NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

NB:
* For this part of the exam you have **30 minutes.**
* You can **NOT** use either books or notes or mobile phones.
* Mark the correct answer in the following table.
* A correct answer gives you 1 pt., an incorrect answer ***decreases*** your grade by 0.25 pts.
* This part of the exam represents **20 %** of the exam grade.

QUESTION 1. For a SJF (Shortest Job First) scheduler when can a context switch occur excepting the cases of a process blocking or terminating?
A.- When another process with a higher priority becomes ready.
B.- In no other case.
C.- When another process becomes ready and has a lower remaining execution time than the current executing process.
D.- When the executing process finishes its share.

QUESTION 2. When is a process sent to execute in background?
A.- When its father dies without calling WAIT.
B.- When the process connects to the father through a pipe and does not wait.
C.- When it is inherited by the process INIT.
D.- When the father executes the process and does not call wait.

QUESTION 3. The file pepe has the rights rwxr-xr-x. What command has to be executed in order to allow the file to be only read and executed by the owner and its group and be only read by others?
A.- chmod 766 pepe
B.- chmod 554 pepe
C.- chgrp +rx pepe
D.- chmod 556 pepe

QUESTION 4. The operating system Windows NT is designed as:
A.- A virtual machine system.
B.- Monolithic system.
C.- Multiple layer hierarchy.
D.- Client-Server system.

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

QUESTION 5. Which option is true after the concurrent execution of the following code by four processes finishes, given that the initial value of V is 0?

```
wait(sem);
V++;
signal(sem);
```

A.- V is 1,5.
B.- V is 4.
C.- V is 3.
D.- V is 0.

QUESTION 6. What does characterize most an operating system ?
A.- Services and system calls.
B.- Command interpreter.
C.- The shell.
D.- The commands.

QUESTION 7. The system call "stat(…); " allows obtaining:
A.- The state of the executing process.
B.- Superblock information of the given file system.
C.- Inode information of the given file.
D.- The state of all processes executing in the system.

QUESTION 8. What information share the father A and his son B after executing the following code?

```
Process A
        if (fork()!=0)
                wait (&status);
        else
                execve (B, parameters, 0);
```

A.- Stack segment.
B.- Open file descriptors.
C.- Text segment.
D.- Data segment.

QUESTION 9. Process A opens a file with descriptor fd and size 19.000 bytes. Subsequently, it creates a son B with a fork call. Assuming that the process A executes first the following code

```
#define BEGIN 0
lseek(fd,805,BEGIN);
```

and later process B executes the following code

```
#define BEGIN 0
lseek(fd,975,BEGIN);
```

what is the value of the file pointer after the execution of the second lseek ?
A.- 805.
B.- 1.780.
C.- 975.
D.- The file pointer of process A points to 805 and the file pointer of process B points to 975.

QUESTION 10. Assume that /etc/bin/link is a symbolic link pointing to /usr/bin/cp on the /dev/hd3 device. The /usr/bin/cp file has inode 74 and does not have any additional link. Which of the following affirmations is true?
A.- When deleting /etc/bin/link the link counter of inode 74 of /dev/hd3 is decremented.

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

B.- After deleting /usr/bin/cp, the user can still access the file through the path /etc/bin/link.
C.- After deleting /usr/bin/cp the file is definitely deleted and it is not possible anymore to access it through the path /etc/bin/link,
D.- Even after unmounting /dev/hd3 it is still possible to access the file through the path /etc/bin/link.

Solutions:

A. False

B. False

C. True

D. False

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

NB
* For this part of the exam you have **2 hours and 30 minutes.**
* You can **NOT** use either books or notes or mobile phones.
* Answer each exercise in a distinct sheet.

EXERCICE 1 (2.5 pts.)

a) Write a C program that reads two numbers from the keyboard and prints the sum of the two. The program ***MUST*** have the following characteristics:

- It must have **3 processes**. A father process (F), a child process (C) and a grandchild process (G).
- The father process (F) creates a child and waits for his termination.
- The child (C) creates a grandchild process (G), reads two numbers from the keyboard and sends them to G through a **pipe**. When it reads a number 0 from the keyboard sends a signal SIGUSR1 to G.
- The grandchild process receives the numbers through the pipe and sums them. When it receives the signal SIGUSR1, it must print the sum and exit.

Solution:

```
int total=0;

void PrintResult()
{
    printf("Total= %d\n",total);
    exit(0);
}

int main()
{
    int fd[2];  // pipe
    int pidChild;
    int pidGranchild;
    int num;
    int status;
    struct sigaction a;

    pidChild=fork();
    if(pidChild!=0)
    {
        wait(&status);
    }
    else
    {
        pipe(fd);
        pidGranchild=fork();
        if(pidGrandchild!=0)
        {
            scanf("%d",&num);
            while(num!=0)
```

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

```
            {
                write(fd[1],&num,sizeof(num));
                scanf("%d",&num);
            }
            kill(pid,SIGUSR1);
            wait(&status);
        }
        else
        {
            a.sa_handler=PrintResult;
            a.sa_flags=0;
            sigaction(SIGUSR1,&a,NULL);
            while(1)
            {
                read(fd[0],&num,sizeof(num));
                total+=num;
            }
        }

    }
    return 0;
}
```

b) Indicate what changes have to be made to the program in order to execute ***in background*** the reading of the number from the keyboard and the sum

Solution:

The father does not wait for the child to terminate.

c) Could the child process C send the signal SIGKILL to the grandchild process G in order to indicate him to print the sum and exit? Justify your answer.

Solution:

No. If the C sends SIGKILL to G, G dies immediately, without having the time to print the sum. Catching the SIGKILL is not possible, therefore G has no mechanism for implementing the presented functionality.

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

EXERCISE 2 (2.5 pts.)

You are asked to write a program which simulates the queues for renewing a pass. There is a room where the clients are waiting for an available counter. When the counter becomes available the client enters the counter room, searches a free counter and goes there to be served.

A new thread is created for each client who arrives and wants to be served. There is a waiting time (chosen randomly between 1 and 3 seconds) between creations of two consecutive threads

There are 3 counter and for each of them a thread is created.

The solution must not necessary be fair: it need not enforce that the clients are served in the order of their arrival. When one counter is free, a client must simply pass to counter room and search the free counter. Once there, each client is served for a random time between 2 and 7 seconds. After this time the counter becomes again available for another client and the thread associated to the client finishes.

The student can chose to implement completely the program or it can add the necessary control mechanisms to the following code.

```
#define MAXCLIENTS       10
#define COUNTERS       3
pthread_mutex_t mtxparameter;    /* mutex for controlling the access to the parameter */
pthread_cond_t cdparameter;   /* controls that the value of the parameters has been taken */
int transfered=FALSE;

pthread_mutex_t mutex; /* mutex for controlling the access to the counter room*/
pthread_mutex_t mutexcounter; /* mutex for controlling the access to the counter */
pthread_cond_t counterwait[COUNTERS];   /* controls the waiting to the counters*/
pthread_cond_t serving;   /* controls the waiting to the counter room */
int freecounters=COUNTERS;
int served =0;
int vcounters[COUNTERS];   // one entry per counter (0 indicates free and  1 busy)

void  *counter(void *p)  {
  int i,timeserving, *aux, numcounter;
```

-1-

```
/* Store the o parameter in a local variable numcounter*/
   pthread_mutex_lock(&mtxparameter);       /* acceder al parámetro */
  aux=p;
  numcounter=*aux;
  transfered=1;
  pthread_cond_signal(&cdparameter);
  pthread_mutex_unlock(&cdparameter);

  while (1) {
```

-2-

```
/*Wait until the client arrives at the counter vcounters[numcounter]==1 */
```

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

```
     pthread_mutex_lock(&mutexcounter);
    while (vventanillas[numcounter] == 0)
    pthread_cond_wait(&counterwait[numcounter],&mutexcounter);
    pthread_mutex_unlock (&mutexcounter);

    timeserving=random () %6 + 2;  //simulates the serving time
    sleep (timeserving);
```

-3-

```
/*Indicate that the counter is free vcounters[numcounter]=0 and freecounters++ */
    pthread_mutex_lock (&mutexcouter);  //counter is free
    vcounters[numcounters]=0;
    pthread_mutex_unlock (&mutexcounter);

    pthread_mutex_lock (&mutex);
    freecounters++;
    pthread_cond_signal (&serving);   // wake up a client
    pthread_mutex_unlock (&mutex);

  }
  pthread_exit(0);
}

void  *client(void *p)  {
  int i,*aux,user;
```

-4-

```
/* Store the parameter p into the local variable user */
  pthread_mutex_lock(&mtxparameter);
  aux=p;
  user=*aux;
  transfered=TRUE;
  pthread_cond_signal(&cdparameter);
  pthread_mutex_unlock(&mtxparameter);
```

-5-

```
/* Wait until there are free counters: freecounters > 0 */

  pthread_mutex_lock(&mutex);
  while (freecounters ==0 )
     pthread_cond_wait(&serving, &mutex);
  freecounters--;
  pthread_mutex_unlock(&mutex);


  printf ("Client %d passes to the counter room \n",user);
```

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

-6-

```
/* Search a free counter, i.e. an entry in  vcounters with value 0 and mark it as busy*/
 i=0;
  pthread_mutex_lock (&mutexcounter);
  while ( vcounters[i] == 1 && i<COUNTERS) i++;
  vcounters[i]=1;
  pthread_cond_signal(&cdcounter[i]);
  pthread_mutex_unlock (&mutexcounter);

  printf ("Client %d has been served at counter %d\n", user, i);

  pthread_exit(0);
}

main(int argc, char *argv[]){
   int i, timeclientarrivals;
   pthread_t thp[MAXCLIENTS], thc[COUNTERS];
   pthread_attr_t attrclient,attrcounters;

   pthread_mutex_init(&mtxparameter, NULL);
   pthread_cond_init(&cdparameter, NULL);
   pthread_mutex_init(&mutex, NULL);
   pthread_cond_init(&serving, NULL);

   for (i=0;i<COUNTERS ;i++){
    vcounters[i]=0;  // All counters are free
    pthread_cond_init(&counterwait[i], NULL);
   }

   pthread_attr_init(&attrcounter);
   pthread_attr_setdetachstate( &attrcounter, PTHREAD_CREATE_DETACHED);
   for (i=0;i<COUNTERS;i++){
    pthread_mutex_lock (&mtxparameter);
    pthread_create(&thc[i], &attrcounter, counter, &i);
    while ( ! transfered)
      pthread_cond_wait(&cdparameter, &mtxparameter); /* blocks */
    pthread_mutex_unlock (&mtxparameter);
    transfered=FALSE;
    pthread_mutex_unlock (&mtxparameter);
   }
   printf ("Couters created\n");
   pthread_attr_init(&attrclient);
   pthread_attr_setdetachstate( &attrclient, PTHREAD_CREATE_DETACHED);
   srandom (getpid() );
   for (i=0;i<MAXCLIENTS;i++){
    timeclientarrivals=random () %3 + 1;
    sleep (timeclientarrivals);
    pthread_create(&thp[i], &attrclient, client, &i);
    pthread_mutex_lock (&mtxparameter);
    while ( ! transfered)
```

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

```
      pthread_cond_wait(&cdparameter, &mtxparameter); /* blocks */
     pthread_mutex_unlock (&mtxparameter);
   }

   pause ();
   pthread_mutex_destroy(&mutex);
   pthread_cond_destroy(&serving);
   for (i=0;i<VENTANILLAS ;i++){
    pthread_cond_destroy(&counterwait[i]);
   }

   exit(0);
}
```

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

EXERCISE 3 (3 pts.)

a) Given a Unix system with the following information:

I-node table:

Inode number	1	2	3	4	5
Type	Directorio	Directorio	Fichero		
Physical link counter	3	2	1		
Data block address	11	12	13		
…….					

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

Data blocks:

Block number	11	12	13	
Content	. 1	. 2	File data	
	.. 1	.. 1		
	d 2	f1 3		

Indicate the values of i-nodes and blocks after executing each of the following operations (Draw an i-node and data block table for each operation):

1- mkdir /d1

2- ln –s /d/f1 /d1/f2

3- rm /d/f1

Solution:

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

After mkdir /d1**:**

Inode number	1	2	3	4	
Type	Directory	Directory	File	Directory	
Physical link counter	3	2	1	2	
Data block address	11	12	13		
…….					

Block number	11	12	13	14
Content	. 1	. 2	File data	. 4
	.. 1	.. 1		. . 1
	d 2	f1 3		
	d1 4			

After ln –s /d/f1 /d1/f2**:**

Inode number	1	2	3	4	5
Type	Directorio	Directorio	Fichero	Directory	Symbolic link
Physical link counter	3	2	1	2	1
Data block address	11	12	13		15
…….					

Block number	11	12	13	14	15
Content	. 1	. 2	File data	. 4	/d/f1
	.. 1	.. 1		. . 1	
	d 2	f1 3		f2 5	
	d1 4				

After rm /d/f1 **:**

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010

Inode number	1	2	3	4	5
Type	Directorio	Directorio	Free inode	Directory	Symbolic link
Physical link counter	3	2		2	1
Data block address	11	12			15
…….					

Block number	11	12	13	14	15
Content	. 1	. 2	Bloque de datos Libre	. 4	/d/f1
	.. 1	.. 1		. . 1	
	d 2	~~f1 3~~		f2 5	
	d1 4				

b) Write a C program that reads from the keyboard the name of a directory and prints the names and sizes of the files found in the given directory.

Solution:

Any program that :

```
Decomposes the name of a directory into tokens (eg. /a/b/c/d => (a, b, c, d))
Starting with the inode of the root directory
  while (!finished) {
       get the block number
       lookup the crt directory for the current directory componenent
       if (not found) return error;
       else
              if (the whole directory has been parsed)
                     break;
              else
                     continue;
     }
     print the content of the data block of current directory

 }
```

NAME: _____ NIA: _____

Operating Systems Extraordinary Examination
24th of June 2010