Answers to Even-Numbered Exercises 14

from page 731

1. Assume that you are working with the following history list:

```
37  pine alex
38  cd /home/jenny/correspondence/business/cheese_co
39  less letter.0321
40  vi letter.0321
41  cp letter.0321 letter.0325
42  grep hansen letter.0325
43  vi letter.0325
44  lp letter*
45  cd ../milk_co
46  pwd
47  vi wilson.0321 wilson.0329
```

   Using the history mechanism, give commands to

   a. Send mail to Alex.

   b. Use vi to edit a file named **wilson.0329**.

   c. Send **wilson.0329** to the printer.

   d. Send both **wilson.0321** and **wilson.0329** to the printer.

2. How can you identify all 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 ~/.*'
```

3. How can you prevent a command from sending output to the terminal when you start it in the background? What can you do if you start a command in the foreground and later decide that it should run in the background?

4. What statement can you put in your **.tcshrc** file to prevent yourself from 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.

5. Assume that the working directory contains the following files:

```
adams.ltr.03
adams.brief
adams.ltr.07
abelson.09
abelson.brief
anthony.073
anthony.brief
azevedo.99
```

   What happens if you press TAB after typing the following commands?

   a. **less adams.l**

   b. **cat a**

   c. **ls ant**

   d. **file az**

   What happens if you press CONTROL-D after typing these commands?

   e. **ls ab**

   f. **less a**

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
```

7. Write an alias named **qmake** (**q**uiet **make**) that runs make with both standard output and standard error redirected to the file named **make.log**.

The command **qmake** should accept the same options and arguments as make.

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

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

Advanced Exercises

9. What lines do you need to change in the Bourne Again Shell script **command_menu** (page 654) to make it a TC Shell script? Make the changes and verify that it works.

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. Finally, 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
```

The following 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 preceding 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 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. Finally, the first command gets rid of error messages while the second one displays them on your screen as you are logging out.

11. Modify the **foreach_1** program (page 721) so that it takes the command to exec as an argument.

12. Rewrite the program **while_1** (page 722) 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
0.296u 0.074s 0:00.38 94.7%     0+0k 0+0io 2277pf+0w
$ time while_2 1000
The sum is 500500
0.250u 0.046s 0:00.29 100.0%    0+0k 0+0io 222pf+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
0.003u 0.005s 0:00.00 0.0%      0+0k 0+0io 222pf+0w
```

13. Write your own version of find named **myfind** that writes output to the file **findout** but without the clutter of error messages, such as when you do not have permission to search a directory. The **myfind** command should accept the same options and arguments as find. Can you think of a situation in which **myfind** does not work as desired?

14. When the **foreach_1** script (page 721) 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:`.