9

Answers to Even-numbered Exercises

2. How can you display the aliases currently in effect? Write an alias named **homedots** that lists the names (only) of all invisible files in your home directory.

   Give the command **alias** to list aliases.

   ```
   $ alias homedots 'ls -d ~/.*'
   ```

4. What statement can you put in your **~/.tcshrc** file to prevent accidentally overwriting a file when you redirect output? How can you override this feature?

   Put the command **set noclobber** in your **~/.tshrc** file to keep from overwriting a file with redirected output. Follow the redirect output symbol with an exclamation point (**>!**) to override **noclobber**.

6. Write an alias named **backup** that takes a filename as an argument and creates a copy of that file with the same name and a filename extension of **.bak**.

   ```
   $ alias backup cp \!:1 \!:1.bak
   ```

8. How can you make tcsh always display the pathname of the working directory as part of its prompt?

   ```
   $ set prompt = '%/ ! '
   ```

10. Users often find rm (and even **rm –i**) too unforgiving because it removes files irrevocably. Create an alias named **delete** that moves files specified by its argument(s) into the **~/.trash** directory. Create a second alias named **undelete** that moves a file from the **~/.trash** directory into the working

directory. Put the following line in your **~/.logout** file to remove any files that you deleted during the login session:

```
/bin/rm -f $HOME/.trash/* >& /dev/null
```

These commands create the required aliases:

```
$ alias delete mv \!:\* ~/.trash
$ alias undelete mv ~/.trash/\!:1 .
```

Explain what could be different if the following line were put in your **~/.logout** file instead:

```
rm $HOME/.trash/*
```

There are several differences between this rm command and the one that starts with **/bin/rm**. The first command uses an absolute pathname, ensuring that the system rm command is used as opposed to a potentially bogus command that could just move your files to a directory for inspection by a malicious user. The **–f** option forces files to be removed even if you do not have write permission for the file. The first command gets rid of error messages, while the second one displays them on the screen as you log out.

12. Rewrite the program **while_1** (page 375) so that it runs faster. Use the time builtin to verify the improvement in execution time.

Speed things up by adding the **–f** option to the line that calls tcsh to avoid calling **~/.tcshrc**:

```
$ cat while_2
#!/bin/tcsh -f
...

$ time while_1 1000
The sum is 500500
0.105u 0.083s 0:00.19 94.7%     0+0k 0+0io 0pf+0w
$ time while_2 1000
The sum is 500500
0.092u 0.055s 0:00.14 100.0%    0+0k 0+0io 0pf+0w
```

You can avoid the **while** loop altogether by using an equation to speed up the calculation:

```
$ cat while_4
#!/bin/tcsh -f
@ sum = ($argv[1] * $argv[1] + $argv[1]) / 2
echo "The sum is $sum"
$ time while_4 1000
The sum is 500500
0.000u 0.002s 0:00.00 0.0%      0+0k 0+0io 0pf+0w
```

14. When the **foreach_1** script (page 374) is supplied with 20 or fewer arguments, why are the commands following **toomany:** not executed? (Why is there no exit command?)

    The exec builtin runs the *command* you specify by overwriting the current process with the process that *command* starts. Execution never reaches **toomany**:.