UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR10079 OPERATING SYSTEMS

Tuesday 2nd May 2023

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.

CALCULATORS MAY BE USED IN THIS EXAMINATION.

Year 3 Courses

Convener: D.Armstrong
External Examiners: J.Bowles, A.Pocklington, R.Mullins

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. (a) Describe the differences between:

   i. *full-virtualization* and *paravirtualization*, specifically hint at what are the benefits and drawbacks of each approach; [*2 marks*]

   ii. *type 1* and *type 2* hypervisors; [*2 marks*]

   iii. *machine* and *language* virtualization, provide an example for each. [*2 marks*]

   (b) UNIX-like operating systems use the `fork()` syscall to create a new process.

   i. How many processes are created by the following snippet of code (including the main process)? Explain or illustrate your answer, assume `fork()` returns with no error.

```
#include <stdio.h>
#include <unistd.h>

int main() {
  fork();
  fork();
  fork();
  fork();
  fork();
  return 0;
}
```

[*2 marks*]

   ii. Now, let's consider the **next** snippet of code. Write down what you expect to be printed to the console by the `printf()` function, and explain your reasoning in detail. Assume that all library calls (`fork()`, `wait()`, and `printf()`) return without any error, children's ids starts from 136, and are allocated sequentially – i.e., 136, 137, 138, etc.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main() {
  int ret, i, j = -1;
  for (i = 0; i < 5; i++) {
    ret = fork();
    if (ret > 0) { //parent
      wait(0);
      printf(''ret is: %d %d\n'', ret, j);
      break;
    }
    j = i;
  }
  return 0;
}
```

[*3 marks*]

QUESTION 1 CONTINUES ON NEXT PAGE

(c) In general, multithreading improves the overall performance and efficiency of programs in execution.

i. Provide two examples where using multithreading **improves** the performance of a single-threaded program. Explain why each example can improve the performance. [*2 marks*]

ii. Provide two examples of programs where multithreading **fails** to improve on the performance of a single-threaded solution. Explain why each example **fails** to improve the performance. [*2 marks*]

iii. Explain or illustrate the main difference between user-level-threading and kernel-level-threading. In addition:

A. Describe one case in which user-level-threading **is more convenient** (or provides better performance) than kernel-level-threading.

B. Describe one case in which user-level-threading **is less convenient** (or provides lower performance) than kernel-level-threading. [*3 marks*]

(d) When multiple processes access a limited number of resources it is important to identify if the system of processes may deadlock or no.

i. In such context explain with your own words when a system is said to be in a safe state. Describe how to determine if a system is in a safe state. [*2 marks*]

ii. Is the following system of four (4) processes with two (2) resources deadlocked? Explain your reasoning. [*3 marks*]

Current **Allocation Matrix**

	R1	R2
P1	1	3
P2	4	1
P3	1	2
P4	2	0

Current **Needed Matrix**

	R1	R2
P1	1	2
P2	4	3
P3	1	7
P4	5	1

Current **Availability Vector**

R1	R2
2	4

iii. What if we assume the following **Availability Vector** instead? [*2 marks*]

R1	R2
1	4

2. (a) About I/O devices.

    i. Describe the difference between blocking and non-blocking input/output operations. Provide an example use-case for each of them. [*2 marks*]

    ii. Describe the difference between Programmed I/O (PIO), and Direct Memory Access (DMA). Explain when it is more convenient to use PIO instead than DMA, and the contrary. [*2 marks*]

    iii. Briefly illustrate in writing or with a schema how device access with memory-mapped IO differs from access using IO ports. [*2 marks*]

   (b) Consider the following set of processes, with their corresponding execution times (in arbitrary time units) and priorities (the lower the number, the higher the priority):

Process	Execution Time	Arrival Time	Priority
P1	4	0	2
P2	3	0	3
P3	1	0	4
P4	8	3	1
P5	1	6	2

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0, but P4 arrives at time 3, and P5 arrives at time 6.

    i. Illustrate the execution of these processes using the following scheduling algorithms (use the tables in the answer sheets, put an X or ink a cell when a process is executing): [*5 marks*]

    - *First-come-first-served* (FCFS)
    - *Non-preemptive shortest job first* (SJF)
    - *Preemptive shortest job first* (SJF)
    - *Round-robin* (with a time quantum of 4 units)
    - *Non-preemptive priority*

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

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

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

QUESTION 2 CONTINUES ON NEXT PAGE

(c) Four processes, *Pa*, *Pb*, *Pc*, and *Pd*, have the following sequential execution patterns:

Pa: [CPU $4ms$; I/O $2ms$; CPU $4ms$; I/O $2ms$; CPU $4ms$]

Pb: [CPU $3ms$; I/O $2ms$; CPU $3ms$; I/O $2ms$; CPU $3ms$]

Pc: [CPU $1ms$; I/O $2ms$; CPU $1ms$; I/O $2ms$; CPU $1ms$; I/O $2ms$; CPU $1ms$]

Pd: [CPU $3ms$; I/O $3ms$; CPU $3ms$]

I/O operations for the two processes do not interfere with each other and are blocking.

i. If the processes are run consecutively one after another, what is the elapsed time for all to complete? [*2 marks*]

ii. Sketch the execution pattern under non-preemptive scheduling and determine the total elapsed time for completion. All processes arrive at the same time. You may assume that processes are scheduled in the order in which they become ready to run and that in the event of a tie

- *Pa* has priority over *Pb*, *Pc*, and *Pd*;
- *Pb* has priority over *Pc* and *Pd*; while
- *Pc* has priority over *Pd*.

You may further assume that the scheduler algorithm and the context switching take negligible time. [*2 marks*]

(d) Mechanical disks (HDDs), or simply disks, still play an important role in storage systems despite the introduction of SSDs. For the sake of this exercise consider a disk drive with 64 cylinders, numbered from 0 to 63. The request queue has the following composition:

```
23  1  27  14    35  30  39  8    51  40  31  62
```

If the current position is 15, and the previous request was served at 13, compute the total distance (in cylinders) that the disk arm would move for each of the following scheduling algorithms:

- FIFO,
- SSTF,
- SCAN, and
- C-SCAN. [*4 marks*]

3. (a) Consider an operating system that uses hardware paging to provide virtual memory to applications.

   i. How does the use of virtual memory improve system utilization? Provide at least an example of that. *[2 marks]*

   ii. What is demand paging? Can you also describe an alternative to demand paging? *[2 marks]*

   iii. What is the (virtual memory) working set of a process? Mention at least a use-case of the working set size. *[2 marks]*

(b) Suppose an OS implements virtual memory paging. A virtual address is 40 bits long and a page comprises 1kB (2^{10} bytes). A hierarchical *three-level* page table is used. The first-level (outer), second-level, and third-level (inner) page tables have 1024 (2^{10}) , 1024 (2^{10}), and 1024 (2^{10}) entries respectively.

If a process uses the first 2GB (2^{31}) of its virtual address space, how much space would be occupied by the page tables for that process, when:

   i. Each page table entry occupies 4 bytes? *[2 marks]*

   ii. Each page table entry occupies 8 bytes? *[1 mark]*

Now suppose that instead of a hierarchical three-level page table, a single-level page table is used, and again a process uses the first 2GB (2^{31}) of its 40 bits virtual address space. How much space would be occupied by the page table for that process when:

   iii. Each page table entry occupies 4 bytes? *[2 marks]*

   iv. Each page table entry occupies 8 bytes? *[1 mark]*

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

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

For the *FIFO*, *LRU*, and *second chance* page replacement policies:

   i. Show the sequence of page swaps, and *[3 marks]*

   ii. compute the number of page faults. *[3 marks]*

QUESTION 3 CONTINUES ON NEXT PAGE

(d) Suppose you have a file system where the block size is 8kB, a disk address is 4 bytes, and an i-node structure contains the disk addresses of: (a) 13 direct blocks, (b) 1x single indirect blocks, (c) 1x doubly indirect blocks, and (d) 1x triply indirect blocks. Moreover, the rest of the i-node can be used to store the very first bytes of the file – hence, if the file is small, it can be fully contained in the inode.

(In answering the following questions, you do not need to simplify arithmetic expressions, but you should show the math and explain your calculations.)

i. What is the maximum file size supported by this system? [*2 marks*]

ii. What is the maximum file system partition supported by this system? [*2 marks*]

iii. Assuming no information other than that the file inode is already in main memory, how many disk accesses are required to access either the byte in position (a) 17, or (b) in position 23,423,956? [*3 marks*]