**Chapter 8**

1. Write a program called valid

if [ -n "$(echo $1 | sed -n "/^[\_A-Za-z][\_A-Za-z0-9]\*$/p")" ]

then

echo yes

else

echo no

fi

1. Write a program called t

echo $(date +"%I:%M %p")

1. Write a program mysed

if sed $1 $2 > $2.tmp

then

mv $2.tmp $2

else

rm $2.tmp

fi

1. Write a program isyes

if [[ $1 == Y\* || $1 == y\* ]]

then

exit 0

echo "0"

else

exit 1

echo "1"

fi

1. Use the date and who commands

loginHour=$(who | grep $1 | cut -c32-33)

loginMin=$(who | grep $1 | cut -c35-36)

currentDate=$(date)

hour=$(echo "$date" | cut -c12-13)

minutes=$(echo "$date" | cut -c15-16)

mins=$(( (hour \* 60 + minutes) - (loginHour \* 60 + loginMin) ))

if [ $mins -lt 0 ]

then

mins=$(( deltamins + 24\*60 ))

fi

echo $1 has been logged on for $((mins / 60)) hours $((mins % 60)) minutes

**Chapter 9**

1. Modify the prargs

num=1

while [ "$#" -ne 0 ]

do

echo $num: "$1"

shift

num=$((num + 1))

done

1. Modify the mon program

if [ "$#" -ne 1 ]

then

echo "Usage: mon user"

exit 1

fi

user="$1"

#

# Check every minute for user logging on

#

until who | grep "^$user " > /dev/null

do

sleep 60

done

#

# When we reach this point, the user has logged on

#

echo "$user logged on tty13 "

# Wait until a specified user logs on

#

if [ "$#" -ne 1 ]

then

echo "Usage: mon user"

1. Add a –f option to mon

filename=""

mailopt=FALSE

interval=60

# process command options

while getopts f:mt: option

do

case "$option"

in

f) filename=$OPTARG;;

m) mailopt=TRUE;;

t) interval=$OPTARG;;

\?) echo "Usage: mon [-m] [-t n] user"

echo " -f means check for file"

echo " -m means to be informed by mail"

echo " -t means check every n secs."

exit 1;;

esac

done

# handle file name and user name checking seprately

if [ -n "$filename" ]

then

#

# Check for file

#

until [ -e $filename ]

do

sleep $interval

done

#

# When we reach this point, the file exists

#

if [ "$mailopt" = FALSE ]

then

echo "$filename exists"

else

runner=$(who am i | cut -c1-8)

echo "$filename exists" | mail $runner

fi

else

# Make sure a user name was specified

if [ "$OPTIND" -gt "$#" ]

then

echo "Missing user name!"

exit 2

fi

shiftcount=$((OPTIND - 1))

shift $shiftcount

user=$1

#

# Check for user logging on

#

until who | grep "^$user " > /dev/null

do

sleep $interval

done

#

# When we reach this point, the user has logged on

#

tty=$(who | grep "^$user " | cut -c12-19)

if [ "$mailopt" = FALSE ]

then

echo "$user has logged onto $tty"

else

runner=$(whoami | cut -c1-8)

echo "$user has logged onto $tty" | mail $runner

fi

fi

1. Add a -n option to mon that inverts the monitoring function

Hello world

1. Write a program called

interval=600

while getopts t: option

do

case "$option"

in

t) interval=$OPTARG;;

\?) echo "Usage: collect [-t n]"

echo " -t is the interval (in secs) at which check needs to be done."

exit 1;;

esac

done

while true

do

sleep $interval

num\_users=`whom | wc -l`

echo "$num\_users users are logged in"

done

1. Write a shell program call wgrep

USAGE=" Usage: wgrep <pattern> <filename>"

if [[ "$#" != 2 || ! -f $2 ]] ; then

echo $USAGE ; exit 1

fi

window\_size=1

for line in $(grep -n "$1" $2 | cut -f1 -d:)

do

if [ "$line" -le "$window\_size" ]

then

first=1

else

first=$((line - window\_size))

fi

last=$((line + window\_size))

sed -n "${first},${last}p" < $2

done

1. Modify wgrep to take option -w

window\_size=1

while getopts w: option

do

case "$option"

in

w) window\_size=$OPTARG;;

\?) echo "Usage: wgrep [-w n]"

echo " -w specifies print window."

exit 1;;

esac

done

shiftcount=$((OPTIND - 1))

shift $shiftcount

for line in $(grep -n "$1" $2 | cut -f1 -d:)

do

if [ "$line" -le "$window\_size" ]

then

first=1

else

first=$((line - window\_size))

fi

last=$((line + window\_size))

sed -n "${first},${last}p" < $2

done

1. Modify the wgrap to take variable number of filenames as arguments.

if [[ "$#" < 2 ]] ; then

echo $USAGE ; exit 1

fi

wsize=1

case $1 in

-w) wsize=$2 ; shift 2;;

-?) echo $USAGE

exit 1;;

esac

pattern=$1 ; shift

for file in $\*

do

last=$(cat $file | wc -l)

matched=$(awk '/'$pattern'/ {print NR }' $file)

(( up = matched - wsize ))

(( down = matched + wsize ))

if (( up < 1 )) ; then

for line in $(sed -n "1,$down p" $file) ; do

echo $file:$line

done

elif (( down > last )) ; then

for line in $(sed -n "$up,$last p" $file) ; do

echo $file:$line

done

else

for line in $(sed -n "$up,$down p" $file) ; do

echo $file:$line

done

fi

done

**Chapter 10**

1. What happens to mycp if one or more of the files to be copied doesn't exist

if [ -e "$from" ]

then

if [ -e "$tofile" ]

then

...

fi

else

echo "$from not found"

fi

1. What happens to mycp if one of the filenames contains a character that has a special meaning to the shell such as ; or |?

The file isn’t found, and error message is displayed based on the name of file

1. Write a program called mymv that does with the mv command what mycp does with the cp command

Just use the same mycp, and change the command from cp to mv

1. Modify mycp to prompt for arguments if none are supplied

numargs=$#

filelist=

copylist=

while [ "$#" -gt 1 ]

do

filelist="$filelist $1"

shift

done

to="$1"

if [ "$numargs" -gt 2 -a ! -d "$to" ]

then

echo "Usage: mycp file1 file2"

echo " mycp file(s) dir"

exit 1

fi

if [ "$numargs" -eq 0 ]

then

echo "Source "

read filelist

echo "Destination "

read to

fi

if [ "$numargs" -eq 1 ]

then

filelist=$to

echo "Destination file "

read to

fi

echo mycp $filelist $to

for from in $filelist

do

if [ -d "$to" ]

then

tofile="$to/$(basename $from)"

else

tofile="$to"

fi

1. Add a -n option to mycp that suppresses the normal check for the existence of the destination files

filelist=

copylist=

nopt=FALSE

#

# check for -n argument

#

if [ "$#" -ge 1 -a "$1" = "-n" ]

then

nopt=TRUE

shift

fi

numargs=$# # save this for later use

#

# Process the arguments, storing all but the last in filelist

#

while [ "$#" -gt 1 ]

do

filelist="$filelist $1"

shift

done

to="$1"

#

# If less than two args, or if more than two args and last arg

# is not a directory, then issue an error message

#

if [ "$numargs" -gt 2 -a ! -d "$to" ]

then

echo "Usage: mycp file1 file2"

echo " mycp file(s) dir"

exit 1

fi

#

# if no args given, prompt for from and to file names

#

if [ "$numargs" -eq 0 ]

then

echo "Source file name? \c"

read filelist

echo "Destination file name? \c"

read to

fi

#

# if just one arg given, prompt for to file name

#

if [ "$numargs" -eq 1 ]

then

filelist=$to # previously set to first arg

echo "Destination file name? \c"

read to

fi

echo mycp $filelist $to

#

# Sequence through each file in filelist

#

for from in $filelist

do

#

# See if destination file is a directory

#

if [ -d "$to" ]

then

tofile="$to/$(basename $from)"

else

tofile="$to"

fi

#

# Add file to copylist if file doesn't already exist

# or if user says it's okay to overwrite

#

if [ -e "$tofile" -a "$nopt" = FALSE ]

then

echo "$tofile already exists; overwrite (yes/no)? \c"

read answer

if [ "$answer" = yes ]

then

copylist="$copylist $from"

fi

else

copylist="$copylist $from"

fi

done

#

# Now do the copy -- first make sure there's something to copy

#

if [ -n "$copylist" ]

then

cp $copylist $to # proceed with the copy

fi

1. Modify mycp to use sed instead of the while loop to process the arguments typed on the command line

count=$( grep “$1” $2 | wc –l )

regexp=”$1”

file=”$2”

grep “$regexp” $file > match.tmp

while read line

do

if grep “$line” match.tmp > /dev/null

then

echo $( sed –n “/$line/p” $file )

echo –e “\n

else

:

fi

done < $file

echo $count matches

1. Modify the rem program used by rolo so that if multiple entries are found, the program will prompt the user for the entry to be removed

Echo “enter name to remove”

read name

if [ -z "$name" ]

then

echo "Removal ignored"

else

rem "$name"

fi;;

1. Modify rolo so that the menu is redisplayed after each selection is made and processed.

do

#

# Display menu

#

echo 'Would you like to:

1. Look someone up

2. Add someone to the phone book

3. Remove someone from the phone book

Please select one of the above (1-3): \c'

read choice

…

done

1. What happens to the rolo program if just an Enter is given as the name for the add, look up, or remove options?

add: an entry with a blank name field will be created

lookup: depending on the local implementation of grep, either an error will be produced by grep or all the phone book entries will be displayed

remove: depending on the local implementation of grep, either an error will be produced by grep and the "I couldn't find $name" message displayed, or all the phone book entries will be found and the "More than one match" error message displayed

1. Modify lu to use printf to print the name and phone number so that they line up in columns for names up to 40 characters in length

printf “%40s” $1 phonebook

**Chapter 11**

1. Write a program called myrm that takes as arguments the names of files to be removed

if [ $# -gt 0 ]

then

# if it is, prompt user before removing files

echo "Remove $# files (y/n)? \c"

read reply

if [ "$reply" = y ]

then

rm "$@"

fi

else

echo "error: no of files less than zero"

fi

1. What output would you expect after typing the following:
   1. $ e2=20; export e2
   2. $ e4=40 prog1

Output: 100 20 300 40

1. Modify rolo from this chapter so that a person running the program can keep his or her phone book file in any directory and not just in the home directory

if [ -z "$PHONEBOOK" ]

then

PHONEBOOK=$HOME/phonebook

export PHONEBOOK

fi

**Chapter 12**

1. What will be the results of the following commands?
   1. echo ${EDITOR}

/bin/vi

* 1. echo ${DB:=/users/pat/db}

/users/pat/db

* 1. echo ${EDITOR:-/bin/ed}

/bin/vi

* 1. echo ${PHONEBOOK:?}

nothing to print (empty line)

* 1. echo ${DB:-/users/pat/db}

/users/pat/db

* 1. ed=${EDITFLAG:+${EDITOR:-/bin/ed}}

/bin/vi

1. Rewrite the home program from Exercise 5 in Chapter 7 to use the set command and the IFS to extract the home directory from /etc/passwd

$ vi home

IFS=:

set -IFS

grep "^$1$" /etc/passwd | cut –d$IFS -f6

$ chmod 744 home

1. Using the fact that the shell construct ${#var} gives the number of characters stored in var, rewrite wc in the shell

if [ $# -eq 0 ]

then

set -- ""

fi

OIFS=$IFS

NIFS='

'

for file

do

totallines=0

totalwords=0

totalchars=0

IFS=$NIFS

# read line with new IFS, preserve whitespace

cat $file | while read -r line

do

# parse line into words with original IFS

IFS=$OIFS

set -- $line

IFS=$NIFS

# now add up lines, words, and chars

totallines=$((totallines + 1))

totalwords=$((totalwords + $#))

totalchars=$((totalchars + ${#line} + 1))

done

echo " $totallines $totalwords $totalchars $file"

done

1. Write a function called rightmatch that takes two arguments as shown:

rightmatch value pattern

rightmatch () {

echo "${1%$2}"

}

1. Write a function called leftmatch that works similarly to the rightmatch function developed in Exercise 4

leftmatch() {

echo "${2#$1}"

}

1. Write a function called substring that uses the leftmatch and rightmatch functions developed in Exercises 4 and 5 to remove a pattern from the left and right side of a value

rightmatch $(leftmatch $1 $2) $3

1. Modify the substring, leftmatch, and rightmatch functions developed in the previous exercises to take options that allow you to remove the largest possible matches of the specified pattern from the left or right side of the specified value

rightmatch () {

echo "${1%%$2}"

}

leftmatch() {

echo "${2##$1}"

}

**Chapter 13**

1. Using eval, write a program called recho that prints its arguments in reverse order.

arg=$#

while [ $arg -gt 0 ]

do

eval echo \"\$$arg \ \"

arg=$((arg - 1))

done

1. Modify the shar program presented in this chapter to handle directories

shar () {

for file

do

dir=$(dirname "$file")

echo "if [ -e $dir -a ! -d $dir ]"

echo "then"

echo " echo $dir already exists and is not a directory!"

echo " echo extraction quits!"

echo " exit 1"

echo "fi"

echo "mkdir -p $dir"

if [ -d "$file" ]

then

shar $file/\*

continue

fi

if [ -f "$file" ]

then

echo

echo "echo Extracting $file"

echo "cat >$file <<\THE-END-OF-DATA"

cat $file

echo "THE-END-OF-DATA"

size=$(wc -c < "$file")

echo "size=\$(wc -c < $file)"

echo "if [ \$size -ne $size ]"

echo "then"

echo "echo $file: expected $size characters, extracted \$size."

fi

done

}

echo "#"

echo "# To restore, type sh archive"

echo "#"

shar "$@"

1. Modify shar to include in the archive the character count for each file and commands to compare the count of each extracted file against the count of the original file

shar () {

for file

do

dir=$(dirname "$file")

echo "if [ -e $dir -a ! -d $dir ]"

echo "then"

echo " echo $dir already exists and is not a directory!"

echo " echo extraction quits!"

echo " exit 1"

echo "fi"

echo "mkdir -p $dir"

if [ -d "$file" ]

then

shar $file/\*

continue

fi

if [ -f "$file" ]

then

echo

echo "echo Extracting $file"

echo "cat >$file <<\THE-END-OF-DATA"

cat $file

echo "THE-END-OF-DATA"

size=$(wc -c < "$file")

echo "size=\$(wc -c < $file)"

echo "if [ \$size -ne $size ]"

echo "then"

echo "echo $file: expected $size characters, extracted \$size."

fi

done

}