Computer Systems

Unassessed Course Work

Department of Computing, Imperial College London

Autumn 2013

Multi-programming

1 Give a brief explanation of multi-programming (or multi-processing).

2 Why do some Operating Systems provide multi-programming?

3 Give a suitable argument for the provision of multi-programming in a single user computing system.

4 Explain why multi-programming systems require:
   a) Hardware interrupts from `I/O` devices
   b) Independent Direct Memory Access channel

5 Explain the term 'multi-user system'. Why do such systems implement time-slicing?

6 Describe the following systems and discuss the implications for the operating system on such systems: *Parallel systems*, *Distributed systems*, and *Real-time systems*.

Synchronisation

7 What is a critical region?

8 In the context of processes that run concurrently, when protecting access to critical regions, the following instruction can be used: `tsl reg,lock`. What are '`tsl`' and '`lock`' here? Give the implementation of protection of critical regions, i.e. give assembler specifications of both '`enter_cr`' and '`leave_cr`', using `lock`s.

9 A computer has an indivisible instruction `swap` which exchanges the contents of two store locations. Show that the following program gives mutual exclusion between `n` processes. Comment on the efficiency of this solution.

```
boolean free = TRUE;
boolean turn[n]; for (i=0;i<n;i++) turn[i] = FALSE;

process Process_i () {
  for (;;) {
    while (!turn[i]) {
      swap(free,turn[i]);
    }
    inside critical region
    swap(free,turn[i]);
    outside critical region
} }
```

10 Explain in brief terms why the *server paradigm* solves the problem of mutual exclusion.

11 A new operating system has only two processes `P_0`, `P_1` which have process ids 0 and

1. Each process has a local variable **pid** which contains its process id. Assume further that the operating system provides the following implementation of locks:

```
type LockStatus:  enum (open, closed)
var LockArray:  array 0..1 of LockStatus

procedure lock()
  if (LockArray[(pid + 1) mod 2] = open) then
    LockArray[pid] = closed
  else
    LockArray[pid] = closed
    loop
      exit when LockArray[(pid + 1) mod 2] = open
    end loop
  end if
end lock


procedure unlock()
  LockArray[pid] = open
end unlock
```

The procedures **lock** and **unlock** are divisible.

a) Can this implementation of **lock** and **unlock** be used to implement mutual exclusion? Explain your answer with examples if necessary.

b) Can this implementation of **lock** and **unlock** lead to deadlocks? Explain your answer with examples if necessary.

12 Assume that in a museum the access to an exhibition is controlled by a computer. Visitors enter the exhibition through a turnstile and leave the exhibition through another turnstile. The entrance turnstile is controlled by a process **Enter** and the exit turnstile is controlled by a process **Exit**.

a) Describe how both processes can keep track of the total number of visitors currently inside the exhibition using a shared variable **numberOfVisitors**. Use a semaphore instead of a lock to ensure mutual exclusion and show how the semaphore must be initialised.

b) Modify both processes in such a way that the total number of visitors currently inside the exhibition never exceeds 100. Again use semaphores to solve the problem and show how the semaphores must be initialised. Discuss whether deadlocks or starvations can occur in your solution.

13 Suppose the next generation of semaphores supports a new operation **query(s)** which would allow processes to query whether the operation **wait(s)** would block or not. Show how the operation **query(s)** may be implemented and discuss whether the calling process could use this to avoid blocking.

14 What is the difference between a binary semaphore and a general semaphore?

15 Describe an implementation for semaphores, i.e. give its structure together with queue, and describe in pseudo code the operations **down(s)** and **up(s)**.

16 Describe a suitable data structure for a general semaphore and outline the following operations in terms of the operations **down(s)** and **up(s)** on a binary semaphore **s**.

(**init(value,gs)**): initialises the general semaphore **gs** to **value**.
the **down** operation on a general semaphore **gs**.
the **up** operation on a general semaphore **gs**.

17 Give a solution for the Readers and Writers problem using semaphores; this problem

arises in the context of allowing access to shared database, where multiple readers are permitted, but only 1 writer.

Interrupts

18 How would the fetch and execute cycle of a processor look if the processor allows the enabling and disabling of interrupts using special instructions? Assume that the processor has a flag **`EF`** which is part of the processor status word and which indicates whether interrupts are enabled or not. Write down the solution as a flow diagram and as an algorithm (in pseudo-code).

19 Why are timer interrupts an essential requirement for a multitasking operating systems?

20 Why are system calls implemented via interrupts?

21 Explain how mutual exclusion for access to a shared data structure by a number of processes running on a single processor can be obtained by locking out interrupts.

22 Describe how it is possible on the Intel Pentium processor to implement a switching between *user-mode* and *system-mode* (with the use of interrupts).

Processes

23 Explain how the scheduler works, how its gets activated, and what activates it. Give a diagram that describes the Round Robin Scheduling Algorithm.

24 Describe the 3 main states of a process in an operating system and the additional process states needed to support swapping of processes between main memory to backing store. This may occur when the system is short of main memory to run all the user processes.

Give a state diagram to show the main state changes and what actions typically cause state changes to take place.

25 What is the main rôle of the scheduler? Give the situations in which it will be invoked in an operating system that uses a message passing mechanism for communication between processes, and a round-robin scheduling algorithm.

26 Using the system calls **`fork`** and **`exceve`** write down the pseudo-code for an algorithm which will start the program **`/usr/local/bin/netscape`** as new child process.

27 Using the system call **`fork`**, give the pseudo-code for an algorithm which spawns three new child processes. Which of the child processes is the oldest?

28 What are the essential differences between *threads* and *processes*.

29 Consider the following four processes which arrive at $t = 0$ in the order **`P_1`**, P2, P3, P4 and whose execution time is given in seconds:

Process ID	Execution time
1	6
2	2
3	9
4	3

Draw a diagram showing the execution of the processes and calculate the average turnaround and waiting time for

a) shortest job first scheduling

b) round-robin scheduling (with time slice $\Delta t = 3$s)

30 Which characteristics (preemptive/non-preemptive) should scheduling algorithms in a multi-user operating system have and why?

31 Consider the following four processes `P_1`, P2, P3, P4 whose execution and arrival time is given in seconds:

Process ID	Arrival time	Execution time	Priority
1	0	6	9
2	1	7	5
3	2	4	8
4	3	3	3

Draw a diagram showing the execution of the processes and calculate the average turnaround and waiting time for

a) *preemptive priority scheduling (processes are preempted if a process with higher priority arrives)*

b) *non-preemptive priority scheduling*

Remember that high numerical values mean low priority.

32 A process has a fixed execution time. The process runs on an operating system which uses a round robin scheduling with a fixed time slice. After the designer of the operating system reduces the time slice, the turnaround time for the process reduces significantly. How can this happen?

33 A specific operating system requires 1 ms to handle interrupts, 2 ms to schedule the next process and 1 ms to dispatch the next process. Assume that the clock chip produces interrupts with a frequency of 50 Hz.

a) Approximate the percentage of processor time the operating system will spend as latency time.

b) Approximate the percentage of processor time the operating system will spend as latency time if the time for interrupt handling, scheduling and dispatching doubles but the clock chip produces interrupts with a frequency of 5 Hz.

34 Round robin schedulers normally maintain a list of all runnable processes, with each process occurring exactly once in the list. What would happen if a process occurred twice in the list? Can you think of any reason for allowing this?

35 During execution, an OS maintains a variable `proc_ptr` that point to the process table entry for the current process. Why?

36 How does scheduling for real-time systems differ from scheduling for multi-access systems?

Messages

37 Why is it important to be able to specify a limit on the time that one is prepared to wait for a message?

38 The main loop `floppy_task()` in a floppy disk driver in a message-passing OS contains the call `receive(ANY,&mess)`, where messages sent to the driver are caught. However, in other positions in the driver (notably `start_motor()`, `seek()`, `transfer()`, `recalibrate()`, `reset()`), the driver waits from messages from a specified source. Explain. Why is this the typical structure for a driver like that for the floppy

disk, and will you not find this in the terminal driver?

39 In an OS, for each interrupt a message is sent to the task that controls the device the interrupt came from; this OS does not buffer messages. Explain how this latter design decision causes problems with clock and keyboard interrupts. Explain how this can be solved.

40 Explain the principle of a *mailbox*, and the discuss its behaviour in the context of both *synchronous* and *asynchronous* sends.

41 A mailbox is a data structure for exchanging messages which is independent of both senders and receivers. The following three mailbox operations are implemented within the kernel (assuming the pre-defined type **`process`**)

(**`AllocMB(mailbox MailBox)`**): Allocates a mailbox, returning a pointer in **`MailBox`**, with buffer space to hold 10 messages. Negative value for **`MailBox`** indicates no spare mailboxes to allocate. Kernel must initialise mailbox data structure.

(**`send(mailbox MailBox, message send_msg)`**): An asynchronous send, so the messages must be buffered by the kernel; returns a result

(**`receive(mailbox MailBox, message rec_msg)`**): the receiver is blocked until a message is available.

Assume messages have fixed length of 100 bytes.

a) Suggest possible error returns for the send result parameter.

b) The situation could arise that a receiver blocks waiting for a message which is not sent because the sender has failed. Suggest a suitable extension to the above primitives to cope with this.

c) Describe the data structures required within the kernel to represent a mailbox with its message buffers.

d) Give pseudo-code implementations for the **`AllocMB`**, **`send`** and **`receive`** procedures specified above, ignoring extensions you suggest in (a), but specifying the types **`mailbox`** and **`message`**.

(No need for detailed queue manipulation code e.g. pseudo-code insert at end of Q or remove from front of Q. Your pseudo code can ignore type checking if necessary.)

File Systems

42 Let *bestand* be a file that occupies the disk blocks 12, 456, 60, 34 and 523 (in that order). Give the names of three methods that connect blocks to files, and show for each, in a diagram, how these methods would represent the file *bestand*.

43 Explain why the information for an open file, kept in main memory, is larger than that for a closed file, kept on disk, and what that information represents.

44 Briefly explain what is meant by implementing a tree-structured directory system above a flat basic file system.

45 Assume that an operating system provides a tree-structured file system, outline the operations which must be performed to open a file by means of the following procedure:

where:

(**`pathname`**): filename of the form "**`abc/def/ghijk/lmnop.xyx`**".

(**`mode`**): **`read`**, or **`write`** (only sequential access allowed).

return parameter used for subsequent access to the file.

Describe the operations performed by both the directory system and the flat file system.

Assuming that a file can be shared for reading by multiple users, outline the information that the file system must hold for each open file and for each user.

46 Give the design (in pseudo code) of the procedures **`send`** and **`receive`**, in the context of synchronous message passing.

47 Give the design (in pseudo code) of the procedures **`send`** and **`receive`**, in the context of asynchronous message passing.

48 Describe, step by step, the actions that take place when opening the file **`/usr/home/test.c`**. Focus on i-node handling and block search.

49 What feature is called **`mount`**? What are the consequences for the design of a filesystem when this service is added to a system (e.g., discuss the features that UNIX provides to deal with **`mount`**)?

50 Explain the functionality of a mount operation in a file system. Explain how, in a system like Unix, mount could be implemented, i.e. discuss the features that need to be added to the file system to adequately deal with mount?

Memory management

51 An operating system uses a buddy system to deal with memory requests. Assuming that the computer has 32 MB memory, draw a diagram showing the memory layout after each of the following requests:

- A = Request 11 MB
- B = Request 1.5 MB
- C = Request 7 MB
- D = Request 2.5 MB
- Release B
- Release A
- Release D
- Release C

52 A computer uses a paged memory system with a page size of 1024 bytes. The page table looks as follows:

page no.	frame no.
0	1
1	4
2	0
3	10
4	3
5	7

For each of the following virtual memory addresses calculate their physical addresses: 2024, 2100, 4097, 64.

53 What is the advantage of a paged memory system with small page sizes?

54 What is the advantage of a paged memory system with large page sizes?

55 Calculate the access time for a five-level paging system assuming a **TLB** hit ratio of 80% and 98%. Assume that time for a memory access is 100 ns and for **TLB** access 20 ns. How does this compare to a single-level paging system?

General issues

56 Which kinds of hardware protection mechanisms does the Intel Pentium processor support and how can they be used for protection by the operating system ?

57 Discuss whether it is possible to construct a secure and reliable operating system without any hardware protection. Try to find arguments for and against. Do you know of any examples?

58 Describe the meaning of the following terms and their relevance for operating systems: *Spooling*, *Throughput*, *Response time*, and *Turn-around time*.

59 What are the potential benefits of a microkernel architecture for operating systems compared to a traditional monolithic kernel architecture? Do you know any operating systems which use a microkernel architecture?

60 What is print spooling? Why is this useful for a multi-user system?

61 How does a *distributed operation system* (DOS) distinguish itself from a number of independent computers connected via a local network? Give some characteristics of the DOS *name / directory* service, that distinguishes it from a stand-alone system?

62 Specify the four different layers in I/O handling, together with their function and abstraction mechanisms.

Minix

63 Explain how the fact that messages are not buffered can create problems for the clock and terminal interrupts.

64 In the process administration of the Unix-like operating system Minix, process 0 (zero) corresponds to the pseudo-process **HARDWARE**, that serves as the source of the messages which are sent as the direct result of an interrupt. Although this pseudo-process has a number, it does not have a slot in the Process Table (hence the predicate 'pseudo'). Explain.

65 Discuss the UNIX/Minix file sharing model. In your answer, describe briefly the data that are kept in the datastructures involved, as well as how opening a file will change, and what the effect of the systemcall **fork** is.