TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA

UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2006

MID-YEAR

COMP 305
Operating Systems

Time Allowed: 3 Hours

Instructions: There are six questions.

Each question is worth 25 marks and should take you about 25 minutes.

Paper foreign language dictionaries are permitted.

Non-programmable calculators without full alphabetic keys are permitted.

Electronic dictionaries and programmable calculators are not permitted.

continued...

Question 1. Synchronization and Deadlock [25 marks]

(a) [5 marks] What are the two main types of waiting found in an operating system? How do they differ in terms of efficiency?

Busy waiting means that a process is waiting for a condition to be satisfied in a tight loop without relinquish the processor. This means that the CPU is doing work for no purpose. Alternatively, a process could wait by relinquishing the processor, and block on a condition and wait to be awakened at some appropriate time in the future. Busy waiting can be avoided but incurs the overhead associated with putting a process to sleep and having to wake it up when the appropriate program state is reached.

(b) [5 marks] Disabling interrupts frequently could affect the operating system clock. Explain why it could and how such effects could be minimised.

The system clock is updated at every clock interrupt. If interrupts were disabled particularly for a long period of time it is possible the system clock could easily lose the correct time. The system clock is also used for scheduling purposes. For example, the time quantum for a process is expressed as a number of clock ticks. At every clock interrupt, the scheduler determines if the time quantum for the currently running process has expired. If clock interrupts were disabled, the scheduler could not accurately assign time quantums. This effect can be minimized by disabling clock interrupts for only very short periods.

(c) [10 marks] A multiuser operating system can be designed to limit the number of concurrent users to a maximum number. For example, as soon as N users are logged into the system, the operating system could block further logins until an existing user logs out. Show how semaphores can be used to implement this scheme.

A semaphore is initialized to the number of allowable concurrent users, say N. When a user attempts to login, the acquire() method is called, when a user logs out, the release() method is called. If the system reaches the number of allowable concurrent users, subsequent calls to acquire() will block until an existing user logouts the release method is invoked. This is because the sempahore will be equal to zero and so any subsequent call will result in the calling thread being put to sleep. Assuming Nachos style semaphores where the thread to wake is the earliest thread put asleep then logins will proceed in the order of arrival at the system. As scheme involving condition variables is a little overcomplicated and will receive fewer marks.

(d) [5 marks] In a real computer system, resources may be added or replaced at different points in the system's lifecycle. Apply the principles of the banker's algorithm to determine if adding or removing a resource will leave a system in a safe state.

Adding new resources - increases availability, keepts it in a safe state before then there were enough resources to satisfy all claims given a particular sequence of process completions. Removing resources may mean it is not in a safe state but it depends upon the *particular maximum claims made by the processes. Distinguish from a simple discussion of deadlock, different concept.*

Question 2. CPU Scheduling [25 marks]

(a) [5 marks] Describe why the cost of a thread context switch is lower than a process context switch.

Less state to save and restore. PCB includes things like file tables and requires the TLB and other architecture specific data to flushed or saved whereas thread only requires registers (including stack pointer) to be saved.

(b) [5 marks] Identify the main points in the lifecycle of a thread where CPU scheduling decisions may take place.

1. Switches from running state to waiting state (result of IO request or invocation of wait); 2. Switches from running state to the ready state (interrupt occurs); 3. Switches from waiting state to ready state (completion of IO); 4. Terminates 1. Describe the differences among short-term, medium-term, and long-term scheduling (3.1).

(c) Consider the following set of jobs, with the length of the CPU-burst time given in milliseconds:

Job	Arrival Time	Burst Time	Priority
1	0	10	3
2	4	1	1
3	6	2	3
4	9	1	2

(i) [5 marks] Draw a Gantt chart illustrating the execution of these jobs using preemptive SJF and a nonpreemptive priority scheduling (note that a smaller priority number implies a higher priority).

Expect two diagrams. Main difference is process one runs to completion under nonpremptive priority whereas SJF delays its finish right until the end. . Diagram (1 mark - its a Gantt chart; 1.5 mark - right except for got two jobs in wrong order; 2.5 marks - correct).

(ii) [5 marks] Calculate the turnaround time and waiting time for each job for each of the scheduling algorithms in part (i).

This answer should reflect the gannt chart drawn above (unless the gannt chart is trivial).

SJF: 9,0,2,1,4 and Priority: 6,0,16,18,1 for waiting time. turnaround is 19,1,4,2,9 and 16,1,18,19,6 respectively.

(d) [5 marks] Explain the main drawback of priority scheduling and outline an algorithm that would address this drawback.

Starvation is a problem, low priority threads may never execute. Extend by everytime that a thread is chosen, increment the priority of the remainig threads.

Question 3. Memory Management [25 marks]

(a) [5 marks] Describe the relationship between a logical address and a physical address.

Logical address runs from 0 to the end of the program, logical addresses used within instructions. Physical address runs from 0 to the end of physical memory, same logical address may map to multiple physical addresses.

(b) [5 marks] What is the cause of thrashing and how can an operating system detect thrashing?

Thrashing is caused by underallocation of the minimum number of pages required by a process (1 mark), forcing it to continuously page fault (1 mark). The system can detect thrashing by evaluating the level of CPU utilization as compared to the level of multiprogramming (2 marks).

(c) [8 marks] Consider a process consisting of seven pages a, b, c, d, e, f, g. A working set algorithm is used to calculate the process' working set. The working-set window is 3 and we assume that the working set is initially empty. Given the sequence of memory references shown below, show how the working set changes over time. The reference string is:

hghffehgfe

The following table shows the working set over time.

1	*h*
2	*h g*
3	*h g*
4	*h g f*
5	*h f*
6	*f e*
7	*h e f*
8	*e h g*
9	*h g f*
10	*f g e*

Have they: (1) calculated working set including time 1-3 while program is loaded into memory (2 marks); (2) realised that it is the active pages within the window that stay in the working set and others are ejected (5, 6, 8, 9, 10) (2 marks); (3) realised when new pages come into the working set (7,9,10) (2 marks); (4) ended up with the correct final set (2 marks) or made a reasonable but flawed attempt.

(d) [7 marks] A page-replacement algorithm should minimize the number of page faults. We can do this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. Define a page-replacement algorithm that follows this scheme. Specifically address the problem of how the page to be replaced is selected and the mechanism for tracking frame use.

- *Initial value of the counters is zero.*
- *Counters are increased, whenever a new page is associated with that frame.*
- *Counters are decreased, whenever one of the pages associated with that frame is no longer required.*
- *How the page to be replaced is selected is find a frame with the smallest counter. Use FIFO for breaking ties.*

Marking based upon how closely approximate this scheme and simplicity of description.

Question 4. I/O Subsystems [25 marks]

(a) [5 marks] Briefly describe the role that a device driver plays with respect to the IO subsystem.

Device drivers present a uniform interface to the IO subsystem. Each piece of hardware has a specific device driver written for it that hide the implementation details (how we actually drive hardware to talk to the device) from the IO subsystem. It permits new hardware devices to be added without having to change the kernel.

(b) [5 marks] Contrast blocking with non-blocking system calls. Why do most operating systems use blocking system calls?

Blocking system calls suspend the process until the IO completes whereas non-blocking system calls returns immediately with whatever is available. Blocking application code is easier for programmers to use and understand whereas non-blocking requires a more complicated polling style of programming.

(c) Assume a disk drive has 3,000 cylinders numbered 0 to 2,999. The drive is currently serving a request at cylinder 143 and the previous request was at cylinder 153. The queue of pending requests in order of arrival is:

$$86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130$$

For each disk scheduling algorithm specified below, compute the seek schedule and identify the last cylinder accessed. **Always start from the current head position.**

(i) [5 marks] the LOOK disk-scheduling algorithm, and

(ii) [5 marks] the SSTF disk-scheduling algorithm.

One mark for getting the order right i.e. that moving towards smaller numbers first, one mark for not going to the edge of the disk, one mark for recognising that the head reads in the direction it is travelling and then returns and reads on the way back. Or full marks for getting the entire schedule right: first part: 130, 86. Second part: 913, 948, 1022, 1470, 1509, 1750, 1774.

(d) [5 marks] Discuss whether it is better for the operating system or the disk controller to take responsibility for disk scheduling.

Operating system: can impose scheduling on more than just optimisation of seek to meet higher level requirements such as reliability. Disk controller: can implement scheduling algorithms that require knowledge of the current position of the head. Most systems go for a mix of both

Question 5. File Systems [25 marks]

(a) [5 marks] Define the term *file* and briefly identify a file's main attributes.

A file is a logical storage unit. It usually has a name, identifier, type, location, size, protection, time, date and user identification. A good answer should at least identify name/type/location/size as important attributes.

(b) [5 marks] Briefly describe three operations typically performed on a directory.

Search for a file. *Search for a directory entry by file name.*

Create a file. *Create a new file and add entry to directory.*

Delete a file. *When a file is no longer needed, remove it from the directory.*

List a directory. *List the files in a directory and contents of directory entries.*

Rename a file. *Change file name, update the directory.*

Traverse the file system. *Access every directory and very file within a directory structure*

(c) Consider the contiguous, linked and indexed (single-level) allocation strategies. Assume the following conditions hold for each problem posed below.

- The file consists of five data blocks numbered zero to four.
- Both the file control block and the block information to be stored are already in memory.
- In the case of indexed allocation, the index block is stored on the disk.
- In the contiguous-allocation case, assume that there is empty space available at the end of the file.

(i) [5 marks] For each strategy, sketch the on-disk structure of the file blocks.

Contiguous: five contiguous blocks (1 mark). Linked: five blocks, pointers connecting them (2 marks). Indexed: six blocks on disk including the index block, pointers from index block to other blocks (2 marks).

(ii) [5 marks] For each strategy, calculate the number of disk I/O operations required to insert a block before block four (which is currently the last block in the file).

Contigious: read block four, write block four, write new block (3), Linked: find block 3 (3 reads), update block 3 to point to block 4, write new block = 5, Indexed: 1 (read from disk) and update in memory

(d) [5 marks] Discuss whether the 4.2 BSD Unix File System disk allocation policy would suit a database system characterised by a small number of large files and a predominantly random access pattern.

> *The UFS tries to locate inodes and data blocks in same cylinder for quick access, large files have their data blocks and index blocks split across nearby cylinders. Because these are large files there will be a lot of splitting of the data blocks over nearby cylinders and the indirect blocks asosciated with the files will also be split. This will lead to poor performance, the UFS scheme favours systems with lots of small files and generally sequential access.*

Question 6. Security [25 marks]

(a) [5 marks] Define the term *principle of least privilege* and give an example of its application.

> *Programs, users and systems should be given just enough privileges to perform their tasks. This means that the damage done by a compromised program is limited. For example, a trojan horse is limited in the damage it can do to those rights possessed by your user account.*

(b) [5 marks] Describe and compare the structure of a capability list with an access control list.

> *An access list is a list for each object consisting of the domains with a nonempty set of access rights for that object whereas capability list is a list of objects and operations allowed on those objects for each domain. The access list and capability list are essentially inverses of each other.*

(c) [5 marks] A capability list is usually kept within the address space of the user. How does the system ensure that the user cannot modify the contents of the list?

> *A capability list is considered a protected object and can only be access indirectly by the user. The operating system ensures the user cannot access the capability list directly. Another approach is to use encryption, the capability must be signed by a third-party.*

(d) [10 marks] You are working in a team with two other programmers called Robert and Sophie. You have been asked to design a one-time password authentication for a banking system. There are two possible designs: (i) one-time pad; and, (ii) algorithmic passwords. Briefly describe both schemes, and analyse the strengths and weaknesses of each design.

> *Five marks for discussion of each. Algorithmic has longer description whereas one-time password has more disadvantages.*

Algorithmic password. System selects a random integer and presents it to the user. The user applies the function plus a shared secret and replies with the correct result. The system compares the result with its own calculation that uses the shared secret and the random integer.

Advantages: More secure than a one-time pad.

Disadvantages: Reasonably complicated, need to distribute function as well as shared secret.

One-time pad. List of single use passwords. Each password is used just once.

Advantages: Simple to implement.

Disadvantages: Security depends on difficulty of guessing the next password based upon observation of previous password. Requires true randomness and this is hard to use in practice. The one-time pad also may be quite large, this could make protecting it difficult.

****