UNIVERSITY OF ABERDEEN SESSION 2014/2015

Examination in CS3026 Operating Systems

15 December 2014 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) When a process executes on a computer system, it will switch between three basic states. Describe (i) each of these three basic processing states and (ii) the transitions between them. (iii) Provide a drawing to show these transitions.

[5]

(b) What is the so-called "Critical Section Problem"? What are the three basic requirements for any solution to the critical section problem? Why are these requirements needed?

[5]

(c) The following implementation for a semaphore shows an outline of the wait(S) and signal(S) functions in pseudo-code:

```
typedef struct { int value; Queue queue; } Semaphore;
Semaphore S;
```

```
wait(S) {
   if(S.value > 0) S.value-- ;
   else {
       add this process to S.plist;
       block process P;
   }
}
```

```
signal(S) {
   if(S.queue is empty) S.value++;
   else {
      remove one of the processes P from S.queue;
      wakeup process P;
   }
}
```

Change the implementation of wait() and signal(), so that also the waiting processes are counted.

[4]

PLEASE TURN OVER

(d) In order to solve the Critical Section problem, we can introduce a mechanism that enforces a strict alternation ("turn taking") between two processes that compete for entering their critical section. However, it does not fulfil all the requirements for solving the Critical Section problem. Describe one problem of this algorithm and why it occurs.

[3]

(e) A bounded buffer with N places, also called a "ring buffer", may be used between a producer and a consumer of information to exchange data. A monitor can be used to synchronize the write and read operations of the two processes. Consider the following fragment of a program, written in a pseudo code:

```
Producer:                          write(x)
while(TRUE)                        {
{                                     b[in] = x;
   x = produce();                     in = (in+1) % N;
   write(x);                          count++;
}                                  }

Consumer:                          read()
while(TRUE)                        {
{                                     y = b[out];
   y = read();                        out = (out+1) % N;
   consume(y);                        count--;
}                                     return y;
                                   }
```

In this program fragment, a producer uses the write() function to write data items into the buffer b[N] (of length N), and the consumer uses the read() function to retrieve data items from the buffer. A variable "count" counts the number of data items currently in the buffer.

Extend the functions read() and write() so that they show the following behaviour with the help of a monitor: a producer can only add data items, if the buffer is not completely full (count == N) and a consumer process can only read, if the buffer is not completely empty (count == 0). Add instructions to both read() and write() that use the variable "count" and monitor-specific concepts to regulate the read / write behaviour of producer and consumer.

[6]

(f) Both the concept of a semaphore and a monitor can be used to guarantee mutual exclusion between processes. However, there is a difference in terms of how they have to be used in programs – what is the difference?

[2]

2. (a) Explain the concept of paging. How does it support the implementation of virtual memory? You can use a drawing to support your answer.

[5]

(b) Explain the problem of fragmentation in the context of allocating memory for processes. What kind of fragmentation can occur when paging is used?

[2]

(c) Let's assume that a (very small process) consists of 5 pages. We also assume that our physical memory only allows an allocation of three page frames that may hold pages. As there are less frames than pages, page faults may occur – this will lead to the loading of a page into memory and result in the replacement of a page currently occupying a frame. We assume that pages with ID's 1 – 5 are referenced in the following sequence:

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

We assume that the loading of a page into an empty frame is not counted as a page fault, we only count the replacement of a page currently loaded into a frame as a page fault. How many of these page faults will occur, if we use the replacement strategy "Least recently used" (LRU)? For the whole page reference sequence, show for each page reference, in which frame the page referenced is stored and when page replacements ("Page faults") occur. 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												

[8]

(d) Explain how the Clock Algorithm implements an efficient paging mechanism for virtual memory.

[4]

(e) What is a Working Set and how can the operating system be tuned by monitoring the working set? How is it related to the principle of "locality"?

[4]

(f) Virtual memory management is supported with a hardware concept called a "Translation Lookaside Buffer" (TLB). Explain what it is used for and why. What has to be done with this buffer when a context switch occurs?

[2]

PLEASE TURN OVER

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

[5]

(b) Consider the scheduling of processes P1, P2, P3 and P4. These processes have the following arrival times, where they become ready for execution: for P1 = 0, for P2 = 2, for P3 = 4, and for P4 = 5 time units. We also assume that these processes will have the following CPU execution times after their arrival: for P1 = 7, for P2 = 4, for P3 = 1, and for P4 = 4 time units.

Process	Arrival Time	Execution Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

(i) Consider the use of a First-Come-First-Serve (FCFS) scheduling policy:

- Calculate the average waiting time for this batch of processes

[3]

(ii) Consider the use of a Shortest-Job-First (SJF) policy:

- Show in what sequence the processes will be scheduled
- Calculate the average waiting time for this batch of processes

[6]

(iii) Point out a problem that may occur when a priority scheduling policy, such as SJF, is used. What can be done to counteract this problem?

[2]

(c) A periodic real-time scheduling algorithm is effective when schedulability is guaranteed – all processes will meet deadlines. 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? Justify your answer by showing how this can be calculated.

[4]

(d) A 4GB flash drive is used as an external block replaceable device, with a block size of 4KB. The flash drive is formatted with a file system that uses a File Allocation Table (FAT) to assign disk blocks to files stored on the flash drive. The FAT also has to be stored on the flash drive. How many disk blocks have to be reserved for the FAT itself? Show your calculations.

[5]

END OF PAPER