

Ex. 1	
Ex. 2	
Ex. 3	
Ex. 4	
Ex. 5	
Ex. 6	
Tot.	

# Operating Systems

## Examination task

### Examples of questions of previous exams

ID number \_\_\_\_\_ Surname \_\_\_\_\_ Name \_\_\_\_\_

Professor: ☐ Scanzio

**It is not possible to consult texts, notes or to use calculators. The only material allowed consists in the forms distributed by the professor. Solve the exercises in the reserved spaces. Additional sheets are permitted only when strictly necessary. Report the main steps for solving exercises.**

**Duration:** 100 minutes.

#### 1. Exam 2019/02/01 Exercise 1:

Suppose that the hard disk of a small embedded system is composed of 24 blocks of 1 MByte each, which are numbered from 0 to 23. Suppose that the operating system keeps track of the free (occupied) blocks indicating them in a vector with the value 0 (1), and that the current situation of the disk is represented by the following vector:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	1	0	0	0	0	1	0	0	1	1	1	0	1	0	0	1	0	1	0	0	1	0	0

With reference to the file allocation methodologies *contiguous*, *linked*, *FAT* and *indexed*, indicate how the files File1, File2 and File3 (with dimension 2.4, 1.6 and 3.9 Mbyte, respectively) can be allocated.

Schematically report the information stored in the entry directory and the relative pointers for each strategy.

### 3. Exam 2019/02/01 Exercise 4:

A file has the following format:

```
# prog1
f1.c
func1.c
main1.c
my_func.c
END
# prog3
main2.c
func2_2.c
my_func.c
END
...
```

Implement a BASH script that:

- Receive the name of a file from command line. This file has the format previously indicated.
- Compile the files whose names are reported in the lines that do not begin with the character #. It must generate an executable whose name is reported in the line of the file that starts with the character # and precedes the name of the source files to be compiled. Lines containing the string END indicate the termination of the list of source files to be compiled.

The script must also generate a directory named `log` (in the case it does not already exist), and copy in this directory the output of all the compilation commands, each in a file with the same name as the executable, but with extension `.log`.

For example, by running the script on the previous source file, the following commands should be generated:

```
gcc -Wall -o prog1 f1.c func1.c main.c my_func.c
gcc -Wall -o prog3 main2.c func2_2.c my_func.c
...
```

and their output must be stored in the files `./log/prog1.log` and `./log/prog3.log`, respectively.

#### 4. Exam 2019/02/01 Exercise 5:

In linear algebra, matrices multiplication is an operation that produces a matrix  $C$ , by performing the product rows by columns of two given matrices  $A$  and  $B$ .

More in details, if matrix  $A$  has dimension  $[r, x]$  and matrix  $B$  has dimension  $[x, c]$ , then  $C$  has dimension  $[r, c]$ , and each element in position  $(i, j)$  can be computed as:

$$C[i][j] = \sum_{k=0}^{x-1} A[i][k] \cdot B[k][j]$$

Write a multi-thread function

```
void mat_mul (int **A, int **B, int r, int x, int c, int **C);
```

that can compute the product matrix  $C$  executing a thread for each of its elements.

Each thread has to compute the value of one element, executing the row by column product previously reported.

Define the data structure needed for the execution of each thread.

5. **Exam 2019/02/01 Exercise 6:**

Consider the following set of processes:

Process	Arrival Time	Burst Time	Priority
P <sub>0</sub>	0	22	2
P <sub>1</sub>	0	26	3
P <sub>2</sub>	5	19	1
P <sub>3</sub>	11	17	4
P <sub>4</sub>	13	18	5

Represent using a Gantt diagram the execution of these processes using the scheduling algorithms PS (Priority Scheduling), RR (Round Robin), and SRTF (Shortest Remaining Time First).

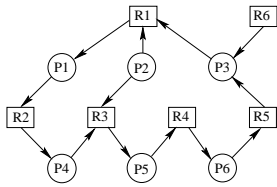
Compute the average *waiting time* for each process and for the global set of processes. Consider a temporal *quantum* of 10 units.

Illustrate which other evaluation metrics can be used in order to compare the previously indicated scheduling algorithms.

6. **Exam 2019/02/27 Exercise 6:**

Illustrate the terms “*safe state*”, “*safe sequence*”, and (with an example) “*Resource trajectory plot*”.

For the management of the deadlock, two operating systems use the graph reported on the left, and the table reported on the right of the following figure.



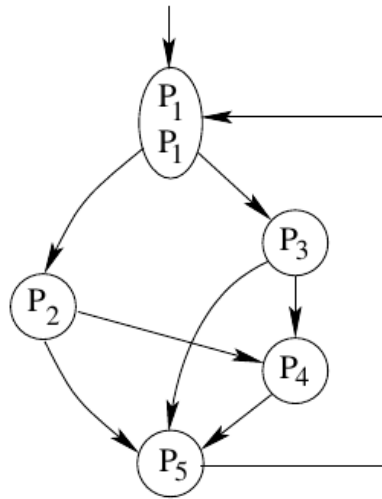
Process	Finish	Allocation			Max			Need			Available		
		$R_1$	$R_2$	$R_3$	$R_1$	$R_2$	$R_3$	$R_1$	$R_2$	$R_3$	$R_1$	$R_2$	$R_3$
P <sub>1</sub>	F	1	0	1	4	3	4				3	2	2
P <sub>2</sub>	F	0	0	1	4	4	2						
P <sub>3</sub>	F	0	1	1	2	3	3						
P <sub>4</sub>	F	0	0	0	5	4	5						
P <sub>5</sub>	F	0	1	0	1	4	4						

Considering the two cases separately, indicate whether the states represented are safe, and in the case reports a possible safe sequence.

7. **Exam 2016/02/28 Exercise 2:**

Given the following precedence graph, implement it by using the minimum number of semaphores. Processes must be **cyclic** (i.e., function with body **while(1)**).

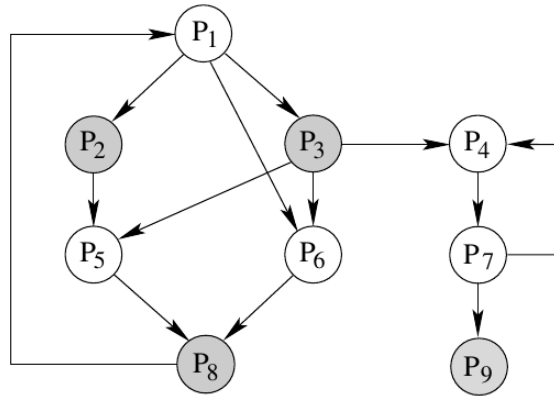
Use the primitives `init`, `signal`, `wait` and `destroy`, indicate and eliminate any unnecessary arc, report the main part of processes ( $P_1, \dots, P_5$ ), and the initialization of semaphores. Note that process  $P_1$  must perform two iterations in sequence on its while loop for each global iteration over the entire precedence graph.



8. Exam 2014/09/08 Exercise 3:

Given the following precedence graph, implement it by using the minimum number of semaphores. White processes are **acyclic**, while gray processes are **cyclic** (i.e., function with body **while(1)**).

Use the primitives `init`, `signal`, `wait` and `destroy`. Report semaphores initialization and destruction, and the main part of processes ( $P_1, \dots, P_9$ ).



## 9. Exam 2019/02/27 Exercise 1:

Suppose that compiling the following program

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
int main (int argc, char *argv[]) {
    int i = 2;
    if (!strcmp (argv[0], "/home/quer/p")) {
        i++;
        fprintf (stdout, "exec 1 %d\n", i);
        execlp ("echo", "./p", "/home/quer/p", "i", NULL);
    }
    if (!strcmp (argv[0], "/home/quer/20190227")) {
        fprintf (stdout, "exec 2 %d\n", i);
        execlp ("./p", "/home/quer/p", "/home/quer/20190227", "++i", NULL);
    }
    fprintf (stdout, "i = %d\n", i);
    return (1);
}
```

an executable with name /home/quer/20190227 has been obtained.

Then, suppose to run the following script:

```
/home/quer/20190227
ln /home/quer/20190227 p
/home/quer/p
```

Report the *control flow graph (CFG)* and the *process generation tree* produced by its execution. In addition, indicate what it produces on the screen and for what reason.