UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Departamento de Informática de Sistemas y Computadoras (DISCA)

fSO

Ejercicio de Evaluación
22 de Diciembre de 2014

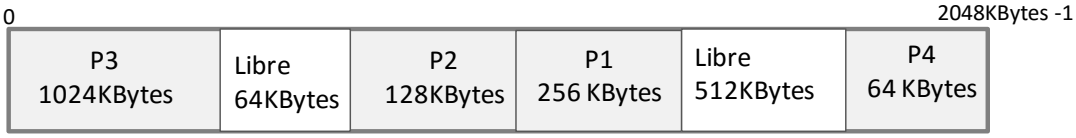
etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

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 10 cuestiones, en cada una de ellas se indica su puntuación.**

---

1. En un sistema con **asignación contigua** de memoria, gestionado con **particiones variables,** el estado de la memoria en un momento dado es el siguiente:



a) Indique cuál sería el contenido de los registros de la MMU que permitirían traducir las direcciones lógicas emitidas por la CPU para los procesos P1, P2, P3 y P4 a direcciones físicas.

b) Suponiendo una lista de huecos ordenada hacia direcciones creciente, indique en qué hueco se ubicaría un nuevo proceso de 28Kbytes para cada de las estrategias de asignación: Best Fit, Worst Fit y First Fit.

c) Justifique si en este tipo de sistema podría darse fragmentación externa o interna.

0.75 puntos

1

a)
En estos tipos de gestión de memoria la MMU utiliza dos tipos de registro: Registro Base y Registro Límite.
El Registro Base contiene la primera dirección física a partir de la que está ubicado el proceso. Mientras que el Registro Limite puede contener, según se implemente la comprobación de direcciones en la MMU, el tamaño del proceso o la última dirección física donde se ubica el proceso.
Proceso P1→ Registro Base=1216K, Registro Limite= 1216K+256K=1472K
Proceso P2→ Registro Base=1088K, Registro Limite= 1088K+128K=1216K
Proceso P3→ Registro Base=0, Registro Limite= 1024K
Proceso P4→ Registro Base=1984K, Registro Limite= 1984K+64K=2048K

b)
Best Fit--→ Hueco de 64KBytes
Worst Fit→ Hueco de 512KBytes
First Fit → Supongo que los huecos están ordenados hacia direcciones crecientes →Hueco de 64KBytes

c)
En este tipo de sistemas puede darse fragmentación externa ya que los huecos se ajustan al tamaño del proceso →con lo cual podría haber suficiente memoria libre para ubicar nuevos proceso pero al no estar contigua la memoria libre no podrían ubicarse→ en estos casos se podría aplicar técnicas de compactación para que toda la memoria libre fuese contigua

2. En un sistema con una Memoria Principal de 1GBytes y una capacidad de direccionamiento lógico de 4GBytes se quiere implementar un sistema basado en **tres niveles de paginación**. Las tablas del primer nivel contienen 16 descriptores mientras que las tablas del segundo y tercer nivel contiene el mismo número de descriptores. El tamaño de página es de 16KBytes y en todas las tablas de páginas, cada descriptor de página ocupa 8Bytes.

Indique de forma razonada:

a) Formato de la dirección lógica y de la dirección física exponiendo sus campos y números de bits

b) Tamaño de las tablas de páginas y número de tablas que conllevan

c) Si está ubicado en Memoria Principal un proceso de 16MBytes, indique el número de descriptores de página que sería necesario consultar para referencia todas las páginas de dicho proceso.

1.0 puntos

2

a)Formato de dirección lógica y física

Memoria Principal de 1GByte → 30 bits para la Dirección Física (DF)
Capacidad de direccionamiento lógico de 4GBytes → 32 bits para la dirección Lógica (DL)
Tamaño de Páginas de 16KBytes → 14 bits para desplazamiento de página (DP) o de marco (DM)
Formato de la dirección Física: 30 bits = 16 bits Número marco + 14 bits Desplazamiento

Formato de la dirección Lógica: 32 bits = 4 bits Nº Páginas 1er nivel + 7 bits Nº Páginas 2º nivel + 7 bits Nº Páginas 3º nivel + 14 bits desplazamiento de página

b) Tamaño de las tablas de pagina y número de páginas que conlleva

Tamaño de Tabla de Página de 1er nivel: 2^4 descriptores * 8Bytes =2^7 Bytes = 128Bytes
Tamaño de Tabla de Página de 2º nivel: 2^7 descriptores * 8Bytes =2^{10} Bytes = 1KBytes
Tamaño de Tabla de Página de 3er nivel: 2^7 descriptores * 8Bytes =2^{10} Bytes = 1KBytes
En total se pueden generarse 1 Tabla de 1er nivel + 16 Tablas de 2º nivel + 16*128 Tablas de 3er nivel

c)

Un proceso de 16Mbyte ocuparía 16MBytes/ 16Kbytes → 1024 páginas → 1024 descriptores de páginas

El sistema con 3 niveles de paginación las tablas del 3er nivel pueden contener 128 descriptores de página cada una→ Necesitaremos 8 tablas de 3 nivel (8*128=1024), 1 tabla de 2º nivel que tendrá 8 descriptores ocupados y una tabla de primer nivel que tendrá un único descriptor.

3. Sea un sistema de tiempo compartido que en un instante dado está ejecutando tres procesos (A, B y C). En dicho sistema se ha implementado el modelo de área activa para evitar situaciones de hiperpaginación, con una ventana de área activa de 5. Considere los instantes t_1 y t_2, marcados y la secuencia de referencias a páginas siguiente:

A2	B4	B3	C2	A6	C2	C1	B2	B4	B1	A2	A1	C2	B1	A4	C3

t_1

A2	B1	A3	A4	C6	B0	B5	A2	A3	B1	C7	C2	A2	A2	B1	B4

t_2

Indique de forma justificada si son correctas o no cada una de las siguientes afirmaciones:

a) El área activa para los procesos A, B y C en t_1 es : $AA_{t1}(A)=\{1,4\}$, $AA_{t1}(B)=\{1\}$, $AA_{t1}(C)=\{2,3\}$
b) El tamaño del área activa del proceso A en t_2 es 3 ($TAA_{t2}(A)=3$)
c) Si disponemos de 11 marcos (para cualquiera de los dos instantes de tiempo) podemos asegurar que no habrá hiperpaginación

0.75 puntos

3

a) El área activa para los procesos A, B y C es : $AA_A=\{1,4\}$, $AA_B=\{1\}$, $AA_C=\{2,3\}$

***FALSO**, para calcular el area active hay que ver las Δ últimas referencias de los tres procesos, por lo que realmente el área active es la siguiente:*

$AA_{t1}(A)=\{1,2,4,6\}$, $AA_{t1}(B)=\{1,2,3,4\}$, $AA_{t1}(C)=\{1,2,3\}$

b) El tamaño del área activa del proceso A en t2 es 3

***CIERTO**, ya que si calculamos el AA para A en el instante t_2 su valor es:*

$AA_{t2}(A)=\{2,3,4\}$ y por tanto sólo necesitamos 3 marcos, es decir:

$TAA_{t2}(A) = 3$

c) Si disponemos de 11 marcos (para cualquiera de los instantes de tiempo) podemos asegurar que no habrá hiperpaginación

***CIERTO**, ya que si calculamos el AA para A, B y C en los instantes t_1 y t_2 tendremos:*

$AA_{t1}(A)=\{1,2,4,6\}$, $AA_{t1}(B)=\{1,2,3,4\}$, $AA_{t1}(c)=\{1,2,3\}$
$AA_{t2}(A)=\{2,3,4\}$, $AA_{t2}(B)=\{0,1,4,5\}$, $AA_{t2}(C)=\{2,3,6,7\}$

Esto nos da unos Tamaños para las áreas activas de:
$TAA_{t1}(A)=4$, $TAA_{t1}(B)=4$, $TAA_{t1}(c)=3$
$TAA_{t2}(A)=3$, $TAA_{t2}(B)=4$, $TAA_{t2}(C)=4$

Por tanto, en ambos instantes, necesitamos 4+4+3 = 11 marcos totales para garantizar que las localidades de los procesos están cubiertas y por tanto poder asegurar que no se producirá hiperpaginación

4. El sistema operativo de cierto computador gestiona la memoria virtual mediante paginación con páginas de 4 KB. El espacio de direccionamiento lógico es de 64MB y el físico de 1MB Se asignan marcos libres, de forma global por orden creciente de direcciones físicas, empezando en la dirección 0x11000.

a) Indique el formato, con campos y número de bits, de las direcciones lógicas y físicas.

b) Dos procesos Y y Z realizan la secuencia de accesos a direcciones lógicas que se indica a continuación (todos los números en hexadecimal):

`Y:1008C24, Y:1008C28, Y:2007143, Y:1008C2C, Y:2007157,`
`Z:1008C24, Z:1008C28, Z:1008C2C, Z:200A100, Z:200B012,`
`Y:2007158, Y:1009000, Y:1009004, Y:2007158, Y:1009008,`
`Z:1008C30, Z:200A017, Z:1058120, Z:1058124, Z:200B012`

Indique la serie de referencias correspondiente a dicha secuencia de acceso.

c) Suponga que hay suficiente marcos libres para contener todas las páginas solicitadas e indique el contenido de las tablas de páginas de ambos procesos, indicando únicamente los descriptores de páginas correspondientes a las páginas que se referencian.

(1,5 puntos)

4

a) Indica el esquema de distribución de direcciones lógicas y físicas

Dirección lógica

25 — 12	11 — 0
Página (14 bits)	Desplazamiento (12 bits)

Dirección física

19 — 12	11 — 0
Marco (8 bits)	Desplazamiento (12 bits)

b) Escribir la serie de referencias correspondiente a la secuencia de acceso

Y:1008, Y:2007, Y:1008, Y:2007,
Z:1008, Z:200A, Z:200B,
Y:2007, Y:1009, Y:2007, Y:1009,
Z:1008, Z:200A, Z:1058, Z:200B

c) Tablas de páginas de ambos procesos:

Nº Página	Tabla de Páginas Proceso Y
1008	11
2007	12
1009	16

Nº Página	Tabla de Páginas Proceso Z
1008	13
200A	14
200B	15
1058	17

5. En un sistema que gestiona la memoria virtual mediante paginación se genera la siguiente serie de referencias al ejecutar los procesos Y y Z (número de páginas en hexadecimal, ceros a la izquierda omitidos): **Y:108, Y:207, Z:119, Z:20A, Y:207, Y:108, , Z:20A, Y:207, Z:20B**
En este caso el sistema sólo dispone de 3 marcos, del 0 al 2, que inicialmente se encuentran vacíos.
a) Indique la evolución del contenido de la memoria física y el número de fallos de página que se producen si se aplica un algoritmo de reemplazo **LRU con reemplazo global**.
b) indique la evolución del contenido de la memoria física y el número de fallos de página que se producen si se aplica un algoritmo de reemplazo de **Segunda Oportunidad con reemplazo global**.
(1,0 punto)

5

a) LRU con reemplazo global. Indicad instante de tiempo del último acceso
Y:108, Y:207, Z:119, Z:20A, Y:207, Y:108, , Z:20A, Y:207, Z:20B

marco	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8
0	Y108 (t=0)	Y108 (t=0)	Y108 (t=0)	Z20A (t=3)	Z20A (t=3)	Z20A (t=3)	Z20A (t=6)	Z20A (t=6)	Z20A (t=6)
1		Y207 (t=1)	Y207 (t=1)	Y207 (t=1)	Y207 (t=4)	Y207 (t=4)	Y207 (t=4)	Y207 (t=7)	Y207 (t=7)
2			Z119 (t=2)	Z119 (t=2)	Z119 (t=2)	Y108 (t=5)	Y108 (t=5)	Y108 (t=5)	Z20B (t=8)
	FP	FP	FP	FP(R)		FP(R)			FP(R)

Entre paréntesis se ha puesto el instante de referencia. Para elegir víctima se selecciona aquella que hace más tiempo que hemos referenciado. Total 7 Fallos de página 4 de ellos con reemplazo.

b) Segunda Oportunidad con reemplazo global. Indicad valor del bit R y el próximo candidato
Y:108, Y:207, Z:119, Z:20A, Y:207, Y:108, , Z:20A, Y:207, Z:20B

marco	t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8
0	Y108 →1	Y108 →1	Y108 →1	Z20A 1	Z20A 1	Z20A →1	Z20A →1	Z20A →1	Z20B 1
1		Y207 1	Y207 1	Y207 →0	Y207 →1	Y207 0	Y207 0	Y207 1	Y207 →1
2			Z119 1	Z119 0	Z119 0	Y108 1	Y108 1	Y108 1	Y108 1
	FP	FP	FP	FP(R)		FP(R)			FP(R)

Nomenclatura usada en la solución: → el próximo candidato y Bit de Referencia

6. Suponga que el código de *redir.c*, se ejecuta sin errores y que el fichero "*listado.txt*" no existe y responda a las siguientes cuestiones:

a) Indique el contenido de las tablas de descriptores en los puntos */***(1) Tabla descriptores***/* y */***(2) Tabla descriptores***/* para cada uno de los procesos que alcancen dichas líneas.

b) Indique de forma justificada qué se muestra por pantalla al ejecutar el código de *redir.c* y las ordenes del Shell a las que equivalen dicha ejecución.

c) Describa y justifique que pasaría al ejecutar *redir.c* si se elimina la llamada wait() del programa.

```
/******Codigo de redir.c*******/
#include <todo lo necesario.h>
#define NEWFILE (O_WRONLY | O_CREAT | O_TRUNC)
#define MODE644 (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)
#define FILE "listado.txt"
int main(int argc, char *argv[]) {
   int fd;

   if (fork() == 0) {
      if ((fd = open(FILE, NEWFILE, MODE644)) == -1)
         exit(-1);
      dup2(fd, STDOUT_FILENO);
      close (fd);
      /***(1)Tabla descriptores***/
      execlp("ls", "ls", "-la", NULL);
      fprintf(stderr,"La ejecucicón de ls falló");
      exit(1);
   }
   wait(NULL);

   if ((fd = open(FILE, O_RDONLY)) == -1)
      exit(-1);
    dup2 (fd, STDIN_FILENO);
   /***(2)Tabla descriptores***/
  close (fd);
   execlp("wc", "wc", NULL);
   fprintf(stderr, "La ejecución de wc falló");
   exit(1);
}
```

(1,0 punto)

6 a)

Donde está el comentario /*(1) tabla de descriptores*/ corresponde al proceso hijo y al /*(2) tabla de descriptores*/ al padre. Las tablas de descriptores quedarían como sigue:

Hijo	
0	STDIN
1	fd
2	STDERR
3	
4	

Padre	
0	fd
1	STDOUT
2	STDERR
3	fd
4	

b) El resultado que se obtiene es el de contar el número de líneas, palabras y letras que tiene el listado del directorio donde se ejecuta la orden.
No hay intercomunicación directa entre los procesos, sino que se realiza el redireccionamiento a un fichero auxiliar (listado.txt). Desde el shell sería equivalente a realizar las siguientes ordenes:

```
>>> ls –la > listado.txt
>>> wc < listado.txt
```

c) La llamada wait, lo que asegura es que el padre espere a que se termine la ejecución de la orden del hijo (ls) y por lo tanto, el fichero resultante se habrá creado y se podrá utilizar como entrada a la orden wc.
Si se elimina la llamada a wait, puede ocurrir que el proceso padre continúe su ejecución antes de que se haya generado el fichero listado.txt y la apertura del mismo fallaría, con lo que el proceso padre finaliza sin haber ejecutado la orden wc.

---

7. El filtro tee realizar una copia de la salida estándar en uno o más archivos. Por ejemplo, la orden:

$ls –la | tee listado.txt

muestra por pantalla el listado del directorio actual y además lo copia en el fichero listado.txt.
A continuación se adjunta el código (incompleto) de una orden denominada *mitee* que requiere: un primer parámetro que es el fichero de salida y uno o más parámetros que son interpretados como la orden a ejecutar y los parámetros con que se ejecuta. El resultado es que *mitee* ejecuta la orden, realizando una copia de la salida estándar en el fichero indicado. Por ejemplo,

$mitee listado.txt ls –la

tendría el mismo efecto que ejecutar *$ls –la | tee listado.txt.* En el código proporcionado, la duplicación de la salida se quiere implementar en la función *tee_out* copiando desde la entrada a la salida la estándar y al fichero salida.

a) Indique los descriptores de archivos que corresponden a las etiquetas con los números **1** a **8** en la función main.
b) Indique los descriptores de archivos que corresponden a las etiquetas con los **9** a **11**.

```
#include <todo lo necesario.h>
#define NEWFILE (O_WRONLY | O_CREAT | O_TRUNC)
#define MODE644 (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)

void tee_out(const char *file_name) {
   int fdout,count;
   char buf[10];

   if ((fdout = open(file_name, NEWFILE, MODE644)) == -1)
      exit(-1);
   while ((count= read(**9**, buf, sizeof(buf)))>0)
      { if (write(**10**, buf, count) != count)
           {fprintf(stderr,"Error escritura STDOUT_FILE\n");
             exit(-1); }
        if (write(**11**, buf, count) != count)
           {fprintf(stderr,"Error escritura FICHERO\n");
             exit(-1); }
       }
     close(fdout);
     exit(1);
}

int main(int argc, char *argv[]) {
   int fd[2];
   if (argc < 3) {
      fprintf(stderr, "Error argumentos\n");
```

```
      exit(-1);
   }
   pipe(fd);
   if (fork() == 0)
   { dup2(**1**,**2**);
     close(**3**);     close(**4**);
      tee_out(argv[1]);
   }

    dup2 (**5**,**6**);
    close(**7**);   close(**8**);
   execvp(argv[2], &argv[2]);
   fprintf(stderr, "Fallo ejecución de %s", argv[2]);
   exit(1);
}
```

(1,0 puntos)

7

a)

```
dup2(**1**,**2**);  dup2( fd[0],  STDIN_FILENO );

close(**3**);       close(fd[0] );

close(**4**);       close(fd[1]  );

dup2 (**5**,**6**); dup2( fd[1],  STDOUt_FILENO );

close(**7**);        close(fd[0] );

close(**8**);       close(fd[1]  );
```

b)

```
while ((count= read(**9**  STDIN_FILENO, buf, sizeof(buf)))>0)
    {  if (write(**10**   STDOUT_FILENO, buf, count) != count)
          {fprintf(stderr,"Error escritura STDOUT_FILE\n");
           exit(-1); }
     if (write(**11**       fdout, buf, count) != count)
          {fprintf(stderr,"Error escritura FICHERO\n");
           exit(-1); }
      }
```

8. El listado de un directorio de un sistema de archivos de Linux es el siguiente:

```
total 3216
drwxr-xr-x  38 pau  admin    1292 12 dic 18:46 .
drwxr-xr-x  11 pau  admin     374  9 dic 16:51 ..
drwxr-x---  32 pau  admin    1088 11 dic 16:35 fig
-rw-r--r--   1 pau  admin    6616 12 dic 09:46 data.txt
-rwsr-xr-x   1 pau  admin     201 12 dic 09:45 reader
-rw-r-sr--   1 pau  admin     889 12 dic 09:45 writer
-rw-r--r--   2 pau  admin   44507 12 dic 09:46 log
```

Suponga que los usuarios que se citan más abajo se conectan y escriben las órdenes indicadas cuando el contenido de su directorio por omisión es el mostrado. Indique con qué UID y GID efectivos se ejecuta cada orden y si el sistema operativo permitirá el acceso a los archivos o directorios mencionados. Los archivos *reader* y *writer* son ejecutables que aceptan como argumento el nombre de un archivo; *reader* abre el archivo para lectura y *writer* para escritura. Las órdenes *cat* y *cp* son las comunes del shell, con la accesibilidad y permisos habituales.

(1,0 puntos)

8

Usuario	**Grupo**	**Orden**	***¿Ejecución* permitida?**	**eUID**	**eGID**	**Acceso a archivos y directorio permitido?**
valero	*admin*	`cp data.txt fig`	***Sí***	***valero***	***admin***	***No***
pau	*admin*	`writer data.txt`	***No***			
boi	*sports*	`cat data.txt`	***Sí***	***boi***	***sports***	***No***
boi	*sports*	`reader data.txt`	***Sí***	***pau***	***admin***	***Sí***
boi	*sports*	`writer data.txt`	***No***			

Corrección: 0.2 puntos/caso

9. Teniendo en cuenta qué el número de enlaces del archivo *log* es dos (2ªcolumna del listado) y su tamaño es de 44507 Bytes, indique de forma justificada qué espacio se liberaría en el disco al ejecutar el usuario *pau* la orden $*rm log*.

(0,5 puntos)

9

Con esta orden no se elimina más que una entrada de directorio, que es uno de los dos enlaces a *log*. Para borrar los datos del archivo y liberar espacio en el disco duro, será necesario borrar el otro enlace.

10.Un disco con una capacidad de 512MBytes, se formatea con una versión de MINIX y la siguiente estructura:

Bloque de Arranque	Super bloque	Mapa de bits Nodos-i	Mapa de bits Zonas	Nodos- i	Zonas de datos

Los tamaños y valores usados en el formateo son:

- Nodo-i de 32Bytes con: 7 punteros directos, 1 indirecto y 1 doble indirecto
- Puntero a zona de 16 bits
- Entradas de directorio de 16 Bytes.
- 1 Bloque = 1KBytes
- 1 zona = 2^3 bloques= 8KBytes
- 8.192 nodos-i

Se pide:

a) Indique de forma detallada el número de bloques que ocupa el Mapa de bits nodos-i, el Mapa de bits Zonas y los Nodos-i.
b) En dicho sistema han sido creados un total 10 directorios (incluido el raíz), 250 archivos regulares, 20 enlaces simbólicos y 40 enlaces físicos a archivos regulares que ya existían. Indique de forma justificada el número de nodos-i que estarán marcados como ocupados.
c) Del directorio raíz cuelgan 3 directorios, 10 archivos regulares y 5 enlaces físicos a archivos regulares. Indique de forma justificada el valor del campo "número de enlaces del nodo-i=1

(1,5 puntos= 0,75+0,5+0,5)

10

a)

Mapa de bits de nodos-i:

Para 8192 nodos-i, se necesita un mapa de bits con 8192 bits como mínimo. Como los bloque son de 1Kbyte → 8192 bits/1Kbyte= 8*1024 bits/ 1024 Byte= 8*1024 bits/8*1024 bits = 1 bloque

Mapa de bits de zonas:

Disco de 512 MBytes= 2^9*2^{20}Bytes → dividiendo por el tamaño de zona que es de 8KBytes tenemos →2^9*2^{20}Bytes/2^3*2^{10}Bytes= 2^6*2^{10}Zonas= 65536 Zonas.
Se necesita 1 bit por cada zona → 65536 bit /1bloque= 2^6*2^{10} bit/ 2^{10} Byte= 2^6*2^{10} bit/ 2^3*2^{10} bit =2^3 = 8 bloques

Nodos-i: Los nodos-i son de 32 byte y se requiere 8192 nodos-i, serán necesarios 8192 * 32 Bytes =2^3 2^{10} 2^5=2^{18} bytes → 2^{18} Bytes /2^{10} Bytes = 2^8 = 256 bloques.

b)

Por cada archivo que haya en el sistema habrá un nodo-i ocupado.
Los directorios y los archivos regulares tendrán asignado un nodo-i →10+250= 260 nodos-i
Los enlaces físicos son entradas de directorio que relaciona un nombre con un nodo-i pero no consumen nodos-i.
Los enlaces simbólicos son archivo cuyo contenido se interpretan como un path o ruta → Cada uno tiene asignado un nodo-i →20
Total nodos-i ocupados 10+250+20=380 nodos-i

c)

El número de enlaces del nodo-i=1 será de 5 y corresponden a:
- dos entradas asociadas al nodo-i 1, que son "." y "..", que hay en el raíz
- tres entradas ".." una en cada uno de los directorios que cuelgan del raíz .