

THE UNIVERSITY OF HONG KONG

FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS

CSIS0230A Principles of Operating Systems

Date: January 7, 2002

Time: 9:30 am–12:30 pm

Candidates may use any calculator which fulfils the following criteria: (a) it should be self-contained, silent, battery-operated and pocket-sized; (b) it should have numeral-display facilities only and should be used only for the purpose of calculation; (c) it should not have any printing device, alphanumeric keyboard, or graphic display; and (d) it should not contain any recorded data or program. It is the candidate's responsibility to ensure that the calculator operates satisfactorily and the candidate must record the name and type of the calculator on the front page of the examination scripts. Lists of permitted/prohibited calculators will not be made available to candidates for reference, and the onus will be on the candidate to ensure that the calculator used will not be in violation of the criteria listed above.

Answer all questions in the answer book provided.

1. (*OS Interface, 15%*) Most computers have an **alarm** device, which can be configured to send an interrupt to the CPU after some number of mini-seconds have elapsed. In a multi-user, multi-programmed operating system, the device has to serve all programs and users.
 - a. Write a plausible interface for user programs to use the device, e.g., to wait for some time defined by the user.
 - b. What parts of the operating system need code to deal with this device and support this interface?
 - c. For each of the operating system parts you answered in (b), write pseudo-code for the section that handles the device.
2. (*Multi-threading, 10%*) Many operating systems support multi-threading, meaning more than one flow of control can be occurring in a **single address space** at the same time.
 - a. The mappings between processes and threads are different between Linux and Sun. Explain their differences.
 - b. Give one application that is easy using the implementation of Linux but impossible in that of Sun, and vice-versa.
3. (*Virtual memory, 15%*) One of the techniques used in the virtual memory system is called **copy on write**. When the OS needs to copy a memory region, it does not copy the memory, but instead modify the page table.
 - a. Give three different situations in which this VM mechanism can be used. Specify what needs to be copied, and when is the copying (i.e., writing) actually done.
 - b. Consider the C++ language construct `string`. Explain why the VM mechanism cannot be used when a string is copied to another.
 - c. Is it possible to use copy-on-write on C++ `string`? If so, how? If not, why?

4. (*Scheduling, 15%*) In Linux, process scheduling is controlled by a per-process counter called the **remaining time quantum**. When a process calls *fork()*, the remaining time quantum is split into halves, one half is given to the newly created process.
- Consider the following alternative: *the remaining time quantum is copied to the newly created process*. This is not a good strategy, and can lead to system crash. How to make a system crash if this is the strategy taken by Linux? Write code to support your claim.
 - Consider the following alternative: *the remaining time quantum of the newly created process is always zero, while the time quantum of the old process is unchanged*. Why this is not a good strategy? Which measure(s) of scheduling efficiency will be affected? Support your answers with a sequence of system calls.
 - Why such concerns are not addressed in the Linux *Round-Robin* and *First-In-First-Out* real-time processes?
5. (*Process synchronization, 15%*) Use **monitor** to solve the following synchronization problem: In a busy mini-bus station, passengers queue up in front of the station. When a minibus comes, it queues in the minibus line until it is the first in it, waits until 16 passengers are on the minibus, and leaves. When a passenger comes, he waits in the queue. When a passenger is in the front of a queue, he waits until a minibus comes and gets up into the minibus.

Implement the system so that the processes are the minibuses and the passengers. **Hints:** In the monitor, records the number of passengers waiting in the queue, the number of passengers already in the bus, and the number of bus already in the station. Have three conditions, which are signalled when a seat becomes available while somebody is in queue, when a bus becomes full, and when a bus leaves the station. Make sure that a waiting passenger or minibus will leave the monitor temporarily (or deadlock will occur).

6. (*Deadlocks, 10%*) Consider a situation in which three processes, named 1, 2 and 3, require three different number of resources A, B and C. There are 7 copies of A, 5 copies of B and 3 copies of C in the system, and no other process contends for the resources.

The processes declares their maximum use as follows:

Process	Maximum requirements
1	6 A, 2 B, 1 C
2	3 A, 5 B, 2 C
3	5 A, 3 B, 3 C

They perform it in the following order:

Process	Operation
1	Request 5 copies of A
2	Request 3 copies of B
2	Request 3 copies of A
3	Request 1 copy of C
3	Request 2 copies of C
1	Request 1 copy of C
1	Free all resources (remember to recheck delayed operations!)
3	Request 2 copies of B
1	Request 2 copies of B

- a. Why it is not good to prevent deadlock in such scenario by attacking the circular wait requirement?
 - b. Suppose deadlock avoidance is used instead to maintain the system in states that deadlock is not possible. What is the sequence in which the operations are performed? For each operation performed, show a possible schedule that guarantees to complete all jobs. For each operation delayed, show the conflicting processes.
7. (*Filesystems, 10%*) In a Unix filesystem, disk data blocks associated with a file can be efficiently located by using an **index-node (inode)**.
- a. For the ext2 filesystem of Linux, write an algorithm which can be used for finding the location of the data block given an offset to a file and an inode. Illustrate your algorithm with the following case: block size = 1024, length of block numbers = 32 bits, number of direct blocks = 12, and file offset (in bytes) = 123456789.
 - b. Explain why it is impossible to efficiently find the name of a file given an inode (and nothing else). Name one in-memory kernel data structure that is more appropriate for that purpose.
8. (*User protection, 10%*) Consider a world readable and executable **SUID root program** which does the following at the beginning of *main()*:

```
system("helper");
```

The C library implements *system()* by invoking the `/bin/sh` shell program.

- a. Why this program is not secure in an environment in which multiple untrusted parties can login the machine? How an intruder gains access to the system?
- b. Suppose the helper really needs to be executed at the beginning of the program, but actually needs no special privilege. On the other hand, some code after the call would require the privilege of the `root` user. How you would modify the above code to make it secure?

END OF PAPER