10. Concurrency (16 p)

Consider the following processes P1 and P2 that update the value of the shared variables, `x` and `y`, as follows:

Process P1 performs the operations `x := x * y; y ++`

```
LOAD R1, X      // (A)
LOAD R2, Y      // (B)
MUL R1, R2      // (C)
STORE X, R1     // (D)
INC R2          // (E)
STORE Y, R2     // (F)
```

Process P2 performs the operations `x ++; y := x * y`

```
LOAD R3, X      // (G)
INC R3          // (H)
LOAD R4, Y      // (I)
MUL R4, R3      // (J)
STORE X, R3     // (K)
STORE Y, R4     // (L)
```

Assume that the initial values of `x` and `y` are 2 and 3, respectively.

1. What would the values of `x` and `y` be after the execution, given that P1 and P2 execute serially (one after the other without any interleaving or concurrent execution)? (8 p)

2. What would the values of `x` and `y` be after the execution of all instructions, given that P1 and P2 are interleaved and the trace of their execution is the following: `(A) -> (B) -> (G) -> (H) -> (I) -> (C) -> (D) -> (E) -> (F) -> (J) -> (K) -> (L)`? (8 p)

Scripting and Programming (80 p)

Write all of your code on the answer sheet.

11. Bash scripting (20 points)

Write a bash script that gets a path to a directory as input and copies everything to a folder and adds `backup-` prefix to the name of the files. The name of the folder should be `backup-<name of folder>-DD:MM:YY`. (Hint: you can use `timestamp=$(date +"%d:%m:%y")` to get the time in the required format). Your code should be a simple bash script, there is no need for error handling. Assume that the given folder exists and it only contains files (there is no need for recursion).

Example: Input structure

```
src
|── a.txt
|── b.txt
```

output structure

```
backup-src-30:05:23
|── backup-a.txt
|── backup-b.txt
```

12. Drivers (20 p)

Message from a driver

Assume that you are designing a kernel driver. You are asked to write a very simple `chdev_open` function. The only thing you have to do in this function is to output "driver is now opened!" to the kernel logs. Please fill the given function below to achieve this goal:

```
/*Write your solution on the answer sheet*/
static int chdev_open(struct inode *inode, struct file *file)
{
    ... // YOUR SOLUTION
    return 0;
}
```

13. Process Control (20 p)

Whenever an operating system is launching a process, it needs to initialize a specific data structure (i.e. process control block). In this exercise we will use a very simple data structure that includes only the ID and two flags about the state of the process. Moreover, we will initialize the data operated by the process within the same structure (this is usually not done this way). The process operates on data that is organized in a matrix of `m` rows and `n` columns, and the matrix should be initialized like below:

$$matrix_{ij} = \begin{cases} 2 & \text{if } i == j \\ 1 & \text{o.w.} \end{cases}$$

```
typedef struct {
    char* process_id;
    bool is_finished;
    bool run_parallel;

    int m; // number of rows of matrix
    int n; // number of columns of matrix
    int** matrix;
} Process;
```

Write the code necessary for allocating (`init_job` in Part A) and freeing (`free_job` in Part B) the process control structure. You have to use dynamic allocation (`malloc` and `free`) for the matrix (i.e. do not use arrays).

Part A (10 p)

```
/*Write your solution on the answer sheet*/
Process* init_job(char* process_id, bool run_parallel, int m, int n) {
    ... // YOUR SOLUTION
}
```

Part B (10 p)

```
/*Write your solution on the answer sheet*/
void free_job(Process* job) {
    ... // YOUR SOLUTION
}
```

14. Synchronization (20 points)

Mutex (10 p)

Two threads are sharing a variable called `counter`. The variable is updated atomically by each thread. Write the `incrementCounter` function to increment the `counter` in a loop of `1000` steps. Make sure to synchronize the critical section by protecting it with `mutex`.

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

int counter = 0;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

/*Write your solution on the answer sheet*/
void *incrementCounterMutex(void *arg) {
    ... // YOUR SOLUTION
    return NULL;
}

int main() {
    pthread_t thread1, thread2;

    pthread_create(&thread1, NULL, incrementCounterMutex, NULL);
    pthread_create(&thread2, NULL, incrementCounterMutex, NULL);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    printf("Counter: %d\n", counter);

    pthread_mutex_destroy(&mutex);

    return 0;
}
```

Semaphore (10 p)

In this exercise you do not have access to mutexes and would like to use semaphores for synchronizing the threads. Use semaphore synchronization methods in the incrementation function.

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>

int counter = 1;
sem_t semaphore;

/*Write your solution on the answer sheet*/
void *incrementCounterSem(void *arg) {
    ... // YOUR SOLUTION
    return NULL;
}
```

CS-C3140 Operating Systems

Exam Sheet - 19 February 2024

Instructions

- The latest version of the Aalto University Examination Guidelines is applicable.
- No electronic devices are allowed (e.g. laptops, smart watches, phones) -- only exception is non-programmable calculators.
- **Write all your answers on the answer sheets. Answers written on the exam sheet will not be considered.**
- Return both the exam and answer sheets at the end of the exam.

Single Choice Questions (10p)

Each of the following questions has a single correct answer. Write down the letter corresponding to the answer choice on the answer sheet.

1. What are four main elements of a computer? (2 p)

a. Processor, Main Memory, I/O Modules, System Bus

b. Monitor, Keyboard, Mouse, Printer

c. Processor, Hard Drive, RAM, Monitor

d. Main Memory, Monitor, Keyboard, Mouse

2. Which statement best describes a distributed operating system? (2 p)

a. An operating system that could be deployed on any device that has a network interface.

b. A system which has multiple components spread across different machines, which coordinate and communicate over a link to appear as a single coherent working system.

c. A system that is specifically designed to be modular and deployable on multiple embedded devices, linked over a network.

3. What does it mean to preempt a process? (2 p)

a. To temporarily suspend the process and resume it later.

b. To terminate the process and remove it from memory.

c. To forcibly interrupt the execution of a process and allocate the CPU to another process.

d. To modify the priority of the process based on its resource requirements.

4. Why cannot you disallow mutual exclusion in order to prevent deadlocks? (2 p)

a. There are some resources (like printers) that are inherently non-shareable, and it is impossible to disallow mutual exclusion.

b. Some resources, such as files, should always have exclusive access for writes even if reads are not exclusive.

c. Mutual exclusion can be disallowed, because deadlocks are not a concern in practice but mostly a theoretical concept.

5. What is the difference between pages and frames in the context of memory management? (2 p)

a. Pages and frames are the same and are used interchangeably.

b. Pages are fixed-size blocks of main memory, while frames are blocks of a process that can vary in size.

c. Pages are fixed-size blocks of a process that is in secondary memory, while frames are the fixed-size blocks of main memory.

d. Frames are pieces of a program in execution, while pages are physical sections of the main memory.

Concepts and Algorithms (70p)

6. Memory (14p)

Assume a task is divided into four equal-sized segments, and the system builds an eight-entry page descriptor table for each segment. Thus, the system has a combination of segmentation and paging. Assume also the page size is 2 kiB.

1. What is the maximum size of each segment? (2p)
2. What is the maximum logical address space for the task? (2p)
3. Assume an element in physical location 00021ABC is accessed by this task. What is the maximum physical address space for the system? (4p)
4. Assume an element in physical location 00021ABC is accessed by this task. What is the format of the logical address that the task generates for it? Hint: The logical address is broken down into three parts: segment, page, and offset. Determine the size of each part. (6p)

7. Processes and Threads (16p)

1. What are the five different states of a process as discussed in the lecture? Explain each state with one line. (10p)
2. What is a thread? (2p)
3. How do threads function as part of the process execution model? (4p)

8. Scheduling (12p)

The processes from the table below will be scheduled using a Round Robin scheduler having the following characteristics:

- the time quanta q = 4 units of time
- the context switch s = 1 units of time

Calculate the average turnaround time of the processes.

Process	Arrival Time	Processing Time
P1	0	12
P2	2	6
P3	8	18
P4	10	4

9. Deadlocks (12p)

1. Which are the four conditions of deadlock (4p)?
2. Show that the four conditions of deadlock apply to the situation depicted below (8p)