DISEÑO Y ADMINISTRACIÓN DE SISTEMAS OPERATIVOS **Enero de 2015**

Material permitido: **Calculadora NO programable.** Tiempo: **2 horas.** N1	**Aviso 1:** Todas las respuestas deben estar razonadas. **Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.** **Aviso 3: No use *Tipp-ex*** o similares (atasca el escáner).

ESTE EXAMEN CONSTA DE 5 PREGUNTAS

1. (*2p*). Explique razonadamente si las siguientes afirmaciones son VERDADERAS o FALSAS

i. (*1p*). Múltiples procesos pueden compartir de forma transparente el mismo fichero.

ii. (*1p*). Cuando se produce una interrupción que no esté bloqueada o enmascarada, el proceso invoca al algoritmo `inthand()`.

2. (*2p*). Explique los motivos por los que el núcleo puede invocar a un driver de dispositivo. Señale y describa las partes en las que se divide un driver.

3. *(1p)*. Indique el resultado de la siguiente ejecución:

```
% chmod g+r prueba
```

4. (*2p*). Explique razonadamente el significado de las sentencias enumeradas ([ ]) del siguiente programa escrito en lenguaje C:

```
#include <signal.h>
main()
{
[1] long mask0;
[2] mask0=sigsetmask(sigmask(SIGUSR1)|sigmask(SIGUSR2));
[3] sigblock(sigmask(SIGINT));
[4] sigsetmask(mask0);
}
```

(PREGUNTA 5 EN LA SIGUIENTE PÁGINA)

5. *(3 p)* Conteste razonadamente a los siguientes apartados:
a) *(1.5 p)* Explicar el significado de las sentencias enumeradas (**[ ]**) de este programa.
b) *(1.5 p)* Explicar el funcionamiento del programa asumiendo que el pid del primer proceso creado es 1000, los pids se asignan secuencialmente y no se produce ningún error.

```
     #include <sys/types.h>
     #include <sys/ipc.h>
     #include <sys/sem.h>
     #include <sys/shm.h>
     #include <stdio.h>
     #include <stdlib.h>

     void main()
     {
          int *x;
          int pid, shmid, semid;
[1]       struct sembuf op[1];
[2]       semid=semget(IPC_PRIVATE,1,IPC_CREAT|0600);
          if (semid==-1)
               {perror("fallo en semget");
               exit(1);}
[3]       shmid=shmget(IPC_PRIVATE,sizeof(int),IPC_CREAT | 0600);
          if (shmid==-1)
               {perror("fallo en shmget");
               exit(2);}
[4]       x=shmat(shmid,0,0);
          *x=0;
[5]       if ((pid=fork())==-1) exit(2);
               if (pid==0)
               {
[6]                 sleep(5);
                    *x=10;
[7]                 shmdt(x);
                    op[0].sem_num=0;
                    op[0].sem_op=1;
                    op[0].sem_flg=0;
[8]                 if (semop(semid, op, 1) == -1)  perror("Error1:");
                    exit(3);
               }
               else
               {
                    struct sembuf op[1];
                    op[0].sem_num=0;
                    op[0].sem_op=-1;
                    op[0].sem_flg=0;
                    if (semop(semid, op, 1) == -1)  perror("Error2:");
                    printf("El proceso %d hace que x=%d\n",pid,*x);
               }
     }
```