UNIVERSITY OF ABERDEEN SESSION 2015/2016

Examination in CS3026 / CS3008 Operating Systems

08 December 2015 Time: 09.00am – 11.00am

Candidates are not permitted to leave the Examination Room during the first or last half hours of the examination.

Answer TWO out of the three questions.

Each question is worth 25 marks; the marks for each part of a question are shown in brackets.

1. (a) Explain the concept of (a) a process and (b) a thread and how they are related to each other.

[5]

(b) What are the advantages as well as disadvantages of using a thread management approach, where the kernel is not involved?

[4]

(c) Describe the two concepts "mode switch" and "context switch": what are these concepts and in what situations of a process execution do they play a role?

[4]

(d) When concurrently executing processes access shared resources (such as shared variables), a race condition may occur. Explain what a race condition is. What has to be guaranteed for concurrently executing processes, so that race conditions are avoided?

[2]

(e) We can use semaphores to regulate the information exchange between two processes, a "producer" and a "consumer", via a so-called "ring buffer". Two functions, read() and write(), are used to write to such a bounded buffer with N places and read from it. The following code fragments show the initialisation of the buffer and the read() and write() function in pseudocode:

```
b[N], in = 0, out = 0
```

```
write(x)
{
   b[in] = x;
   in = (in+1) % N;
}
```

```
read()
{
   y = b[out];
   out = (out+1) % N;
   return y;
}
```

An array b[N] represents the buffer, and two index variables "in" and "out" the next read and write positions in this buffer of a consumer and a producer process.

(i) Why do we have to use a synchronisation mechanism, such as semaphores, to regulate the read and write behaviour of producer and consumer? Explain, under what circumstances read and write operations may take place, and under what circumstances a process has to be stopped from reading from the buffer or writing to it.

[4]

(ii) The following code fragments show a producer using the write() function and a consumer. The Producer and Consumer programs are not complete – they lack the necessary synchronisation between the producer and consumer to regulate their concurrent access to the shared buffer.

```
Producer:                          Consumer:
while(TRUE)                        while(TRUE)
{                                  {
   x = produce();                     y = read();
   write(x);                          consume(y);
}                                  }
```

Extend and complete these two code fragments for a consumer and producer with semaphores. (i) Explain what semaphores have to be used to complete the producer and consumer and (ii) state how the semaphores needed have to be initialised, e.g. you can state with a function "init('semaphore', 'initialvalue')" for each semaphore its initial value.

[6]

2. (a) Explain the following concepts in the context of virtual memory management:

(i) Resident set.

[1]

(ii) Page fault.

[1]

(iii) Thrashing.

[1]

(b) Explain what the Principle of Locality is, and why it is important for the efficient management of virtual memory.

[2]

(c) In virtual memory management, page replacement algorithms decide which pages of virtual memory already loaded into physical memory frames will be replaced by new pages demanded by a process. We assume that the operating system allows an allocation of **3 frames** of physical memory to a process. We also assume that our process requires **5 pages** of virtual memory. As there are less frames than pages, page faults may occur, resulting in the loading of a page into physical memory and possibly a replacement of a page currently occupying a frame. We assume that pages with ID's 1 – 5 are referenced in the following sequence by the process:

Page reference sequence: 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2

(i) How many replacing page faults will occur with this page reference sequence, if we use a FIFO (First-in-First-out) strategy for replacing pages in memory? (For your answer, only count those page faults that replace a page in memory, the initial loading of pages into empty frames is not counted). You can use a table like the one below to illustrate your answer.

	2	3	2	1	5	2	4	5	3	2	5	2
Frame1												
Frame2												
Frame3												
Page faults												

[3]

(ii) Explain, why FIFO is (a) a poor strategy for page replacement, and (b) which problem may occur, if FIFO is used.

[2]

(iii) The Clock algorithm extends the FIFO strategy to achieve a better page replacement performance: describe in detail how this algorithm operates and point out the central concept underlying this algorithm.

[6]

(d) Explain:

(i) What is Working Set, and how is it related to the Principle of Locality?

[2]

(ii) The operating system may monitor the size of the Working Set, what can be concluded from this information?

[3]

(e) A File Allocation Table (FAT) records which disk blocks of a hard disk belong to a file. The FAT itself has to be stored on the hard disk. How many disk blocks will be used for storing the FAT itself, if a USB hard drive with a capacity of 8GB is formatted with a FAT-based file system, where each block is of size 2KB? (please note: 1KB = 2^{10} bytes)

[4]

3. (a) For each of the following scheduling performance criteria, state whether it should be maximised or minimised, and provide a brief explanation.

(i) Throughput

[1]

(ii) Response time

[1]

(iii) Turnaround time

[1]

(b) Four processes P1, P2, P3, P4 are started in this sequence on a computer system. The operating system records the following arrival times for these processes, where they become ready for execution: P1 arrives at time point 0, P2 arrives at time point 4, P3 at time point 6, and P4 at time point 8. We also assume that these processes have the following execution times: P1 executes for 7 time units, P2 for 4 time units, P3, for 1, and P4 for 5 time units.

Process	Arrival Time	Execution Time
P1	0	7
P2	4	4
P3	6	1
P4	8	5

(i) For a Shortest-Job-First (SJF) scheduling policy:
- show in what sequence the processes will be scheduled
- calculate the average waiting time for this batch of processes

[4]

(c) Explain the concepts non-pre-emptive and pre-emptive scheduling. For the First-Come-First-Serve (FCFS) scheduling policy and for the Shortest-Job-First (SJF) scheduling policy, point out whether they are a pre-emptive or non-pre-emptive scheduling policy.

[4]

(d) Explain the concept of Lottery Scheduling. How can it be used to prioritise processes? How does it differ from "Fair Share Scheduling"?

[5]

(e) With Round-Robin scheduling, each process gets a fixed unit of CPU time (a time quantum). If the time quantum is 20ms and there are 10 processes waiting to be scheduled for execution – what is the maximum time a process can expect to wait?

[3]

(f) Let's assume that a computing system has to handle three video streams at the same time:

- Stream 1 is handled by process A: every 30 msec a frame arrives to be processed by process A, process A needs 20 msec CPU time to decode a frame
- Stream 2 is handled by process B: every 40 msec, a frame arrives to be processed by process B, process B needs 20 msec to decode a frame
- Stream 3 is handled by process C: every 50 msec, a frame arrives to be processed by process B, process C needs 20 msec to decode a frame

Given the processing times for each process, is the performance of the computer system good enough to schedule all three streams without delay? In your answer, explain (a) how you can calculate whether a computer system's performance is good enough to handle a real-time scheduling task, and (b) provide an answer for this particular example.

[6]

END OF PAPER