Material permitido: **Ninguno**

Tiempo: **120 minutos**

N1

Aviso 1: Todas las respuestas deben estar debidamente razonadas.

Aviso 2: Escriba con buena letra y evite los tachones.

Aviso 3: Solución del examen y fecha de revisión en <https://www.dia.uned.es/71023016/>

1. Conteste **razonadamente** a las siguientes apartados:

- a) (1 p) Explicar razonadamente la planificación en BSD 4.3.
- b) (1 p) Explicar el funcionamiento del contador de referencias de un nodo-v en los SOBUNIX. Enumerar las circunstancias en que se incrementa o se decrementa este contador.
- c) (1 p) Enumerar y explicar brevemente las estructuras de datos que utiliza Linux para gestionar la memoria principal.
- d) (1 p) Enumerar las características principales del sistema operativo MS-DOS.

2. (2 p) Explicar **razonadamente** que información es necesario mantener por cada hilo del núcleo en los SOBUNIX ¿Quién gestiona dicha información? ¿Dónde se almacena?

3. (2 p) Explique **razonadamente** cuál sería el resultado de la ejecución de las siguientes órdenes:

- a) `ps -L`
- b) `pmap 1564`
- c) `sort < T1 > T2`
- d) `chmod 4666 temp`
- e) `PATH=$PATH:.`

Material permitido: **Ninguno**

Tiempo: **120 minutos**

N1

Aviso 1: Todas las respuestas deben estar debidamente razonadas.

Aviso 2: Escriba con buena letra y evite los tachones.

Aviso 3: Solución del examen y fecha de revisión en <https://www.dia.uned.es/71023016/>

4. (2 p) La máscara de modo simbólica del archivo ejecutable `e2025` que resulta de compilar el programa que se muestra en la Figura 1 es `-rwsrwxrwx` y dicho archivo pertenece al usuario `Paco` (UID=720, GID=103). El archivo `fichero1.txt` pertenece también al usuario `Paco` y su máscara de simbólica es `-rw-----`. El archivo `fichero2.txt` pertenece al usuario `David` (UID=725, GID=105) y su máscara de simbólica es `-rw-----`. Explique **razonadamente** el funcionamiento de este programa e indique los mensajes que aparecen en la salida estándar en los siguientes casos: a) El programa lo ejecuta `Paco`. b) El programa lo ejecuta el usuario `David`.

```
#include <fcntl.h>
#include <stdio.h>
main()
{
    int x,y,fd1,fd2,fd3,fd4;

    x=getuid();
    y=geteuid();

    printf("\n[1] S1= %d, S2= %d",x,y);

    fd1=open("fichero1.txt", O_RDONLY);
    fd2=open("fichero2.txt", O_RDONLY);
    printf("\n[2] fd1= %d, fd2= %d", fd1, fd2);

    setuid(x);
    printf("\n[3] S1= %d, S2= %d",getuid(),geteuid());

    fd3=open("fichero1.txt", O_RDONLY);
    fd4=open("fichero2.txt", O_RDONLY);
    printf("\n[4] fd3= %d, fd4= %d", fd3, fd4);

    setuid(y);
    printf("\n[5] S1= %d, S2= %d\n",getuid(),geteuid());
}
```

Figura 1

AMPLIACIÓN DE SISTEMAS OPERATIVOS (Cód. 71023016)

Solución Examen Enero 2025

Solución Ejercicio 1

- a) BSD 4.3 es un sistema operativo de núcleo no expropiable que utiliza como unidad de planificación a los *procesos*, ya que el núcleo no soporta hilos del núcleo. Cada proceso tiene asignada una *prioridad* de ejecución que es un número entero positivo comprendido dentro del rango de valores $[0,127]$, siendo 0 la prioridad más alta posible y 127 la prioridad más baja. La prioridad de un proceso ejecutándose en *modo usuario* solo puede encontrarse dentro del rango $[50, 127]$. Los valores comprendidos dentro del rango $[0, 49]$ están reservados para la ejecución del núcleo.

En modo núcleo la prioridad de un proceso no se recalcula en función del tiempo sino que se asigna de forma directa. Un proceso cuando invoca a una llamada al sistema y pasa a ejecutarse en modo núcleo continua ejecutándose en principio con la misma prioridad que tenía en modo usuario hasta que entra en el estado dormido (bloqueado) en espera de que se produzca algún evento. Cuando el núcleo despierta a un proceso en el estado dormido porque se ha producido el evento por el que se había bloqueado, el núcleo aumenta temporalmente su prioridad asignándole un determinado valor dentro del rango $[0, 49]$ en función del evento por el que había entrado en el estado dormido.

En BSD 4.3 existen 32 colas de procesos en el estado preparado para ejecución. Cada cola se implementa como una lista doblemente enlazada de estructuras `proc`. Cada cola i ($i = 0, 1, 2, \dots, 31$) está asociada al rango de prioridades $[4 \cdot i, 4 \cdot i + 3]$ formado por cuatro valores contiguos de prioridad. Así la cola 0 está asociada al rango de prioridades $[0, 3]$, la cola 1 al rango $[4, 7]$, etc. Solamente pueden formar parte de una determinada cola aquellos procesos en el estado preparado para ejecución cuya prioridad esté dentro del rango de prioridades asociada a dicha cola.

Siempre se planifica para ejecución el proceso situado más cerca de la cabecera de la cola i asociada al rango de prioridades más altas. Luego un proceso ubicado en una cola i se ejecutaría antes que uno situado en la cola $i + 1$.

- b) Un *nodo virtual* o nodo-v es una estructura de datos que representa a un archivo activo. El núcleo asigna un nodo-v cuando se crea o se abre un archivo durante el tratamiento de las llamadas al sistema `create` u `open`. El descriptor del archivo tiene un puntero al objeto de archivo abierto, el cual tiene a su vez un puntero al nodo-v asociado al archivo.

Nótese que si dos procesos A y B abren el mismo archivo (o si el mismo proceso abre dos veces el mismo archivo) existirán dos descriptors de archivos y dos estructuras de objeto abierto distintas, pero solo una estructura nodo-v. El nodo-v mantiene un contador de referencias con el número de estructuras de datos que lo referencian. En este caso el contador de referencias contendrá el valor 2. Si un proceso ejecuta una llamada al sistema `close` sobre dicho archivo el contador de referencias se decrementará en una unidad.

En general cada vez que el núcleo referencia, a través de alguna de sus estructuras de datos, al nodo-v de un archivo se incrementa en una unidad el contador de referencias del nodo-v. Por el contrario, cuando el núcleo elimina una referencia al nodo-v el contador de referencias se decrementa en una unidad. Solo cuando el contador de referencias de un nodo-v toma el valor 0 entonces el nodo-v puede ser desasignado (liberado) y las estructuras de datos (nodo-i) y bloques de datos del archivo cargadas en memoria principal pueden ser eliminadas.

Aparte de la apertura de un archivo existen otras circunstancias en las que es necesario asignar un nuevo nodo-v o referenciar a un nodo-v ya creado, y en consecuencia incrementar su contador de referencias:

- Un proceso mantiene una referencia al nodo-v de su directorio de trabajo actual. Cuando un proceso cambia su directorio de trabajo tiene que referenciar al nodo-v del nuevo directorio de trabajo y eliminar la referencia al nodo-v del directorio de trabajo antiguo.
- Cuando se monta un nuevo sistema de archivos se crea una referencia al nodo-v del directorio utilizado como punto de montaje.
- Cuando el núcleo analiza un nombre de ruta absoluta o relativa introducido como argumento de alguna llamada al sistema o comando tiene que referenciar al nodo-v de cada directorio intermedio que aparece en la ruta. Dicha referencia se crea para poder buscar en un directorio. Cuando termina la búsqueda en dicho directorio la referencia a su nodo-v se elimina.

c) Linux divide la memoria principal en *nodos de memoria*. El tiempo de acceso a una posición de memoria de un determinado nodo es siempre el mismo.

Por otra parte, Linux considera las particularidades de direccionamiento (tamaño de registros, líneas de direccionamiento para DMA, etc) de cada arquitectura hardware descomponiendo cada nodo de memoria en *zonas*, que son regiones contiguas de memoria principal.

El núcleo de Linux mantiene un *descriptor de página* por cada marco de página *j* existente. Cada descriptor de página se implementa mediante una estructura de tipo `page` de 32 bytes que contiene distintos datos sobre un marco.

Todos los descriptores de página de un mismo nodo se organizan en un *array de descriptores de página* denominado `mem_map`.

El núcleo mantiene también un *descriptor de zona* por cada zona existente en un nodo. Cada descriptor de zona se implementa mediante una estructura de tipo `zone` que contiene datos sobre una zona.

Finalmente, el núcleo mantiene un *descriptor de nodo* por cada nodo de memoria existente. Cada descriptor de nodo se implementa mediante una estructura de tipo `pg_data_t` que contiene datos sobre un nodo.

d) Las características principales del sistema operativo MS-DOS son:

- Es un sistema monousuario, es decir, solo puede atender a un único usuario simultáneamente.
- Es un sistema monotarea, es decir, solo se puede ejecutar un proceso y hasta que no termina su ejecución no se puede ejecutar otro. Contrariamente a la creencia general MS-DOS no es un sistema monoprogramado ya que en la memoria principal pueden estar cargados varios procesos.
- El núcleo está escrito en lenguaje ensamblador.
- El núcleo posee una estructura de tipo monolítica o simple.
- No implementa un planificador de procesos. Los procesos son ejecutados en orden de creación y hasta que no termina un proceso no se puede ejecutar otro.
- Administra la memoria principal mediante la técnica de particionamiento dinámico.
- Considera a los archivos como una secuencia de bits y delegan en las aplicaciones la interpretación de los mismos.
- Soporta tanto el acceso secuencial como el acceso aleatorio a los archivos.
- En los nombres de archivos no distingue entre minúsculas y mayúsculas.
- Cada sistema de archivos presente en memoria secundaria es tratado como una unidad o volumen lógico independiente, cada una de los cuales posee su propia estructura de directorios. MS-DOS asigna una letra (C, D, E,...) a cada unidad.

- Dentro de cada volumen soporta una estructura de directorios de árbol de directorios. El directorio raíz de un volumen se denota por el carácter ' \ ', que además se usa para separar los componentes del nombre de ruta de un archivo. Para acceder al sistema de archivos de otra unidad se debe especificar al comienzo de la ruta la letra de la unidad seguido de dos puntos, por ejemplo: C : \.
- En sus primeras versiones únicamente soportaba sistemas de archivos FAT. Aunque en sus últimas versiones también pasó a soportar sistemas de archivos de discos ópticos (CD-ROM) como el ISO 9660.
- La interfaz con el usuario es del tipo línea de comandos.

Solución Ejercicio 2

En los SOBUNIX que soportan hilos del núcleo el sistema operativo debe mantener en su espacio de direcciones una estructura de datos y una pila del núcleo por cada hilo del núcleo. La estructura de datos asociada a un hilo del núcleo contiene la siguiente información:

- *Identificador del hilo núcleo.* Es un número entero positivo que identifica de manera unívoca a cada hilo del núcleo.
- *Contexto hardware salvado.* Son los valores que contenían los registros del computador cuando se interrumpió la ejecución del hilo del núcleo debido a un cambio de contexto. El contexto hardware será restaurado cuando el hilo del núcleo vuelva a ser planificado y se continúe con la ejecución del hilo.
- *Parámetros de planificación.* Es la información que utiliza el planificador del núcleo para decidir qué hilo del núcleo planificar, como por ejemplo, la prioridad de planificación del hilo o el tiempo de uso del procesador.
- *Punteros para implementar diferentes colas de hilos del núcleo.* Como colas de hilos preparados para ejecución, colas de hilos dormidos, etc.
- *Puntero a la pila del núcleo asociada al hilo.*
- *Información sobre el proceso ligero asociado.* Como por ejemplo un puntero a la estructura `lwp` del proceso ligero al que está asociado y un puntero a la estructura `proc` del proceso del que forma parte el proceso ligero. Si el hilo del núcleo no está asociado a ningún proceso ligero, entonces estos punteros contienen valores `NULL`.

Por su parte, una *pila del núcleo* es una estructura que contiene los marcos de pila de las funciones invocadas por el hilo durante su ejecución en modo núcleo. Esta pila está vacía cuando el hilo del núcleo se ejecuta en modo usuario.

Solución Ejercicio 3

- a) Muestra un listado con los procesos asociados al intérprete de comandos incluyendo información sobre los procesos ligeros (-L) existentes en cada proceso.
- b) Muestra información (permisos de acceso, tamaño y dirección de inicio de las regiones) del espacio de direcciones virtuales del proceso con PID=1564.
- c) Ordena alfabéticamente el contenido del archivo T1 y lo guarda en el archivo T2.
- d) Cambia la máscara de modo del archivo temp al valor 4666, es decir, establece permisos de lectura y escritura para el propietario del archivo, el grupo de usuarios propietario del archivo y el resto de usuarios. Además activa el bit -S_ISUID.
- e) Incluye el directorio de trabajo actual en la variable de entorno PATH.

Solución Ejercicio 4

- a) El usuario Paco (UID=720) es el propietario del archivo e2025. De acuerdo con su máscara de modo simbólica el propietario del archivo puede ejecutar el archivo. Al proceso A que se crea en el sistema asociado a la ejecución de e2025 se le asigna UID=720 y EUID=720.

Al ejecutar el proceso A en primer lugar se invoca a la llamada al sistema `getuid` para asignar el valor del UID del proceso a la variable `x`. Luego se invoca a la llamada al sistema `geteuid` para asignar el valor del EUID del proceso a la variable `y`. A continuación se imprime en la salida estándar, por defecto la pantalla, el mensaje

```
[1] S1= 720, S2= 720
```

Posteriormente se invoca a la llamada al sistema `open` para abrir con permiso de solo lectura el archivo `fichero1.txt`. La máscara de modo de este archivo únicamente permite leerlo al propietario del mismo, que es el usuario Paco cuyo UID=720. Puesto que el proceso A tiene un EUID=720 que coincide con el UID del propietario del archivo la llamada al sistema se ejecuta con éxito y en `fd1` se almacena el descriptor del archivo.

Luego se vuelve a invocar a la llamada al sistema `open` para abrir con permiso de solo lectura el archivo `fichero2.txt`. La máscara de modo de este archivo únicamente permite leerlo al propietario del mismo, que es el usuario David cuyo UID=725. Puesto que el proceso A tiene un EUID=720 que es distinto del UID del propietario del archivo la llamada al sistema devuelve un error y en `fd2` se almacena el valor `-1`. A continuación se imprime en la salida estándar el mensaje

```
[2] fd1= 3, fd2= -1
```

Posteriormente se invoca a la llamada al sistema `setuid` para asignar al UID y al EUID del proceso A el valor de `x` (720). Como el EUID del proceso A no es el del superusuario y el valor de `x` coincide con el valor del UID del proceso A entonces solamente el EUID del proceso A se configura con el valor de `x`. En resumen el resultado de esta llamada al sistema es configurar el EUID del proceso A al valor EUID=720, valor que por otra parte ya tenía sin necesidad de haber ejecutado esta llamada al sistema. A continuación se imprime en la salida estándar el mensaje

```
[3] S1= 720, S2= 720
```

Luego se invoca a la llamada al sistema `open` para abrir con permiso de solo lectura el archivo `fichero1.txt`. La máscara de modo de este archivo únicamente permite leerlo al propietario del mismo, que es el usuario `Paco` cuyo `UID=720`. Puesto que el proceso A tiene un `EUID=720` que coincide con el `UID` del propietario del archivo la llamada al sistema se ejecuta con éxito y en `fd3` se almacena el descriptor del archivo.

Luego se vuelve a invocar a la llamada al sistema `open` para abrir con permiso de solo lectura el archivo `fichero2.txt`. La máscara de modo de este archivo únicamente permite leerlo al propietario del mismo, que es el usuario `Davis` cuyo `UID=725`. Puesto que el proceso A tiene un `EUID=720` que es distinto del `UID` del propietario del archivo la llamada al sistema devuelve un error y en `fd4` se almacena el valor `-1`. Acto seguido se imprime en la salida estándar, el mensaje

```
[4] fd3= 4, fd4= -1
```

Posteriormente se invoca de nuevo a la llamada al sistema `setuid` para asignar al `UID` y al `EUID` del proceso A el valor de `y` (`720`). Como el `EUID` del proceso A no es el del superusuario y el valor de `y` coincide con el valor del `UID` del proceso A entonces solamente el `EUID` del proceso A se configura con el valor de `y`. En resumen el resultado de esta llamada al sistema es configurar el `EUID` del proceso A al valor `EUID=720`, valor que por otra parte ya tenía sin necesidad de haber ejecutado esta llamada al sistema. Finalmente se imprime en la salida estándar, el mensaje

```
[5] S1= 720, S2= 720
```

- b) Como la máscara de modo simbólica del archivo `e2025` tiene el bit `S_ISUID` activado entonces al ejecutarlo el usuario `David` al proceso A que se crea en el sistema asociado a la ejecución de `e2025` se le asigna un `EUID` igual a `UID` del propietario del archivo, que recuérdese que es `Paco` `UID=720`. Luego el proceso A tiene `UID=725` y `EUID=720`.

Al ejecutar el proceso A en primer lugar se invoca a la llamada al sistema `getuid` para asignar el valor del `UID` del proceso a la variable `x`. Luego se invoca a la llamada al sistema `geteuid` para asignar el valor del `EUID` del proceso a la variable `y`. A continuación se imprime en la salida estándar, por defecto la pantalla, el mensaje

```
[1] S1= 725, S2= 720
```

Posteriormente se invoca a la llamada al sistema `open` para abrir con permiso de solo lectura el archivo `fichero1.txt`. La máscara de modo de este archivo únicamente permite leerlo al propietario del mismo, que es el usuario `Paco` cuyo `UID=720`. Puesto que el proceso A tiene un `EUID=720` que coincide con el `UID` del propietario del archivo la llamada al sistema se ejecuta con éxito y en `fd1` se almacena el descriptor del archivo.

Luego se vuelve a invocar a la llamada al sistema `open` para abrir con permiso de solo lectura el archivo `fichero2.txt`. La máscara de modo de este archivo únicamente permite leerlo al propietario del mismo, que es el usuario `Davis` cuyo `UID=725`. Puesto que el proceso A tiene un `EUID=720` que es distinto del `UID` del propietario del archivo la llamada al sistema devuelve un error y en `fd2` se almacena el valor `-1`. A continuación se imprime en la salida estándar el mensaje

```
[2] fd1= 3, fd2= -1
```

Posteriormente se invoca a la llamada al sistema `setuid` para asignar al `UID` y al `EUID` del proceso A el valor de `x` (`725`). Como el `EUID` del proceso A no es el del superusuario y el valor de `x` coincide con el valor del `UID` del proceso A entonces solamente el `EUID` del proceso A se configura con el valor de `x`. En resumen el resultado de esta llamada al sistema es configurar el `EUID` del proceso A al valor `EUID=725`. A continuación se imprime en la salida estándar el mensaje

[3] S1= 725, S2= 725

Luego se invoca a la llamada al sistema `open` para abrir con permiso de solo lectura el archivo `fichero1.txt`. La máscara de modo de este archivo únicamente permite leerlo al propietario del mismo, que es el usuario `Paco` cuyo `UID=720`. Puesto que el proceso A tiene un `EUID=725` que es distinto del `UID` del propietario del archivo la llamada al sistema devuelve un error y en `fd3` se almacena el valor `-1`.

A continuación se invoca a la llamada al sistema `open` para abrir con permiso de solo lectura el archivo `fichero2.txt`. La máscara de modo de este archivo únicamente permite leerlo al propietario del mismo, que es el usuario `David` cuyo `UID=725`. Puesto que el proceso A tiene un `EUID=725` que coincide con el `UID` del propietario del archivo la llamada al sistema se ejecuta con éxito y en `fd4` se almacena el descriptor del archivo. Acto seguido se imprime en la salida estándar, el mensaje

[4] `fd3= -1, fd4= 4`

Posteriormente se invoca de nuevo a la llamada al sistema `setuid` para asignar al `UID` y al `EUID` del proceso A el valor de `y` (`720`). Como el valor de `y` coincide con el valor del `UID` del propietario de `e2025` y el bits `S_ISUID` esta activado entonces el `EUID` del proceso A se configura con el valor de `y`, `EUID=720`. Finalmente se imprime en la salida estándar, el mensaje

[5] S1= 725, S2= 720