Australian National University

Second Semester Examination 2016

Operating Systems Implementation

(COMP3300/COMP6330)

Writing Period: 3 hour duration
Study Period: 15 minutes duration
Permitted Materials: One A4 page with notes on both sides. Note also the standard lab tools are available including: gedit, kate, man, ...
NO calculator permitted.

Please Read The Following Instructions Carefully.

This exam will be marked out of 100 and consists of 4 questions. Questions are of unequal value. The value of each question is shown in square brackets. Questions that are partitioned into parts show the number of marks given to each part within square brackets.

Students should attempt all questions. Answers must be saved into the question's directory (Q1, Q2, Q3, Q4) using the file(s) described in the question statement. Marks may be lost for giving information that is irrelevant. Questions Q2iii and Q3iii are different for COMP3300 and COMP6330 students – only attempt the part relevant to your course.

Network traffic may be monitored for inappropriate communications between students, or attempts to gain access to the Internet.

The marking scheme will put a high value on clarity so, as a general guide, it is better to give fewer answers in a clear manner than to outline a greater number in a sketchy, half-answered fashion.

A VirtualBox disk image is available at: /scratch/LinuxCOMP3300ev.vdi (user account with sudo access is: comp3300 with password: comp3300)
Linux source code is available at: /usr/src/linux-source-4.4.0

DO NOT COPY THESE FILES INTO YOUR HOME DIRECTORY

Please remember that anything stored in /scratch will be lost at the completion of the exam. So edit your answers in the Q1, Q2, Q3 and Q4 directories in your home directory and just copy parts into either the VirtualBox machine or /scratch for testing your answer.

Question 1 [40 marks]

Highest marks are gained by providing: clear, concise, and short answers. Save your answers in the file 'Q1Answers.txt' in the directory Q1. This file can be edited using a text editor such as 'gedit' or 'kate'. Please make certain that this file is saved both as you progress through the exam and before the exam ends.

- i. [4 marks] What is an operating system?
- ii. [4 marks] What is the distinction between a short term CPU scheduler, mid-term scheduler and a long term scheduler? In what way do these schedulers effect the degree of multiprogramming?
- iii. [4 marks] In UNIX what do the 'fork' and 'execve' system calls do?
- iv. [4 marks] What is free space management for filesystems? List two common data structures used for free space management. What are some advantages/disadvantages for these two approaches?
- v. [4 marks] Generally an operating system will enable programs to change between different domains of protection. List some mechanisms that are often used to achieve this change between domains of protection. What approach is used by the “sudo” program in Linux.
- vi. [4 marks] Explain the “convoy effect”. Which CPU scheduler exhibits this problem?
- vii. [4 marks] In Linux what is the difference between the system calls “unlink” and “unlinkat”? What is their relationship within the Linux kernel source code?
- viii. [4 marks] Explain how a “lottery scheduler” works. What type of scheduler is a “lottery scheduler”? What are some advantages of such a scheduler?
- ix. [4 marks] What is Read Copy Update (RCU)? Explain how it works. In what situations would the use of RCU be most advantageous? When would it be least advantageous?
- x. [4 marks] What is the purpose of the “skbuff” within Linux's network kernel code. Which network layer(or layers) is the “skbuff” associated with. What are the advantages/disadvantages of the approach taken by Linux with respect to its network implementation?

Question 2 [20 marks]

Your answer for this question must be placed in 'counter.c'. To get you started the “hello_module.tar.gz” has been unpacked and renamed ‘counter.c’ within the directory.

(i) [5 marks] In Linux what are the parameters of the “write” system call? Find the location within the kernel where this system call is implemented (hint - the name of the function within the kernel is “sys_write”, noting the function signature is generated by a macro). In the second part of this question you will create a “write” file operation. The “write” system call will find its way down to your “write” file operation within the kernel. What functions are called along its path? How do the parameters change from the “write” system call to the “write” file operation? Give your answers as a comment in the 'counter.c' file.

(ii) [15 marks] Create a kernel module that adds a proc entry called “/proc/counter”. The proc entry counts the total number of bytes written to it. Note that the data written to this proc entry is not stored but just lost (like /dev/null). When you read this proc entry the total number of bytes written to this proc entry is reported.

So once the module has been inserted you could have the following interaction with the proc entry:

```
$ cat /proc/counter
0
$ echo "Hello!" > /proc/counter
7
$ echo "Bye" > /proc/counter
11
```

Put your solution in the file called 'counter.c' within the Q2 directory. Remember any testing you do within /scratch or within the VirtualBox image must be copied back to the Q2 directory.

Question 3 [20 marks]

A new file system has been implemented, called the “dfs”. It is based on the “vvsfs”. However, the approach used for storing the files has not been recorded within a comment. Also a key part of the “write” file operation has been deleted. Your task is to understand the file structure and also re-implement the deleted code. All your answers to this question must go within the “dfs_file_write” function (either as a comment or code) in “dfs.c” within the Q3 directory.

(i) [5 marks] Explain the file structure used within the “dfs”. What is the maximum file size (explain)? What is the maximum number of files that can be created (noting the ability to make a new directory has not been implemented for the file system – so in answering this question assume you only have the root directory to store files)? What would be the maximum number of files if the ability to create new directories was implemented? What are some disadvantages of such a file system? Put your answers within a comment in the “dfs_file_write” function in “dfs.c” within the Q3 directory.

(ii) [15 marks] Fix the “dfs_file_write” file operation by re-implementing the deleted code. Although this is the only part of the code you should change, you will need to read over other parts of the code to understand what to do.

Remember, any testing you do within scratch or within the VirtualBox image must be copied back to the Q3 directory. Note you should only modify code/add comments within the “dfs_file_write” function in “dfs.c” within the Q3 directory.

Question 4 [20 marks]

Save your answer for this question in a file called "Q4answers.txt" in the Q4 directory.

(i) [10 marks] Suppose that a disk has 30 tracks (to make adding up easy). The head is currently servicing a request at track 9. The previous request was serviced at track 8. The queue of outstanding request tracks is:

8,4,11,20,28,29

You may assume that no more requests arrive until all these have been serviced.
For the disk scheduling algorithms FCFS, SSTF, C-SCAN, answer the following questions:

- In what order are these requests serviced?
- What is the total number of tracks the head has moved?
- Do any of the above algorithms perform optimally in this case?
- What is unusual about how these different approaches compare?

(ii) [5 marks] Say a particular MMU has logical addresses that are 10 bits and frames that are 128 bytes in size. Given the following inverted page table (numbers are given in binary):

index	PID	Page
000	A	010
001	-	-
010	A	000
011	B	001
100	-	-
101	A	011
110	B	000
111	A	001

For **Process A** what are the physical addresses for the following logical addresses?

- 0011100111
- 0010000000
- 0101010101
- 1110000000

Given this approach what is the largest amount of memory a single process could be allocated, assuming it is the only process?

(iii) [5 marks]   From OSC Silberschatz Q9.5 page 331 "Suppose we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds.

Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds? "