8

Answers to Even-numbered Exercises

2. What are two ways you can execute a shell script when you do not have execute access permission for the file containing the script? Can you execute a shell script if you do not have read access permission for the file containing the script?

   You can give the name of the file containing the script as an argument to the shell (for example, **bash** *scriptfile* or **tcsh** *scriptfile*, where *scriptfile* is the name of the file containing the script).

   Under bash you can give the following command:

   ```
   $ . scriptfile
   ```

   Under both bash and tcsh you can use

   ```
   $ source scriptfile
   ```

   Because the shell must read the commands from the file containing a shell script before it can execute the commands, you must have read permission for the file to execute a shell script.

4. Assume that you have made the following assignment:

   ```
   $ person=jenny
   ```

   Give the output of each of the following commands:

   a. **echo $person**

   jenny

   b. **echo '$person'**

   $person

   c. **echo "$person"**

   jenny

6. Assume that the **/home/jenny/grants/biblios** and **/home/jenny/biblios** directories exist. Give Jenny's working directory after she executes each sequence of commands given. Explain what happens in each case.

   a.

```
$ pwd
/home/jenny/grants
$ CDPATH=$(pwd)
$ cd
$ cd biblios
```

   After executing the preceding commands, Jenny's working directory is **/home/jenny/grants/biblios**. When **CDPATH** is set and the working directory is not specified in **CDPATH**, cd searches the working directory only after it searches the directories specified by **CDPATH**.

   b.

```
$ pwd
/home/jenny/grants
$ CDPATH=$(pwd)
$ cd $HOME/biblios
```

   After executing the preceding commands, Jenny's working directory is **/home/jenny/biblios**. When you give cd an absolute pathname as an argument, cd does not use **CDPATH**.

8. Give the following command:

```
$ sleep 30 | cat /etc/inittab
```

   Is there any output from sleep? Where does cat get its input from? What has to happen before the shell displays another prompt?

   There is no output from sleep (try giving the command **sleep 30** by itself). The **/etc/inittab** file provides input for cat (when cat has an argument, it does not check standard input). The sleep command has to run to completion before the shell displays another prompt.

10. Write a shell script that outputs the name of the shell that is executing it.

    There are many ways to solve this problem. The following solutions are all basically the same. These scripts take advantage of the **PPID** shell variable, which holds the PID number of the shell that is the parent of the process using the variable. They also use the fact that echo changes multiple sequential SPACEs to a single SPACE. The cut utility interprets multiple sequential SPACEs as multiple delimiters so, without echo, the script does not work properly.

```
$ cat a
pid=$PPID
line=$(ps | grep $pid)
echo $line | cut --delimiter=" " --fields=4

$ cat a2
pid=$PPID
echo $(ps | grep $pid) | cut --delimiter=" " --fields=4

$ cat a3
echo $(ps | grep $PPID) | cut --delimiter=" " --fields=4
```

The easy solution is to give the following command:

```
$ echo $0
```

The **$0** is the first command line token, which is usually the name of the script or program that is running (page 481). In some cases, such as when you call the script with a relative or absolute pathname, this outcome may not be exactly what you want.

12. Add the exit status of the previous command to your prompt so that it behaves similarly to the following:

```
$ [0] ls xxx
ls: xxx: No such file or directory
$ [1]
```

The following command sets up the prompt described in the question:

```
PS1='$ [$?] '
```

14. Implement the basename utility, which writes the last component of its pathname argument to standard output, as a bash function. For example, given the pathname **a/b/c/d**, basename writes **d** to standard output:

```
$ basename a/b/c/d
d
```

The following function is named **bn** so that you know that you are not running the **/bin/basename** utility. It behaves the same way as basename.

```
$ function bn () {
> if [ $# = 0 ]; then
>         exit 1
>     elif [ "$1" = "/" ]
>         then
>         echo /
>     else
>         echo $1 | sed 's:.*/::'
> fi
> }
```