SISTEMAS OPERATIVOS

Dobles Grados en Ingeniería Informática – Curso 2020-21

Prueba individual de los Temas 3 y 4 - incidencias (18/1/2021)

Apellidos y nombre: Gálvez López, Inmaculada

D.N.I.:20226294W

Observaciones:
- La duración de la prueba es de 60 minutos (que incluyen la operatoria para subir las respuestas a Prado).
- La ponderación de cada ejercicio aparece al inicio del enunciado. La calificación de cada tema se recogen en la Guía de la Asignatura.
- Utiliza esta hoja para responder a las cuestiones planteadas. Al finalizar y antes de que acabe el plazo deberá subir las soluciones a Prado en este documento, que se pasará por *Turnitin*.

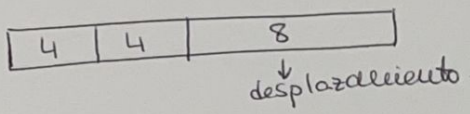
Tema 3:

1. [50%] Suponed un sistema con una memoria principal de 8192 Bytes y que utiliza paginación a dos niveles. Las direcciones lógicas son de 16 bits con la estructura: 4 bits para direccionar la tabla de páginas de primer nivel, 4 bits para las tablas de páginas de segundo nivel y 8 bits para el desplazamiento. El espacio de direcciones virtuales de un proceso tiene la estructura: código de la dirección 0 a 255, datos de la dirección 256 a la 675, y la pila que tiene 256 B de tamaño y la base en la dirección 62.976. Representad gráficamente las tablas de páginas de primer y segundo nivel y sus contenidos, suponiendo que que todas las páginas válidas de este proceso están cargadas en memoria principal en el orden que queráis (todos los marcos de página de memoria principal están libres). Dada esa asignación traduce la dirección virtual (a) 47 y la dirección virtual (b) 6523.

INMACULADA GÁLVEZ LÓPEZ

① MP - 8192 B

Direcciones - 16 bits



CÓDIGO - 0 a 255

DATOS - 256 a 675

PILA - 256 B, con base en 62976

Tamaño de página: $2^8 = 256$ B

Código: 256 B - 1 página
Datos: 420 B - 2 páginas (sobra espacio)
Pila: 256 B - 1 página.

•) Vemos dónde acaban el código y los datos: de la 0 a la $3 \cdot 256 - 1 = 767$

Cálculos:

$\frac{767}{2^4 \cdot 256}$ → Cociente: 0, Resto: 767

$\frac{767}{256}$ → Cociente = 2, Resto = 255

•) Vemos la pila:

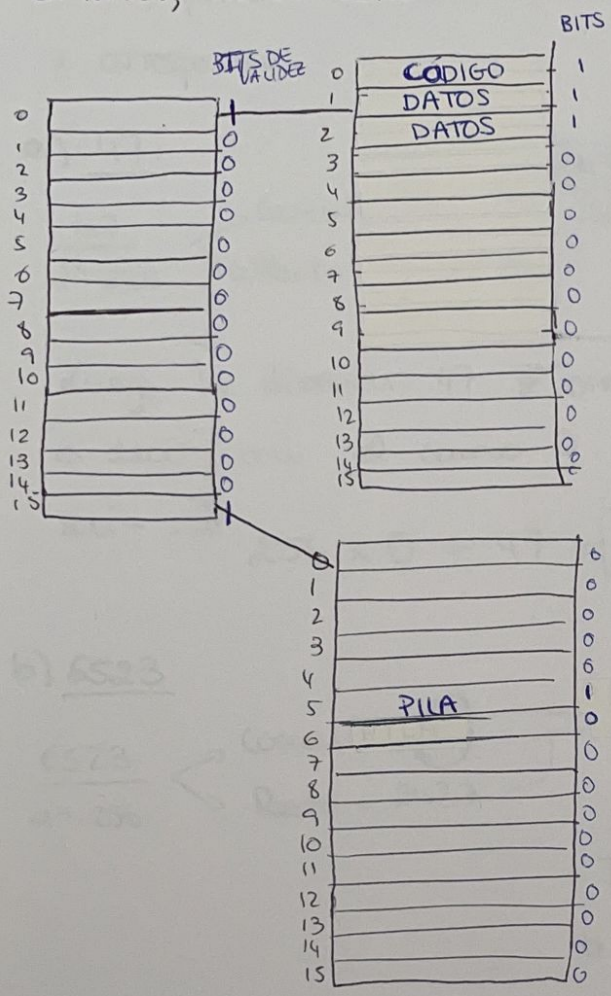
Base: $\frac{62976}{2^4 \cdot 256}$ → Cociente: 15, Resto: 1536

$\frac{1536}{256}$ → Cociente = 6, Resto = 0

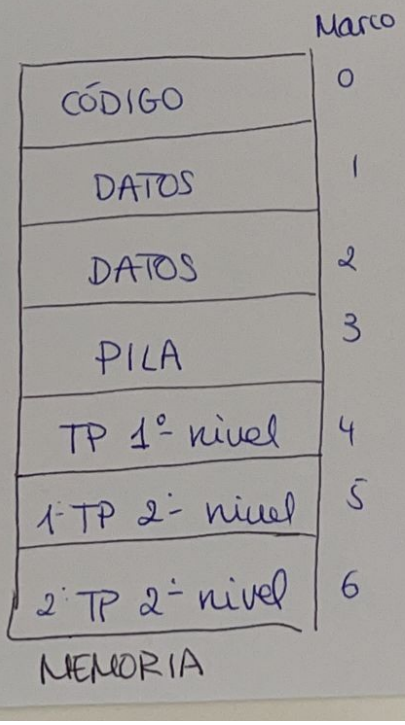
Inicio: $\frac{62721}{2^4 \cdot 256}$ → Cociente: 15, Resto: ~~761~~ 1281

$\frac{1281}{256}$ → Cociente = ~~6~~ 5, Resto = ~~255~~ 1

Entonces, la estructura de los TP queda



Vamos a poner los marcos en MP en el siguiente orden:



Ahora, para traducir las direcciones, vemos a qué marco se corresponden.

a) 47:

$\frac{47}{2^4 \cdot 256}$ → Cociente = 0, Resto = 47

$\frac{47}{256}$ → Cociente = 0, Resto = 47 (offset)

Luego la dirección 47 se corresponde con la página de código, es decir, con el marco 0. Por tanto, la dirección resultante sería:

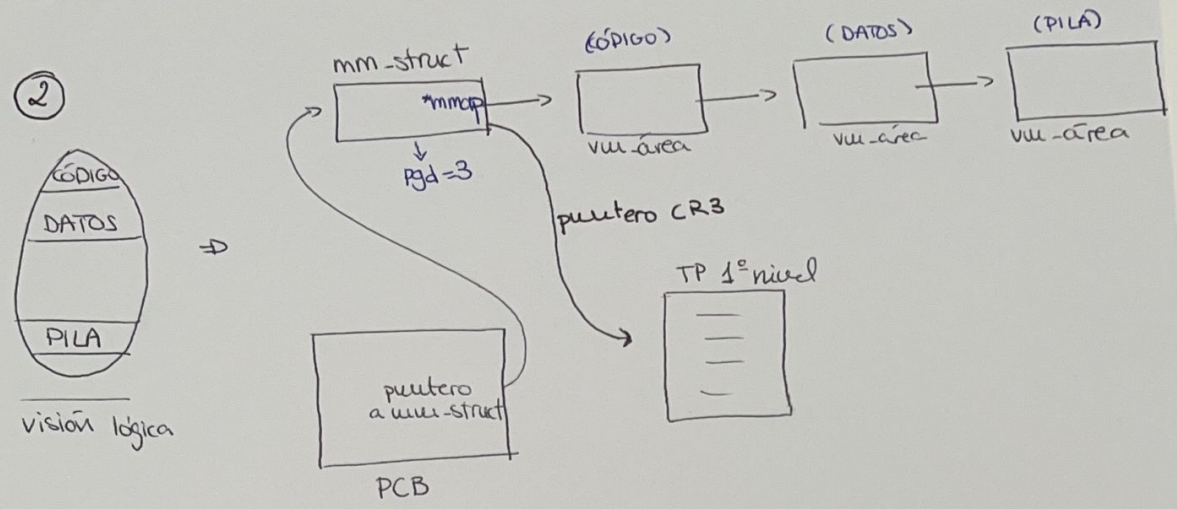
$$256 \times 0 + 47 = \boxed{47}$$

b) 6523

$\frac{6523}{2^4 \cdot 256}$ → Cociente = 1, Resto = 2427

No se corresponde con ninguna página válida, porque la TPE 1 de la TP de 1º nivel tiene el bit de validez a 0.

2. [50%] Para el proceso anterior, (a) dibujar las estructuras de datos que usa el sistema operativo Linux para describir su espacio de direcciones rellenado en ellas aquellos campos que sea posible. (b) ¿Cómo se modificaría la descripción del espacio dada en el apartado (a) de esta pregunta si el proceso solicita al sistema operativo 512B a través de la función `malloc()`?.



•) En la vm_área correspondiente al código:

* DIR_INICIO = 0
* DIR_FIN = 255

•) En la vm_área de datos:

* DIR_INICIO = 256
* DIR_FIN = 675

•) Vm_área de pila:

* DIR_INICIO = 62721
* DIR_FIN = 62976

b) Si el proceso solicita 512B a través de malloc, habría que construir una nueva vm_área, ~~reservando~~ reservando [redacted] dos nuevos marcos.

Tema 4:

1. Indicar cómo quedan las estructura de datos de memoria, y los contenidos de las mismas, relativas a la gestión de archivos en el sistema operativo Linux tras las ejecución por parte de un proceso de las siguientes llamadas al sistema:

```
...
fd-ent = open (“archivo”, O_RDONLY);      /* abrir “archivo” de lectura */
fd-sal = open (“archivo”, O_WDONLY);      /* abrir “” de escritura*/
lseek(fd-sal, 8, SEEK_SET);               /* salta los 8 primeros bytes */
close(0);                                 /* cierra entrada estandar */
dup (fd-sal);                             /* duplica fd-sal en 1 */
fork();                                   /* crea un hijo */
```

Como abrimos el mismo archivo dos veces, pero con distintos permisos, se crean dos estructuras de archivo abierto (una cin permiso de lectura y otra con permiso de escritura). Por tanto, en el files_struct tenemos dos punteros, uno apuntando a cada archivo abierto. Sin embargo, ambas estructuras archivo abierto apuntan al mismo inodo en memoria, ya que se trata del mismo archivo.

Cuando se crea un hijo con fork(), se heredan los descriptores de archivo abiertos por el padre.

2. Qué aspecto tienen las estructuras de datos de una entrada de directorio en los sistemas FAT y ext2. Indicar las ventaja/inconvenientes de una y otra.

Las entradas de directorio de fat32 apuntan al primer bloque de la tabla fat, y cada bloque apunta al siguiente, acabando con un EOF.

Ventajas: cuando monto un SA, lo que hago es leer en memoria la fat, y a partir de ahí, para reconstruir el acceso a un fichero, la FAT y por tanto los punteros, están en memoria, y el acceso es mucho más rápido.

Inconvenientes: cuando modifico el Sistema de archivos, se modifica primero en la FAT y por tanto, desde que modifico esta hasta que físicamente modifico el SA en disco, si el sistema cae, queda inconsistente.

Además, si borramos una entrada puede perderse toda la información

En cambio, en el caso de ext2, se divide el disco duro en conjuntos de bloques del mismo tamaño, en los que se almacenan metadatos.

Ventajas: a la hora de borrar, no hay peligro porque se pierda información