Operating Systems – Laboratory

Hristo Trifonov

20/March/2019

For SDip in Embedded Systems Engineering

Laboratory Assignment #5

OBJECTIVES:

The objectives of this laboratory assignment are as follows:

1) Learn how to use a bash shell **array** in a script program
2) Learn how to use **dd** command and **bc** utility from the bash shell
3) Learn how to measure UNIX **elapsed time** to high resolution
4) Experimentally measure **file transfer** rates

INSTRUCTIONS:

- Students will provide individual submissions (however, learning/study cooperation is encouraged).
- A short report **document** file must be submitted to describe the operation of your program and to comment on any problems etc. (see Addendum for details).

SUBMISSION:

Students will submit via SULIS (EE5012 page) by 23:55 hours, Wednesday 27th March 2019. Late reports will not be accepted. The submitted files will be the:

- a **report** file (pdf) – see format in the Addendum
- script solution code for **copy_rate**

Please put all of the above in a folder named **"Assignment5_yourIdNumber"** and compress/archive with **zip**, **tar** or whichever program you prefer before submitting to SULIS.

The student name and ID number is to be on the heading comments of any script file. The programs are to be commented for readability. Individual submissions only will be accepted. The student can be asked to demonstrate the working programs in the lab.

Assignment assessment weightings:

Assignment #1	10% of module
Assignment #2	10% of module
Assignment #3	10% of module
Assignment #4	10% of module
Assignment #5	**10% of module …. this assignment**

There will be a compulsory exam question in the final exam based on the laboratory assignments.

INSTRUCTIONS

Please complete the following exercise.

STEPS:

1) Study the UNIT 5 – Tutorial.
2) Make sure that you know how to create and use a bash shell **array**.
3) Exercise using the **date**, **dd** and **bc** commands in the shell.

SCRIPT EXERCISE:

Write a shell script called **copy_rate** to do the following:

- Make an **array** of 5 file names.

- Create five actual files, using **dd**, with various sizes ranging from 10 Kbytes to 50 Mbytes.

- Your script must contain a **function** to do the following:

    - The *function* is called with a **filename** parameter (positional parameter $1)
    - Copy (**cp**) a file (represented by $1) to any file name (e.g. _temp)
    - Calculate the **size** of the file in ***Kilobytes*** that is copied using the **wc** command.
    - Measure the **elapsed time** in ***milliseconds*** for the file copy operation (please use the **date** command to read the time)
    - Calculate the **data transfer** rate for the file copy operation in **Kbytes / s** (data transfer rate is: (file size) / (elapsed time))

- For each file name in the array, call the function to do a file copy iteration.

- Print out to the terminal a summary output to display the following for each file:

    - **file name, file size, copy(elapsed) time, transfer rate**

Note – you can display the information from within your function; or from outside your function if you use global variables for argument passing.

Report the results for the five files in your **report** document.

HINTS

HINT 1 - *MAKING FILES OF ARBITARY SIZES*

First make some test files of various sizes ranging from a small file (e.g. 10kB) to a large file (e.g. 50MB) can be achieved using the UNIX dd utility.

The original purpose of the UNIX **dd** (data definition) command-line utility was to convert and copy files. However, the command has a much wider range of application including disk imaging for forensics purposes and making files of arbitrary sizes etc.

For experimental purposes we can make files of arbitrary size using the **dd** command as in the following examples:

```
dd if=/dev/zero of=myFile_1 bs=1024k count=5     # creates myFile_1 of 5 MBytes

dd if=/dev/zero of=myFile_2 bs=1k count=10       # creates myFile_2 of 10 kiloBytes
```

In a script file you will not want your **dd** command’s output text details cluttering up your screen, so you can supress such outputs using the **status=none** option, as follows:

```
dd if=/dev/zero of=myFile_2 bs=1024k count=5  status=none
```

HINT 2 - *COPYING ONE FILE TO ANOTHER*

You can use the **cp** command to copy a file as follows:

```
cp  <file>  <file>
```

HINT 3 - *FINDING THE SIZE OF A FILE*

There are many ways to find the size of a file but an easy (crude) way to get the size of a file is to count the number of characters in the file using **wc** as follows:

```
size=$(wc -c  <  myFile)
```

HINT 4 – *MEASURE TIME TO HIGH RESOLUTION*

To read time to millisecond resolution, you could take the following approach:

```
date  +%s%N                # this will show current time to nanosecond resolution

date  +%s%N /1000000    # this will convert to millisecond resolution
```

Here is an example script to time how many nanoseconds it takes for the **sleep** command to sleep for a nominal one second:

```
#! /bin/bash
# measure how long a nominal 1 second sleep takes in nanoseconds
################################################################

t1=$(( $(date +%s%N) ))      # read time in nanoseconds

sleep 1                        # sleeps for a nominal 1 second

t2=$(( $(date +%s%N) ))       # read time again in nanoseconds

elapsedTime=$(( t2 - t1 ))      # compare time differences

echo "The one second sleep took an actual $elapsedTime nanoseconds"

exit
```

ADDENDUM

The document file

The submission for this laboratory assignment will include a document. Note, the document does not at all need to be very long and wordy – but must be of good quality, to the standard of a small technical report, and presented as listed below:

1) The file will be submitted as a **PDF** file.

2) The document will have the following information and sections

Front page
Title page with student name, ID, date, module code, assignment number (e.g. *Assignment #1*)

Requirements
Briefly summarise the assignment requirements from the assignment instructions in the handout.

Description of solution
Describe your solutions noting any special problems or issues.

Testing and results
State how you tested your program and record any results, using **screenshots** to show actual outputs.

Statement of completion
Briefly make a statement saying that you have completed all of the requirements, or summarise any aspects that you could not complete.

Source code
Include your source code as part of this document. You will also submit a separate plain text source code file or files as stated on the cover page of this assignment.

NOTE: A student can lose up to 30% of assignment marks for a bad report.