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

**ESTE EXAMEN CONSTA DE 5 PREGUNTAS**

**1**. (*2 p*) Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas.

i) *(1 p)* Las interrupciones son atendidas en modo usuario dentro del contexto del proceso que se encuentra actualmente en ejecución, aunque dicha interrupción no tenga nada que ver con la ejecución de dicho proceso.

ii) (*1 p*) Si un usuario normal ejecuta la orden:

```
$ cat /etc/shadow
```

se le solicitará la contraseña de administración.

**2.** (*1.5 p*) Conteste razonadamente las siguientes cuestiones:

a) *(1 p)* ¿Qué es el superbloque?

b) *(0.5 p)* Explique cómo procede núcleo cuando la lista parcial de nodos-i libres del superbloque se vacía.

**3.** (*1.5 p*) Responda razonadamente a las siguientes cuestiones:

a) *(0.75 p)* ¿Qué es un proceso ligero? ¿Puede un proceso ligero aprovechar el paralelismo en un sistema multiprocesador?

b) *(0.75 p)* ¿Cuáles son las principales limitaciones de los procesos ligeros?

**4**. *(2 p)* Responda razonadamente a las siguientes cuestiones. Supóngase que en el directorio de trabajo se ejecuta la orden:

```
$ ls -l
```

Que muestra en pantalla la siguiente línea:

```
drw-r----- 1 addison users 1028 Lu 30 12:40 editoriales
```

a) *(0.5 p)* ¿Cuál es el significado de cada uno de los elementos de la línea que se muestra en pantalla?

b) *(1 p)* Explique los permisos de acceso del fichero `editoriales` para propietario, los miembros del grupo al que pertenece el propietario del fichero y para otros usuarios.

c) *(0.5 p)* Muestre y explique la expresión en modo octal de la máscara en modo simbólica.

**5.** *(3 p)* Conteste razonadamente a los siguientes apartados:

a) *(2 p)* Explicar el significado de las sentencias enumeradas (**[ ]**) del programa que se muestra en la página siguiente.

b) *(1 p)* Explicar el funcionamiento del programa si se compila en el ejecutable `programa`, se invoca con la orden `./programa DyASO` y no se produce ningún error.

**La pregunta continua en la página siguiente**

```
        #include <stdio.h>
        #include <stdlib.h>
        #include <sys/types.h>
        #include <sys/stat.h>
        #include <fcntl.h>
        #include <unistd.h>
        #include <string.h>
        #include <sys/wait.h>

        void manejador(int sig)
        {
[1]        execl("/bin/cat","cat","texto.txt",NULL);
           perror("Error execl");
           printf("¡Incorecto!\n");
           exit(1);
        }

        int main(int argc, char *argv[])
        {
           int fd, pid, salida, status;
           if (argc!=2) {printf("Argumentos incorrectos\n"); exit(2);}

[2]        if ( (pid=fork())==-1 ) {printf("Error fork\n"); exit(3);}
           if ( pid==0 )
               {
[3]            if (signal(SIGUSR1,manejador)==SIG_ERR)
[4]                {perror("Error signal"); exit(4);}
[5]            pause();
               printf("¡Erroneo!\n");
               }
           else
                {
                sleep(1);
[6]             fd=open("texto.txt", O_RDWR|O_CREAT|O_TRUNC, 0666);
                if (fd==-1) {perror("Error open"); exit(5);}
                write(fd,"Ejercicio 5 ",12);
[7]             write(fd,argv[1],strlen(argv[1]));
[8]             close(fd);
[9]             kill(pid,SIGUSR1);
[10]            wait(&status);
                printf(" ¡Correcto!\n");
[11]            unlink("texto.txt");
                }
           exit(0);
        }
```

| | |
|---|---|
| Material permitido:<br>**Calculadora NO programable.**<br>Tiempo: **2 horas.**<br>N2 | **Aviso 1:** Todas las respuestas deben estar razonadas.<br>**Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.**<br>**Aviso 3: No use *Tipp-ex*** o similares (atasca el escáner). |

**ESTE EXAMEN CONSTA DE 5 PREGUNTAS**

**1**. *(2 p)* Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas.

i) *(1 p)* El núcleo puede emplear sus mecanismos de protección para proteger unas hebras de usuario de otras.

ii) *(1 p)* El uso de las colas de mensajes en la transferencia de grandes cantidades de datos no afecta al rendimiento del sistema.

**2** (*1.5 p*) Supóngase que el fichero `/etc/group` contiene, entre otras, las siguientes líneas:

```
root:*:0:
estudiantes:*:10: TUTOR, ESTUDIANTE1
profesores:x:50:DIRECTOR
```

a) (*0.25 p*) ¿Cuántos miembros tiene el grupo `root`?

b) (*0.25 p*) ¿Cuál es el GID del grupo `profesores`?

c) (*0.25 p*)¿Qué usuarios tienen acceso en el grupo `estudiantes`?

d) (*0.75 p*) ¿Tiene contraseña de entrada el grupo `profesores`? ¿Y el grupo `estudiantes`? En caso afirmativo, ¿dónde está almacenada?

**3.** *(1.5 p)* Responda razonadamente a las siguientes cuestiones:

a) (*0.75 p)* ¿Qué son los *callouts*? Relaciona algunos usos de los *callouts.*

b) *(0.75 p)* Describe las formas de implementar la lista de *callouts.*

**4**. (*2 p*) Supóngase que la lista parcial de i-nodos libres del superbloque está vacía, su i-nodo recordado es 25 y su variable índice puede tomar como máximo el valor 4. Además, existen los siguientes i-nodos libres en la tabla de i-nodos: 95, 45, 88, 37, 32, 30 y 73. Dibuje la lista parcial de i-nodos libres del superbloque una vez que ha sido rellenada por el núcleo. ¿Cuál sería ahora el i-nodo recordado?

**5.** *(3 p)* Conteste razonadamente a los siguientes apartados:

a) *(2 p)* Explicar el significado de las sentencias enumeradas (**[ ]**) del programa que se muestra en la página siguiente.

b) *(1 p)* Explicar el funcionamiento del programa si se compila en el ejecutable programa, se invoca con la orden `./programa DyASO` y no se produce ningún error.

**La pregunta continua en la página siguiente**

```
        #include <stdio.h>
        #include <stdlib.h>
        #include <sys/types.h>
        #include <sys/stat.h>
        #include <fcntl.h>
        #include <unistd.h>
        #include <string.h>
        #include <sys/wait.h>

[1]     int main(int argc, char *argv[])
        {
           int fd, pid, salida, status;

           if (argc!=2) {printf("Argumentos incorrectos\n"); exit(1);}
[2]        fd=open("fichero1.sh", O_RDWR|O_CREAT|O_TRUNC, 0666);
[3]        if (fd==-1) {perror("Error open"); exit(2);}
           write(fd,"#!/bin/bash\necho Ejercicio 5 ",29);
[4]        write(fd,argv[1],strlen(argv[1]));
[5]        close(fd);
[6]        chmod("fichero1.sh",0700);

[7]        unlink("fifo");
[8]        if (mknod("fifo",S_IFIFO|0666,0)==-1)
              {perror("Error mknod"); exit(3);}

[9]        if ( (pid=fork())==-1 ) {printf("Error fork\n"); exit(4);}
           if ( pid==0 )
               {
[10]            salida=system("./fichero1.sh > fifo");
                if (salida==0) printf("¡Correcto!\n");
                unlink("fifo");
                unlink("fichero1.sh");
                }
           else
                {
[11]            execl("/bin/cat","cat","fifo",NULL);
                perror("Error execl");
                printf("¡Incorecto!\n");
                exit(5);
                }
           exit(0);
        }
```

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

**ESTE EXAMEN CONSTA DE 5 PREGUNTAS**

**1.** (*2 p*). Señale **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

I. (*1p*) La rutina `spltty()` invocada cuando se produce una interrupción por parte de un terminal, disminuye el *npi* al valor de *npi* anteriormente almacenado.

II. (*1 p*) En UNIX los atributos de un fichero se encuentran almacenados en la entrada del directorio al que pertenece.

**2.** (*1.5 p*). Conteste **razonadamente** a las siguientes cuestiones:

a) *(0.5 p)* ¿Qué es la tabla de regiones por proceso?
b) *(1 p)* Señale al menos cinco operaciones con regiones implementadas por el núcleo

**3.** (*1.5 p*) Conteste **razonadamente** a las siguientes cuestiones:

a) *(0.5 p)* ¿Qué parámetros de entrada requiere el algoritmo `sleep()`?
b) *(1 p)* Dibuje un diagrama con las principales acciones que realiza el núcleo durante la ejecución del algoritmo `sleep ()`.

**4.** *(2 p)* Considérese un sistema GNU/Linux, con la tabla de montaje ubicada en `/etc/fstab`. El contenido de este archivo es el que se muestra a continuación:

```
# device        directory       type        options
/dev/hda1       /               ext2        defaults
/dev/hda2       /usr            ext2        defaults
/dev/hda3       none            swap        sw
/dev/sda1       /dosc           msdos       defaults
/proc           /proc           proc        none
```

Describa la información que se obtiene a partir de esta contenido.

**5.** *(3 p)* Conteste **razonadamente** a los siguientes apartados:

a) (1.5 p) Explique brevemente el significado de las sentencias enumeradas ([ ]) del programa mostrado en la página siguiente.

b) (1.5 p) El programa es compilado produciendo el fichero ejecutable `Ejercicio`.
Explique la ejecución del programa y su salida si se invoca desde la línea de comandos la orden "`./Ejercicio cinco`", y no se produce ningún error.

**Nota:** `fflush` sirve para vaciar el buffer de salida estándar haciendo que la salida de `printf` se imprima inmediatamente.

**La pregunta continua en la página siguiente**

```
     #include <stdio.h>
     #include <stdlib.h>
     #include <unistd.h>
     #include <string.h>
     #include <wait.h>
     #include <sys/msg.h>
     #include <sys/types.h>
     #include <sys/ipc.h>
     #include <sys/shm.h>
     void main(int argc, char* argv[]){
        int pid, msqid, shmid, par, estado;
        int tam=(strlen(argv[1])+1)*sizeof(char);
        char* texto;
        key_t llave;
[1]     llave=ftok(argv[0], 'X');
[2]     if (argc!=2) {printf("Argumentos incorrectos\n"); exit(1);}

[3]     shmid=shmget(llave,100*sizeof(char),IPC_CREAT | 0600);
        if (shmid==-1){perror("Error en shmget");exit(2);}
[4]     texto=shmat(shmid,0,0);

[5]     msqid=msgget(llave, IPC_CREAT | 0600);
        if (msqid==-1){perror("Error en msgget");exit(3);}
        struct{
           long tipo;
           char cadena[tam];
           } mensaje;
        int longitud=sizeof(mensaje)-sizeof(mensaje.tipo);

[6]     if ((pid=fork())==-1) {perror("Error en Fork"); exit(4);}
        if (pid==0){
[7]          msgrcv(msqid, &mensaje,longitud,1,0);
             printf("%s ",mensaje.cadena);
             fflush(stdout);
             strcpy(texto,"correcto.");
[8]          shmdt(texto);
           }
        else {
[9]          sleep(1);
             strcpy(texto,"mal resuelto.");
             printf("%s ",argv[0]);
             fflush(stdout);
             mensaje.tipo=1;
             strcpy(mensaje.cadena,argv[1]);
[10]         msgsnd(msqid, &mensaje,longitud,0);
[11]         par=wait(&estado);
             printf("%s\n",texto);
[12]         msgctl(msqid, IPC_RMID,0);
             shmdt(texto);
             }
     }
```

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

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1.** (*2 p*) Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

I) (*1 p*) En un intérprete de comandos de UNIX la pulsación de las teclas [`control` + `w`] permite borrar todo el contenido de la línea de órdenes.

II) (1 *p*) El superbloque de un sistema de ficheros s5fs de UNIX contiene metadatos del propio sistema de ficheros y una copia de la lista de nodos-i.

**2**. (*2 p*). Dibuje dos esquemas, **adecuadamente rotulados**, uno de la arquitectura y otro de la estructura del sistema operativo UNIX.

**3.** (*2 p*). Explique **razonadamente** la relación existente entre regiones y páginas en el sistema UNIX SVR3

**4.** (*1.5 p*) Explique **razonadamente** cual sería el resultado de la ejecución de las siguientes órdenes desde la línea de comandos ($) de un terminal UNIX:

a) *(0.5 p)* `$ sort < fA > fB`
b) *(0.5 p)* `$ ls -l > asd`
c) *(0.5 p)* `$ ls -i >> wert23`

**5.** *(2.5 p)* Al compilar el código C de este programa se crea el ejecutable `s18`. Supóngase que al invocar un programa desde la línea de órdenes del terminal (`$`) se le asocia el *pid* 1000 y que la asignación de los *pid* de los procesos hijos, si se llegaran a crear, se realizaría incrementando en una unidad el *pid* del proceso padre. Conteste **razonadamente** a los siguientes apartados:

a) *(1 p)* Explicar el significado de las cinco sentencias enumeradas (`[]`) de este programa.

b) (*1.5 p*) Explicar el funcionamiento de `s18` al escribir de forma secuencial las siguientes tres órdenes:

1) `$ ./s18 &`
2) `$ kill -SIGUSR1 1002`
3) `$ kill -SIGUSR2 1002`

**La pregunta continua en la página siguiente**

```
#include <signal.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

void f1(int sig);

int main(void)
{
          int a,b;

[1]       signal(SIGUSR1,SIG_DFL);
[2]       if(fork()==0)
          {

[3]            signal(SIGUSR1,SIG_IGN);
[4]            pause();
[5]            exit(5);
          }
          else
          {
               a=wait(&b);
[6]            printf("\n a=%d b=%d \n",a,b);
          }
exit(0);
}
void f1(int sig)
{
          printf("\nMensaje 1: \n");
}
```

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

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1.** (*2 p*) Conteste razonadamente si las siguientes afirmaciones son verdaderas o falsas:

a) (*1 p*). La llamada al sistema `lseek` tiene tres parámetros de entrada. Uno de ellos indica el número de bytes que se va a desplaza el puntero de lectura/escritura. Por tanto, dicho parámetro solo puede ser un número entero positivo.

b) (*1 p*) Para transferir la abstracción de hebra enteramente al nivel de usuario, debe intervenir el núcleo.

**2.** (*1.5 p*) Describe qué son las variables de condición y para qué se utilizan.

**3.** (*1.5 p*) Conteste razonadamente las siguientes cuestiones:

a) *(0.5 p)* ¿Para qué usa el núcleo el algoritmo `syscall()`?

b) (*1.5 p*) Dibuje un diagrama adecuadamente rotulado que resuma las principales acciones que realiza este algoritmo.

**4.** *(2 p)* Considérese una cache de buffers de bloques en un determinado sistema de ficheros que utiliza una sola partición. Esta cache consta de cuatro colas de dispersión. En un cierto instante de tiempo *t*, esta cache contiene una copia de 12 bloques de disco, cuyos números son: 11, 74, 85, 32, 8, 13, 2, 31, 41, 56, 47, 82. Dibuje un posible esquema adecuadamente rotulado de la cache de buffers de bloques en dicho instante t.

**5.** *(3 p)* Conteste razonadamente a los siguientes apartados:

a) *(1.5 p)* Explicar el significado de las sentencias enumeradas (**[ ]**) del programa que se muestra en la página siguiente.

b) *(1.5 p)* Si el programa anterior se compila produciendo el ejecutable DyASO y se ejecuta con la orden `./DyASO`. Explique detalladamente el funcionamiento del programa así como su salida suponiendo que el primer proceso creado tiene pid=1000, los pids de los procesos se asignan consecutivamente y no se produce ningún error.

**NOTA:** la función `snprintf (buffer_de_salida, tamaño_del_buffer, cadena_de_control, Arg1, … ArgN )` formatea una cadena de texto usando una cadena de control y un conjunto de argumentos del mismo modo que `printf`. Pero, en lugar de escribirlo en la salida estándar, lo hace en el buffer que se le pasa como primer argumento y limitando la salida (si fuese necesario) al tamaño especificado en el segundo argumento.
**NOTA2**: La función `fflush(stdout)` obliga a que la escritura por salida estándar se realice inmediatamente impidiendo condiciones de carrera en la impresión del buffer de salida estándar `stdout`.

**La pregunta continua en la página siguiente**

```
        #include <stdio.h>
        #include <stdlib.h>
        #include <string.h>
        #include <unistd.h>
        #include <sys/msg.h>
[1]     void main(int argc, char *argv[])
        {
        key_t llave;
        struct{ long tipo;
             char cadena[2];
             } mensaje;
        int i, pid, msqid, N, longitud=sizeof(mensaje)-sizeof(mensaje.tipo);
        char Ntexto[10], texto[]="Ejercicio 5 DyASO ";
[2]     if ( (llave=ftok(argv[0],'M'))==-1) {perror("Error ftok"); exit(-1); }
[3]     if ( (msqid=msgget(llave, IPC_CREAT | 0600))==-1 )
                             {perror("Error msgget"); exit(-2); }

        if (argc>2) {printf("Argumentos incorrectos\n"); exit(-3); }
        if (argc==1) {
             N=strlen(texto);
             for(i=0; i<=N; i++) {
                  mensaje.tipo=1;
                  mensaje.cadena[0]=texto[i];
                  mensaje.cadena[1]=0;
[4]               msgsnd(msqid, &mensaje,longitud,0);
                  }
             }
        else      N=atoi(argv[1]);

[5]     if ( (pid=fork())==-1 ) {perror("Error fork"); exit(-4);}
        if (pid==0) {
             if (N>0) {
                  printf("Proceso hijo %d\n",getpid());
                  fflush(stdout);
                  snprintf(Ntexto, 10 , "%d",N-1);
[6]               if(execl(argv[0],argv[0],Ntexto,NULL)==-1)
                        {perror("Error en execl"); exit(-4);}
                  }
             printf("Fin ultimo hijo %d\n",getpid());
             fflush(stdout);
[7]          sleep(1);
             }
        else {
             wait();
[8]          msgrcv(msqid, &mensaje,longitud,1,0);
[9]          printf("%s",mensaje.cadena);
             fflush(stdout);
             if (argc==1) printf("correcto\n");
             }
        }
```

| | |
|---|---|
| Material permitido:<br>**Calculadora NO programable.**<br>Tiempo: **2 horas.**<br>N2 | **Aviso 1:** Todas las respuestas deben estar razonadas.<br>**Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.**<br>**Aviso 3: No use *Tipp-ex*** o similares (atasca el escáner). |

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1.** (*2 p*) Explique razonadamente si las siguientes afirmaciones son verdaderas o falsas:

a) (*1 p*). En UNIX; para visualizar pantalla a pantalla el contenido del fichero `host.conf` situado dentro del directorio `/etc`, se debe teclear la orden

```
$ cat /etc/host.conf
```

b) (*1 p*) Un proceso puede incurrir en un fallo de validez cuando intenta escribir una página cuyo bit *copiar al escribir* esté activado.

**2.** (*1.5 p*) Conteste razonadamente las siguientes cuestiones:

a) (*1 p*) ¿Qué es el superbloque?

b) (*0.5 p*) Explique cómo procede núcleo cuando la lista parcial de nodos-i libres del superbloque se vacía.

**3.** (*1.5 p*) Describa qué es un cerrojo con bucle de espera y explique cómo funciona.

**4.** (*2 p*) Supóngase que el ladrón de páginas de un cierto sistema UNIX debe transferir a un dispositivo de intercambio 50 páginas del proceso A, 30 páginas del proceso B, 40 páginas del proceso C, y 60 páginas del proceso D; y que en una operación de escritura puede transferir 56 páginas al dispositivo de intercambio. Determinar la secuencia de operaciones de intercambio de operaciones que tendría lugar si el ladrón de páginas examina las páginas de los procesos en el orden A, B, C y D.

**5.** *(3 p)* Conteste razonadamente a los siguientes apartados:

a) *(1.5 p)* Explique el significado de las sentencias enumeradas (**[ ]**) del programa mostrado en la página siguiente.

b) *(1.5 p)* El programa es compilado produciendo el fichero ejecutable `prog`. Explique la ejecución del programa su salida si se invoca desde la línea de comandos la orden `./prog DyASO`

**La pregunta continua en la página siguiente**

```
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include <sys/types.h>
    #include <sys/stat.h>
    #include <fcntl.h>
    #include <unistd.h>

    void main(int argc, char *argv[])
    {
    int i, pid, fd;
    char texto[]="OSAyD 5 oicicrejE";
    int tam=strlen(texto)-1;

    if (argc!=2){printf("Argumentos incorrectos\n"); exit(-1);}
[1] unlink(argv[1]);
[2] if (mknod(argv[1],S_IFIFO|0666,0)==-1){perror("Error mknod"); exit(-2);}
[3] if((fd=open(argv[1],O_RDWR))==-1){perror("Error open"); exit(-3);};
    if((pid=fork())==-1) {perror("error en primer fork"); exit(-4);}
    if(pid==0)
         {
         for(i=0;i<=tam;i++)
              {
[4]           if ((pid=fork())==-1) {perror("error en fork 2"); exit(-3);}

              if(pid==0)
                   {
[5]                if(i==tam) {close(fd);break}
                   }

              else
                   {
[6]                wait();
[7]                sleep(1);
                   write(fd,&texto[i],1);
                   if (i==0) write(fd,"\n",1);
                   close(fd);
                   break;
                   }
              }
         }
    else
         {
         close(fd);
[8]      execl("/bin/cat","cat","argv[1]",NULL);
         printf("Incorrecto\n");
         }
    }
```

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

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1.** (*2 p*) Conteste razonadamente si las siguientes afirmaciones son verdaderas o falsas:

a) (*1 p*) Las entradas de un directorio FFS son invariables en cuanto a longitud.

b) (*1 p*) Un proceso en segundo plano puede enviar las salidas al monitor.

**2.** (*1.5 p*) Responda a las siguientes cuestiones razonadamente: a) (*0.5 p*) ¿Qué es un intérprete de comandos? b) (*1 p*) ¿Qué diferentes tipos de intérpretes de comandos existen en función de su forma de invocación?

**3**. (*1.5 p*) Responda razonadamente las siguientes cuestiones: a) (*0.5 p*) ¿Qué son las señales? b) (*1 p*) ¿Cuáles son las principales fuentes de generación de señales?

**4**. (*2 p*). Supóngase que la lista parcial de i-nodos libres del superbloque está vacía, su i-nodo recordado es 750 y su variable índice puede tomar como máximo el valor 5. Además, existen los siguientes i-nodos libres en la tabla de i-nodos: 875, 765, 782, 773, 810, 793, 850 y 825. Dibuje la lista parcial de i-nodos libres del superbloque una vez que ha sido rellenada por el núcleo. ¿Cuál sería ahora el i-nodo recordado?

**5.** *(3 p)* Conteste razonadamente a los siguientes apartados:

a) *(1.5 p)* Explique el significado de las sentencias enumeradas (**[ ]**) del programa mostrado en la página siguiente.

b) *(1.5 p)* El programa es compilado produciendo el fichero ejecutable `programa`.
Explique la ejecución del programa y su salida si se invoca desde la línea de comandos la orden "`./programa correcto`", el SO asigna al proceso creado el pid 1000 y no se produce ningún error.

**Nota:** `fflush` sirve para vaciar el buffer de salida estándar haciendo que la salida de `printf` se imprima inmediatamente.

**La pregunta 5 continúa en la siguiente página**

```
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include <sys/types.h>
    #include <sys/stat.h>
    #include <fcntl.h>
    #include <unistd.h>

    int main(int argc, char *argv[]){
    char buffer, texto[]=" OSAyD 5 oicicrejE";
    int i, pid, tub[2], tam=strlen(texto)-1;
[1] if (argc!=2){printf("Argumentos incorrectos\n"); exit(-1);}
[2] if(pipe(tub)==-1) {perror("Error open:");exit(-2);}
[3] if((pid=fork())==-1) {perror("error en primer fork:"); exit(-3);}
    if(pid==0) {
[4]     printf("Hijo 0, PID=%d\n",getpid()); fflush(stdout);
        close(tub[0]);
        for(i=0;i<=tam;i++) {
          if ((pid=fork())==-1) {perror("error en siguente fork:"); exit(-4);}

              if(pid==0) {
                   printf("Hijo %i, PID=%d\n",i+1,getpid()); fflush(stdout);
[5]                if(i==tam) {close(tub[1]); break;}
                   }
              else{
[6]                wait();
[7]                sleep(1);
[8]                write(tub[1],&texto[i],1);
                   if (i==0) write(tub[1],argv[1],strlen(argv[1]));
                   close(tub[1]);
                   break;}
              }
        }
    else {
         printf("Padre 0, PID=%d\n",getpid()); fflush(stdout);
         close(tub[1]);
[9]      while((read(tub[0],&buffer,1))>0) {
              printf("%c",buffer); fflush(stdout);
              }
         printf(".\n");
         close(tub[0]);
         return 0;
         printf("Incorrecto\n");
         }
    }
```

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

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1.** (*2 p*) Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

I) (*1 p*) En UNIX BSD4.3 la prioridad de planificación de un proceso es un valor comprendido entre 0 y 127. Las prioridades 0 a 63 están reservadas para el núcleo y las restantes para los procesos en modo usuario.

II) (*1 p*) El núcleo del UNIX SVR3 utiliza el algoritmo `attachreg()` para ligar una región al espacio de direcciones virtuales de un proceso.

**2**. *(2 p)* Enumere y explique **brevemente** los atributos de un fichero comúnmente soportados por los sistemas de ficheros de UNIX.

**3.** (*2 p*) Supóngase que el directorio de trabajo actual de un usuario contiene dos ficheros ordinarios (`carta.txt, programa.sh`) y tres subdirectorios (`Escritorio, Descargas, Basura`).

a) (0.5p) ¿Qué orden se debe teclear desde la línea de comandos ($) para conocer la ruta absoluta del directorio de trabajo actual?

b) (0.5p) ¿Qué orden se debe teclear para mover al subdirectorio Basura los ficheros carta.txt y programa.sh?

c) (0.5p) ¿Qué orden se debe teclear para conocer los ficheros que se encuentran en la carpeta Descargas?

d) (0.5p) ¿Qué orden se debe teclear para que el directorio de trabajo actual pase a ser el subdirectorio fotos?

**4.** (*1.5 p*) Explique **brevemente** los principales servicios realizados por el núcleo de UNIX.

**5.** (*2.5 p*) Al compilar el código C del programa que se muestra en la página siguiente se crea el ejecutable `prog`. Supóngase que 1) al invocar `prog` desde la línea de órdenes del terminal (`$`) se le asocia el *pid*=1120. 2) que la asignación de los *pid* se realiza secuencialmente a partir del pid del padre, es decir, *pid*=1121 es el primer hijo creado, *pid=1122* es el segundo hijo creado, etc. 3) El intérprete de comandos desde donde se lanza `prog` tiene asociado el *pid* 1000. Conteste razonadamente a los siguientes apartados:

a) (*1 p*) Explique el significado de las cinco sentencias enumeradas (`[]`) del código.

b) (*1.5 p*) Explique **detalladamente** el funcionamiento de este programa si se invoca desde el intérprete de comandos mediante la orden `$ ./prog 4`

**La pregunta 5 continúa en la siguiente página**

```
    #include <stdio.h>
    #include <stdlib.h>

[1] main(int argc, char *argv[])
        {

            int a,b=0,c,d;

            if (argc!=2)
            {
                exit(3);
            }
            else
            {
[2]             a=atoi(argv[1]);
[3]             while (fork()==0 && b!=a)
                {
[4]                 printf("\nMensaje 1[%d]\n", getpid());
                    b=b+1;
                }
            }
[5]         c=wait(&d);
[6]         printf("\nMensaje 2[%d]=%d\n", getpid(),getppid());
    }
```

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

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1.** Responda a las siguientes cuestiones:

a) *(0.5 p)* Explique el funcionamiento del siguiente comando:

```
grep patrón fichero1 fichero2 … ficheroN
```

b) *(0.5 p)* Explique el funcionamiento de la siguiente llamada al sistema y el significado de sus parámetros:

```
resultado = mount(dispositivo,dir,flags);
```

**2.** *(2 p)* Describa las acciones que realiza el núcleo cuando un proceso realiza la llamada al sistema `shmget`.

**3.** (*2 p*) Dibuje un diagrama, **adecuadamente rotulado**, que esquematice las principales acciones que realiza el núcleo durante la ejecución del algoritmo `exec()`.

**4.** *(2p)* Escriba un programa en C que realice la creación de un conjunto de 3 semáforos asociados a la llave creada a partir del fichero "esllave" y la clave 'J'. El conjunto de semáforos se crea con permisos de lectura y modificación para el usuario.

**5.** *(3 p)* Conteste razonadamente a los siguientes apartados:

a) *(1.5 p)* Explique el significado de las sentencias enumeradas (**[ ]**) del programa que se muestra en la página siguiente.

b) *(1.5 p)* Explique el funcionamiento del programa explicando detalladamente su salida asumiendo que no hubiese otros programas que compitan con él por el uso de CPU.

**La pregunta continua en la página siguiente**

```
      #include <signal.h>
      #include <stdio.h>
      #include <stdlib.h>
      #include <unistd.h>
      #include <time.h>
      #include <sys/times.h>

      void despertar(int sig);
      void continuar(int sig);
      void acabar(int sig);

      void main(void)
      {
           int pid;
[1]        signal(SIGCHLD,despertar);
           signal(SIGCONT,continuar);
           signal(SIGTERM,acabar);
[2]        if ((pid=fork())==-1){
                perror("Error en fork:");
                exit(1);
           }
           if (pid!=0) {
[3]             pause();
                printf("del examen DyASO ");
[4]             while(1);
                printf("erroneo.\n");
[5]             exit(0);
           }
           else{
[6]             sleep(1);
                printf("Ejercicio 5 ");
[7]             kill(getppid(),SIGCONT);
                sleep(1);
           }
      }

      void despertar(int sig)
      {
      struct tms T;
      time_t cpu;
[8]   times(&T);
      printf("correcto");
      cpu=T.tms_utime + T.tms_stime + T.tms_cutime + T.tms_cstime;
      printf(" tras %ld ms de ejecución activa.\n",
                                          (1000*cpu)/sysconf(_SC_CLK_TCK));
[9]   raise(SIGTERM);
      }

      void continuar(int sig) {}
      void acabar(int sig) {exit(0);}
```

| | |
|---|---|
| Material permitido:<br>**Calculadora NO programable.**<br>Tiempo: **2 horas.**<br>N2 | **Aviso 1:** Todas las respuestas deben estar razonadas.<br>**Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.**<br>**Aviso 3: No use *Tipp-ex*** o similares (atasca el escáner). |

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1.** Responda a las siguientes cuestiones:

a) *(0.5 p)*. Explique el resultado de ejecutar la orden:

```
$ cat < /proc/1/status.
```

b) *(0.5 p)*. Explique el funcionamiento del comando:

```
$ fsck [-opciones] [sistema...]
```

**2.** a) *(1.5 p)* Describa qué son las señales, indicando sus fases. b) *(0.5 p)* Indique qué señal se emplea para finalizar un proceso.

**3.** *(2 p)* Describa las principales acciones que realiza el núcleo durante la ejecución del algoritmo **`sleep()`.** ¿Qué parámetros de entrada requiere? ¿Qué ocurre cuando el proceso puede ser interrumpido por señales?

**4.** *(2 p)* Escriba un programa en C que envíe el mensaje "hola" desde un proceso padre a su hijo y a través de una tubería sin nombre.

**5.** *(3 p)* Conteste razonadamente a los siguientes apartados:

a) *(1.5 p)* Explique el significado de las sentencias enumeradas (**[ ]**) del programa mostrado en la página siguiente.

b) *(1.5 p)* El programa es compilado por **root** en el fichero ejecutable `proc` y a continuación establece la máscara de modo de `proc` a 4711. Explique la ejecución del programa y muestre su salida si es ejecutado por el usuario sistemas con `uid`=1000 y `gid`=1000.

**Nota:** Recordamos que el fichero shadow tiene permiso para el usuario root pero no para el resto de usuarios.

**La pregunta continua en la página siguiente**

```
      #include <stdio.h>
      #include <stdlib.h>
      #include <unistd.h>
      #include <sys/types.h>

      int main(int argc,char *argv[])
      {
      uid_t uid, euid, gid, egid;
[1]   uid=getuid();
[2]   euid=geteuid();
[3]   gid=getgid();
[4]   egid=getegid();

[5]   if (euid==0 || egid==0) {
           printf("Tengo acceso al fichero shadow");
           printf(" (uid=%d euid=%d gid=%d egid=%d)\n",uid,euid,gid,egid);
           printf("Y voy a demostrarlo:\n");
[6]        if (fork()==0) {
[7]             if (execl("/bin/cat","cat","/etc/shadow", NULL)!=0){
                      perror("Error en la lectura:");
                }
           }
           else {
[8]            seteuid(uid);
               setegid(gid);
               euid=geteuid();
               egid=getegid();
[9]            wait();
               printf("¿Y el usuario inicial?");
           printf(" (uid=%d euid=%d gid=%d egid=%d):\n",uid,euid,gid,egid);
                if (execl("/bin/cat","cat","/etc/shadow", NULL)!=0) {
                     perror("Error en la lectura:");
                }
           }
      }
      else {
      printf("No tengo acceso al archivo.\n");
      }
      }
```

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

**ESTE EXAMEN CONSTA DE 5 PREGUNTAS**

**1.** (*1.5 p*). Responda a las siguientes cuestiones:

a) (*0.75 p*). Explique el resultado de ejecutar la orden:

```
$ cat < /etc/host.conf
```

b) (*0.75 p*). Explique el funcionamiento de la siguiente llamada al sistema y el significado de sus parámetros.

```
resultado=nice(incremento);
```

**2.** (*1.5 p*). Explique razonadamente si las siguientes afirmaciones son verdaderas o falsas:

a) (*0.75 p*). Un proceso hijo puede acceder a una tubería sin nombre creada por su proceso padre.

b) (*0.75 p*). El núcleo puede acceder directamente a los campos del *área U* del proceso que se está ejecutando pero no al *área U* de otros procesos.

**3.** (*2 p*). Conteste a las siguientes cuestiones:

a) (*0.5 p*). Defina qué son las hebras de usuario.

b) (*1.5 p*). Describa sus ventajas e inconvenientes.

**4.** (*2 p*). Escriba un programa en C que cree una zona de memoria compartida privada de tamaño 1024 bytes, donde solo el usuario va a tener permisos de lectura y escritura.

**Pregunta 5 en la página siguiente**

**5.** (*3 p*) Conteste razonadamente a los siguientes apartados:

a) (*1.5 p*) Explicar el significado de las sentencias enumeradas (**[ ]**) del siguiente programa.

b) (*1.5 p*) El programa es compilado enlazando la librería `lpthread` y creándose el ejecutable `Examen`. Describir el **funcionamiento** así como la **salida** obtenida salida cuando se invoca la orden "`$ ./Examen`". Supóngase que el sistema asigna PIDs a los procesos consecutivamente comenzando con pid=1000 y que no se produce ningún error.

```
        #include <pthread.h>
        #include <stdlib.h>
        #include <stdio.h>
        #include <string.h>

        char cadena1[]="Problema 5", cadena2[]="incorrecto.";

[1]     void *fun1(void *arg)
        {
[2]     wait();
        printf("PidB=%d\n", getpid());
[3]     strcpy(cadena2,"correcto.");
        }

        void main(void)
        {
        int pid;
[4]     if ((pid=fork())==-1){perror("Fallo en fork:"); exit(1);}
        if (pid==0)
             {
[5]          sleep(1);
             strcpy(cadena1,"Ejercicio imposible");
             printf("PidA=%d\n", getpid());
             }
        else
             {
             pthread_t hilo;
[6]          if (pthread_create(&hilo,NULL,fun1,NULL))
                  {
                  printf("Error creando pthread");
                  exit(1);
                  }
[7]          pthread_join(hilo,NULL);
[8]          printf("PidC=%d\n", getpid());
             printf("%s del examen DyASO %s\n",cadena1,cadena2);
             }
        }
```

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

**ESTE EXAMEN CONSTA DE 5 PREGUNTAS**

**1.** (*1.5 p*) Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

I) (*0,75 p*) En el planificador del UNIX BSD4.3 la prioridad de planificación de un proceso es un valor entre 0 y 127. Los procesos en modo usuario tienen las prioridades entre 60 y 127.

II) (*0.75 p*) Una de las principales desventajas de los enlaces simbólicos es que el análisis de las rutas de acceso es más lento que en el caso de los enlaces duros.

**2**. *(2p)* Enumere **brevemente** la información que se almacena en un nodo-i de un sistema de archivos s5fs de UNIX.

**3.** (*2p*) Explique **razonadamente** para qué sirven los siguientes comandos de UNIX:

a) (*0.5 p*) Comando `ps`

b) (*0.5 p*) Comando `top`

c) (*0.5 p*) Comando `jobs`

d) (*0.5 p*) Comando `ls`

**4.** (*1.5 p*) Dibuje un diagrama, **adecuadamente rotulado**, que esquematice las principales acciones que realiza el núcleo durante la ejecución de la rutina asociada a la llamada al sistema `fork()`.

**5.** *(3 p)* Conteste razonadamente a los siguientes apartados:

a) (*1.5p*) Explique el significado de las sentencias enumeradas (`[ ]`) del código que se muestra en la página siguiente.

b) (*1.5p*) Si el programa se compila y el resultado es el ejecutable `prog`. Explique detalladamente el funcionamiento si se invoca desde el intérprete de comandos (`$`) mediante la orden:

```
$ prog DyA_Sistemas_Operativos
```

**La pregunta 5 continua en la página siguiente**

```
    #include <stdlib.h>
    #include <stdio.h>
    #include <string.h>

[1] main(int b, char *c[])
    {
         int pfd[2];
         int a;
         char buf;
         if (b == 2)
         {
[2]           if (pipe(pfd) == -1){
                   fprintf(stderr,"E1\n");
                   exit(-1);
              }
[3]           a=fork();
              if (a == 0) {
[4]                close(pfd[1]);
[5]                while (read(pfd[0], &buf, 1) > 0) write(1, &buf, 1);
                   write(1, "\n", 1);
                   close(pfd[0]);
                   }
              else {
                   printf("\nMensaje:\n");
                   close(pfd[0]);
                   write(pfd[1], c[1],strlen(c[1]));
                   close(pfd[1]);
[6] wait();
              }
         }
         else {
              fprintf(stderr,"E2\n");
              exit(-1);
         }

[7]      exit(0);
    }
```

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

| | |
|---|---|
| Material permitido:<br>**Calculadora NO programable.**<br>Tiempo: **2 horas.**<br>N1 | **Aviso 1:** Todas las respuestas deben estar razonadas.<br>**Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.**<br>**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);
               }
     }
```

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

| | |
|---|---|
| Material permitido:<br>**Calculadora NO programable.**<br>Tiempo: **2 horas.**<br>N2 | **Aviso 1:** Todas las respuestas deben estar razonadas.<br>**Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.**<br>**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*) La actuación de los descriptores de fichero reduce el rendimiento del sistema.
ii. (*1p*) Cuando se produce una interrupción que no esté bloqueada o enmascarada, el núcleo añade una capa de contexto en la pila de capas de contexto de un proceso.

**2**. (*2p*). Enumere y describa las características comunes de los mecanismos IPC del System V.

**3.** (*1p*). Describe el resultado de la siguiente operación:

mutsem=semget(2471, 3, IPC_CREAT | 0666);

**4.** (*2p*)**.** El ladrón de páginas de un cierto sistema UNIX debe transferir a un dispositivo de intercambio 25 páginas del proceso A, 100 páginas del proceso B, 50 páginas del proceso C y 75 páginas del proceso D. En una operación de escritura puede transferir 30 páginas al dispositivo de intercambio.

a) (*1.5 p*). Determinar la secuencia de operaciones de intercambio de páginas que tendría lugar si el ladrón de páginas examina las páginas de los procesos en el orden C, B, D y A.
b) (*0.5 p*). ¿Cuál es el último proceso que transfiere completamente todas sus páginas al dispositivo de intercambio?

**(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 suponiendo que al primer proceso creado al ejecutarlo se le asigna el pid 1000, los pids se asignan en secuencia, no se produce ningún error y los identificadores textuales de las señales de usuario son "User defined signal 1" y "User defined signal 2" respectivamente.

```
     #include <signal.h>
     #include <string.h>
     #include <stdio.h>
     #include <stdlib.h>
     void fun1(int sig)
          {
          printf("%d recibe: %s \n",getpid(),strsignal(sig));
          }
     void main()
          {
          int pid;
          signal(SIGUSR1,fun1);
          signal(SIGUSR2,fun1);
[1]       if ((pid=fork())==-1) {  perror("Error en fork():"); exit(2);}
          if (pid==0)
               {
               printf("Primera ronda\n");
[2]            signal(SIGUSR1,SIG_IGN);
[3]            kill(getppid(),SIGUSR1);
[4]            sleep(1);

               printf("Segunda ronda\n");
[5]            sigblock(sigmask(SIGUSR2));
               signal(SIGUSR1,fun1);
               kill(getppid(),SIGUSR1);
               sleep(1);

               printf("Tercera ronda\n");
[6]            sigsetmask(0);
               sleep(1);
               exit(3);
               }
          else
               pause();
               kill(pid,SIGUSR1);
               kill(pid,SIGUSR2);
[7]            pause();
               kill(pid,SIGUSR1);
               kill(pid,SIGUSR2);
[8]            wait();
               exit(3);
               }
          }
```

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

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1.** (*1 p*). Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

a) (*0.5 p*) `specfs` mantiene un `nodo-s` común por cada proceso que quiere acceder a un dispositivo.

b) (*0.5 p*) En el UNIX SVR3, para tratar un fallo de validez el proceso invoca al manipulador de fallos de validez, que necesita como argumento de entrada la dirección virtual ha provocado dicho fallo.

**2.** (*2 p*). Responda razonadamente las siguientes preguntas:

a) (*1 p*). Indique en detalle el funcionamiento y sintaxis del programa `fsck`.

b) (*1 p*). Señale las inconsistencias que revisa `fsck`.

**3.** (*2 p*). Responde razonadamente las siguientes preguntas:

a) (*1 p*) ¿Qué son las hebras? Señala los tipos de hebras que existen.

b) (*1 p*) Describa qué son las hebras de núcleo.

**4.** (*2 p*) Se tiene un directorio con un único fichero de texto "`disnak.txt`" cuyo contenido es "`manantial`". El propietario del fichero invoca las siguientes acciones en un terminal:

```
$ ln disnak.txt texto.txt
$ ln -s disnak.txt notas.txt
$ rm disnak.txt
```

Responde razonadamente a las siguientes preguntas:

i) (*1 p*) ¿Qué es un enlace duro? ¿Qué es un enlace simbólico?
ii) (*0.5 p*) ¿Cuánto vale el contador de referencias de "`notas.txt`"
iii) (*0.5 p*) Señala la sentencia que hay que invocar para que se imprima el contenido original del fichero `disnak.txt`, ("manantial").

**Pregunta 5 en la página siguiente.**

**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*) El programa es compilado bajo el nombre `Examen` y se invoca la orden "`$ mknod yAS p`" sin producirse ningún error. Describir el funcionamiento así como la salida obtenida salida cuando se invoca a continuación "`$ ./Examen yAS`".

```
       #include <stdio.h>
       #include <string.h>
       #include <stdlib.h>
       #include <sys/types.h>
       #include <sys/stat.h>
       #include <fcntl.h>
       #include <unistd.h>
       #define MAX 256

[1]    void main(int argc, char *argv[])
            {
[2]         if (argc!=2)        {
                 printf("Argumentos incorrectos \n");
                 exit(1);
                 }
            int fd, pid;
            char buffer[MAX];
[3]         if((fd=open(argv[1],O_RDWR))==-1){perror("Error open:"); exit(2);}
[4]         if((pid=fork())==-1) {perror("Error en fork:"); exit(2);}
            if(pid==0)
                 {
[5]              if(read(fd,buffer,4)>0)
                      {
                      printf("%s", buffer);
                      }
                 else
                      {
                      perror("error read:");
                      close(fd);
                      exit(3);
                      }
                 }
            else
                 {
[6]              sleep(2);
                 printf("\n%s D", argv[0]);
                 fflush(stdout); //fuerza impresion imediata
[7]              if(write(fd,argv[1],strlen(argv[1])+1)<0)
                      {
                      perror("error write:");
[8]                   close(fd);
                      exit(4);
                      }
[9]              wait();
                 printf("O.\n\n");
                 }
            close(fd);
            }
```

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

| | |
|---|---|
| Material permitido: **NINGUNO.**<br>Tiempo: **2 horas.**<br>R | **Aviso 1:** Todas las respuestas deben estar razonadas.<br>**Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.**<br>**Aviso 3: No use Tipp-ex** o similares (atasca el escáner). |

**1.** (*1.5 p*). Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:
a) (*0.75 p*) El mayor problema de seguridad en un sistema de ficheros s5fs proviene del superbloque.
b) (*0.75 p*) Una de las principales limitaciones del planificador del UNIX BSD4.3 es que presenta inversión de prioridades.

**2.** (*2 p*) Explique **razonadamente** la estructura de una partición de disco para un sistema de ficheros s5fs.

**3.** (*2 p*) Explique **brevemente** los principales servicios realizados por el núcleo de UNIX.

**4.** (*2 p*). Conteste razonadamente a los siguientes apartados: a) (*0.5 p*) ¿Qué es el ladrón de páginas?
b) (*1.5 p*) Explique los posibles casos que se pueden presentar cuando el ladrón de páginas pretende realizar una transferencia de una página al dispositivo de intercambio.

**5.** *(2.5 p).* Cuando se compila el siguiente código C se crea un ejecutable que se llama `prog`, en el mismo directorio se encuentran los ficheros `fich1.c fich2.c prueba.txt` y el usuario posee los permisos adecuados, conteste razonadamente a los siguientes apartados:
a) (*1 p*) Explicar el significado de las cinco sentencias enumeradas (**[ ]**) de este código.
b) (*1.5 p*) Explicar el funcionamiento de este programa si se invoca desde la línea de órdenes (`$`) de las siguientes formas: 1) `$ ./prog fich1.c fich2.c` 2)`$ ./prog prueba.txt`

```
     #include <fcntl.h>
     #include <stdlib.h>
     #include <stdio.h>
[1]  main(int argc, char *argv[])
     {
          int var1,var2;
          char var3[50];
[2]       if (argc==1||argc>2)
          {
               printf("\nMensaje 1\n");
[3]            exit(0);
          }
          else
          {
               var1=open(argv[1],O_RDONLY);
               if (var1==-1)
               {
[4]                 perror("e1:")
                    printf("\nMensaje 2\n");
[5]                 if (fork()==0)
                    {
                         printf("\nMensaje 3\n");
                         for(;;);
                    }
               }
               else
               {
[6]                 if (read(var1,var3,50)==-1) printf("\nMensaje 4\n");
                    close(var1);
                    exit(1);
               }
          }
     }
```

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

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

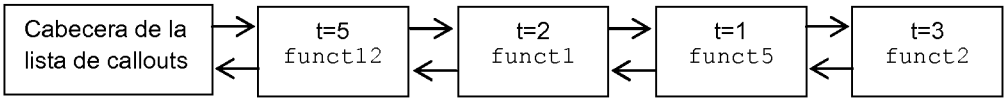
**1**. (*1p*) Señala qué máscara representa un fichero regular donde únicamente el propietario del fichero puede leer, escribir y ejecutar el fichero; y además el bit `S_ISUID` está activado y los bits `S_ISGID` y `S_ISVTX` están activados.

**2**. (*2p*) Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

i) (*1p*) Una tubería sirve para transmitir datos a múltiples procesos de forma simultánea.
ii) (*1p*) Se tiene un computador con una memoria principal de capacidad $C_{MP}$= 16 Mbytes y un tamaño de página $S_P$= 1 Kbyte. El número total de marcos de página es 8192.

**3.** (*2 p*) Conteste **razonadamente** a las siguientes preguntas:
a) (*0.5 p*) ¿Qué es una región de un proceso?
b) (*0.5 p*) ¿Cuáles son las principales regiones en que se descompone un proceso?
c) (*1 p*) ¿Cuál es el contenido y las características de las regiones en que se descompone un proceso?

**4.** (*2p*) En la Figura se muestra la lista de *callouts* del núcleo del UNIX BSD en un cierto instante de tiempo.




Se pide:

a) (*0.5 p*) Explicar brevemente que es un *callout*.
b) (*0.75 p*) Determinar el tiempo de disparo (en tics) de `func1`, `func2`, `func5` y `func12`.
c) (*0.75 p*) Supuesto que ha transcurrido un tic, dibujar la lista de *callout*.

**(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 y describir su salida asumiendo que es ejecutado por el administrador del sistema, en el directorio actual existe un directorio *dir1* y dentro de él un único archivo *prueba.txt*, y además que la raíz del sistema de archivos es de tipo *ext4* y se encuentra alojada en el dispositivo *sda1*.

```
     #include <stdio.h>
     #include <stdlib.h>
     #include <sys/mount.h>
     #include <unistd.h>

     void main(void)
     {
          int res,res2,pid;
[1]       if ((res=mount("/dev/sda1","./dir1","ext4",0,NULL))==0)
               {
[2]            if ((pid=fork())==0)
                    {
                    printf("\nPrimer ls:\n");
[3]                 execl("/bin/ls","ls","dir1",NULL);
[4]                 perror("fallo en execl");}
               else
                    {
[5]                 wait();
[6]                 sleep(1);
[7]                 if (umount("./dir1")==1) perror("fallo en umount");
                    printf("\nSegundo ls:\n");
[8]                 system("ls dir1");
                    printf("Fin.\n");
                    }
               }
          else
               {
               perror("Error en mount");
               }
     }
```

| | |
|---|---|
| Material permitido:<br>**Calculadora NO programable.**<br>Tiempo: **2 horas.**<br>N2 | **Aviso 1:** Todas las respuestas deben estar razonadas.<br>**Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.**<br>**Aviso 3: No use *Tipp-ex*** o similares (atasca el escáner). |

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1**. (*1p*). Señala cuál de las siguientes afirmaciones es verdadera y **razona** la respuesta. El algoritmo de planificación usado en BSD4.3:

a) Permite que un proceso de baja prioridad quede abandonado mientras existan procesos de mayor prioridad.

b) Establece la prioridad de un proceso mediante la llamada al sistema `priocntl`.

c) Permite que cualquier proceso aumente su prioridad mediante el uso de la llamada al sistema `nice`.

d) Es estrictamente no expropiable lo que hace que pueda aparecer inversión de prioridades.

**2**. (*2p*) Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

i) (*1p*) Si en el directorio actual existe un archivo `uno.txt` con número de nodo-i 4096 que posee una sola referencia y se invocan consecutivamente los comandos "`ln uno.txt dos.txt`" y "`ln -s uno.txt tres.txt`"; el contador de referencias del archivo `tres.txt` vale 1.

ii) (*1p*) Una mejora de FFS respecto a s5fs es que Intenta asignar bloques secuenciales de un fichero en posiciones rotacionalmente óptimas.

**3.** (*2p*) Conteste a las siguientes cuestiones:
a) (*0.5 p*) Explique **brevemente** el funcionamiento de un intérprete de comandos de UNIX.
b) (*1.5 p*) Enumere y explique los tipos de órdenes que de forma general se pueden ejecutar en un intérprete de comandos.

**4**. (*2p*) En un sistema UNIX la máscara de modo del fichero ordinario resultados es 6644.
a) (*1p*) Explique razonadamente el significado de todos los bits de la máscara de modo.
b) (*0.5 p*) Escriba la máscara de modo simbólica asociada a este fichero.
c) (*0.5 p*) ¿Qué orden se debe teclear desde la línea de comandos (`$`) para que la máscara simbólica del fichero pase a ser `-rwx r-- --T`?

**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 ejecutable `prog` resultante de compilar el código siguiente cuando se invoca desde la ventana de comandos: `./prog SO`.

```
     #include <stdio.h>
     #include <string.h>
     #include <stdlib.h>
     #include <sys/types.h>
     #include <unistd.h>
     #define MAX 256
[1]  void main(int argc,char *args[])
          {
          int tuberia[2];
          int pid;
          char mensaje[MAX];
          if (argc!=2)
               {printf("Argumentos incorrectos\n");
               exit(-1);}
[2]       if (pipe(tuberia)==-1)
               {
[3]            perror("Error en pipe");
[4]            exit(-1);
               }
[5]       if ((pid=fork())==-1)
               {
               perror("Error en fork");
               exit(-1);
               }
          else if (pid==0)
               {
[6]            if (read(tuberia[0], mensaje, MAX)==0)
                    {perror("Error de lectura"); exit(-2);}
               printf("Examen %s ",mensaje);
[7]            if (write(tuberia[1],args[1],strlen(args[1])+1)==0)
                    {perror("Error de escritura"); exit(-2);}
[8]            close(tuberia[0]);
               close(tuberia[1]);
               exit(0);
               }
          else
               {
               if (write(tuberia[1],"DyA",4)==0)
                    {perror("Error de escritura"); exit(-2);}
[9]            wait();
               if (read(tuberia[0], mensaje,MAX)==0)
                    {perror("Error de lectura"); exit(-2);}
               printf("%s\n",mensaje);
               close(tuberia[0]);
               close(tuberia[1]);
               exit(0);
               }
          }
```

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

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1**. *(1p)* Describa el resultado de la orden `sem=semget(llave, 3, IPC_CREAT | 0600);`

**2**. *(2p)* Explique brevemente para qué usa el núcleo la llamada al sistema `wait` y escriba un pseudocódigo asociado con las principales acciones.

**3**. *(2p)* Cuando se produce una interrupción, ¿qué algoritmo invoca el núcleo para el tratamiento de las interrupciones? Describa las principales acciones que realiza dicho algoritmo.

**4** *(2p)* Conteste razonadamente a los siguientes apartados:
a) *(1p)* ¿Qué concepto se implementa a partir de las siguientes líneas de código? Explica para qué sirve, su funcionamiento y las principales ventajas.

```
void spin_lock (spinlock_t *s)
      {
      while (test_and_set(s)!=0)
      }
void spin_unlock (spinlock_t *s)
      {
      *s=0;
      }
```

b) *(1p)* Describe el comportamiento de las funciones que son invocadas en el código anterior.

**(PREGUNTA 5 EN LA SIGUIENTE PÁGINA)**

**5.** *(3p)* Conteste razonadamente a los siguientes apartados:
a) *(2p)* Explicar el significado de las sentencias enumeradas (**[ ]**) de este programa.
b) *(1p)* Si el programa es compilado bajo el nombre DyASO y describir su salida cuando se invoca de la siguiente forma "`./DyASO hola`". Suponga que los PIDs de los procesos se asignan de forma consecutiva comenzando desde 1000.

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

[1]     void main(int n, char* arg[])
        {
[2]     if (n!=2) printf("Argumentos incorrectos");
        else {
             int pid, shmid, par, estado;
             char *cadena;
             key_t llave;
[3]          llave=ftok(arg[0], 'M');
[4]          shmid=shmget(llave,50*sizeof(char),IPC_CREAT | 0600);
             if (shmid==-1)
                  {
[5]               perror("fallo en shmget");
                  exit(1)
                  };
[6]          cadena=shmat(shmid,0,0);
[7]          if ((pid=fork())==-1) exit(2);
                  if (pid==0)
                  {
                       sleep(2);
[8]                    strcpy(cadena,arg[1]);
[9]                    shmdt(cadena);
                       exit(3);
                  }
                  else
                  {
[10]                   par=wait(estado);
                       printf("El proceso %d dice: %s DyASO\n",pid,cadena);
                  }
             }
        }
```

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

## ESTE EXÁMEN CONSTA DE 5 PREGUNTAS

**1**. Explique razonadamente si las siguientes afirmaciones son verdaderas o falsas.

i) *(1 p)* La ejecución de los procesos en un sistema UNIX está dividida en dos modos de ejecución: un modo de mayor privilegio denominado *modo núcleo o supervisor* y otro modo de menor privilegio denominado *modo usuario*.

ii) *(1 p)* Las interrupciones son atendidas en modo usuario dentro del contexto del proceso que se encuentra actualmente en ejecución, aunque dicha interrupción no tenga nada que ver con la ejecución de dicho proceso.

**2.** *(1.5 p)* Explique las principales ventajas y desventajas que presentan las tuberías FIFO frente a las tuberías sin nombre.

**3**. *(2 p)* Explique qué es un sistema de ficheros transaccional

**4.** (*2 p*) El ladrón de páginas de un cierto sistema UNIX debe transferir a un dispositivo de intercambio 25 páginas del proceso A, 100 páginas del proceso B, 50 páginas del proceso C y 75 páginas del proceso D. En una operación de escritura puede transferir 30 páginas al dispositivo de intercambio. Determinar la secuencia de operaciones de intercambio de páginas que tendría lugar si el ladrón de páginas examina las páginas de los procesos en el orden C, B, D y A.

**Quinta pregunta en la página siguiente.**

**5.** Conteste razonadamente a los siguientes apartados:

a) (*1 p*) Explicar el significado de las sentencias enumeradas (**[ ]**) de este programa.

b) (*1.5 p*) Explicar el funcionamiento del programa mostrando la salida en pantalla.

```
     #include <sys/ipc.h>
     #include <sys/msg.h>
     #include <string.h>
     #include <stdio.h>
     #include <stdlib.h>
     void main(void)
     {
          int pid, msqid;
          struct{
               long tipo;
               char cadena[50];
               } mensaje;
          int longitud=sizeof(mensaje)-sizeof(mensaje.tipo);
          key_t llave;
[1]       creat("./archivo",0777);
[2]       llave=ftok("archivo", 'M');
[3]       msqid=msgget(llave, IPC_CREAT | 0600);
          if (msqid==-1){exit(1);}
[4]       if ((pid=fork())==-1)
               {
[5]                 exit(1);
               }
               else if (pid==0)
               {
[6]                 mensaje.tipo=2;
                    strcpy(mensaje.cadena,"dos");
[7]                 msgsnd(msqid, &mensaje,longitud,0);
                    mensaje.tipo=1;
                    strcpy(mensaje.cadena,"uno");
                    msgsnd(msqid, &mensaje,longitud,0);
               }
               else
               {
[8]                 msgrcv(msqid, &mensaje,longitud,1,0);
                    printf("mensaje %s \n",mensaje.cadena);
                    msgrcv(msqid, &mensaje,longitud,2,0);
                    printf("mensaje %s \n",mensaje.cadena);
               }
     }
```

| | |
|---|---|
| Material permitido: **NINGUNO.**<br>Tiempo: **2 horas.**<br>N2 | **Aviso 1:** Todas las respuestas deben estar razonadas.<br>**Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.**<br>**Aviso 3: No use Tipp-ex** o similares (atasca el escáner). |

**1**. Explique razonadamente si las siguientes afirmaciones son verdaderas o falsas:

i) *(1 p)* Un *proceso ejecutándose en modo usuario* puede acceder a otras partes de su propio espacio de direcciones, como aquellas reservadas para estructuras de datos asociadas al proceso usadas por el núcleo.

ii) *(1 p)* El núcleo de UNIX realiza la invocación del algoritmo `wakeup()` únicamente dentro de los algoritmos asociados a las llamadas al sistema.

**2.** *(1.5 p)* Describa las principales limitaciones que presentan las tuberías.

**3** *(2 p)* Explique qué diferencia existe entre una copia de seguridad y un *snapshot* en un sistema que implementa *copy-on-write*

**4.** *(2 p)* Explique razonadamente el significado de la siguiente llamada al sistema `y=times(&x);`

**5.** Conteste razonadamente a los siguientes apartados:

a) *(1 p)* Explicar el significado de las sentencias enumeradas (**[ ]**) de este programa.

b) *(1.5 p)* Explicar el funcionamiento del programa mostrando la salida en pantalla.

```
     #include <signal.h>
     #include <stdio.h>
     #include <math.h>

     long t1;

     void fun1(int sig);
     void fun2(int sig);

     void main(void)
     {
[1]        signal(SIGVTALRM, fun1);
           signal(SIGALRM, fun2);
           printf("Comienza:\n");
           time(&t1);
[2]        alarm(3);
[3]        pause();
[4]        kill(getpid(),SIGVTALRM);
           printf("Acaba.\n");
     }

     void fun1(int sig)
     {
           long t2;
[5]        time(&t2);
[6]        printf("fun1:Han pasado %d segundos\n",(int)(t2-t1));
           kill(getpid(),SIGTERM);
     }
     void fun2(int sig)
     {
           long t2;
           time(&t2);
           printf("fun2:Han pasado %d segundos\n",(int)(t2-t1));
           kill(getpid(),SIGTERM);
     }
```

| | |
|---|---|
| Material permitido: **NINGUNO.**<br>Tiempo: **2 horas.**<br>N | **Aviso 1:** Todas las respuestas deben estar razonadas.<br>**Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.**<br>**Aviso 3: No use *Tipp-ex*** o similares (atasca el escáner). |

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1.** (*2 p*) Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

a) (*1 p*) La llamada al sistema `msgrcv` permite que un proceso pueda enviar un mensaje.

b) (*1 p*) Un nodo-i se modifica solamente cuando se cambia el contenido de un archivo.

**2.** *(1.5 p)* Describe las principales limitaciones que presentan las señales.

**3.** (*1.5 p*) Supóngase la siguiente orden tecleada por un usuario desde el intérprete de comandos (`$`) de un sistema UNIX:

```
$ man 2 mount
```

a) (*0.75 p*) Explique **razonadamente** si dicha orden está bien escrita o contiene algún error de sintaxis.

b) (*0.75 p*) Si considera que la orden está bien escrita explique su significado. Por el contrario, si considera que está mal escrita, indique cómo se debería escribir correctamente.

**4**. (*2.5 p*) Supóngase un computador con una memoria principal de capacidad $C_{Mp}$=4 MiB y un tamaño de página $S_P$= *4* KiB. Calcular el contenido en binario y en decimal de cada uno de los campos en que se descompondría la dirección física $DIR_F$=*2020220* expresada en decimal. Suponer que cada posición de memoria contiene una palabra y que esta tiene un tamaño de 1 byte.

Ayuda: 1 MiB=$2^{20}$ bytes, 1 KiB=$2^{10}$ bytes.

(**PREGUNTA 5 EN LA SIGUIENTE PÁGINA**)

**5.** (*2.5 p*) Conteste razonadamente a los siguientes apartados:
a) (*1 p*) Explicar el significado de las sentencias enumeradas (**[ ]**) de este programa destacando la diferencia entre **[2]** y **[3]**.
b) (*1.5 p*) El programa una vez compilado se convierte en el ejecutable `pr1`. Explicar el funcionamiento del programa cuando se invoca desde la línea de comandos (`$`) de la siguiente manera: `$ ./pr1 archivo1 archivo2`.

```
     #include <fcntl.h>
     #include <stdio.h>
     #include <stdlib.h>
     void main(int np, char* a[])
     {
     int fd1, fd2;
     FILE* fd3;
     char buf[1], mensaje[]="nemaxe";

     if(np==3)
     {
[1]  fd1=creat(a[1],0600);
[2]  fd2=open(a[1],O_RDONLY);
[3]  fd3=fopen(a[2],"w");

[4]  if (fd1==-1 || fd2==-1 || fd3==NULL)
           {printf("Error creando los ficheros");}
     else
           {
[5]              write(fd1,mensaje,6);
                 close(fd1);

                 lseek(fd2,0,SEEK_END);

                 while(0<=lseek(fd2,-1,SEEK_CUR))
                 {
[6]                    read(fd2,buf,1);
[7]                    lseek(fd2,-1,SEEK_CUR);
                       fwrite(buf,1,1,fd3);
                 }

           }
[8]  close(fd2);
     fclose(fd3);

     }
     else {printf("Argumentos incorrectos");}

     }
```

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

## ESTE EXAMEN CONSTA DE 5 PREGUNTAS

**1.** (*2 p*) Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

a) (*1p*) En el UNIX SVR3 el único caso en que se puede producir un fallo de protección es cuando un proceso intenta acceder a una página cuyos bits de protección no permiten acceder a la página.

b) (*1p*) El `nodo-im` permite el acceso rápido a la información del archivo.

**2.** (*1.5 p*) Explica qué son los *shell scripts.*

**3.** (*2 p*) Explique **razonadamente** qué es la interfaz *nodo-v/sfv* de UNIX y qué ventajas proporciona su uso.

**4.** (*1.5 p*) Describa el funcionamiento del siguiente programa escrito en C:

```
     #include <stdio.h>
     #include<unistd.h>
     main()
     {
[1]  if (fork()==0)
           {
[2]        execl("/bin/date","date",NULL);
           }
[3]        printf("\nFinalizar\n");
     }
```

**(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 y mostrar la salida en pantalla asumiendo que la llamada al sistema `fork()` devuelve 0 y 4141.

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

     void main(void)
     {
          int pid, shmid,par, estado;
          char *cadena;
          key_t llave;
[1]       if (creat("./archivo",0600)==-1) exit(1);
[2]       llave=ftok("archivo", 'M');
[3]       shmid=shmget(llave,50*sizeof(char),IPC_CREAT | 0600);
          if (shmid==-1) exit(1);
[4]       cadena=shmat(shmid,0,0);
[5]       if ((pid=fork())==-1) exit(2);
          if (pid==0)
          {
               sleep(2);//espero un poco antes de escribir
               strcpy(cadena,"hola");
[6]            shmdt(cadena);
               exit(3);
          }
          else
          {
[7]            par=wait(estado);
               printf("El proceso %d dice %s \n",pid,cadena);
          }
     }
```