VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

EXAMINATIONS — 2001

MID-YEAR

COMP 305

OPERATING SYSTEMS

Time Allowed: 3 Hours

Instructions: There are five questions.
Answer **ALL** questions.
Each question is worth 20 marks and should take you about 30 minutes.
Marks for individual parts of questions are shown below.
Paper foreign language dictionaries are allowed.
Electronic dictionaries and calculators are not allowed.

Question 1. Memory Management. [20 marks]

A new operating system and computer architecture have the following memory characteristics:

a 32 bit virtual address space,
4 segments in the virtual memory, and
a page frame size of 2k bytes.

(a) Describe the following characteristics of this system's virtual memory.

(i) [2 marks] Draw a picture showing the layout of a *virtual memory address.*

From left to right 2 bits for the segment, 19 bits for page within segment and 11 bits for address within page.

(ii) [2 marks] What is the maximum size of each segment in bytes?

*2**30 or one gigabyte.*

(iii) [2 marks] If each segment is represented by its own page table what is the maximum number of entries the page table might contain?

*2**19 or 512k*

(b) [4 marks] Show how a virtual address is translated to a physical address.

The leftmost two bits are used to select the page table for the segment. The next 19 bits index the page table. The entry in the page table is concatenated with the remaining 11 bits of the virtual address.

(c) [5 marks] What information should be held in, or easily accessible from, each page table entry?

presence bit
lock bit (e.g. for I/O)
physical address (if present)
disk address
replacement information

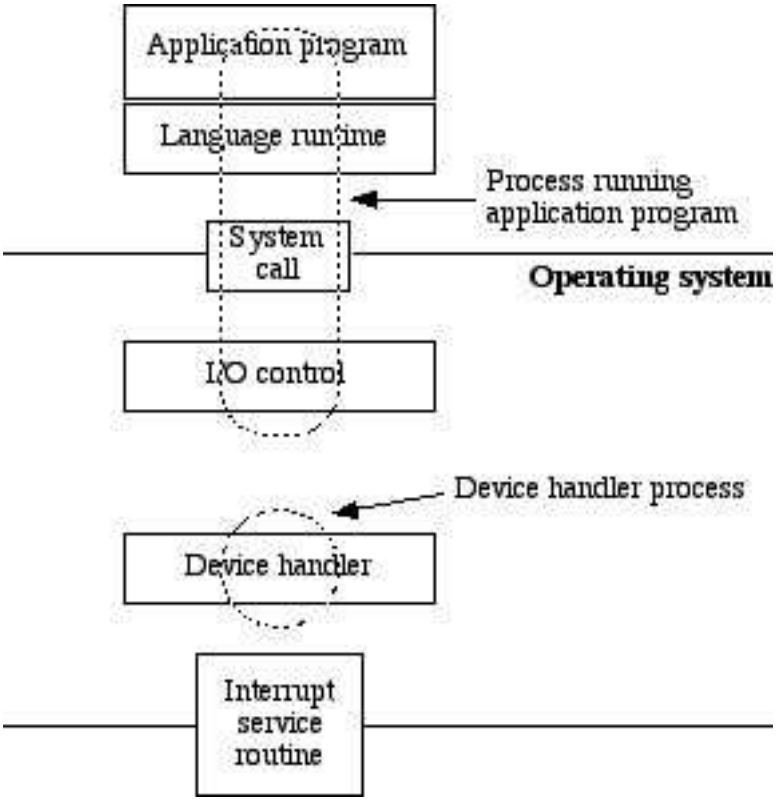
(d) [5 marks] The designers are arguing whether to place the operating system kernel in segment 0 leaving segments 1, 2 and 3 for application programs, or whether to devote all four segments to application programs. What are the advantages of each choice?

Placing the kernel in segment 0 means that the kernel is accessed on the same basis as the application programs making system calls much more efficient as no exception is required. All processes would see the same segment 0.

The advantage of not placing the kernel in segment 0 is the better protection of the kernel space. A user process is not able to fabricate a virtual address that addresses the kernel. There is also another segment for the user.

Question 2. Operating System Structure. [20 marks]

The figure below shows a typical operating system structure for handling input/output requests from application programs. Using this figure as a guide answer the following questions.



(a) [5 marks] The thread executing in the application program makes a *system call* to request an I/O operation. Explain the nature of a system call emphasising how it differs from a procedure or method invocation.

> *A system call changes protection domains from the program to the OS. This requires it to disable memory mapping. The application shares the OS with other programs/processes so we must be careful to handle the sharing correctly. Strict checking of parameters. A system call is usually accomplished using the exception mechanism.*

(b) The figure shows a seperate thread running in the device handler.

(i) [5 marks] Explain how the application thread's I/O request is actually accomplished.

> *The I/O control will request data from the device handler in units appropriate to the device, e.g. disk blocks or lines. These are held in a buffer between the two. A producer consumer structure is used to communicate requests and the data.*

(ii) [5 marks] What is the advantage of this structure compared with the application's thread executing the device handler code?

Using a seperate thread allows the device handler code to be written for a single thread removing the need for critical sections in code that is responding to interrupts. For some devices such as disks it also allows optimisation of the device requests.

(c) [5 marks] The application program is multi-threaded. What are the implications of the system providing kernel supported threads as opposed to threads provided by the programming language's run-time system?

If the threads are supported at the programming language level the kernel will see only a single process and when the thread waits in the I/O Control the entire process will be blocked. If the threads are supported by kernel threads other application threads will be able to run.

Question 3. Program Design. [20 marks]

Your first assignment in your new job is to evaluate the design of a process offering a network service. The design team has studied the problem and concluded that the average service request requires:

> 2 disk accesses at an average of 4 msec. each, and
> 4.5 msec. of processor time (including all overheads).

The team's current design is a single threaded server that accepts a request, services it sending the reply and then accepts a second request, etc.

(a) [2 marks] If the average network transit time from client to server is 100 msec. what is the expected latency for a client if the server is lightly loaded.

> *212.5 msec. = 100 + 12.5 + 100.*

(b) [3 marks] What is the maximum server throughput that you would expect the current design to achieve?

> *A top notch answer will recognise that the server waits for its response to be acknowledged thus each request will consume 212.5 msec. and the maximum will be about 4.5 requests per second. They haven't done 306 so we should expect 1000/12.5 = 80 requests per second.*

(c) [5 marks] Give a concise evaluation of this design.

> *This design will not scale because it does not support any concurrency amongst the requests. In particular the server is idle while the response is being sent and there is no concurrency between disk and cpu use.*

(d) [10 marks] Recommend how the design should be changed by explaining and justifying the change or changes to be made and indicating the theoretical increase in server throughput that you would expect from your changes.

> *The first change is multi-threading which will allow new requests to be handled while previous requests are in progress. The big win here is allowing new requests to be served while waiting for acknowledgements of previous requests. The disk is the bottleneck (8 msec/request) so the theoretical limit is 125 requests per second.*
>
> *Next try caching you disk information to eliminate some of the disk accesses. A 50% percent hit rate would would make the cpu the bottle neck and we would peak at about 200 requests per second.*

Question 4. Synchronisation. [20 marks]

(a) [8 marks] Producer and consumer processes interact via an N-slot cyclic buffer. Semaphores are defined and initialised as follows:

Semaphore lock = new Semaphore(1);
Semaphore spaces = new Semaphore(N);
Semaphore items = new Semaphore(0).

For the following program segments indicate where mutual exclusion and condition synchronisation are being attempted and explain how the system might fail.

producer code	consumer code
produce item	lock.P()
lock.P()	items.P()
spaces.P()	*remove item*
insert item	spaces.V()
items.V()	lock.V()
lock.V()	*consume item*

The lock semaphore is being used to obtain mutual exclusion between lock.P() and lock.V(). The spaces and items semaphores are being used for condition synchronisation. This fails because it waits on the condition synchronisation while holding mutual exclusion, i.e. spaces.P() and items.P() while lock is held. This will lead to deadlock.

(b) [12 marks] Write a monitor to manage the N-slot buffer. Discuss why the problems you pointed out in the previous part do not arise in the monitor implementation.

```
PC Monitor begin {
    int items = 0;
    int spaces = N;

    procedure void put(item) {
        while (spaces == 0) wait();
        items++;
        insert item
        spaces–;
        notify();
    }

    procedure item get() {
        while (items == 0) wait();
        spaces++;
        items–;
        remove item
        notify();
    }
} end Monitor
```

Students might use condition variables, but they are used to java so expect a solution without.

The problem encountered with the semaphore solution is avoided as the wait on the cv releases the mutual exclusion. This is handled by the run-time support so it minimises the opportunities for errors by the programmer.

Question 5. File Systems.

File systems are made up of a directory service and a storage service.

(a) [5 marks] A graph is a more convenient naming structure than a tree because it facilitates sharing. Define the two types of links that are commonly used to construct graph structures and give the advantages and disadvantages of each.

> *A hard link is the fundamental type of link. Subsequent links are indistinguishable from the original. The advantage is the transparency and speed of access. The disadvantage is the possibility of cycles in the naming structure.*
>
> *A symbolic link places the path of the file in the secondary entry. Since a symbolic link can be distinguished from a hard link cycles are avoided. The down side is the slower access and the need for correct permissions on the full path to the file.*

(b) [5 marks] What information is held in a file's meta-data?

> *Typically this would include the owner of the file, any security information (groupr, world access, acl), various attributes such as size, last modified, accessed times, created time and of course the location of the data itself.*

(c) [5 marks] In a typical file system structure each process has its own process open file table. Each entry in the process open file table points to an entry in a single *system open file table.*
What are the major responsibilities of the system open file table?

> *The system open file table is responsible for sharing access to files amongst those processes that have it open. This includes RW control, file locking, buffer management, fail safe writing, and access control.*

(d) [5 marks] A proposed design of a network file server is to provide the interface of a basic file system over the network. The interface would have two idempotent procedures:

```
DiskBlock b = get(DiskIndex i); // read a DiskBlock from the server
put(DiskBlock b, DiskIndex i);  // write a DiskBlock to the server
```

What are the advantages and disadvantages of this design for a file server?

> *The advantages are small. It is a simple interface and being idempotent is useful. The disadvantages are major. The system open file tables in clients can no longer control sharing of the files as they will hold it in seperate buffers and will not be able to run their RW protocol.*

****