–t** option do?

The **–t** option touches files—instead of running the build commands, make updates access times on the targets to simulate a successful build.

b. If you have files named **makefile** and **Makefile** in the working directory, how can you instruct make to use **Makefile**?

```
$ make -f Makefile
```

c. Give two ways to define a variable so that you can use it inside a makefile.

You can export the variable in the shell that calls make (**export CC=gcc**), set the variable on the make command line (**make CC=gcc**), or edit **Makefile** and add the line **CC=gcc**.