

Assignment 3 and 4: Optional Scripting

Complete assignment due: Thursday, Sep. 25th, 8:00 p.m.

The Korn shell scripts in this assignment explore the ways to handle options on the command-line. These are the “dash” options, such as **-i** or **-q**. The Korn shell supports an expanded version of **getopts**, which is a tool that aids in processing the options. We will see the use of **getopts** in C programs later in the semester.

The first script uses a simpler version of **getopts**, that allows single character options. The options can occur individually on the command line, as in:

```
ksh fileOpts.ksh -c -f filename1 filename2
```

or they can be grouped together, as in

```
ksh fileOpts.ksh -cf filename1 filename2
```

The second script uses an expanded version of **getopts** that can handle **--** arguments, such as

```
now.ksh --before --month
```

which has the same meaning as

```
now.ksh -b -m
```

The expanded version also supports the **--help** and **--man** options.

Problem 1: File Options

Write a Korn shell script named **fileOpts.ksh**. The script will perform various actions on each of its arguments. Which operations are performed will be determined by the options specified. The synopsis of the command is:

```
ksh fileOpts.ksh [-c] [-l] [-p #] [-f] [-r filename] [ file ... ]
```

The four command-line options have the following meanings:

- **-c** Run **wc** on each of the file arguments
- **-l** Run **ls -l** on each of the file arguments
- **-p** Print some or all of the contents of each of the file arguments. There will be a numeric argument provided. When the numeric argument is **0**, use **cat** to print the entire file. When the argument is positive, use **head** to print the first part of the file. When the argument is negative, use **tail** to print the last part of the file. For head and tail, the numeric argument specifies the number of lines to print.
- **-f** Run **file** on each of the file arguments.
- **-r** Send the standard output of all the requested operations to the specified file.

When multiple options are chosen and multiple filenames are specified, **fileOpts.ksh** will perform all of the options on the first file, followed by all of the options on the second file, etc.

Some examples:

```
$ export books="/home/cs352/fall108/books/"
$ ksh fileOpts.ksh -c $books/sawyer.txt $books/gettysburg.txt
8303 71910 400480 /home/cs352/fall108/books/sawyer.txt
23 280 1495 /home/cs352/fall108/books/gettysburg.txt
```

```

$ ksh fileOpts.ksh -l $books/sawyer.txt $books/gettysburg.txt
-rw-rw-r-- 1 patrick cs352f08 400480 2007-02-05 17:26 /home/cs352/fall08/
books/sawyer.txt
-rw-rw-r-- 1 patrick cs352f08 1495 2008-08-26 15:19 /home/cs352/fall08/books/
gettysburg.txt
$ fileOpts.ksh -f $books/*
/home/cs352/fall08/books/gettysburg.txt: ASCII English text
/home/cs352/fall08/books/hesperus.txt: ASCII English text
/home/cs352/fall08/books/lgDictionary.txt: ASCII text
/home/cs352/fall08/books/longfellow.txt: ASCII English text
/home/cs352/fall08/books/mobydick.txt: ASCII English text
/home/cs352/fall08/books/mohicans.txt: ISO-8859 English text
/home/cs352/fall08/books/musketeers.txt: ASCII English text
/home/cs352/fall08/books/noSpace.txt: ASCII text
/home/cs352/fall08/books/raven.txt: ASCII English text
/home/cs352/fall08/books/sawyer.txt: OS/2 REXX batch file text
/home/cs352/fall08/books/words: ASCII C program text
$ ksh fileOpts.ksh -p 3 $books/gettysburg.txt $books/raven.txt
Gettysburg Address

```

Four-score and seven years ago, our fathers brought forth on this
Edgar Allan Poe: The Raven

```

$ ksh fileOpts.ksh -p -4 $books/gettysburg.txt $books/raven.txt
measure of devotion - that we here highly resolve that these dead shall
not have died in vain - that this nation, under God, shall have a new
birth of freedom - and that government of the people, by the people, for
the people, shall not perish from the earth.
And his eyes have all the seeming of a demon's that is dreaming,
And the lamp-light o'er him streaming throws his shadow on the floor;
And my soul from out that shadow that lies floating on the floor
Shall be lifted - nevermore!
$ ksh fileOpts.ksh -clfp 4 $books/gettysburg.txt $books/raven.txt
 23 280 1495 /home/cs352/fall08/books/gettysburg.txt
-rw-rw-r-- 1 patrick cs352f08 1495 2008-08-26 15:19 /home/cs352/fall08/books/
gettysburg.txt
Gettysburg Address

```

Four-score and seven years ago, our fathers brought forth on this
continent a new nation, conceived in liberty and dedicated to the

```

/home/cs352/fall08/books/gettysburg.txt: ASCII English text
 127 1128 6261 /home/cs352/fall08/books/raven.txt
-rw-rw-r-- 1 patrick cs352f08 6261 2006-08-21 20:58 /home/cs352/fall08/books/
raven.txt
Edgar Allan Poe: The Raven

```

Once upon a midnight dreary, while I pondered, weak and weary,

```

/home/cs352/fall08/books/raven.txt: ASCII English text
$ ksh fileOpts.ksh -cftp -2 -r zap.out $books/gettysburg.txt $books/raven.txt
$ cat zap.out
 23 280 1495 /home/cs352/fall08/books/gettysburg.txt

```

birth of freedom - and that government of the people, by the people, for the people, shall not perish from the earth.

/home/cs352/fall08/books/gettysburg.txt: ASCII English text

127 1128 6261 /home/cs352/fall08/books/raven.txt

And my soul from out that shadow that lies floating on the floor

Shall be lifted - nevermore!

/home/cs352/fall08/books/raven.txt: ASCII English text

The following script gives a simple example of how to use **getopts** in a Korn shell script. In addition to the example, the manpage for **ksh** provides an overview. Search for the string “**getopts**”; the third occurrence will be the section that describes it. The manpage also describes the **\${OPTIND}** and **\${OPTARG}** variables.

This **simpleOpts.ksh** script is available in **~cs352/fall08/UnixExamples**.

```
#!/bin/ksh
```

```
a="not chosen"
```

```
b="not chosen"
```

```
b_arg="not chosen"
```

```
c="not chosen"
```

```
c_arg="not chosen"
```

```
d="not chosen"
```

```
# The while loop will execute until there are no more command-line
# arguments that start with a hyphen. The options can be grouped
# together, as in: ksh simpleOpts.ksh -ad
# The # following b indicates that when -b is present, it has a
# required numeric argument following it.
# The : following c indicates that when -c is present, it has a
# required argument following it. Typically, this will be a string,
# such as a filename.
# In the case statement, the variable ${OPTARG} is the argument
# following the current option being processed. It can be used to
# get a numeric or text argument following a parameter.
while getopts "ab#c:d" optchar ; do
    case ${optchar} in
        a) a="selected"
           ;;
        b) b="selected"
           b_arg=${OPTARG}
           ;;
        c) c="selected"
           c_arg=${OPTARG}
           ;;
        d) d="selected"
           ;;
    )
        # When getopts encounters an option that is not listed in its
        # options string, ${optchar} will be assigned the ? character.
```

```
# getopt will print to stderr an error message about the
# invalid option. This arm of the case statement can be used
# to print additional error messages, such as a Usage statement.
# Depending on circumstances, the programmer may, or may not,
# choose to exit at this point.
\?) echo "error message goes here"
    exit 1
    ;;
esac
done

# The shift statement following the while loop will shift the
# options that have been processed. ${OPTIND} is the index of the
# next command-line argument to be processed.
shift $(( ${OPTIND} - 1 ))

# Some print statements to show values that might have been acquired
# while processing options.
echo a is ${a}
echo b is ${b}
echo b_arg is ${b_arg}
echo c is ${c}
echo c_arg is ${c_arg}
echo d is ${d}

# Since we shifted above, $1 is now the first command-line argument
# following the options. Here is a simple loop to print those
# arguments, if any.
echo "The remaining command-line arguments are:"
for i in "$@"; do
    echo "'${i}'"
done

exit 0;
```

Turnin: Use **turnin** to turn in your completed **fileOpts.ksh** file. The command is:

```
turnin 352assign3 fileOpts.ksh
```

See the man page for the **turnin** program for details on what turn in can do and how you can confirm that your file was turned in.

Problem 2: Now Options

Write a Korn shell script named **now.ksh**. The script will print the day and day-of-the-week for the current date or for a specified (future or past) date. The script has 6 possible command-line options. Three of the options are used to specify how the date is to be printed. The other three can be used to specify the desired date.

Here is the --man output from my version of now.ksh:

NAME

now.ksh -- display day-of-the-week information

SYNOPSIS

now.ksh [options]

DESCRIPTION

The **now.ksh** script will print the requested date in one of four formats: the entire week that contains the date, the month showing the dates up to and including the requested date, the month starting with the requested date to the end of the month, or the day-of-the-week and date. The **-w**, **-b**, and **-a** options select the first three choices. The absence of all three options selects the last choice.

The date used by the program will default to today.

The **-d**, **-m**, and **-y** options allow choice of a different day, month, or year, respectively. If any of the three are missing, the script uses the corresponding values from today. "Today" here refers to the date on which the script is being run.

OPTIONS

-w, --week print the whole week
-b, --before print the month up to and including the requested day
-a, --after print the month starting from the requested day to the end of the month or week
-d, --day=day requested day
 1-31
 e default value is 15.
-m, --month=month
 month of requested day
 1-12
 e default value is 9.
-y, --year=year year of requested day. The default value is 2008.

SEE ALSO

cal(1), cut(1), date(1), egrep(1), fgrep(1), tr(1)

IMPLEMENTATION

version now.ksh, version 1.0
author Patrick Homer
copyright Copyright (c) September 25, 2008

Note: There are two “mistakes” above that are not mistakes. For the **-d** option, the phrase “**e default value is 15.**” should actually be “**The default value is 15.**” There is an

analogous mistake for **-m**. Fedora on lectura (and Mac OS X) drops the “**Th**” that should be at the beginning of each phrase. The “**Th**” is actually in my script (and should be in yours).

Note: The default values listed in the **--man** output above for **-d**, **-m**, and **-y** come from the day on which I ran the script, 9/15/2008. Your script should fill in the correct values for the day on which it is running.

Here are a few examples from running my script. Note that I ran these examples on September 15, 2008:

```
$ ksh now.ksh
Mo
15
$ ksh now.ksh -d 7 -m 12 -y 1941
Su
7
$ ksh now.ksh -d 7 -m 12 --week
Su Mo Tu We Th Fr Sa
7 8 9 10 11 12 13
$ ksh now.ksh -w -m 12 -y 1941
Su Mo Tu We Th Fr Sa
14 15 16 17 18 19 20
$ ksh now.ksh -m 12 -y 1941 --before
December 1941
Su Mo Tu We Th Fr Sa
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15
$ ksh now.ksh --month 10 -d 31 --week
Su Mo Tu We Th Fr Sa
26 27 28 29 30 31
$ ksh now.ksh --month -d 31 --week
now.ksh: --month: numeric month value expected
Usage: now.ksh [-wba] [-d day] [-m month] [-y year]
$
```

Hints:

This assignment consists of the expanded **getopts** features, and the printing of the desired date results. You can get the expanded **getopts** features working without having to address the date results. This will enable you to pass test cases that ask for **--man**, **--help**, **--html**, and various error cases.

The **phaser4expanded.ksh** script, located in **~cs352/fall08/UnixExamples**, is taken from Appendix B of *Learning the Korn Shell*. I have added many comments to the file to explain the various parts.

You can extract the information needed for printing the requested date by making use of **cal** and **date**. Tools such as **cut** and **egrep/fgrep** are also useful. We will cover the use of regular expressions in class, which will help with **egrep**. The manpage for **cut** will prove useful.

Turnin: Use **turnin** to turn in your completed **now.ksh** file. The command is:

```
turnin 352assign3 now.ksh
```

See the man page for the **turnin** program for details on what turn in can do and how you can confirm that your file was turned in.

Final Summary:

There are two files that you will need to turnin for full credit on this assignment:

```
fileOpts.ksh
```

```
now.ksh
```