| APELLIDOS | | NOMBRE | | Grupo |
|---|---|---|---|---|
| DNI | | Firma | | |

- **No desgrape las hojas.**
- **Conteste exclusivamente en el espacio reservado para ello.**
- **Utilice letra clara y legible. Responda de forma breve y precisa.**
- **El examen consta de 9 cuestiones, cuya valoración se indica en cada una de ellas.**

**1.** Sea un sistema con gestión de memoria paginada, con páginas de 4KB, direcciones lógicas de 28 bits y 4GB de memoria física. En dicho sistema se está ejecutando un proceso del que se muestra **parte** del contenido de su tabla de páginas (en hexadecimal):

| Página | Marco | Bv |
|---|---|---|
| 0 | 521F5 | 1 |
| 1 | 112D2 | 1 |
| 2 | 43231 | 1 |
| 3 | | 0 |
| 4 | 231F2 | 1 |
| 5 | 231F3 | 1 |
| 6 | 231F4 | 1 |
| ... | ... | ... |

**(1,0 puntos = 0,5 + 0,5)**

**1**

**a)** Calcule, en hexadecimal, la dirección física que se genera si el proceso emite la dirección lógica 0002F1F (en hexadecimal). Exponga de forma concisa cómo la obtiene.

La dirección **0002**F1F, tiene como página 0002 y desplazamiento F1F, por lo que el marco será 43231 y su dirección física **43231**F1F.

**b)** Calcule, en hexadecimal, la dirección lógica de dicho proceso que genera un acceso a la dirección física 231F2288 (en hexadecimal). Exponga de forma concisa cómo la obtiene.

La dirección **231F2**288, tiene como marco 211F2 y desplazamiento 288, por lo que la página es la 4 y la dirección lógica **0004**288.

**2.** Un sistema operativo debe gestionar de forma eficiente la memoria de la máquina. Piense en la problemática que genera la gestión de memoria y responda:

**(1,0 puntos = 0,5 + 0,5)**

| | |
|---|---|
| **2** | **a)** ¿Qué problema trata de resolver la técnica de compactación y en qué tipo o tipos de asignación contigua de memoria tienen utilidad su aplicación?<br>Resuelve el problema de la fragmentación externa. La fragmentación externa consiste en tener suficiente espacio en memoria para ubicar un nuevo proceso, pero este espacio se encuentra disperso en huecos no contiguos y por tanto no se puede ubicar el nuevo proceso. Las técnicas de compactación permiten desplazar los procesos ubicados en memoria de unas posiciones a otras, pudiendo agrupar de forma contigua el espacio libre de memoria. Estas técnicas se pueden aplicar siempre que los procesos sean reubicables en tiempo de ejecución.<br>La fragmentación externa se presenta en particiones variables, donde el sistema ajusta el espacio de memoria asignado, al tamaño del proceso que lo solicita. |
| | **b)** ¿Qué utilidad tiene la MMU (Unidad de Manejador de Memoria) y qué problemas trata de resolver?<br>La MMU es la encargada de traducir direcciones lógicas a físicas resolviendo así el problema de reubicación y proporcionando protección.<br>Traducir direcciones lógicas a física: Este sistema permite utilizar espacios de direcciones lógicos (DL) para cada proceso con independencia del espacio de direcciones físicos (DF), de manera que en tiempo de ejecución los procesadores emiten direcciones lógicas que son traducidas por la MMU ( unidad de manejador de memoria) a direcciones físicas.<br>Problema de reubicación: Los procesos deben poder ejecutarse correctamente con independencia de donde estén ubicadas sus instrucciones en memoria principal (físicamente). Esto debe ser así incluso cuando esta ubicación cambia durante la ejecución de un proceso. Para ello es importante considerar los espacios lógicos de cada proceso de forma independiente, de manera que estos espacios no dependan de las particularidades de la máquina ni de la situación en cada momento de la memoria principal del sistema .<br>Proporcionando protección: La MMU es la encargada de verificar que las direcciones emitidas por los procesos les pertenecen y forman parte de su espacio, así como el tipo de acceso (R,W) que solicitan realizar en las mismas, en caso contrario debería enviar una excepción al procesador. |

**3.** Considerando el mapa de memoria de un proceso UNIX, diga si las siguientes sentencias son verdaderas o falsas:

**(0,6 puntos)**

| **3** | SENTENCIA | V/F |
|---|---|---|
| | La región de datos sin valor inicial del mapa de memoria que se genera al ejecutar un programa, es una región que no tiene soporte en ningún fichero. | V |
| | La región de código con soporte en el archivo ejecutable del proceso tiene permisos de lectura, escritura y de ejecución. | F |
| | Cuando un proceso mapea un archivo en memoria se crea una nueva región en el mapa de memoria de ese proceso a la que se accede mediante operaciones de entrada/salida. | F |
| | La utilización de bibliotecas dinámicas se basa en la técnica de mapeo de las librerías en memoria durante la ejecución del proceso que las utiliza. | V |

**4.** Sea un sistema operativo que gestiona la memoria virtual mediante paginación con páginas de 4 KB. Los marcos libres se asignan por orden creciente de direcciones físicas utilizando políticas de reemplazo locales. A cierto proceso nuevo el sistema le asigna 3 marcos (del 0 al 2) y durante su ejecución se emiten las siguientes direcciones lógicas (en hexadecimal):

400000, 400014, 800000, 800024, 600030, 600034, 800280, 40060c, 200000, 400c10, 600f24

Indique qué páginas quedarán en la memòria física después del último acceso y cuál es la última dirección física a la que se accede en los siguientes casos. (Indique los marcos libres u ocupados con una página inválida con '—').

**(2,0 puntos = 0,5 + 0,5 + 0,5 + 0,5)**

**4**

**a)** Algoritmo de reemplazo LRU

| marco | Página |
|---|---|
| 0 | 400 |
| 1 | 600 |
| 2 | 200 |

Última dirección física: 1f24

**b)** Algoritmo de reemplazo segunda oportunidad

| marco | Página |
|---|---|
| 0 | 200 |
| 1 | 400 |
| 2 | 600 |

Última dirección física: 2f24

**c)** Algoritmo de reemplazo óptimo

| marco | Página |
|---|---|
| 0 | 400 |
| 1 | 200 |
| 2 | 600 |

Última dirección física: 2f24

**d)** Suponga ahora que se aplica un algoritmo FIFO y que después del último acceso de la lista, el código invoca a la llamada *exec* de un ejecutable que realiza los siguientes accesos lógicos: 400000, 400010, 600020

| marco | Página |
|---|---|
| 0 | 400 |
| 1 | 600 |
| 2 | — |

Última dirección física: 1020

**5.** En un sistema con direcciones físicas y lógicas de 32 bits, se establece un mecanismo de paginación de dos niveles. Las páginas son de 4KB y las tablas de páginas de segundo nivel residen en memoria ocupando un único marco cada una.

**(1,0 puntos = 0,5 + 0,5)**

**5**

**a)** Determine el número de entradas que tendrán las tablas de páginas de segundo nivel. Considere que cada entrada de la tabla o descriptor de página utiliza 12 bits para los bits Bv, Br, Bm, R, W, X, etc.

Cada entrada de la tabla de páginas de segundo nivel ocupa: 20 bits para el número de marco (hay $2^{32}/2^{12} = 2^{20}$ marcos) + 12 bits adicionales (Bv, Br, etc.) = 32 bits -> 4 bytes por entrada.
Por tanto tendrán 4KB/4B = 1K entradas (1024).

**b)** Describa la estructura de las direcciones lógicas y de las direcciones físicas de este sistema, así como el tamaño en bits de cada uno de los elementos que la componen.

Direcciones lógicas: Se requieren

- 12 bits para el desplazamiento (páginas de 4KB).
- 10 bits para el número de entrada a la tabla de segundo nivel (tablas con 1K = $2^{10}$ entradas)
- 10 bits (32-12-10) para referenciar la entrada a la tabla de primer nivel

```
-------------------------------------------------------------
|   p1 (10 bits)   |  p2 (10 bits)   |   desplazam. (12 bits) |
-------------------------------------------------------------
```

Direcciones físicas: Se requieren

- 12 bits para el desplazamiento
- 20 bits para el número de marco

```
-------------------------------------------------------------
|            marco (20 bits)       |  desplazam. (12 bits)  |
-------------------------------------------------------------
```

**6.** El contenido de las tablas de descriptores de archivos abiertos correspondientes a tres procesos en un instante dado es el siguiente:

| Tabla inicial | |
|---|---|
| 0 | STDIN |
| 1 | STDOUT |
| 2 | STDERR |
| 3 | |
| 4 | |
| 5 | |

| Tabla del proceso P1 | |
|---|---|
| 0 | STDIN |
| 1 | STDOUT |
| 2 | STDERR |
| 3 | "result" |
| 4 | "fd_pipe[0]" |
| 5 | "fd_pipe[1]" |

| Tabla del proceso P2 | |
|---|---|
| 0 | STDIN |
| 1 | "fd_pipe[1]" |
| 2 | STDERR |
| 3 | |
| 4 | |
| 5 | |

| Tabla del proceso P3 | |
|---|---|
| 0 | "fd_pipe[0]" |
| 1 | "result" |
| 2 | STDERR |
| 3 | |
| 4 | |
| 5 | |

Complete el fragmento de código en C con las primitivas POSIX necesarias para que al ejecutarlo se creen los tres procesos y el contenido de sus tablas de descriptores en los puntos marcado como /*** Tabla …….***/ sea el mostrado

**(1.0 puntos)**

**6**

```
int fd_pipe[2]; /*descriptor de tubo*/
int fd;      /* descriptor archivo regular**/
 main (int argc, char *argv[])
{
/*** Tabla Inicial ***/
  fd=open("result",O_WRONLY |O_CREAT|O_TRUNC,0666);
  pipe(fd_pipe);
  /*** Tabla del proceso P1 ***/
  if (!(pid=fork())) {
      dup2(fd_pipe[1],1);
      close(fd_pipe[0]);
      close(fd_pipe[1]);
      close (fd);
  /***Tabla del proceso P2 ***/
    execlp("/bin/cat", "cat", "fich1", NULL);
  }
  if(!(pid=fork())){

       dup2(fd_pipe[0],0);
       dup2(fd,1);
       close(fd_pipe[0]);
       close(fd_pipe[1]);
       close(fd);

  /***Tabla del proceso P3 ***/
    execlp("/usr/bin/wc", "wc", "-l",NULL);
  }
  close(fd_pipe[0]); close(fd_pipe[1]);
  close (fd);
  while(pid != wait(&status));
}
```

**7.** Dado el siguiente listado de un directorio en un sistema POSIX:

```
i-nodo   permisos    enlaces usuario  grupo tamaño     fecha       nombre
2021662 drwxr-xr-x   2       sterrasa  fso  4096       dec  9 2015  .
2021611 drwxr-xr-x   8       sterrasa  fso  4096       sep 10 2015  ..
2021663 -rwxr-sr-x   1       sterrasa  fso  1139706    dec  9 2015  copia
2021664 -rw-rw-r--   1       sterrasa  fso  9706       dec  9 2015  f1
2021665 -rw-rw-r--   2       sterrasa  fso  4157       dec  9 2015  f2
2021665 -r--r--rw-   2       sterrasa  fso  4157       dec  9 2015  f3
2021666 lrwxrwxrwx   1       sterrasa  fso     2       dec  9 2015  f4->f1
```

**(1.25 puntos = 0.75 + 0.5)**

**7**

**a)** Teniendo en cuenta que el programa `copia` copia el contenido del archivo que recibe como primer argumento en otro cuyo nombre se indica como segundo argumento. Es decir, "`copia a b`" copia el contenido del archivo `a` al archivo `b`, si `b` no existe lo debe crear y después realizar la copia. Rellene la tabla, indicando en caso de éxito cuáles son los permisos que se van comprobando y, en caso de error, cuál es el permiso que falla y por qué.

| (UID,GUI) | ORDEN | ¿FUNCIONA? | JUSTIFICACIÓN |
|---|---|---|---|
| (pepe, fso) | copia f1 f5 | No | Para que la orden tenga éxito se debe tener permisos de ejecución para copia de lectura sobre f1 y de escritura sobre el directorio, ya que f5 no existe.<br>Falla al intentar escribir en el directorio, ya que como (pepe, fso) se accede al mismo por la segunda tripleta y no tiene permisos de escritura |
| (sterrasa, fso) | copia f1 f3 | No | Para que la orden tenga éxito se debe tener permisos de ejecución para copia de lectura sobre f1 y de escritura sobre f3<br>Falla al intentar escribir en f3, ya que si siquiera el propietario (sterrasa) tiene permisos para poder escribir (primera tripleta) |
| (ana, etc) | copia f3 f4 | SI | Para que la orden tenga éxito se debe tener permisos de ejecución para copia de lectura sobre f3 y de escritura sobre f4<br>(Ana, etc) puede ejecutar copia (permiso x en la tercera tripleta) y con eso cambia su eGID, pasando a ser (Ana, fso). Con este nuevo eGID, tiene permisos de lectura sobre f3 y de escritura sobre f1, que es finalmente al fichero al que se accede por el enlace simbólico f4 |

**b)** Indique qué valores de la columna *enlaces* de listado anterior se verían afectados si se elimina la entrada de directorio f3 (`rm f3`). ¿Y si se elimina la entrada f4?. Justifique sus respuestas

Al ejecutar la orden "rm f3", se elimina la entrada a directorio correspondiente . Además si nos fijamos f3 y f2 son enlácesfísicos al mismo conteido, esto lo sabemos por que comparten nodo-i y además en el campo de número de enlaces aparece un 2. Si eliminaramos f3, a parte de eliminar la entrada, en el campo número de enlaces de f2 **aparecería un 1** indicando así que sólo queda una entrada a directorio para poder acceder al contenido del fichero con nodo-i 2021665

Sin embargo al eliminar la entrada a directorio f4, **no se modificaría nada en la columna de enlaces**, ya que f4 es un enlace simbólico sobre el fichero f1, y en el campo nº enlaces del nodo -i, no se lleva la cuenta de los enlaces simbólicos, solo de los físicos

**8.** Una partición de 32MBytes, de los cuales 2Mbytes están ocupados con estructuras propias del sistema de archivos, está organizada en bloques de 512 bytes, el puntero a bloque es de 32 bits. Calcule:

**(1,0 puntos = 0.5 + 0.5)**

**8**

**a)** Tamaño máximo de un archivo si se utiliza asignación indexada simple, con un solo bloque de índices por archivo

Los punteros a bloque son de 32 bits --> 4 bytes
Con punteros de 32 bits el número máximo de bloques que podemos direccionar son : $2^{32}$= 4 x 1024 x 1024 x 1024 bloques, cualquier tamaño máximo de archivo debe ser igual o inferior a este número de bloques.
Cada bloque de 512 bytes contiene 512/4 = 128 punteros a bloque.
Por tanto 128 x 512= $2^{7}$ x $2^{9}$= $2^{16}$=64 KBytes

**b)** Tamaño máximo de un archivo si se utiliza asignación enlazada

El espacio que podemos considerar libre en el disco para crear el archivo es de 32MBytes-2 MBytes =30 MBytes
Cada bloque es de 512bytes, por tanto el número total de bloques del disco es:
30 MBytes/512Bytes = 30 $2^{20}$ /$2^{9}$= 30 * $2^{11}$= 60 x 1024 bloques
En cada bloque tendremos 512 bytes= 4 bytes para puntero al siguiente+508 bytes de información propia del archivo → Tamaño máximo para datos del archivo= 508 x 60 x 1024=
30480 KBytes

**9.** Un disco con una capacidad de 6GB, se formatea con una versión de MINIX, cuyos tamaños son los siguientes:

- El bloque de arranque y el superbloque ocupan 1 bloque cada uno.
- El tamaño del nodo-i es de 32 bytes (7 punteros directos, 1 indirecto,1 doble indirecto).
- Con punteros a zona de 16 bits.
- Cada entrada de directorio ocupa 16 bytes.
- 1 zona = 1 bloque = 1 KByte
- Al formatear se ha reservado espacio en la cabecera para para 4.096 nodos-i

El esquema de los diferentes elementos del disco es el siguiente:

| Arranque | Superbloque | Mapa de bits Nodos-i | Mapa de bits Zonas | Nodos- i | Zonas de datos |
|---|---|---|---|---|---|

Se pide:

a) Calcule el número de bloques que ocupa el Mapa de bits nodos-i, el Mapa de bits Zonas y los Nodos-i.

b) El contenido de los directorios es el que se muestra, suponiendo que los archivos regulares ocupan 10 KBytes cada uno ¿cuántos i-nodos están ocupados?¿cuántas zonas de datos están ocupadas?

| | |
|---|---|
| 1 | . |
| 1 | .. |
| 4 | usr |
| 10 | bin |

| | |
|---|---|
| 4 | . |
| 1 | .. |
| 20 | list |
| 12 | alumno |

| | |
|---|---|
| 12 | . |
| 4 | .. |
| 26 | prac |

| | |
|---|---|
| 10 | . |
| 1 | .. |
| 35 | calc |

**(1,15 puntos = 0,5 + 0,65)**

**9**

a)

Nº bloques mapa i-nodos = 4096 / (1024*8) = 0,5 -> 1 bloque

Nº bloques mapa zonas = (6G / 1024) / (1024*8) = 768 bloques

Nº bloques i-nodos = (4096*32) / 1024 = 128 bloques

b)

Se tienen 4 directorios con los i-nodos: 1, 4, 12 y 10, y 3 archivos regulares con los i-nodos 20, 26 y 35. En total hay pues 7 i-nodos ocupados.

Las zonas ocupadas se pueden dividir en:

- Contenido de directorio: 4 zonas
- Contenido de archivo regular: 30 zonas
- Punteros: Con los 7 punteros directos se direccionan 7KBytes, para direccionar los restantes 3KBytes hay que utilizar el puntero simple-indirecto, por lo que se requiere una zona de punteros por archivo, o sea 3 zonas.

En total hay pues 37 zonas de datos ocupadas.