

## Lab #3: More UNIX Shell Scripts


*The purpose of this lab is to write a useful, complex UNIX shell script for Solaris or Linux. We will start with a skeleton shell script and add code to produce a full-featured suite of tools.*

---

### CREATING AND RUNNING A SHELL SCRIPT

Start two **Console** or **Terminal** (*This Host*) windows. One window will be used to test your shell scripts; the other to view manual pages.

Start an editor, such as the **Text Editor** (**dtpad**). The editor will be used to create and edit your shell scripts.

Type the following example into your editor. Save the file as **skel**. At each line marked  save your shell script and run it, either with **bash skel** or with **chmod +x skel** (*you only have to run this command once*) and **./skel**. Experiment with **-x**, **set -x**, **set +x**, **pause**, and passing various command line parameters to see what the shell script is doing at each stage. See the **Examples** section for command lines to try out.

```
#!/usr/bin/bash
#
#       CTEC1430/2009W Shell Programming Assignment
#       Skeleton code for myrm
#
```



```
# function to pause
```

```
pause( )
{
    echo -n Press Enter to continue
    read junk
}
```



```
# variable for "recycling bin" directory
```

```
RecycleBin="${HOME}/.Recycled"
```

### Lab #3: More UNIX Shell Scripts

```
# function to display program usage
```

```
usage( )
{
    status=${1-0}

    echo "usage: $0" '[-i] [--] filename [...]'
    echo "          $0 -c"

    exit $status
}
```



```
# check for the minimum number of arguments
```

```
if [ $# -lt 1 ]
then
    usage 1
fi
```



```
# set switches to default mode (off)
```

```
interactive_mode=0
```

```
# parse the command line arguments
```

```
while [ "$1" != "" ]
do
    case "$1" in
        -i)    interactive_mode=1
               echo "set interactive mode ($interactive_mode)"
               ;;
    esac
done
```

**Lab #3: More UNIX Shell Scripts**

```
--)    shift    # skip --
        break
        ;;

-*)    echo "unknown switch '$1'"
        usage 2
        ;;

*)     break    # this is a file:  no more switches
        ;;

esac

shift

done
```



```
# make sure at least one filename has been passed
```

```
if [ "$1" == "" ]
then
    usage 3
fi
```



### Lab #3: More UNIX Shell Scripts

```
# function to get user confirmation (interactive mode)
```

```
confirm( )
{
    echo -n $* '[y/n]? '
    read confirmation junk
    confirmation=`echo $confirmation | cut -c 1`
    if [ "$confirmation" == "y" -o \
        "$confirmation" == "Y" ]
    then
        confirmed=1
    else
        confirmed=0
    fi
}
```



```
# start processing the files
```

```
for f in $*
do
    echo $f
done
```

```
exit 0
```



### Lab #3: More UNIX Shell Scripts

Next, put this if statement before the echo \$f in the for loop:

```
if [ "$f" == "." -o "$f" == ".." ]
then
    echo "can't delete current or parent directories"
fi
```



Next, add this if statement after the previous if statement, but before the echo \$f:

```
if [ $interactive_mode -eq 1 ]
then
    confirm remove $f
else
    confirmed=1
fi
```



Finally, replace the echo \$f with the following code:

```
if [ $confirmed -eq 1 ]
then
    echo mv "${f}" "${RecycleBin}"
fi
```



## Lab #3: More UNIX Shell Scripts

**Examples** (test cases). For all examples below, *somefile* and *someotherfile* are the names of files that may or may not exist, *somedir* and *someotherdir* are the names of directories that may or may not exist.

```
./skel
./skel -i
./skel -q
./skel -h
./skel --help
./skel --
./skel -- -i
./skel -- -q
./skel somefile
./skel somefile someotherfile
./skel somedir
./skel somedir someotherdir
./skel somefile somedir
./skel somedir somefile
./skel -i somefile
./skel -i somefile someotherfile
./skel -i somedir
./skel -i somedir someotherdir
./skel -i somefile somedir
./skel -i somedir somefile
./skel -q somefile
./skel *
```

---

## Lab #3: More UNIX Shell Scripts

### Shell Programming Assignment

**Due date: Thursday, March 12, 2009 by 5:30 PM**

#### Introduction

It's always easier not to make a mistake than to recover from it once the mistake is made. The first rule to avoid such mistakes is to avoid certain irrevocable commands. UNIX makes it remarkably easy to simply remove entire directory structures without even a warning. Now, this power is impressive, if that is what you are sure you really want to do. But if that wasn't what you had in mind, then going "oops" won't help.

#### Instructions

The goal of this assignment is to write a shell script to replace the **rm** command and provide undelete capability similar to MS-DOS.

Write a shell program to duplicate the UNIX **rm** command with the following features:

1. It will have a switch **-i** that will act in the same manner as in the **rm** command.
2. Instead of deleting the files, it will move them to a **wastebasket** directory. If the file already exists in the **wastebasket** directory, then the existing file (in the **wastebasket**) will have the version number zero appended to it and the newly deleted file will have version number one appended to it. If the version number setup is already in use for that file then the newest one will simply have the next version number in the series appended to it. It's your choice as to what system to use for this. Just be sure that it is a unique system and will not be part of any normal file naming convention you might use.
3. It will have a switch **-c** that will clear the entire wastebasket after asking for confirmation. The **-i** and **-c** switches may be combined to ask for confirmation on individual files.

Don't forget that wildcards and multiple files are allowed on the command line. The switches when combined may be in any order but must be before any filenames. Your program must also catch any operator errors.

## Lab #3: More UNIX Shell Scripts

### Evaluation

The program is worth 10 marks. Out of the 10 marks, 2 marks will be assigned for documentation (comments), and 2 marks for style (indenting, etc).

After March 12, a late penalty of 10% (1 mark) per day late will be assessed. Scripts handed in after March 19 will receive a zero.