UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

FACULTY OF SCIENCE & ENGINEERING

DEPARTMENT OF ELECTRONIC AND COMPUTER ENGINEERING

MODULE CODE:	**ET4725**
MODULE TITLE:	**Operating Systems 1**
SEMESTER:	**Semester 2 - 2016/17**
DURATION OF EXAM:	**2.5 Hours**
LECTURER:	**Dr. D. Heffernan**

IMPORTANT INSTRUCTIONS TO CANDIDATES:

- **Answer any THREE questions**
- **This exam represents 70% of the full module assessment**
- **All questions are of equal weight**
- **If you answer more than three questions you will be marked on the three best answers only**
- **The addendum contains a list of common bash shell commands**

Q1 **33 marks**

Write a **bash** shell script to do the following.

- Make an array of **four** file names.
- Create the four actual files, with various file sizes ranging from 10kBytes to 500MBytes.
- For each of the four files named in the array, call the **f_copy()** function.
- The **f_copy() function** will do the following:
    - The function is called with one file name parameter argument
    - Copy (**cp**) the specified file to any file name
    - Calculate the size of the file that is copied
    - Measure the elapsed time in **milliseconds** for the file copy operation
    - Calculate the data transfer **rate** for the file copy operation (i.e. file size/elapsed time)
    - Print a summary output to show: file name, file size, copy time, transfer rate.

NOTE – In the addendum of this paper there is a list of common bash shell commands.

Q2

a) **6 marks**

Briefly state the meaning of the following terms in relation to computer virtualisation:

Type-1 hypervisor	Type-2 hypervisor	Virtual machine
Virtual appliance	Para-virtualisation	Thin client

b) **8 marks**

Consider the following two architectural models for **implementing multiple applications** on a single host system:

Virtual machine (VM) model **Docker container** model

With the aid of a diagram, briefly explain the concept for each of the two models. Clearly highlight any advantages and disadvantages for each model

c) **19 marks**

Figure Q2 shows a physical computer hardware layout. You are required to configure a virtualised system on to this layout so that it includes the following:

- The host 1 will have **one guest VM** (virtual machine), based on a **Linux** OS.
- The host 2 will have **two guest VMs** (virtual machines), each based on a **BSD** OS.
- Part of the physical disk storage will be assigned to a **single data store** that is configured as a **clustered** file system.
- For **host 1** its **VM** will have access to **two** virtual SCSI disk drives within the data store.
- For **host 2**, each of its **VMs** will have access to a **single** virtual SCSI disk drive within the data store

Draw a **block diagram** to represent the required virtual system. In your diagram label the following items clearly: hosts, VMs, guest OS, clustered file system, virtual SCSI controllers, virtual disks, physical disk storage system.

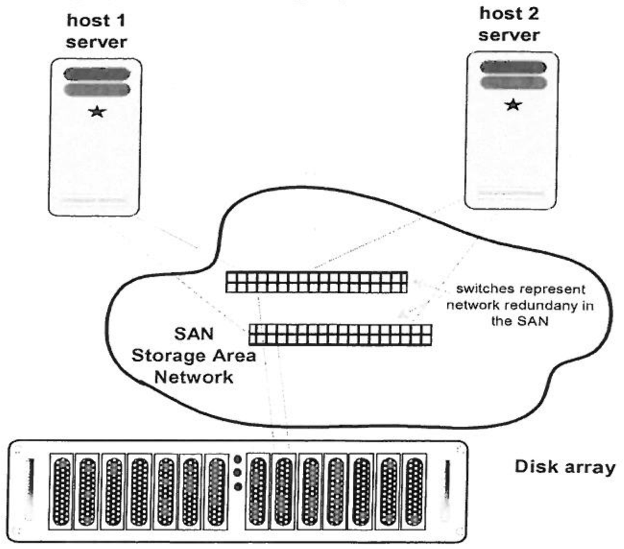

Figure Q2 Physical computer system

Q3

a) **15 marks (5, 5, 5)**

Answer the following in relation to the **UNIX/Linux** operating system:

- Draw a simple **block diagram of the Linux operating system** and clearly label each block.
- With the aid of a state diagram show the various **states for a process** in a multitasking system. Label clearly all state transitions.
- Draw a diagram for a **round robin** scheduler and state one advantage and one disadvantage for this type of scheduler.

b) **18 marks**

Write a **bash** shell script program to do the following:

- Display **how many processes** exist in the system
- List the **command** name and **PID** for the busiest process
- **Kill** the busiest process

Assume the output of a **ps –aux** command is like as follows:

```
USER      PID  %CPU %MEM  VSZ  RSS  TTY    STAT START  TIME   COMMAND
root      3321  0.0  0.0  1876  408 tty1   Ss+   2007  0:00   /sbin/mingetty tt
root      3340  0.0  0.0  2484  408 tty2   Ss+   2007  0:03   /sbin/mingetty tt
donal    17205  0.0  0.0  4420 1468 pts/2  Ss   08:31  0:25   -bash
joe      19168  0.0  0.0  2928  776 pts/2  R+   09:30  0:00   ps au
```

NOTE – In the addendum of this paper there is a list of common bash shell commands.

Q4

a) **15 marks (5, 5, 5)**

Answer the following in relation to operating system processes:

- Briefly state what is meant by a **thread** in the context of an operating system. Briefly summarise the key differences between a **process** and a **thread**, highlighting any advantages for threads.

- Briefly describe a **signal** in the context of the UNIX/Linux operating system.

- A UNIX/Linux operating system supports **named pipes** and **unnamed pipes** as interprocess communication mechanisms. Briefly describe each of these pipe types, highlighting the differences between them.

b) **18 marks**

Write a short **bash** script program that uses a **trap** to act on the SIGINT signal.

Your **main** program in the script can be any simple program that runs in a continuous loop.

The **trap** code is to be written as a **function**. When a SIGINT signal is received the trap's function will do the following operations:

- check the amount of disk space that is available on the local disk volume
- if there is more than **70% of the space** in use, then exit the script
- if there is NOT more than **70% of the space** in use, then the main loop continues

NOTE:
Assume the output format for the **df –h** command is as in this example:

```
Filesystem      Size  Used   Avail   Use%  Mounted on
/dev/sd3a       63G   13G    41G     22%        /
```

ADDENDUM: Commands

Quick Command Reference Chart

The bash shell commands and utilities – a brief summary card (8/Dec/15)

Command/Util	Brief description
awk	Scans a file(s) and performs an action on lines that match a condition. General format: *awk ' condition { action } ' filename* Example: *awk '/University/ {print $3,"\t", $11}' myFile*
bc	Arbitary precision calculator Example: *echo "scale=3; (1 + sqrt(5))/2" | bc* .... calculates phi to 3 places
cal	Display a calendar output
cat	Concatenate file to the standard output
cd	Change directory
chmod	Change file access permissions
chown	Change file owner/group
cp	Copy files and subdirectories
cut	Cut columns from a data file Example: *cut –c 49-59 logfile* ... extract column defined between characters 49 to 59
dd	Copy a file, converting and formatting Example: *dd if=/dev/zero of=myFile bs=1k count=10* ... makes myFile of 10 kiloBytes
date	Display current time, set date etc. Example: *date +%s%N* ...time with nanosecond resolution
df	Display disk space information
diff	Compare files line by line to find differences
du	Display disk usage information
echo	Display a line of text
exit	Exit the process e.g.: exit 0 ... exits with the code 0
find	Search for files Examples: *find / -type d –print* ...find directory files starting at root and display *find . –name "verse"* ...find all files, starting at the current directory, with "verse" string at start of name
grep	Scans text files looking for a string match. Examples: *grep "and" myFile* ... search for lines containing "and" *grep "^The" myFile* ... search for lines that begin with "The" *grep "floor$" myFile* ... search for lines that end with "floor"
head	Display a number of lines at the head of a file
history	Display previous commands
kill	Sends a signal Example: *kill –HUP 43165* ... send HUO signal to process 43165
less	Outputs a file to the console, a page at a time
ls	List directory(s) content ls –l long listing to show file details ls –R list subdirectories recursively ls –a list all files, including ones that start with *a* .
mkdir	Make directories
mkfifo	Make a named pipe Example: *mkfifo mypipe*
more	Outputs a file to the console, a page at a time
mv	Move files (effectively means to rename files)

ps	Show process status ps au show all processes, for all users
pwd	Print the name of the current working directory
read	Read user input
rm rm -R	Remove files and/or directories rm –r (or rm –R) will remove files recursively
rmdir	Remove directories (assuming directory is empty).
sed	A stream editor Example: *sed 's/Jack/Jill' filebook* … substitute the string 'Jill' for 'Jack' in file filebook
seq	Generates a sequence of numbers. Examples: seq 1 9 … generates numbers 1 to 9, line by line seq –s "-" 1 9 … default separator can be changed, using the –s option
set	If no options are used, set displays the names and values of all shell variables Examples: set …. shows all shell variables set | grep "USER" … shows shell variables with a specified string
sort	Sort lines in a text file sort –g general numeric sort sort –r reverse result of sort sort -k sort for a key position sort –n sort to string numerical value
tail	Display a number of lines at the end of a file
tee	Diverts a piped input to a second separate output Example: *cat demo_file1 | sort | tee demo_file1_sorted | more*
trap	Defines actions to take upon receipt of a signal or signals Example: *trap ' echo "This is my trap" ' SIGHUP* …. echo some text on receipt of HUP
uniq	Output a file's lines, discarding all but one successive identical lines
wc	Count number of lines, words, bytes etc. in a file wc –l count number of lines wc -c count number of bytes wc -m count number of characters
wait	Wait for child process to exit before finishing. e.g.: wait

Some common built-in shell variables

Variable	**Description**
$?	Exit status of the previous command
$$	Process ID for the shell process
$!	Process ID for the last background command
$0	Name of the shell or shell script
$PPID	Process ID for the parent process
$UID	User ID of the current process
$HOME	The home directory
$SHELL	The shell

Bash function example

```
# Example script program that uses two function parameters.
# The function calculates the product of the # two arguments:
# #! /bin/bash

# product is declared as a function and defined
product () {
(( product_var = $1 * $2 )) # global variable
}

# The main program

product 22 3 # The product function is called, with two arguments
echo "The answer is: $product_var"
exit
```

Bash array example

```
#! /bin/bash
my_array=("black" "brown" "red" "sea blue")
for colour in "${my_array[@]}"; do
   echo "$colour"
done
exit 0
```