

Surname:.....Forenames:.....

ID:.....

# THE UNIVERSITY OF AUCKLAND

---

SECOND SEMESTER, 2007

Campus: City

---

## COMPUTER SCIENCE & SOFTWARE ENGINEERING

### Operating Systems

(Time allowed: TWO hours)

**NOTE:**

Attempt ALL questions.

Answer the questions in the spaces provided.

Marks for each question are shown and total **100**.

For markers only:

<i>Question 1</i>	<i>/11</i>	<i>Question 6</i>	<i>/10</i>
<i>Question 2</i>	<i>/8</i>	<i>Question 7</i>	<i>/10</i>
<i>Question 3</i>	<i>/10</i>	<i>Question 8</i>	<i>/16</i>
<i>Question 4</i>	<i>/14</i>	<i>Question 9</i>	<i>/16</i>
<i>Question 5</i>	<i>/5</i>		
		<i>Total</i>	

1. System calls and kernels [11 marks]

(a) Briefly explain what a system call (or supervisor call) is and give an example.

A system call is a call to a privileged entry point in the operating system. Any call to a privileged service provided by the operating system would count, e.g. a file read call. Normally the instruction to invoke a system call forces control to jump to a specific address.

**CONTINUED**

Surname:.....Forenames:.....

ID:.....

(3 marks)

(b) How do system calls help to guarantee the safety of kernel data structures?

Because the system call instruction forces control to jump to a particular address within the kernel and changes the processor to kernel mode. As the kernel memory is protected the system call code cannot be modified by the user. Thus the kernel data structures are only accessed by kernel code which has only been entered through the correct entry points.

(3 marks)

(c) Early versions of UNIX did not allow preemption within the kernel. Explain why pre-emption wasn't allowed.

To simplify the kernel. If only one process can be "running" at a time, you don't have to worry about issues of concurrency. Hence all algorithms could assume that the currently running process was the only one modifying any data structure.

(3 marks)

(d) Why is the technique of not allowing preemption in the kernel not used in multiprocessing versions of UNIX?

It is unnecessarily restrictive. We don't want to hold up processes on other CPUs from making system calls just because one process is running in the kernel.

(2 marks)

2. Virtual machine monitors [8 marks]

The lecture notes state "Until 2006, all x86 type CPUs had problems with classical - trap and emulate virtualization".

(a) What is "trap and emulate virtualization"?

User mode processes are not permitted to carry out privileged instructions or have access to privileged areas of memory. As a virtual machine is running in user mode there are many operations it cannot perform. Trap and

**CONTINUED**

Surname:.....Forenames:.....

ID:.....

emulate virtualization is used to capture (trap) calls of privileged instructions, and then emulate the operation of the privileged instruction thus preserving the underlying host system state and devices from incorrect operation. The VM appears to be running on the real hardware as the emulation provides the same result.

(4 marks)

(b) Give one example of a problem that prevented those CPUs using trap and emulate.

Some instructions ran in both user and kernel modes (they just worked differently in kernel mode). So they were not privileged instructions and executing them did not cause a trap into the VMM.

Some instructions allowed the program to determine if it was running in privileged mode. The kernel of a guest OS should be privileged however if it checked it would see it was in user mode, breaking the fidelity requirement.

Even worse there are problems with protecting the page table information and keeping it all consistent.

(2 marks)

(c) Binary translation was one solution to these problems. Briefly explain what binary translation is.

- Code running in kernel mode is translated at run-time into something which doesn't have these problems.
- The translation is very simple (and hence efficient).
- Only translates code which is actually run.
- Much of the code is exactly the same as the original.
- The translated code is cached and reused.
- Uses all sorts of tricks to speed up emulation.

(2 marks)

Surname:.....Forenames:.....

ID:.....

## 3. Processes and Threads [10 marks]

## (a) Why are threads sometimes called "lightweight processes"?

A process includes the resources being accessed by the process, in particular memory. Threads in the same process use the same memory and other resources. Thus this information is not part of the thread, hence threads are lightweight processes. Also acceptable: easier to create, easier to switch between, hence a lightweight process.

(2 marks)

## (b) Describe two advantages user-level threads have over system-level threads.

Works even if the OS doesn't support threads.  
Easier to create - no system call.  
Control can be application specific.  
Easier to switch between - saves two processor mode changes.

(2 marks)

## (c) Describe two advantages system-level threads have over user-level threads.

If one thread blocks the other threads in the process can continue.  
Each thread can be scheduled separately.  
On a multiprocessor different threads can be scheduled on different processors.

(2 marks)

## (d) The following code opens a file for writing and creates a child process. Both parent and child processes then write to the file.

The `file.pos` method returns the current position in the file.

The `file.sync = true` method means that output to the file will be flushed immediately to the operating system.

```
file = open("temp", "w+")
#file.sync = true
file.write("before the fork")
if fork.nil? # in the child
  puts "child: #{file.pos}"
  file.write("after the fork in the child")
  puts "child: #{file.pos}"
else # in the parent
  puts "parent: #{file.pos}"
  file.write("after the fork in the parent")
  puts "parent: #{file.pos}"
end
```

Surname:.....Forenames:.....

ID:.....

The program produces the following output:

```
child: 15
child: 42
parent: 57
parent: 85
```

And the temp file contains:

```
before the forkafter the fork in the childbefore the forkafter the fork in
the parent
```

If the `file.sync = true` line is uncommented it produces:

```
child: 15
child: 42
parent: 42
parent: 70
```

And the temp file contains:

```
before the forkafter the fork in the childafter the fork in the parent
```

Explain why there is a difference in the output.

In the first case the call to `file.write("before the fork")` places the text in a Ruby buffer but doesn't send it to the operating system immediately. When the fork occurs the child process gets a copy of this buffer (and the parent keeps a copy). Both of these copies eventually get transferred to the file. In the second version the data is flushed from the Ruby process before the call to fork and hence only gets transferred to the file once.

(4 marks)

Surname:.....Forenames:.....

ID:.....

4. File system [14 marks]

- (a) Draw a diagram showing the relationships between a Unix program reading from a file, the process open file table, the system wide open file table, the inodes currently in memory and the on-disk inodes.

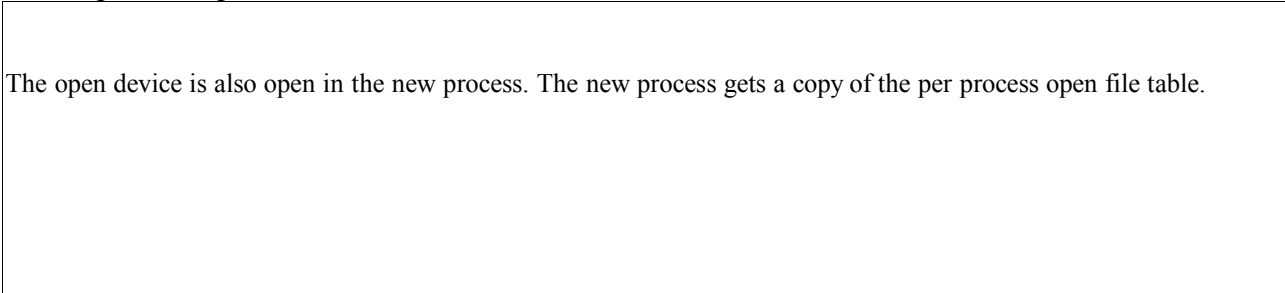
Book work



(4 marks)

- (b) In Unix all devices and open communication channels look like files.
  - (i) What happens to open devices when a process forks? Explain your answer by referring to the process open file table.

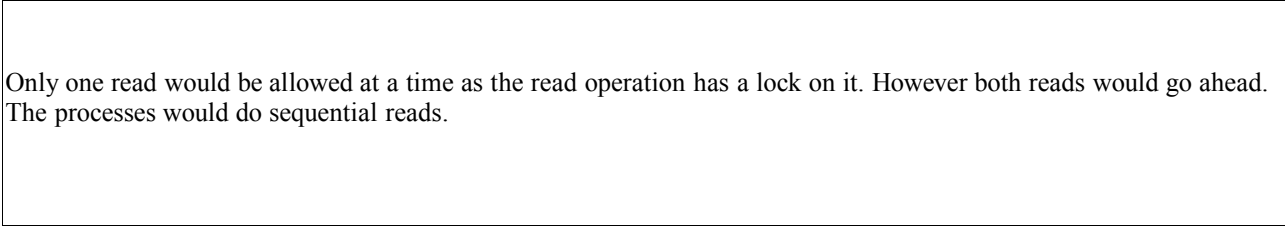
The open device is also open in the new process. The new process gets a copy of the per process open file table.



(3 marks)

- (ii) What would happen if both processes tried to read from the device at the same time?

Only one read would be allowed at a time as the read operation has a lock on it. However both reads would go ahead. The processes would do sequential reads.



(2 marks)

Surname:.....Forenames:.....

ID:.....

(c)(i) During a write to the disk, if the computer crashes before the on-disk inode of a file is updated to the current version of the inode in memory what goes wrong?

Data for the files on the disk and the corresponding inodes could be inconsistent. The on-disk inodes will be used on reconnecting the file system to access the files. This means it is possible that some data written to the disk will not be found as the inode doesn't point to it or include the updated file size. The extra blocks originally allocated will be unusable until a file system check utility is run.

(3 marks)

(ii) Describe a possible solution to this problem.

Journaling of the inode changes. This way on reconnecting the file system all changes can be read from secure storage and either redone or undone to keep the system consistent.

(2 marks)

5. Assignment 2 [5 marks]

a) What is a distributed memory caching system and why would it be used?

A distributed memory caching system uses memory in a number of servers or workstations to cache the results of calculations or database enquiries. Its purpose is to speed up the retrieval of the answers either by eliminating the recalculation or by avoiding the database lookup thereby alleviating the CPU or database load.

(3 marks)

b) What advantages does a distributed memory caching system have over one centralized machine with lots of free memory for caching results?

More memory can be used as we are using several servers. Results would be distributed over all of the servers, therefore there would not be the same bottleneck as on a single server.

(2 mark)

**CONTINUED**

Surname:.....Forenames:.....

ID:.....

## 6. Security 10 marks

(a) What are capabilities and how are they related to the access matrix?

A capability is a key which enables a subject to have access to an object. Each domain has a list of capabilities <object, access rights>. These lists are the rows of the access matrix.

(3 marks)

(b) Systems which use capabilities have a problem with revoking permissions. Explain what the problem is and describe one possible way in which a subject could have access rights removed.

A subject with a capability cannot be forced to give it up. The system may not even know which subjects have a specific capability. One solution is to expire the capabilities after a period of time. All subjects would then have to apply for the capabilities again.

(3 marks)

(c) AFS in the lab uses Kerberos for authentication. If a user gets a Ticket from the login authentication service for 10 hours explain how the user's access to a particular collection of files could be removed in less than 10 hours.

The particular Ticket for the service is provided by a ticket granting server. These tickets are for a shorter period of time. The administrator removes the user's access rights in the ticket granting server database. When the user needs to use the service again Kerberos would go back to the ticket granting server and be refused a new ticket. This happens behind the scenes. The user still only has to authenticate every 10 hours.

(4 marks)



Surname:.....Forenames:.....

ID:.....

## 7. Unix security 10 marks

- (a) A Unix process running with super-user or root privileges is not the same as a process running in kernel mode. Explain the differences.

Root privileges means the process can access all files/devices via the file system. It also allows certain system calls which are refused to non-root processes. Whereas running in kernel mode is a processor mode which allows the running code to perform any instruction including privileged instructions.

(4 marks)

- (b) What is a setuid program?

A program which runs with the privileges of the owner rather than the privileges of the person running the program.

(2 marks)

- (c) Calling `exec` in a setuid program can cause security problems. Explain how this can be a problem.

The `exec'd` program inherits the privileges of the calling process. In particular if the user can call any program they like from within the setuid program they can do anything with files owned by the owner of the setuid program. Particularly bad if the program is setuid root.

(2 marks)

- (d) How can the problem in part (c) be solved?

Revert to the user's privileges before invoking the new program.

(2 marks)

**CONTINUED**

Surname:.....Forenames:.....

ID:.....

8. Memory [16 marks]

- (a) How much memory would be used by a **full** page table on a system with a virtual address space of 64-bits and pages of size 16Kbytes ( $2^{14}$  bytes), and where each page table entry is 16 bytes long? Show your working.

$16K = 2^{14}$  so 14 bit offset. This leaves  $64 - 14 = 50$  bits to index the page table.  
 $2^{50} \times 16\text{bytes} = 2^{50} \times 2^4\text{bytes} = 2^{54}$  bytes or 16 petabytes (or pebibytes).

(3 marks)

- (b) When a page of memory is accessed but not found in physical memory, a frame must be found for it to be stored in. Different page replacement algorithms and different numbers of frames give different results.

Given this reference string of memory requests 1, 2, 3, 4, 1, 4, 2, 5, 1 complete the following tables showing the contents of memory, using the specified algorithms.

- (i) With the First In First Out selection algorithm and 3 frames.

<i>page request</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>1</i>	<i>4</i>	<i>2</i>	<i>5</i>	<i>1</i>
frame 0	1	1	1	4	4		4	5	
frame 1		2	2	2	1		1	1	
frame 2			3	3	3		2	2	

(2 marks)

- (ii) With the First In First Out selection algorithm and 4 frames.

<i>page request</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>1</i>	<i>4</i>	<i>2</i>	<i>5</i>	<i>1</i>
frame 0	1	1	1	1				5	5
frame 1		2	2	2				2	1
frame 2			3	3				3	3
frame 3				4				4	4

(2 marks)

- (iii) With the Least Recently Used selection algorithm and 3 frames.

<i>page request</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>1</i>	<i>4</i>	<i>2</i>	<i>5</i>	<i>1</i>
Frame 0	1	1	1	4	4		4	4	1
Frame 1		2	2	2	1		1	5	5
Frame 2			3	3	3		2	2	2

(2 marks)

Surname:.....Forenames:.....

ID:.....

(iv) With the Least Recently Used selection algorithm and 4 frames.

<i>page request</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>1</i>	<i>4</i>	<i>2</i>	<i>5</i>	<i>1</i>
Frame 0	1	1	1	1				1	
Frame 1		2	2	2				2	
Frame 2			3	3				5	
Frame 3				4				4	

(2 marks)

(c) Which of the four parts above had the smallest number of page replacements?

(iv)

(1 mark)

(d) Implementing the LRU algorithm is very expensive. We can approximate LRU using reference bits. Explain how this can be done.

Keep a byte to represent the accesses to each page of memory. Have a timer set to periodically move the reference bit into the most significant bit of the byte and shifting the other bits down. The reference bits are also cleared. The larger the number in the referenced byte the more recently (and frequently) the page has been used.

(4 marks)

Surname:.....Forenames:.....

ID:.....

## 9. Assignment 3 [16 marks]

The following code uses a correctly functioning `dfs.rb` file from Assignment 3.

```
require 'dfs'

dfs = DistributedFileSystem.create([10, 8, 17, 10])
puts dfs.blocks
file = dfs.open("A.txt")
p dfs.list
file.write(0, "start")
file.write(7, "end")
puts file.read(4, 4)
dfs.display_all
["B.txt", "C.txt", "D.txt", "E.txt"].each do |name|
  dfs.open(name).write(0, "abcde")
  p dfs.list
end
dfs.display_all
dfs.shutdown
```

- (a) Show the output produced by this program. (Remember that the `p` method prints arrays like this `"[item1, item2, item3, ...]"`.)

```
45
["A.txt"]
t_e
A.txt
=====
start end
["A.txt", "B.txt"]
["A.txt", "B.txt", "C.txt"]
["A.txt", "B.txt", "C.txt", "D.txt"]
["A.txt", "B.txt", "C.txt", "D.txt", "E.txt"]
A.txt
=====
start end
B.txt
=====
abcde
C.txt
=====
abcde
D.txt
=====
abcde
E.txt
=====
abcde
```

Surname:.....Forenames:.....

ID:.....

\_\_\_\_\_

(6 marks)

(b) What does the list of numbers passed as a parameter to the `DistributedFileSystem.create` method represent?

The sizes (in blocks) of the distributed disk drives which act as servers.

(2 marks)

(c) Would all files created by the program in part (a) be stored on one server? Explain your answer.

It depends. There are five files created by the program and 10 blocks on the first server. If opening a file uses 3 blocks then the only disk they could all fit in would be the third server. If less blocks are used when opening a file then it may be possible that all files would be on one server/disk. Most people distributed the files round-robin amongst the servers.

(2 marks)

(d) The `DistributedFileSystem` class in Assignment 3 could permanently store file data on a number of distributed disks. Describe one way in which a `DistributedFileSystem` object could find the data of a file when the file is opened.  
Your answer must mention the disk or disks that the file data is stored on.

Lots of different answers possible.

Surname:.....Forenames:.....

ID:.....

--

(6 marks)

---