Chapter 13
Shell Scripts: Writing Applications

This chapter builds on the commands and concepts of the previous chapter

It discusses additional shell programming commands and techniques.

It presents a simple application program.

It shows the process of developing programs using the shell language.

WRITING APPLICATIONS

Shell built-in commands that are covered in this chapter and their availability

Table 13.1
The Shell Built-in Commands

Command	Bourne Shell		
	Bourne Shell	Korn Shell	Bourne Again Shell
trap	sh	ksh	bash
case	sh	ksh	bash

The lock1 Program

Scenario: you want to protect the access to your without logging off and then logging in again

Write a script file that, when invoked, shows a message on the screen and does not go away until you enter the correct password

Figure 13.1
The lock1 Script File

```
$ cat -n lock1
   1 #
   2 # lock1 (lock version 1)
   3 # This program locks the keyboard, and you must type the
   4 # specified password to unlock it.
   5 # logic:
   6 #    1- Ask the user to enter a password.
   7 #    2- Lock the keyboard until the correct password is entered.
   8 #
   9 # DO NOT RUN THIS PROGRAM IF YOU DO NOT KNOW WHAT YOU ARE DOING!!
  10 # Please read the first part of chapter 13.
  11 #
  12 echo "\032"                          # can be replaced by tput clear
  13 echo "\n\nENTER YOUR PASSWORD>"      # ask for password
  14 read pword_1                         # read password
```

The lock1 Program

Scenario: you want to protect the access to your without logging off and then logging in again

```
# Please read the first part of chapter 13.
#
echo "\032"                          # can be replaced by tput clear
echo "\n\nENTER YOUR PASSWORD>"      # ask for password
read pword_1                         # read password
echo "\032"                          # clear screen again
echo "\n\n THIS SYSTEM IS LOCKED...."
pword_2=                             # declare an empty variable
until [ "$pword_1" = "$pword_2" ]    # start of the loop
do
  read pword_2                       # body of the loop
done                                 # end of the until loop
exit 0                               # end of the program, exit
$_
```

The lock1 Program

- Problems with the lock1 Program – not foolproof!
  - While the keyboard is locked, if you press [Del] (the interrupt key), the lock1 script is terminated
  - The password is displayed.
  - The message is hard coded. It always shows.
- To solve these problems, a few more commands must be explored.

UNIX INTERNALS: THE SIGNALS

Several different events that make the kernel send a signal to your process

Table 13.2
Some of the Shell Signals

Signal Number	Name	Meaning
1	**hang up**	Terminal connection is lost.
2	**interrupt**	One of the **interrupt** keys has been pressed.
3	**quit**	One of the **quit** keys has been pressed.
9	**kill**	The **kill -9** command has been issued.
15	**terminator**	The **kill** command has been issued.

UNIX INTERNALS: THE SIGNALS

Trapping the Signals: The **trap** Command

- Commands that are specified to the trap command must be enclosed in single or double quotation marks
- You can specify more than one signal number to be trapped
- Signal numbers are the numbers associated with the signals that you want the trap command to catch

```
trap "echo I refuse to die!" 15
```

UNIX INTERNALS: THE SIGNALS

Resetting the Traps

- Issuing a **trap** command in your script changes the default actions of the signals received by your process
- Using the **trap** command, without the optional commands part, changes the specified signals to their default actions

Setting Terminal Options: The **stty** Command

- Use the command to set and display terminal characteristics

Figure 13.2

An Example of the Terminal Settings

```
$ stty
speed 9600 baud;-parity
erase = '^h' ;kill = '^u';
echo
$_
```

UNIX INTERNALS: THE SIGNALS

Setting Terminal Options

Table 13.3
A Short List of Terminal Options

Option	Operation
echo [-echo]	Echoes [does not echo] the typed characters; the default is **echo**.
raw [-raw]	Disables [enables] the special meaning of the metacharacters; the default is **-raw**.
intr	Generates an interrupt signal; usually the **[Del]** key is used.
erase	**[Backspace]**. Erases the preceding character; usually the # key is used.
kill	Deletes the entire line; usually @ or **[Ctrl-u]** is used.
eof	Generates the (end-of-file) signal from the terminal; usually **[Ctrl-d]** is used.
ek	Resets the **erase** and **kill** keys to # and @ respectively.
sane	Sets the terminal characteristics to sensible default values.

MORE ABOUT TERMINALS

- The Terminals Database: The **terminfo** File
  - A single text file that contains descriptions of many types of terminals
  - A list of capabilities associated with each terminal in the database.
- Setting Terminal Capabilities: The **tput** Command
  - Lets you print out the values of any single capability
  - Makes it possible to use the terminals' capabilities in shell programming

MORE ABOUT TERMINALS

Terminal Capabilities

Table 13.4
A Short List of the Terminal Capabilities

Option	Operation
bel	Echoes the terminal's bell character.
blink	Makes a blinking display.
bold	Makes a boldface display.
clear	Clears the screen.
cup *r c*	Moves cursor to row *r* and column *c*.
dim	Dims the display.
ed	Clears from the cursor position to the end of the screen.
el	Clears from the cursor position to the end of the line.

MORE ABOUT TERMINALS

Solving the **lock1** Program Problems

Figure 13.3
The lock2 Script File

```
$ cat -n lock2
 1 #
 2 # lock2 (lock version 2)
 3 # This program locks the keyboard, and you must type the
 4 # specified password to unlock it.
 5 # Logic:
 6 #  1- Ask the user to enter a password.
 7 #  2- Lock the keyboard until the correct password is entered.
 8 #
 9 # DO NOT RUN THIS PROGRAM IF YOU DO NOT KNOW WHAT YOU ARE DOING!!
10 # Please read the first part of chapter 13.
11 #
12 trap " " 2 3                      # ignore the listed signals
13 stty -echo                        # prohibit echoing the input
14 tput clear                        # clear the screen
15 tput cup 5 10; echo "ENTER YOUR PASSWORD> \c" # ask for password
16 read pword_1                      # read password
17 tput clear                        # clear screen again
18 tput cup 10 20 ; echo "THIS SYSTEM IS LOCKED ...
```

MORE ABOUT TERMINALS

Solving the **lock1** Program Problems

```
# Please read the first part of chapter 13.
#
trap " " 2 3                          # ignore the listed signals
stty -echo                            # prohibit echoing the input
tput clear                            # clear the screen
tput cup 5 10; echo "ENTER YOUR PASSWORD> \c" # ask for password
read pword_1                          # read password
tput clear                            # clear screen again
tput cup 10 20 ; echo "THIS SYSTEM IS LOCKED...."
pword_2=                              # declare an empty variable
until [ "$pword_1" = "$pword_2" ] # start of the loop
do
 read pword_2                         # body of the loop
done                                  # end of the until loop
stty echo                             # enable echo of input characters
tput clear                            # clear screen
exit 0                                # end of program, exit
$_
```

MORE ABOUT TERMINALS

Solving the **lock1** Program Problems

Figure 13.4
The Prompt from the **lock2** Program

```
ENTER YOUR PASSWORD>_
```

Figure 13.5
The Message from the **lock2** Program

```
THIS SYSTEM IS LOCKED....
```

MORE ABOUT TERMINALS

Specifying the Display Message

```
$ lock3 Coffee Break. Will be back in 5 minutes [Return]
```

Figure 13.6
The lock Program,Third Version

```
$ cat -n lock3
   1 #
   2 # lock3 (lock program version 3)
   3 # This program locks the keyboard, and you must type the
   4 # specified password to unlock it.
   5 # Logic:
   6 #   1- Ask the user to enter a password.
   7 #   2- Lock the keyboard until the correct password is entered.
   8 #
   9 # DO NOT RUN THIS PROGRAM IF YOU DO NOT KNOW WHAT YOU ARE DOING!!
  10 # Please read the first part of chapter 13.
  11 #
  12 trap " " 2 3 4                    # ignore the listed signals
  13 stty -echo                        # prohibit echoing the input
  14 if [ $# -gt 0 ]                   # online message is specified
  15 then
  16   MESG="$@"                       # store the specified message
  17 else
  18   MESG="THIS SYSTEM IS LOCKED"    # set to default message
  19 fi
  20 tput clear                        # clear the screen
```

MORE ABOUT TERMINALS

Specifying the Display Message

```
12 trap " " 2 3 4                    # ignore the listed signals
13 stty -echo                        # prohibit echoing the input
14 if [ $# -gt 0 ]                   # online message is specified
15 then
16   MESG="$@"                       # store the specified message
17 else
18   MESG="THIS SYSTEM IS LOCKED"    # set to default message
19 fi
20 tput clear                        # clear the screen
21 tput cup 5 10 ; echo "ENTER YOUR PASSWORD>\c"  # ask for password
22 read pword_1                      # read password
23 tput clear                        # clear screen again
24 tput cup 10 20 ; echo "$MESG"
25 pword_2=                          # declare an empty variable
26 until [ "$pword_1" = "$pword_2" ] # start of the loop
27 do
28   read pword_2                    # body of the loop
29 done                              # end of the until loop
30 stty echo                         # enable echo of input characters
31 tput clear                        # clear screen
32 exit 0                            # end of program, exit
$_
```

MORE COMMANDS

Multiway Branching: The **case** Construct

```
case variable in
  pattern_1)
    commands_1 ;;
  pattern_2)
    commands_2 ;;
  ...
  ...
    *)
  default_commands ;;
esac
```

MORE COMMANDS

Multiway Branching: The **case** Construct

Consider use of a menu program

- Display the possible options.
- Prompt the user to select a function.
- Read the user input.
- Call other programs according to the user input.
- Display an error message if the input is wrong.

MORE COMMANDS

Multiway Branching: The **case** Construct

Figure 13.7

Source Code for a Simple MENU Program

```
$ cat -n MENU
   1 #
   2 # MENU
   3 # A sample program to demonstrate the use of the case construct.
   4 #
   5 echo
   6 echo " 0: Exit"
   7 echo " 1: Show Date and Time"
   8 echo " 2: List my HOME directory"
   9 echo " 3: Display Calendar"
  10 echo "Enter your choice: \c"   # display the prompt
  11 read option                    # read user answer
  12 case $option in                # beginning of the case construct
  13    0) echo Good bye ;;         # display the message
  14    1) date ;;                  # show date and time
  15    2) ls $HOME ;;              # display the HOMEdirectory
  16    3) cal ;;                   # show the current month calendar
  17    *) echo "Invalid input. Good bye." ;; # show error message
  18 esac                           # end of the case construct
  19 echo
  20 exit 0                         # end of program, exit
$_
```

MORE COMMANDS

Revisiting the greetings Program

Figure 13.8
The New Version of the greetings Program

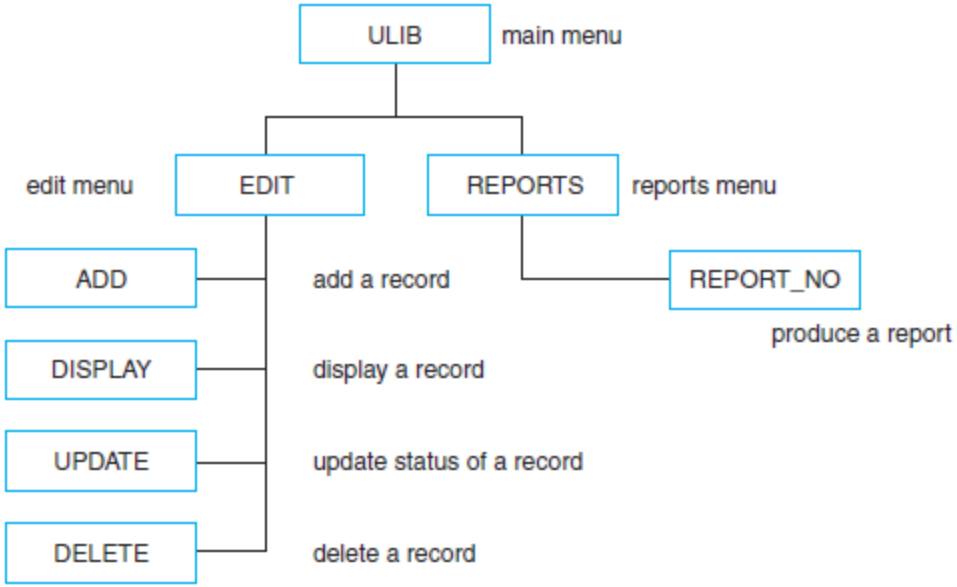
```
$ cat -n greetings3
   1 #
   2 # greetings3
   3 # greeting program version 3
   4 # This version is using the case construct to check the hour of
   5 # and the day and to display the appropriate greetings.
   6 #
   7 echo
   8 bell=`tput bel`                    # store the code for the bell sound
   9 echo $bell$bell                    # two beeps
  10 hour=`date +%H`                    # obtain hour of the day
  11 case $hour in
  12   0?|1[0-1] )    echo "Good Morning";;
  13      1[2-7] )    echo "Good Afternoon";;
  14           * )    echo "Good Evening";;
  15 esac                               # end of the case
  16 echo
  17 exit 0                             # end of the program, exit
$_
```

A MENU-DRIVEN APPLICATION

Scenario: write a menu-driven application that facilitates keeping track of your UNIX books

The Hierarchy Chart

Figure 13.9
The ULIB Program Hierarchy Chart



A MENU-DRIVEN APPLICATION

The ULIB Program

Figure 13.10

The ULIB Program Source Code

```
$ cat -n ULIB
     1 #
     2 # UNIX library
     3 # ULIB: This program is the main driver for the UNIX library application
     4 #       program. It shows a brief startup message and then displays the
               main menu.
     5 #       It invokes the appropriate program according to the user selection.
     6 #
     7 BOLD=`tput smso`        # store code for bold mode in BOLD
     8 NORMAL=`tput rmso`      # store code for end of the bold mode in NORMAL
     9 export BOLD NORMAL      # make them recognized by subshells
    10 #
    11 # show the title and a brief message before showing the main menu
    12 #
    13 tput clear              # clear screen
    14 tput cup 5 15           # place the cursor on line 5, column 15
    15 echo "${BOLD}Super Duper UNIX Library" # show the title in bold
    16 tput cup 12 10          # place the cursor on line 12, column 10
    17 echo "${NORMAL}This is the UNIX library application" # the rest of the title
    18 tput cup 14 10 ; echo "Please enter any key to continue..._\b\c"
    19 read answer             # read user input
```

A MENU-DRIVEN APPLICATION

The ULIB Program

```
#
tput clear                # clear screen
tput cup 5 15             # place the cursor on line 5, column 15
echo "${BOLD}Super Duper UNIX Library" # show the title in bold
tput cup 12 10            # place the cursor on line 12, column 10
echo "${NORMAL}This is the UNIX library application" # the rest of the title
tput cup 14 10 ; echo "Please enter any key to continue..._\b\c"
read answer               # read user input
error_flag=0              # initialize the error flag, indicating no error
while true                # loop forever
do
 if [ $error_flag -eq 0 ]     #check for the error
 then
   tput clear             # clear screen
   tput cup 5 10
   echo "UNIX Library - ${BOLD}MAIN MENU${NORMAL}"
   tput cup 7 20  ; echo "0: ${BOLD}EXIT${NORMAL} this program"
   tput cup 9 20  ; echo "1: ${BOLD}EDIT${NORMAL} Menu"
   tput cup 11 20 ; echo "2: ${BOLD}REPORTS${NORMAL} Menu"
   error_flag=0           # reset error flag
 fi
 tput cup 13 10 ; echo "Enter your choice> _\b\c"
 read choice              # read user choice
```

A MENU-DRIVEN APPLICATION

The ULIB Program

```
    tput cup 9 20 ; echo "1: ${BOLD}EDIT${NORMAL} Menu"
    tput cup 11 20 ; echo "2: ${BOLD}REPORTS${NORMAL} Menu"
    error_flag=0          # reset error flag
  fi
  tput cup 13 10 ; echo "Enter your choice> _\b\c"
  read choice             # read user choice
#
# case construct for checking the user selection
#
  case $choice in         #check user input
       0 ) tput clear ; exit 0 ;;
       1 ) EDIT ;;        # call EDIT program
       2 ) REPORTS ;;     # call REPORT program
       * ) ERROR 20 10    # call ERROR program
         tput cup 20 1 ; tput ed # clear the reset of the screen
         error_flag=1;; # set error flag to indicate error
  esac                    # end of the case construct
done                      # end of the while construct
exit 0                    # exit the program
$_
```

A MENU-DRIVEN APPLICATION

The ULIB Program

Figure 13.11
The ULIB Program Startup Screen

```
Super Duper UNIX Library

This is the UNIX library application
Please enter any key to continue...>_
```

Figure 13.12
The ULIB Program Main Menu Screen

```
UNIX Library - MAIN MENU

 0:EXIT this program
 1:EDIT menu
 2:REPORTS Menu

Enter your choice>_
```

A MENU-DRIVEN APPLICATION

The ULIB Program

Figure 13.13
The ULIB Program Error Message Screen

```
        UNIX Library - MAIN MENU

            0:EXIT this program
            1:EDIT menu
            2:REPORTS Menu

        Enter your choice>_6
    Wrong Input. Try again.
Press any key to continue...>_
```

A MENU-DRIVEN APPLICATION

The ERROR Program

Figure 13.14
The **ERROR** Program Source Code

```
$ cat -n ERROR
   1 #
   2 # ERROR: This program displays an error message and waits for user
   3 #        input to continue. It displays the message at the specified
   4 #        row and column.
   5 #
   6 tput cup $1 $2                    # place the cursor on the screen
   7 echo "Wrong Input. Try again."    # show the error message
   8 echo "Press any key to continue...> _\b\c"    # display the prompt
   9 read answer                       # read user input
  10 exit 0                            # indicate normal exit
$_
```

A MENU-DRIVEN APPLICATION

The EDIT Program

Figure 13.15

The EDIT Program Source Code

```
$ cat -n EDIT
 1 #
 2 # UNIX Library
 3 # EDIT:  This program is the main driver for the EDIT program.
 4 #        It shows the EDIT menu and invokes the appropriate program
 5 #        according to the user selection.
 6 #
 7 error_flag=0              # initialize the error flag, indicating no error
 8 while true                # loop forever
 9 do
10 if [ $error_flag -eq 0 ]  # check for the error
11 then
12   tput clear ; tpup cut 5 10  # clear screen and place the cursor
13   echo "UNIX Library - ${BOLD}EDITMENU${NORMAL}"
14   tput cup 7 20           # place the cursor
15   echo "0: ${BOLD}RETURN${NORMAL}To the Main Menu"
16   tput cup 9 20  ; echo "1:${BOLD}ADD${NORMAL}"
17   tput cup 11 20 ; echo "2:${BOLD}UPDATESTATUS${NORMAL}"
18   tput cup 13 20 ; echo "3:${BOLD}DISPLAY${NORMAL}"
19   tput cup 15 20 ; echo "4:${BOLD}DELETE${NORMAL}"
20 fi
21 error_flag=0              # reset error flag
22 tput cup 17 10 ; echo "Enter your choice>\b\c"
```

A MENU-DRIVEN APPLICATION

The EDIT Program

```
15   echo 0:${BOLD}RETURN${NORMAL}TO THE MAIN MENU
16   tput cup 9 20  ; echo "1:${BOLD}ADD${NORMAL}"
17   tput cup 11 20 ; echo "2:${BOLD}UPDATESTATUS${NORMAL}"
18   tput cup 13 20 ; echo "3:${BOLD}DISPLAY${NORMAL}"
19   tput cup 15 20 ; echo "4:${BOLD}DELETE${NORMAL}"
20 fi
21 error_flag=0              # reset error flag
22 tput cup 17 10 ; echo "Enter your choice> _\b\c"
23 read choice               # read user choice
24 #
25 # case construct for checking the user selection
26 #
27 case $choice in
28    0) exit 0 ;;           # check user input
29    1) ADD ;;              # return to the main menu
30    2) UPDATE;;            # call UPDATE program
31    3) DISPLAY ;;          # call DISPLAY program
32    4) DELETE ;;           # call DELETE program
33    *) ERROR 20 10         # call ERROR program
34       tput cup 20 1 ; tput ed       # clear the rest of the screen
35       error_flag=1 ;;     # set error flag to indicate
36 esac                      # end of the case construct
37 done                      # end of the while construct
38 exit 0                    # end of the program
$_
```

A MENU-DRIVEN APPLICATION

The EDIT Program

Figure 13.16
The EDIT Menu

```
UNIX Library - EDIT MENU

     0: RETURN To The Main Menu
     1: ADD
     2: UPDATE STATUS
     3: DISPLAY
     4: DELETE

Enter your choice>_
```

A MENU-DRIVEN APPLICATION

The ADD Program

Figure 13.17

The ADD Program Source Code

```
$ cat -n ADD
   1 #
   2 # UNIX   library
   3 # ADD: This program adds a record to the library file (U_LIB). It asks the
   4 #        title, author, and category of the book. After adding the
              information to
   5 #        the ULIB_FILE file, it prompts the user for the next record.
   6 #
   7 answer=y                      # initialize the answer to indicate yes
   8 while [ "$answer" = y ]       # as long as the answer is yes
   9 do
  10  tput clear
  11  tput cup 5 10 ; echo "UNIX Library - ${BOLD}ADDMODE"
  12  echo "${NORMAL}"
  13  tput cup 7 23  ; echo  "Title:"
  14  tput cup 9 22  ; echo  "Author:"
  15  tput cup 11 20 ; echo  "Category:"
  16  tput cup 12 20 ; echo  "sys; system, ref; reference, th; textbook",
```

A MENU-DRIVEN APPLICATION

The ADD Program

```
echo "${NORMAL}"
tput cup 7 23  ; echo  "Title:"
tput cup 9 22  ; echo  "Author:"
tput cup 11 20 ; echo  "Category:"
tput cup 12 20 ; echo  "sys; system, ref: reference, tb: textbook"
tput cup 7 30  ; read  title
tput cup 9 30  ; read  author
tput cup 11 30 ; read  category
status=in                    # set the status to indicate book is in
echo "$title:$author:$category:$status:$bname:$date" >> ULIB_FILE
tput cup 14 10 ; echo "Any more to add? (Y)es or (N)o> _\b\c"
read answer
case $answer in              # check user answer
   [Yy]* ) answer=y ;;       # any word starting with Y or y is yes
       * ) answer=n ;;       # any other word indicates no
   esac                      # end of the case construct
done                         # end of the while loop
exit 0                       # end of the program
$_
```

A MENU-DRIVEN APPLICATION

The ADD Program

Figure 13.18
The ADD Program Screen Format

```
UNIX Library - ADD MODE

        Title:
       Author:
     Category:
     sys: system, ref: reference,tb: textbook
```

Figure 13.19
The Example of the Filled in ADD Screen

```
UNIX Library - ADD MODE

        Title:  UNIX Unbounded
       Author:  Afzal Amir
     Category:  tb
     sys: system, ref: reference,tb: textbook

     Any more to add? (Y)es or (N)o>_
```

A MENU-DRIVEN APPLICATION

Delimiter Character

- Fields in each record are stored in sequential order
- Must designate a field delimiter
- [Tab], [Return], and [Spacebar] are *not* good choices
- Better options: **~**, **^**, or **:**

Figure 13.20
The Contents of the ULIB_FILE File

```
$ cat ULIB_FILE
   UNIX Unbounded:Afzal Amir:tb:in::
$_
```

A MENU-DRIVEN APPLICATION

Record Retrieval

- To display a record, you must specify what record you want displayed

The DISPLAY Program

- displays a specified record from the ULIB_FILE file on the screen

Figure 13.22
The Prompt Displayed by the **DISPLAY** Program

```
Enter the Author/Title>_
```

A MENU-DRIVEN APPLICATION

The DISPLAY Program

Figure 13.21
The DISPLAY Program Source Code

```
$ cat -n DISPLAY
   1 #
   2 # UNIX library
   3 # DISPLAY: This program displays a specified record from the ULIB_FILE.
   4 #          It asks the Author/Title of the book, and displays the specified
   5 #          book is not found in the file.
   6 #
   7 OLD_IFS="$IFS"                              # save the IFS settings
   8 answer=y                                    # initialize the answer to indicate yes
   9 while [ "$answer" = y ]                     # as long as the answer is yes
  10 do
  11  tput clear ; tput cup 3 5 ; echo "Enter the Author/Title> _\b\c"
  12  read response
  13  grep -i "$response" ULIB_FILE> TEMP        # find the specified book in the library
  14  if [ -s TEMP ]                             # if it is found
  15  then
  16    IFS=":"                                  # set the IFS to colon
  17    read title author category status bname date < TEMP
  18    tput cup 5 10
  19    echo "UNIX Library - ${BOLD}DISPLAYMODE${NORMAL}"
  20    tput cup 7 22 ; echo "Title: $title"
```

A MENU-DRIVEN APPLICATION

The DISPLAY Program

```
if [ -s TEMP ]                          # if it is found
then
  IFS=":"                               # set the IFS to colon
  read title author category status bname date < TEMP
  tput cup 5 10
  echo "UNIX Library - ${BOLD}DISPLAYMODE${NORMAL}"
  tput cup 7 23 ; echo "Title: $title"
  tput cup 8 22 ; echo "Author: $author"
  case $category in                     # check the category
                  [Tt][Bb]) word=textbook ;;
              [Ss][Yy][Ss]) word=system ;;
              [Rr][Ee][Ff]) word=reference ;;
                         *) word=undefined ;;
  esac
  tput cup 9 20 ; echo "Category: $word" # display the category
  tput cup 10 22 ; echo "Status:$status" # display the status
  if [ "$status" = "out" ]              # if it is checked out
  then                                  # then show the rest of the information
    tput cup 11 14 ; echo "Checked out by: $bname"
    tput cup 12 24 ; echo "Date: $date"
fi
else                                    # if book not found
```

A MENU-DRIVEN APPLICATION

The DISPLAY Program

```
26                                  word=underlined ;;
27    esac
28    tput cup 9 20 ; echo "Category: $word" # display the category
29    tput cup 10 22 ; echo "Status:$status" # display the status
30    if [ "$status" = "out" ]               # if it is checked out
31    then                                   # then show the rest of the information
32      tput cup 11 14 ; echo "Checked out by: $bname"
33      tput cup 12 24 ; echo "Date: $date"
34  fi
35  else                                     # if book not found
36    tput cup 7 10 ; echo "$response not found"
37  fi
38  tput cup 15 10 ; echo "Any more to look for? (Y)es or (N)o> _\b\c"
39  read answer                              # read user's answer
40  case $answer in                          # check user's answer
41     [Yy]* ) answer=y ;;                   # any word starting with Y or y is yes
42         * ) answer=n ;;                   # any other word indicates no
43 esac                                      # end of the case construct
44 done                                      # end of the while loop
45 IFS="$OLD_IFS"                            # restore the IFS to its original value
46 exit 0                                    # exit the program
$_
```

A MENU-DRIVEN APPLICATION

The DISPLAY Program

Figure 13.23
Sample of the **DISPLAY** Program Screen

```
UNIX Library - DISPLAY MODE

   Title : UNIX Unbounded
  Author : Afzal Amir
Category : texbook
  Status : in

Any more to look for?(Y)es or (N)o>_
```

Figure 13.24
Sample Screen Showing the Error Message

```
Enter the Author/Title>XYZ

        XYZ not found
        Any more to look for?(Y)es or (N)o>_
```

A MENU-DRIVEN APPLICATION

The UPDATE Program

Figure 13.25
Source Code for the **UPDATE** Program

```
$ cat -n UPDATE
   1 #
   2 # UNIX library
   3 # UPDATE:This program updates the status of a specified book. It asks the
   4 #        Author/Title of the book, and changes the status of the specified
   5 #        book from in (checked in) to out (checked out), or from out to in.
   6 #        If the book is not found in the file, an error message is displayed.
   7 #
   8 OLD_IFS="$IFS"                        # save the IFS settings
   9 answer=y                              # initialize the answer to indicate yes
```

A MENU-DRIVEN APPLICATION

The UPDATE Program

```
10 while [ "$answer" = y ]
11 do
12    new_status= ; new_bname= ; new_date=      # declare empty variables
13    tput clear                                # clear screen
14    tput clear ; tput cup 3 5 ; echo "Enter the Author/Title > _\b\c"
15    read response
16    grep -i "$response" ULIB_FILE > TEMP      # find the specified book
17    if [ -s TEMP ]                            # if it is found
18    then                                      # then
19       IFS=":"                                # set the IFS to colon
20       read title author category status bname date < TTEMP
21       tput cup 5 10
22       echo "UNIX Library - ${BOLD}UPDATESTATUSMODE${NORMAL}"
23       tput cup 7 23 ; echo "Title: $title"
24       tput cup 8 22 ; echo "Author: $author"
25       case $category in                      # check the category
26               [Tt][Bb] ) word=textbook ;;
27           [Ss][Yy][Ss] ) word=system ;;
28           [Rr][Ee][Ff] ) word=reference ;;
29                      * ) word=undefined ;;
30       esac
31       tput cup 9 20  ; echo "Category: $word" # display the category
32       tput cup 10 22 ; echo "Status: $status" # display the status
```

A MENU-DRIVEN APPLICATION

The UPDATE Program

```
                    *) word=undefined ;;
esac
tput cup 9 20  ; echo "Category: $word" # display the category
tput cup 10 22 ; echo "Status: $status" # display the status
if [ "$status" = "in" ]                 # if it is checked in
then                                    # then show the rest of the
                                          information
  new_status=out                        # indicate the new status
  tput cup 11 18 ; echo "New status: $new_status"
  tput cup 12 14 ; echo "Checked out by: _\b\c"
  read new_bname
  new_date=`date +%D`
  tput cup 13 24 ; echo "Date: $new_date"
else
  new_status=in
  tput cup 11 14 ; echo "Checked out by: $bname"
  tput cup 12 24 ; echo "Date:$date"
  tput cup 15 18 ; echo "New status:$new_status"
fi
grep -iv "$title:$author:$category:$status:$bname:$date" ULIB_FILE> TEMP
cp TEMP ULIB_FILE
echo "$title:$author:$category:$new_status:$new_bname:$new_date">> ULIB_FILE
```

A MENU-DRIVEN APPLICATION

The UPDATE Program

Figure 13.26
The UPDATE Program Display

```
UNIX Library - UPDATE STATUS MODE

          Title:  UNIX Unbounded
         Author:  Afzal Amir
       Category:  textbook
         Status:  in
     New status:  out
 Checked out by:  Steve Fraser
           Date:  12/12/05

Any more to update?(Y)es or (N)o>_
```

A MENU-DRIVEN APPLICATION

The UPDATE Program

Figure 13.27
The Contents of the ULIB_FILE File

```
$ cat ULIB_FILE file
   UNIX Unbounded:Afzal Amir:tb:out:Steve Fraser:12/12/05
$_
```

A MENU-DRIVEN APPLICATION

The UPDATE Program

Figure 13.28
Sample Record Display

```
UNIX Library - DISPLAY MODE

          Title: UNIX Unbounded
         Author: Afzal Amir
       Category: textbook
         Status: out
 Checked out by: Steve Fraser
           Date: 12/12/05
Any more to look for?(Y)es or (N)o>_
```

A MENU-DRIVEN APPLICATION

The DELETE Program

Figure 13.29
Source Code for the DELETE Program

```
$ cat -n DELETE
   1 #
   2 # UNIX library
   3 # Delete: This program deletes a specified record from the ULIB_FILE.
   4 #         It asks the Author/Title of the book, and displays the specified
   5 #         book, and deletes it after confirmation, or shows an error message
   6 #         if the book is not found in the file.
   7 #
   8 OLD_IFS="$IFS"                                # save the IFS setting
   9 answer=y                                      # initialize the answer to indicate yes
  10 while [ "$answer" = y ]                       # as long as the answer is yes
  11 do
  12    tput clear ; tput cup 3 5 ; echo "Enter the Author/Title> _\b\c"
  13    read response
  14    grep -i "$response" ULIB_FILE> TEMP        # find the specified book in the library
  15    if [ -s TEMP ]                             # if it is found
  16    then                                       # then
```

A MENU-DRIVEN APPLICATION

The DELETE Program

```
grep -i "$response" ULIB_FILE> TEMP        # find the specified book in the library
if [ -s TEMP ]                             # if it is found
then                                       # then
  IFS=":"                                  # set the IFS to colon
  read title author category status bname date < TEMP
  tput cup 5 10
  echo "UNIX Library - ${BOLD}DELETEMODE${NORMAL}"
  tput cup 7 23 ; echo "Title: $title"
  tput cup 8 22 ; echo "Author: $author"
  case $category in                        # check the category
           [Tt][Bb] ) word=textbook ;;
        [Ss][Yy][Ss] ) word=system ;;
        [Rr][Ee][Ff] ) word=reference ;;
                   * ) word=undefined ;;
  esac
  tput cup 9 20 ; echo "Category: $word"  # display the category
  tput cup 10 22 ; echo "Status: $status" # display the status
  if [ "$status" = "out" ]                 # if it is checked out
  then                                     # then show the rest of the information
  tput cup 11 14 ; echo "Checked out by: $bname"
  tput cup 12 24 ; echo "Date: $date"
  fi
  tput cup 13 20 ; echo "Delete this book?(Y)es or (N)o> _\b\c"
  read answer
```

A MENU-DRIVEN APPLICATION

The DELETE Program

```
    fi
    tput cup 13 20 ; echo "Delete this book?(Y)es or (N)o> _\b\c"
    read answer
    if [ $answer = y -o $answer = Y ]       # test for Y or y
    then
    grep -iv "$title:$author:$category:$status:$bname:$date" ULIB_FILE> TEMP
    mv TEMP ULIB_FILE
 fi
else                                        # if book not found
 tput cup 7 10 ; echo "$response not found"
fi
tput cup 15 10; echo "Any more to delete? (Y)es or (N)o> _\b\c"
read annswer                                # read user answer
case $answer in                             # check user answer
      [Yy]* ) answer=y ;;                   # any word starting with Y or y is yes
          * ) answer=n ;;                   # any other word indicates no
esac
done                                        # end of the while loop
IFS="$OLD_IFS"                              # restore the IFS to its original value
rm TEMP                                     # delete the TEMP file
exit 0                                      # exit the program
```

A MENU-DRIVEN APPLICATION

The DELETE Program

Figure 13.30
The DELETE Program Display

```
UNIX Library - DELETE MODE

             Title:  UNIX Unbounded
            Author:  Afzal Amir
          Category:  textbook
            Status:  out
    Checked out by:  Steve Fraser
              Date:  12/12/05

Delete this book? (Y)es or (N)o>Y

Any more to delete? (Y)es or (N)o>_
```

A MENU-DRIVEN APPLICATION

The REPORTS Program

Figure 13.31

Source Code for the REPORTS Program

```
$ cat -n REPORTS
 1 #
 2 # UNIX Library
 3 # Reports: This program is the main driver for the REPORTS menu.
 4 #          It shows the reports menu and invokes the appropriate
 5 #          program according to the user selection.
 6 #
 7 error_flag=0           # initialize the error flag, indicating no error
 8 while true                    # loop forever
 9  do
10   if [ $error_flag -eq 0 ]    # check for the error
11   then
12    tput clear ; tput cup 5 10 # clear screen and place the cursor
13    echo "UNIX Library - ${BOLD}REPORTSMENU${NORMAL}"
14    tput cup 7 20              # place the cursor
15    echo "0: ${BOLD}RETURN${NORMAL} To The Main Menu"
16    tput cup 9 20 ; echo "1: Sorted by ${BOLD}TITLES ${NORMAL}"
17    tput cup 11 20 ; echo "2: Sorted by ${BOLD}AUTHOR ${NORMAL}"
18    tput cup 13 20 ; echo "3: Sorted by ${BOLD}CATEGORY ${NORMAL}"
19    fi
20    error_flag=0               # reset error flag
21    tput cup 17 10 ; echo "Enter your choice> _\b\c"
```

A MENU-DRIVEN APPLICATION

The REPORTS Program

```
15     echo "0: ${BOLD}RETURN${NORMAL} to the Main Menu"
16     tput cup 9 20 ; echo "1: Sorted by ${BOLD}TITLES ${NORMAL}"
17     tput cup 11 20 ; echo "2: Sorted by ${BOLD}AUTHOR ${NORMAL}"
18     tput cup 13 20 ; echo "3: Sorted by ${BOLD}CATEGORY ${NORMAL}"
19     fi
20     error_flag=0                # reset error flag
21     tput cup 17 10 ; echo "Enter your choice> _\b\c"
22 read choice                     # read user choice
23 #
24 # case construct for checking the user selection
25 #
26  case $choice in                # check user input
27   0 )  exit 0 ;;                # return to the main menu
28   1 )  REPORT_NO 1 ;;           # call REPORT_NO program, passing 1
29   2 )  REPORT_NO 2 ;;           # call REPORT_NO program, passing 2
30   3 )  REPORT_NO 3 ;;           # call REPORT_NO program, passing 3
31   * )  ERROR 20 10              # call ERROR program
32      tput cup 20 1 ; tput ed    # clear the rest of the screen
33      error_flag=1 ;;            # set error flag to indicate error
34  esac                           # end of the case construct
35 done                            # end of the while construct
36 exit 0                          # end of the program
$_
```

A MENU-DRIVEN APPLICATION

The REPORTS Program

Figure 13.32
The REPORTS Menu

```
UNIX Library - REPORTS MENU

     0: RETURN To The Main Menu
     1: Sorted by TITLE
     2: Sorted by AUTHOR
     3: Sorted by CATEGORY
Enter your choice>
```

A MENU-DRIVEN APPLICATION

The REPORT_NO Program

- activated whenever you select a report number from the REPORTS menu

Options:

- The -f option considers all lowercase letters to be uppercase letters.
- The -d option ignores all blanks or nonalphanumeric characters.
- The -t option indicates which character is to be used as the field separator.
- The number option (+1 and +2) sorts the file on the next field specified by the field

A MENU-DRIVEN APPLICATION

The REPORT_NO Program

Figure 13.33
Source Code for the REPORT_NO Program

```
$ cat -n REPORT_NO
   1 #
   2 # UNIX library
   3 # REPORT_NO:  This program produces report from the ULIB_FILE file.
   4 #             It checks for the report number passed to it on the
   5 #             command line, sorts and produces reports accordingly.
   6 #
   7 IFS=:"                                        # set delimiter to ;
   8 case $1 in                                    # check the contents of
                                                     the $1
   9   1 ) sort -f -d -t : ULIB_FILE > TEMP ;;     # sort on title field
  10   2 ) sort -f -d -t : +1 ULIB_FILE > TEMP ;; # sort on author field
  11   3 ) sort -f -d -t :+2 ULIB_FILE > TEMP ;;  # sort on category field
  12 esac                                          # end of the case
  13 #
  14 # read records from the sorted file TEMP. Format and store them in
  15 # PTEMP.
  16 #
  17 while read title author category status bname
     date                                          # read a record
  18 do
  19  echo "  Title:$title" >> PTEMP               # format title
  20  echo "      Author: $author" >> PTEMP            # format author
  21  case $category in                            # check the category
  22          [Tt][Bb] ) word=textbook ;;
  23          [Ss][Yy][Ss] ) word=system ;;
```

A MENU-DRIVEN APPLICATION

The REPORT_NO Program

```
do
 echo "  Title:$title" >> PTEMP                # format title
 echo "      Author: $author" >> PTEMP            # format author
 case $category in                            # check the category
         [Tt][Bb] ) word=textbook ;;
     [Ss][Yy][Ss] ) word=system ;;
     [Rr][Ed][Ff] ) word=reference ;;
                * ) word=undefined ;;
 esac
 echo "   Category: $word" >> PTEMP            # format category
 echo "   Status: $status\n" >> PTEMP          # format status
 if [ "$status" = "out"                       # if it is checked out
    then                                      # then
     echo " Checked out by:$bname" >> PTEMP   # format bname
     echo "     Date:$date\n" >> PTEMP         # format date
 fi
 echo >> PTEMP
done < TEMP
#
# ready to display the formatted records in the PTEMP
#
pg -c -p "Page %d:" PTEMP                   # display PTEMP page by page
rm TEMP PTEMP                               # remove files
exit 0                                      # end of the program
$_
```

A MENU-DRIVEN APPLICATION

The REPORT_NO Program

Figure 13.34
A Sample Report Produced by REPORT_NO

```
          Title: UNIX Unbounded
         Author: Afzal Amir
       Category: textbook
         Status: out
Checked out by: Steve Fraser
           Date: 1/12/05

          Title: UNIX For All
         Author: Brown David
       Category: reference
         Status: in

          Title: A Brief UNIX Guide
         Author: Redd Emma
       Category: reference
         Status: out
Checked out by: Steve Fraser
           Date:1/12/05
Page 1:
```