UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR10079 OPERATING SYSTEMS

Thursday 12 th May 2022

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 NOT BE USED IN THIS EXAMINATION

Year 3 Courses

Convener: D.Armstrong
External Examiners: J.Bowles, A. Pocklington, H.Vandierendonck

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. (a) Describe the differences between:

   i. an *operating system* and a *hypervisor* (*virtual machine monitor*), specifically distinguishing the type of abstractions offered by each; [*2 marks*]

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

   iii. *trap-and-emulate* and *binary translation*. [*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? Explain or illustrate your answer, assume `fork()` returns without any error.

```c
#include <stdio.h>
#include <unistd.h>
#define LOOPS 5

int main()
{
  int i, ret;
  for (i = 0; i < LOOPS; i++)
    ret = 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.

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#define LOOPS 5

int main()
{
  int ret, i, j = -1;
  for (i = 0; i < LOOPS; i++) {
    ret = fork();
    if (ret > 0) { //parent
      wait(0);
      break;
    }
    printf("ret is: %d %d\n", ret, j);
    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. If an application contains 25% of code that is inherently serial, what will the maximum speed-up be if run on a multicore system with four processors? Explain your reasoning. [*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
1	4

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

R1	R2
2	4

2. (a) About blocking and non-blocking I/O, and DMA.

   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 how Programmed I/O (PIO) works and state its disadvantages. Under what conditions is polling a sensible approach? [*2 marks*]

   iii. What advantages does direct memory access (DMA) provide? Describe its operation as seen by a device driver in the operating system. [*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	Priority
P1	2	2
P2	4	4
P3	1	2
P4	7	1
P5	3	3

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0, but P5 that 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 priority*
   - *Non-preemptive shortest job first* (SJF)
   - *Preemptive shortest job first* (SJF)
   - *Round-robin* (with a time quantum of 2 units)

   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) Two processes, *Pa* and *Pb*, have the following sequential execution patterns:

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

Pb: [CPU 1 ms; I/O 2 ms; CPU 1 ms; I/O 2 ms; CPU 1 ms; I/O 2 ms; CPU 1 ms]

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. 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*. You may further assume that the scheduler and context switches 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:

`32 11 17 24 30 35 39 8 40 51`

If the current position is 26, and the previous request was served at 24, compute the total distance (in cylinders) that the disk arm would move for each of the following algorithms: FIFO, SSTF, SCAN, and C-SCAN scheduling. [*4 marks*]

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

   i. Explain how the hardware and operating system support for paging combine to prevent one process from accessing another's memory. *[2 marks]*

   ii. Explain how space and time overheads arise from use of paging. *[2 marks]*

   iii. Explain how the Translation Lookaside Buffer (TLB) and hierarchical page tables help solving time and space overheads, and in what circumstances. *[2 marks]*

   (b) Suppose an OS implements virtual memory paging. A virtual address is 32 bits long and a page comprises 1kB (2^{10} bytes). A hierarchical two-level page table is used. The first-level (outer) and second-level (inner) page tables have 4096 (2^{12}) and 1024 (2^{10}) entries respectively.

   If a process uses the first 4MB (2^{22}) 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 two-level page table, a single-level page table is used, and again a process uses the first 4MB (2^{22}) of its 32 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:

```
F E D C B A D B 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 *[4 marks]*

   ii. compute the number of page faults. *[2 marks]*

QUESTION 3 CONTINUES ON NEXT PAGE

(d) Suppose you have a file system where the block size is 2kB, a disk address is 4 bytes, and an i-node structure contains the disk addresses of: (a) 12 direct blocks, (b) 2x indirect blocks, and (c) 2x double 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 largest file that can be represented by an i-node? [*2 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 (including i-node related blocks and data blocks)? [*2 marks*]

iii. How many disk blocks must be read to access byte 16 of a file? What about the byte at the 4MB (2^{22}) mark? [*3 marks*]