UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR09047 OPERATING SYSTEMS

May 2020

13:00 to 15:00

INSTRUCTIONS TO CANDIDATES

Answer any TWO of the three questions. If more than two questions are answered, only QUESTION 1 and QUESTION 2 will be marked.

All questions carry equal weight.

This is an OPEN BOOK examination.

Year 3 Courses

Convener: S.Ramamoorthy
External Examiners: S.Rogers, S.Kalvala, H.Vandierendonck

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. (a) Briefly describe the three classic OS architectures. List the key advantages and drawbacks for each, and mention at least one well-known operating system that implements it. [*3 marks*]

   (b) What is the difference between an *operating system* and a *hypervisor* (or *virtual machine monitor*)? [*2 marks*]

   (c) Why is para-virtualisation a more efficient form of hardware virtualisation? [*2 marks*]

   (d) What is the output of the following snippet of code? Write down what you expect to be printed to the console by the `printf()` functions, and explain your reasoning in detail.

   Assume that all library calls (`fork()`, `getpid()`, and `printf()`) return without any error, and that the main process has PID 4224. The OS will assign the next PIDs sequentially (at increments of one), and there are no other processes or threads running. [*7 marks*]

```
1  int main () {
2    int x = 1, y = 2;
3    int pid;
4
5    x = fork();
6    if (x == 0) {
7      printf("x = %d y = %d\n", x, y);
8      pid = getpid();
9      printf("I am process: %d\n", pid);
10   }
11
12   y = fork();
13   if (y == 0) {
14     printf("x = %d y = %d\n", x, y);
15     pid = getpid();
16     printf("I am process: %d\n", pid);
17   }
18
19   return 0;
20 }
```

   (e) Explain the differences between *kernel-level* threading (or *one-to-one* threading), *user-level* threading (or *many-to-one* threading), and *N:M* threading (or *many-to-many* threading). [*3 marks*]

(f) From the following code snippet, identify a sequence of code lines that when executed may result in a deadlock. Write this sequence as a list of line numbers. Explain why this sequence causes the deadlock to occur.

There are multiple sequences that may cause a deadlock, you only have to identify one such sequence.

- Assume the `down()` and `up()` operations are atomic.
- The deadlock may occur on the `down()` operation.

This is the producer-consumer problem with one consumer task, and one producer task. The example has three semaphores: two counting semaphores, and one mutex. Assume that the producer starts executing first (from line 6). *[8 marks]*

```
1  const int N = 3;
2  semaphore full = N, empty = 0;
3  semaphore mtx = 1;
4
5  Producer Function:
6    while (1) {
7      produce an item A;
8      down(mtx);
9      down(full);
10     insert item;
11     up(mtx);
12     up(empty);
13   }
14
15 Consumer Function:
16   while (1) {
17     down(empty);
18     down(mtx);
19     remove item;
20     up(mtx);
21     up(full);
22     consume an item;
23   }
```

2. (a) Of I/O-bound, and CPU-bound programs, which is more likely to incur voluntary context switches (i.e. purposefully give back control of the CPU to the OS), and which is more likely to incur involuntary context switches (i.e. the OS forcibly interrupts the program's execution)? Explain why this is the case. [*2 marks*]

   (b) Draw a diagram that shows the classic process states, and transitions between them. Briefly describe each state and transition. [*2 marks*]

   (c) Explain the *first-come-first-serve* (FCFS), *round-robin*, and *multilevel feedback queue* (MLFQ) scheduling algorithms. Which one is fairer, and why? [*3 marks*]

   (d) Consider the following set of processes, with their corresponding execution times (in time units) and priorities:

Process	Execution Time	Priority
P1	5	4
P2	2	1
P3	1	2
P4	7	3
P5	4	3

   The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

   i. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms:
      - *First-come-first-served* (FCFS)
      - *Non-preemptive shortest job first* (SJF)
      - *Non-preemptive priority* (where a higher priority number implies a higher priority)
      - *Round-robin* (with a time quantum of 2 units).

      [*6 marks*]

   ii. For each of the scheduling algorithms in 2(d)i, what is the turnaround time of each process? [*3 marks*]

   iii. For each of the scheduling algorithms in 2(d)i, what is the waiting time of each process? [*3 marks*]

   iv. Which of the algorithms in 2(d)i results in the minimum average waiting time (over all processes)? [*2 marks*]

(e) The following processes are being scheduled using a preemptive, priority-based, round-robin scheduling algorithm – i.e. multilevel queue scheduling.

Process	Priority	Execution time	Arrival
P1	8	15	0
P2	3	25	0
P3	4	20	20
P4	4	20	25
P5	5	5	45
P6	5	20	55

Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. The scheduler will execute the highest priority process first. For processes with the same priority, a round-robin scheduler will be used with a time quantum of 10 units. If a process is preempted by a higher priority process, the preempted process is placed at the end of the queue.

Draw a Gantt chart that shows the scheduling order of these processes. [*4 marks*]

3. (a) What is the difference between a *virtual page*, and a *page frame*? [*2 marks*]

   (b) What is the difference between a *translation lookaside buffer (TLB) miss*, and a *page fault*? For each concept, explain how it works. [*2 marks*]

   (c) Suppose an OS implements virtual memory with paging. A virtual address is 32 bits long and a page comprises 4kB (2^{12} bytes). A hierarchical two-level page table is used. The first-level and second-level page tables have 1024 (2^{10}) entries.

   If a process uses the first 1MB (2^{20}) of its virtual address space, how much space would be occupied by the page tables for that process? Assume each page table entry occupies 4 bytes. [*5 marks*]

   (d) Consider a hypothetical machine with four pages of physical memory and seven pages of virtual memory (labelled `A`–`G`). Given the following page reference requests:

   `A B C D E F C A A F F G A B G D F F`

   For both the FIFO and LRU page replacement policies:

   i. Show the sequence of page swaps, and

   ii. compute the number of page faults. [*8 marks*]

   (e) Suppose you have a file system where the block size is 1kB, a disk address is 4 bytes, and an i-node structure contains the disk addresses of: (a) 12 direct blocks, (b) a single indirect block, and (c) a double indirect block.

   (In answering the following questions, you do not need to simplify arithmetic expressions.)

   i. What is the largest file that can be represented by an i-node? [*5 marks*]

   ii. Suppose you want to create a new file of the largest file size. How many free blocks on the disk would there need to be, in order to create that file? [*3 marks*]