Nombre alumno:
DNI:

# Examen final de teoría de SO

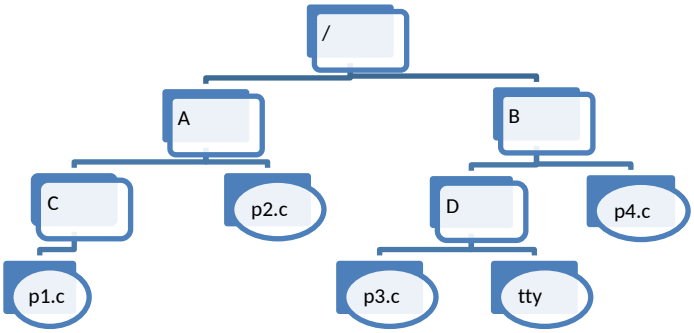
**Justifica todas tus respuestas del examen.** Las respuestas no justificadas se considerarán erróneas.

## Sistema de Ficheros (*2 puntos*)

Tenemos el siguiente sistema de ficheros basado en inodos. Datos:

| | |
|---|---|
| • Los rectángulos representan directorios, los óvalos representan otros elementos | • Las rutas destino de los softlinks se guardan en bloques de datos |
| • El tamaño de bloque es de 4 KB | • p1.c y p4.c son hardlinks al mismo fichero |
| • p3.c es un soft-link a p2.c. Asume que el path no cabe en el inodo. | • p1.c ocupa 2 KB |
| • p2.c ocupa 12 KB | • tty es un dispositivo especial de carácter (una consola) |

a) ***(0,25 puntos)*** asigna a los elementos de la jerarquía siguiente un número de inodo. Indícalo justo a la derecha de cada figura. Usa el cuadro a la derecha para justificar tu asignación.




b) ***(0,75 puntos)*** Completa las siguientes tablas con la información necesaria para representar la jerarquía anterior:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **ID Inodo** | | | | | | | | | | | |
| **#enlaces** | | | | | | | | | | | |
| **Tipo** | | | | | | | | | | | |
| **Tabla de índices a BD** | | | | | | | | | | | |

| **ID BD** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Contenido** | | | | | | | | | | | |

Nombre alumno:
DNI:

c) ***(0,5 puntos)*** Estando situados en el directorio /A ejecutamos el comando "**`mkdir SubA`**". Enumera los cambios que habría que hacer en las tablas anteriores (y en la jerarquía) para completar el comando:

d) ***(0,5 puntos)*** Analiza el siguiente código. Indica para cada línea qué accesos, consultas o modificaciones, se realizan a inodos y bloques de datos del sistema de ficheros. Supón que el sistema tiene **buffer cache de 1000 bloques vacía**. Indica qué accesos serían a través de la buffer cache.

```
1. int fd=open("/B/D/p3.c",O_RDWR);
2. int ret=lseek(fd,0,SEEK_END);
3. int w=write(fd,"a",1);
4. close(fd);
```

Sigue la siguiente nomenclatura, de ejemplo, como GUÍA para escribir tu respuesta:

- I3C significa que se accede al Inodo 3 para Consulta (lectura)
- B6M significa que se accede al Bloque de datos 6 para Modificarlo
- Subraya claramente los accesos que sean en buffer cache

| | **Accesos y justificación** |
|---|---|
| Linea 1 | |
| Linea 2 | |
| Linea 3 | |
| Linea 4 | |

Nombre alumno:
DNI:

## Gestión de memoria (*2 puntos*)

Analiza el siguiente código. Responde a la siguientes preguntas de forma razonada. El código se ejecuta sobre un sistema operativo Linux de 32 bits paginado, con tamaño de página de 4096 bytes, y **no** ofrece la optimización Copy-On-Write pero **sí** ofrece SWAP (memoria virtual). El tamaño del tipo **int** es de 4 bytes. La región de código del programa ocupa 2400 bytes. Suponemos que el tamaño de las regiones para los datos viene dado únicamente por lo definido en este código. Supón que el programa "miprog" existe, la ruta correcta y hay permiso para ejecutarlo.

```
int *a;
int b=4;

int main ()
{
int c, ret;

ret=fork();
------------------PUNTO A
a=sbrk(sizeof(int);


if (ret>0){
   waitpid(-1,NULL,0);
   a=sbrk(sizeof(int);
   *a=1;
   c=(*a)*b;
   }
else if (ret==0)
    execlp("./miprog","miprog", (char*)NULL);
------------------PUNTO B
else error_y_exit("fork");
}
```

A) *(0,5 puntos)* Indica y justifica cuánta memoria física ocupa por región (en marcos de página) la ejecución de este programa en el **punto A**.

B) *(0,25 puntos)* Razona si añadir la optimización COW ofrecería alguna mejora en la gestión de memoria si hacemos avanzar el código desde el punto A hasta el **punto B**.

Nombre alumno:
DNI:

C) *(0, 5 puntos)* Estando en el **punto A**, justifica si con el esquema de gestión de memoria de este sistema se produce algún tipo de fragmentación de memoria, qué tipo de fragmentación sería, en caso afirmativo, y cuánta memoria se desaprovecharía.

D) *(0,5 puntos)* Al ejecutar la instrucción " `*a=1;` " se produce un MAJOR PAGE FAULT. Indica 3 razones que lo puedan provocar:

E) *(0,25 puntos)* Indica las operaciones que realizaría el sistema para solucionar el MAJOR PAGE FAULT

Nombre alumno:
DNI:

## Procesos y signals (*3 puntos*)

Analiza el siguiente código y contesta justificadamente a las preguntas. (se omite el control de errores por simplicidad). El programa "A" existe, es correcto, y no modifica nada relacionado con signals ni hace nada relevante para el ejercicio. El objetivo de este código es crear un nuevo proceso cada vez que se reciba un SIGUSR1 y terminar cuando se reciba un SIGUSR2.

```
1.  uint sig1 = 0, sig2 = 0;
2.  void create_process()
3.  {
4.    int ret;
5.    ret = fork();
6.    if (ret == 0){
7.      execlp("./A", "A", NULL);
8.    }
9.  }
10. void f_sig(int s)
11. {
12.   if (s == SIGUSR1) sig1 = 1;
13.   if (s == SIGUSR2) sig2 = 1;
14. }
15.
16. void main(int argc, char *argv[])
17. {
18.   struct sigaction new_action;
19.   sigset_t        action_mask;
20.
21.   new_action.sa_handler = f_sig;
22.   new_action.sa_flags   = SA_RESTART;
23.   sigfillset(&new_action.sa_mask);
24.   sigaction(SIGUSR1, &new_action, NULL);
25.   sigaction(SIGUSR2, &new_action, NULL);
26.
27.   sigfillset(&action_mask);
28.   sigdelset(&action_mask, SIGUSR1);
29.   sigdelset(&action_mask, SIGUSR2);
30.
31.   while(1){
32.     sigsuspend(&action_mask);
33.     if (sig1) create_process();
34.     if (sig2) exit(0);
35.     sig1 = 0;
36.     waitpid(-1,NULL,0);
37.   }
38. }
39.
```

Nombre alumno:
DNI:

Si el programa se ejecuta en un terminal y devuelve el PID 1000, y en otro terminal ejecutamos la siguiente secuencia:

```
kill -USR1  1000
kill -USR1  1000
kill -USR1  1000
kill -USR2 1000
```

a) (0,5 puntos) ¿Qué pasaría si enviamos los dos SIGUSR1 antes de llegar al sigsuspend?

b) (0,5 puntos) Si enviamos un signal cada minuto y el programa A tarda 5 segundos en ejecutarse, ¿Al recibir el SIGUSR2, cuantos procesos hijos tendrá activos o zombies el proceso 1000?

c) (0,5 puntos) Si movemos la llamada de la función create_process() a la línea 12 (donde se gestiona la recepción del SIGUSR1), ¿Cuál sería la máscara de signals bloqueados de los nuevos procesos?

d) (0,5 puntos) ¿Es necesario modificar el código para evitar que el SIGINT, en ningún momento de la ejecución, termine la ejecución de proceso? Justifica la respuesta y en caso afirmativo indica el código que añadirías y en que líneas.

Nombre alumno:
DNI:

e) (0,5 puntos) Si quisiéramos que la gestión de la finalización de los procesos hijos fuera asíncrona, ¿Qué cambios en el código habría que hacer? Enumera los cambios de forma resumida (lista de cambios).

f) (0,5 puntos) Si queremos hacer una mejora limitando el tiempo de ejecución de los procesos que ejecutan el programa A (por ejemplo 1000 segundos) utilizando signals, indica: (1) Si cambiarias el código del padre o de los hijos, (2) que signal elegirías, y (3) que llamada(s) a sistema añadirías. (solo se pide que termine, no que haga ninguna acción concreta)

## Procesos y pipes (*3 puntos*)

Analiza el siguiente código, que está incompleto, y contesta justificadamente las preguntas. (la función toupper pasa de minúsculas a mayúsculas). El fichero "pipeA" es una pipe. Asume que ejecutamos el programa (B) en línea de comandos de la siguiente forma (el fichero f es un fichero de datos con un texto de 1024 caracteres):

```
$ B < f
```

Nombre alumno:
DNI:

```
1. void process_data()
2. {
3.     char c;
4.     int fd = open("pipeA", O_RDONLY);
5.     while(read(fd, &c, sizeof(char)) > 0){
6.       char uc = toupper(c);
7.       write(1, &uc, sizeof(char));
8.     }
9. }
10.
11.   int main(int argc,char * argv[])
12.   {
13.     int fd, ret, fd_out, c;
14.     char f_name[128], buffer[128];
15.
16.     sprintf(f_name,"f_out.%d", getpid());
17.
18.     fd_out = open(f_name, O_WRONLY|O_CREAT, S_IRUSR|S_IWUSR);
19.
20.     ret = fork();
21.     if (ret == 0){
22.       process_data();
23.     }
24.
25.     ret = fork();
26.     if (ret == 0){
27.       process_data();
28.     }
29.
30.     int turn = 0;
31.     fd     = open("pipeA", O_WRONLY);
32.     while(read(0, &c, sizeof(c)) > 0){
33.       write(fd, &c, sizeof(c));
34.     }
35.
36.     while(waitpid(-1,NULL,0) > 0);
37.     sprintf(buffer, “Ready\n”);
38.     write(1,buffer, strlen(buffer));
39.
40.   }
```

a) (0,25 puntos) Dibuja la jerarquía de procesos que genera este código y el acceso a los ficheros y pipes que harían cada uno de los procesos.

Nombre alumno:
DNI:

b) (0,25 puntos) ¿Deberíamos utilizar el flag O_TRUNC en el open de la línea 18 para asegurar que el fichero de salida contiene únicamente el resultado de cada ejecución?

c) (0,5 puntos) ¿Tendría algún efecto en el comportamiento si movemos el open de la línea 31 a la línea 17?

d) (0,5 puntos) Si queremos que las escrituras de la línea 7 vayan al fichero que hemos abierto en la línea 18, indica que cambios propondrías en la función process_data para garantizarlo.

e) (1 punto) Rellena el estado de las tablas de E/S asumiendo que el proceso inicial está en el waitpid y que los hijos están creados y están justo al inicio del bucle de lectura. Añade las filas o tablas que necesites para reflejar el estado de todos los procesos. Ten en cuenta como se ha ejecutado el programa tal y como se indica al inicio del ejercicio. Incluye el cambio que hayas propuesto en la pregunta d. Las tablas muestran su contenido antes de que la shell cree el proceso que luego ejecutará el programa.

Nombre alumno:
DNI:

**Tabla de Canales**

| Entrada TFA | |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |

**Tabla de Ficheros abiertos**

| | refs | modo | Posición l/e | Entrada T.inodo |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |
| 9 | | | | |
| 10 | | | | |
| 11 | | | | |

**Tabla de iNodo**

| | refs | inodo |
|---|---|---|
| 0 | | |
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |
| 8 | | |
| 9 | | |
| 10 | | |
| 11 | | |

f) (0,5 puntos) ¿Debemos añadir alguna modificación a este código para asegurar que termina? En caso afirmativo indica que código y en que líneas.