Shell Programming Assignment 2

Deadline: **Thurs. Dec. 3rd 2015, 5pm**

Submit: **1.** a fully and meaningfully *commented* version of your program

2. an uncommented plain text version.

Send both as attachments to an email to paul.rothwell@cit.ie I will send a receipt email. If you do not receive an email in reply confirming that I have received your assignment by **Thursday Dec. 3rd 2015, 6pm** then you should **send it again.** Be careful to actually attach the file.

Notes: **The commented program should be commented fully and meaningfully.** There is room for plenty of difference, assignments that look exactly similar will be assumed not to be original work and will be marked as 0%. The assignment is worth 15% of the overall module marks. Award of marks will be subject to a satisfactory ability to explain your code. You are expected to complete an attempt at doing this assignment with help from the notes, the web, books, other people, etc. However, having presented a program for marking you are also expected to understand how it works.

Assignment

Write a non-interactive script that allows to copy all of the ordinary files from an existing directory to another named new directory. The script should expect the user to supply arguments on the command line giving:

- the name of a source directory from which to copy the ordinary files,
- the name of a new destination directory to create and into which to place the copied files.

The script should report errors where:

1. the number of arguments given is below 2 (fatal error);
2. the source directory name given does not name an existing directory (fatal error);
3. the destination directory name given names an already existing directory (fatal error);
4. the destination directory name given names an already existing file (fatal error);
5. the attempt to create the new directory fails (fatal error);
6. the attempt to copy any of the files fails (warning errors).

Examples

Here's how some example runs would look (assume the program is called 'ass2' and **user input in bold**):

A successful run would look something like this (**user input in bold**):

ass2 source destdir
ass2: All ordinary files from directory source have been copied to destdir.

An unsuccessful run would look something like this (**user input in bold**):

ass2
ass2: ERROR: Incorrect number of arguments supplied.
ass2: USAGE: ass2 <source directory> <new destination directory>

Another unsuccessful run would look something like this (**user input in bold**):

ass2 source
ass2: ERROR: Incorrect number of arguments supplied.
ass2: USAGE: ass2 <source directory> <new destination directory>

Another unsuccessful run would look something like this (**user input in bold**):

ass2 sourceNotThere destdir
ass2: ERROR: Source directory SourceNotThere does not exist.
ass2: USAGE: ass2 <source directory> <new destination directory>

A partly unsuccessful run would look something like this (**user input in bold**):

ass2 source destdir
ass2: WARNING: File somefile could not be copied to destdir.
ass2: WARNING: File someotherfile could not be copied to destdir.
ass2: Some ordinary files from directory source have been copied to destdir.

Notes:

Errors should be echoed to the standard error channel and adhere to the standard syntax as before. For example:

```
echo    "${0}: ERROR: Destination directory already exists." 1>&2
echo    "${0}: USAGE: ${0} <source directory> <new destination directory>" 1>&2
exit    4    #if required
```

Which, in an example run would display on the error channel as:

ass2: ERROR: Destination directory already exists.
ass2: USAGE: ass2 <source directory> <new destination directory>

All your exits for errors should be given a unique exit status number 1-255.

Pseudocode for assignment 1:

1. Check number of arguments given.
2. Exit with error message if incorrect.
3. Check that source directory exists.
4. If not then exit with error.
5. Check if name given for destination directory names an existing directory.
6. If so exit with error message.
7. Check if name given for destination directory names an existing file.
8. If so exit with error message.
9. Try to create the destination directory.
10. If fail send error message and exit.
11. Initialise variables for number of copied files and number of failed copies.
12. Enter loop to copy all ordinary files from source directory to destination directory.
    a) Try to copy the next file
    b) If this copy fails send warning message, record extra fail, and continue.
    c) Otherwise record extra copied file and continue.
13. If completely successful send confirmation message and exit.
14. Otherwise, if partial fail send partial confirmation and exit.
15. Otherwise if no files copied send information message and exit.

Sources of required information

You will need to find out:

1. How to handle command line arguments using positional parameters – see lab 8: http://mcom.cit.ie/staff/Computing/prothwell/bbsos/labs/lsp8.pdf 'Positional parameters' and examples.
2. How to use integer variables for counting – see lab 7: http://mcom.cit.ie/staff/Computing/prothwell/bbsos/labs/lsp7.pdf 'Let command'
3. How to write for loops – see lab 9: http://mcom.cit.ie/staff/Computing/prothwell/bbsos/labs/lsp9.pdf 'For loops' & Examples.
4. How to generate a list of files in a directory for a for loop to work on using 'command substitution' – see lab 8: http://mcom.cit.ie/staff/Computing/prothwell/bbsos/labs/lsp8.pdf 'Command Substitution'
5. How to increment variables using the let command '(( ))' - see see lab 7: http://mcom.cit.ie/staff/Computing/prothwell/bbsos/labs/lsp7.pdf 'Let

command'

6. How to write boolean expressions checking values of integer variables – see see lab 7: http://mcom.cit.ie/staff/Computing/prothwell/bbsos/labs/lsp7.pdf 'Let command'

Example Program

The following example program may help:

A program to accept a directory name and count & identify all of the executable files in that directory.

```
#!/bin/bash
if ((${#}<1)) #if arguments not entered (i.e. Number of args < 1)
then
    echo "${0}:ERROR:No argument given." 1>&2
    echo "${0}:USAGE: ${0}  <existing directory>" 1>&2
    exit 1
fi
typeset -i  exefiles=0   #initialise number of executables
for fname in $(ls "${1}")     #for each file in directory named
#$(ls "${1}") generates a list of the files in the directory named
#in ${1}
do
     if [[ -e "${1}/${fname}" ]]   #if the file is executable
     then
          ((exeFiles++)) #increment count of executables
          echo "${0}:INFO: ${fname} is executable." 1>&2
     fi
done
if ((exeFiles>0))
then
     echo "${0}:INFO:${exeFiles} executable files found in ${1}"
1>&2
else
     echo "${0}:INFO:No executable files found in ${1}" 1>&2
fi
```