

Reserved cells

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

Operating Systems

Examination task

05 September 2019

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. Specify the meaning of *Process Control Block* and of *Context Switching*.

Represent and describe the state diagram of a process. Also indicate the meaning of “orphan process” and “zombie process”.

Finally, reports two *precedence graphs* that make use of the system calls `fork` and `wait`, one must be realizable, the other not realizable.

2. Suppose to execute the following program

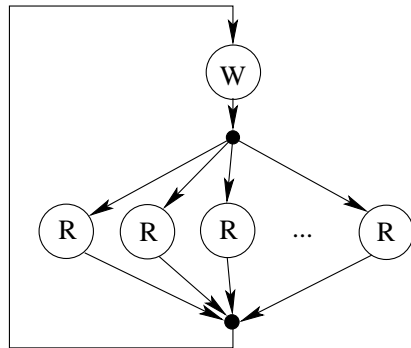
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
void *t1 (void *p){
pthread_t thread;
int *pn = (int *) p;
int n = *pn;
if (n>0) {
printf ("thread: %d\n", n--);
pthread_create (&thread, NULL, t1, &n);
}
pthread_join (thread, NULL);
pthread_exit (NULL);
}
int main (int argc, char *argv[]) {
pthread_t thread;
int n = atoi (argv[1]);
setbuf (stdout, 0);
printf ("main 1: %d\n", n);
if (fork()) {
printf ("main 2: %d\n", -n);
pthread_create (&thread, NULL, t1, &n);
pthread_join (thread, NULL);
} else {
system ("echo main 3: n\n");
execlp ("echo", "bash", "main 4:", "n", NULL);
}
return 1;
}
```

with only one integer parameter equal to 3.

Report the *control flow graph (CFG)* and the *process generation tree* after its execution. Also indicate what it produces on video and for what reason.

3. Illustrate the *Readers and Writers* problem. Report its solution using semaphore primitives in the case of precedence to Readers (illustrate the meaning of each semaphore). What do “precedenza ai Readers” mean?

Starting from the solution of the previous problem, realize the synchronization scheme represented by the following *precedence graph*.



The number of Readers is unknown

Cyclically, the execution of one Writer is followed by the execution of an indefinite number of Readers. At the beginning, it is necessary to run a Writer. At the end of the Writer, it is needed to execute one or more Readers. However, when the last Reader exits the critical section, before another Reader enters, it is necessary that a new Writer is executed.

4. A textual file with name `virus.dat`, stores a list of PID on subsequent lines.

Realize a BASH script that verifies which of the processes listed in the file `virus.dat` are running in the system. For each of these processes in execution, the script has to visualize the name of the owner, and the list of the PID of all its children processes.

Remember that the command `ps -ef` provides an output similar to the following:

```
UID          PID  PPID  C  STIME TTY          TIME CMD
root          1     0  0   giu14 ?           00:00:08 /sbin/init splash
syslog       630     1  0   giu14 ?           00:00:00 /usr/sbin/rsyslogd -n
avahi       665   643  0   giu14 ?           00:00:00 avahi-daemon: chroot helper
quer        938     1  0   giu14 ?           00:00:01 /lib/systemd/systemd --user
quer        946   938  0   giu14 ?           00:00:00 (sd-pam)
...
```

5. Write a multi-thread program able to read and write a matrix of integer numbers as described in the following.

The matrix, statically defined with R rows and C columns, has to be firstly read from standard input and then write on standard output. Both operations have to be performed on a specific order, from row number 0 to row number $R-1$, from column 0 to column $C-1$.

The operations of read and write must be performed by the program using a single function thread that reads, and using a single function thread that writes, both executed R times. At the beginning, the program executes the R threads that read, which synchronize to read the matrix in the indicated order (the first thread reads the first row and the others wait, then the second thread reads the second row and the other wait, etc.). Each thread reads the whole line. After reading, the program executes the R threads that write, which synchronize with each other and write the matrix in the indicated order. Each thread writes the whole row. The program then ends.

6. In reference to the *indexed allocation* in UNIX/LINUX, indicate the meaning of the following terms (possibly using appropriate graphic aids): *directory block*, *directory entry*, *data block* and *i-node*.

Specify the meaning of “*permissions*” (or rights), “*owner*” and “*group*” of a file or a directory in the UNIX system.

How can you modify the “*permissions*”, the “*owner*” and the “*group*” of a file or a directory?

Clarify the meaning of *soft-link* and *hard-link*, reporting the commands to create them.

Let *s1* be a soft-link to the file *s*, *h1* a hard-link to the file *h*, and *d* a directory.

In this circumstance, indicate what happens to the objects and to the links following the 6 following operations, which are (**not**) executed in sequence, but all performed starting from the initial state previously indicated): `cp s1 s2`, `cp h1 h2`, `rm s`, `rm h`, `mkdir d/d1`, `mkdir d/d2`.