Grupo: …… NIA: …… Nombre y apellidos: ……..

Ejercicio 1

El gestor de memoria de un sistema operativo debe permitir que sea posible ejecutar concurrentemente 3 procesos, cuyos tamaños en bytes de cada uno de los segmentos que forman parte de sus imágenes de memoria son los siguientes:

PROCESO	CODIGO	DATOS	PILA
A	16384	8700	8192
B	2048	1000	1024
C	4096	2272	2048

Además, se sabe que se dispone de una memoria física de 16 Kbytes y que el espacio de direcciones del sistema es de 64 Kbytes.

Se pide:

1. Determinar si son viables tamaños de página de 1024 bytes y 512 bytes. (1 punto)
2. En el supuesto de que ambos tamaños de página sean posibles, justificar qué tamaño debería utilizar el sistema de paginación si se tiene en cuenta la fragmentación interna producida. (1 punto)
3. Si se desea que las entradas de la tabla de páginas dispongan de 3 bits para referenciar el marco de página ¿qué tamaño de página debería utilizarse sin tener en cuenta la ejecución de los procesos A, B y C? (0.5 puntos)

Grupo: …… NIA: …… Nombre y apellidos: ……..

Solución

Sabiendo que una página no puede contener partes de dos segmentos diferentes (código, datos o pila), se obtienen los siguientes valores:

PROCESO A	Nº bytes	Nº páginas 1024 bytes	Fragmentación interna	Nº páginas 512 bytes	Fragmentación interna
CODIGO	16384	16	0	32	0
DATOS	8700	9	516	17	4
PILA	8192	8	0	16	0

PROCESO B	Nº bytes	Nº páginas 1024 bytes	Fragmentación interna	Nº páginas 512 bytes	Fragmentación interna
CODIGO	2048	2	0	4	0
DATOS	1000	1	24	2	24
PILA	1024	1	0	2	0

PROCESO C	Nº bytes	Nº páginas 1024 bytes	Fragmentación interna	Nº páginas 512 bytes	Fragmentación interna
CODIGO	4096	4	0	8	0
DATOS	2272	3	800	5	288
PILA	2048	2	0	4	0

Grupo: …… NIA: …… Nombre y apellidos: ……..

- Con páginas de 1024 bytes se necesitan para poder ejecutar los 3 procesos cargar en memoria:

16+9+8+2+1+1+4+3+2 = 46 páginas

Puesto que el sistema tiene un espacio de direcciones de 64 Kbytes (65536 bytes), el número de páginas posibles sería:

65536 / 1024 = 2^{16} / 2^{10} = 2^{6} = 64 páginas.

Para poder ejecutar los procesos se necesitan 46 páginas al menos, y se dispone de 64 por lo que es posible utilizar páginas de 1024 bytes.

- Con páginas de 512 bytes se necesitan para poder ejecutar los 3 procesos cargar en memoria:

32+17+16+4+2+2+8+5+4 = 90 páginas

Puesto que el sistema tiene un espacio de direcciones de 64 Kbytes (65536 bytes), el número de páginas posibles sería:

65536 / 512 = 2^{16} / 2^{9} = 2^{7} = 128 páginas.

Para poder ejecutar los procesos se necesitan 90 páginas al menos, y se dispone de 128 por lo que es posible utilizar páginas de 512 bytes.

Por tanto, es viable utilizar cualquiera de los dos tamaños de página.

2. Teniendo en cuenta la fragmentación interna producida se elegirá aquel tamaño de página en el que se desperdicie menos memoria en las últimas páginas:

Con 1024 bytes ⇒ 516+24+800=1340 bytes desperdiciados

Con 512 bytes ⇒ 4+24+288=316 bytes desperdiciados

Por tanto se elegirán páginas de 512 bytes.

3. El número de marcos de página posibles en el sistema en función del tamaño de página es:

Páginas de 1024 bytes ⇒ 16384 / 1024 = 2^{14} / 2^{10} = 2^{4} ⇒ Se necesitan 4 bits para hacer referencia al marco de página.

Páginas de 512 bytes ⇒ 16384 / 512 = 2^{14} / 2^{9} = 2^{5} ⇒ Se necesitan 5 bits para hacer referencia al marco de página.

Grupo: …… NIA: …… Nombre y apellidos: ……..

Por tanto, ninguno de los dos tamaños de página propuestos es adecuado para los 3 bits de referencia al marco de página. Lo ideal es poder referenciar todos los marcos de página posibles en memoria física, por lo que se diseñaría un sistema con 2^3 = 8 marcos ⇒ 16384 / 8 = 2048 ⇒ Se deberían utilizar **páginas de 2048 bytes**.

Grupo: …… NIA: …… Nombre y apellidos: ……..

Ejercicio 2

Se tiene un computador con direcciones de 32 bits que usa memoria virtual utilizando la técnica de la segmentación paginada por demanda. Este sistema usa páginas de 4 KBytes y segmentos de entre 4 KBytes y 4 MBytes. La memoria RAM dispone de 2 GBytes y se dispone de una partición de 4 GBytes utilizada para implementar un volumen de SWAP con preasignación. El sistema implementa la política *copy-on-write*.

Se pide:

a) Indicar con detalle como es el proceso de traducción de la dirección virtual 0 de un proceso (donde empieza el código del programa), resaltando cómo se obtiene la dirección física y cuáles son las estructuras de datos necesarias.

b) Calcular cuál es el total de memoria física (RAM+disco) del sistema que pueden usar conjuntamente todos los procesos. ¿Y cuánto puede usar un solo proceso?

c) Dado el programa de nombre `multiplicar` con el siguiente código:

```
int ratio = 2;
int *valor;
int main (int argc, char *argv[])
{
    pthread_t pid;
    int aux;
    aux = atoi(argv[1]);  /*convierte de ASCII a entero*/
    pthread_create(pid,NULL,calcular,&aux);
    pthread_join(pid,NULL);
}

int calcular(int *entrada){
    valor = (int *) malloc (sizeof(int));
    (*valor) = ratio * (*entrada);
    sleep(1);
    printf ("El resultado es: %d\n",(*valor));
}
```

Indicar con el máximo detalle posible cual es el mapa de memoria del proceso creado al ejecutar en la `shell` el comando "`multiplicar 8`" y justo antes de comenzar la 1ª instrucción del programa. Es necesario remarcar al menos:

- Las regiones que lo componen (tamaño, posición en el mapa, permisos, etc.)
- Las páginas que lo componen (cantidad, contenido, etc.)

d) Describir con los mismos detalles del apartado de c) el mapa de memoria justo después de ejecutar la sentencia `sleep`.

e) Si la función `main` se cambiara por la siguiente:

```
int main (int argc, char *argv[])
```

Grupo: …… NIA: …… Nombre y apellidos: ……..

```
{
    int pid, aux;
    aux = atoi(entrada);  /* convierte de ASCII a entero */
    pid = fork();
    calcular (&aux);
}
```

Describir con los mismos detalles del apartado de c) el mapa de memoria justo después de ejecutar la sentencia `sleep` tanto en el padre como en el hijo.

f) Partiendo del código del apartado c) y suponiendo que en el momento de ejecutar la instrucción `sleep` se realiza un cambio de contexto y el sistema operativo envía al disco todas las páginas del proceso. Indicar cual es el mecanismo para enviar cada una de estas páginas al disco y, además, indicar donde se guardan cada una de las páginas del proceso en el disco.

g) Por último indicar los cambios producidos en la gestión de memoria del sistema desde que el proceso del apartado c) reanuda la ejecución al terminar el `sleep` hasta que se ejecuta la instrucción `printf`. Indique que estructuras de datos han sido creadas/modificadas, que páginas/regiones han sido creadas/modificadas/destruidas, que cantidad de memoria física (RAM+disco) ha sido reservada/liberada, etc.

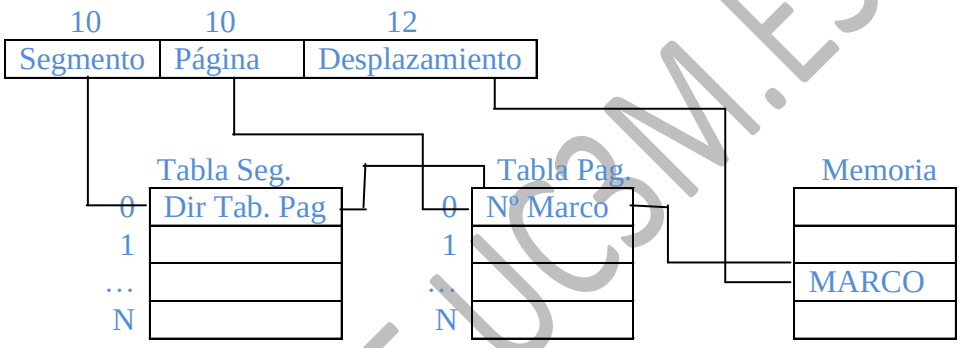
Grupo: …… NIA: …… Nombre y apellidos: ……..

Solución

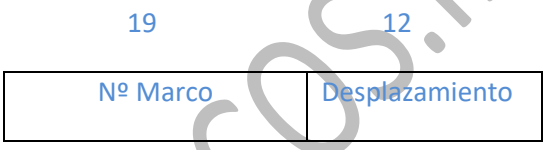
a)

Direcciones de 32 bits
Página 4KB ➔ 12 bits para desplazamiento dentro de la página
Máximo Tamaño segmento: 4MB ➔ 2^{10} páginas ➔ 10 bits para determinar la página de cada segmento
4 GB de Memoria Virtual.

Dirección virtual (32 bits)



Dirección física (31 bits).



Existe una tabla de segmentos por proceso.
Cada segmento tiene una tabla de páginas.
La tabla de segmentos contiene además el límite del segmento.
Si el total (Página, Desplazamiento) es mayor que el límite provoca un error.

b)

Como hay preasignación todos los procesos deben reservas sus páginas en SWAP, luego se desperdicia tanta SWAP como RAM haya. Así:

Total memoria = RAM + (SWAP – RAM) = 2GB + (4GB -2GB) = 4 GB

El total de memoria por proceso solo esta limitado por el tamaño de la dirección (cada segmento tiene como máximo 4MB pero un proceso puede tener innumerables segmentos de ficheros proyectados, memoria compartida, bibliotecas dinámicas, pilas de threads, etc.)

Grupo: …… NIA: …… Nombre y apellidos: ……..

Total memoria que puede usar un proceso = 2^{32} = 4GB

c)

0

Código	1 pagina, contiene el código Permisos de lectura y ejecución, tamaño fijo
Datos con V.I.	1 pagina, contiene la variable global ratio con valor 2 Permisos de lectura y escritura, tamaño fijo
Datos sin V.I.	1 pagina, contiene la variable global valor Permisos de lectura y escritura, tamaño fijo
…..	
Pila	1 pagina, contiene los parámetros del comando ejecutado (2, “multiplicar”, “8”) y la variables locales pid y aux Permisos de lectura y escritura, puede crecer

2^{32}

Grupo: …… NIA: …… Nombre y apellidos: ……..

d)

0

Código	1 pagina, contiene el código Permisos de lectura y ejecución, tamaño fijo
Datos con V.I.	1 pagina, contiene la variable global ratio con valor 2 Permisos de lectura y escritura, tamaño fijo
Datos sin V.I.	1 pagina, contiene la variable global valor Permisos de lectura y escritura, tamaño fijo
Heap	1 pagina, contiene la variable dinámica reservada con malloc y referenciada por el puntero valor Permisos de lectura y escritura, tamaño variable
…..	
Pila del thread	1 pagina, contiene los parámetros de la función calcular (entrada) Permisos de lectura y escritura, tamaño variable
…..	
Pila	1 pagina, contiene los parámetros del comando ejecutado (2, “multiplicar”, “8”)y la variables locales pid y aux Permisos de lectura y escritura, puede crecer

2^{32}

Grupo: …… NIA: …… Nombre y apellidos: ……..

e)

Proceso original

0

Código	1 pagina, contiene el código Permisos de lectura y ejecución, tamaño fijo
Datos con V.I.	1 pagina, contiene la variable global ratio con valor 2 Pagina compartida con proceso nuevo Permisos de lectura, bit de copy-on-write, tamaño fijo
Datos sin V.I.	1 pagina, contiene la variable global valor Permisos de lectura y escritura, tamaño fijo
Heap	1 pagina, contiene la variable dinámica reservada con malloc y referenciada por el puntero valor Permisos de lectura y escritura, tamaño variable
…..	
Pila	1 pagina, contiene los parámetros del comando ejecutado (2, “multiplicar”, “8”), los parámetros de la función (entrada) y la variables locales pid y aux Permisos de lectura y escritura, puede crecer

2^{32}

Proceso nuevo: Exactamente igual que el proceso original, solo cambia el valor de la variable pid.

Grupo: …… NIA: …… Nombre y apellidos: ……..

f) Siguiendo con el proceso del apartado c). En el momento de ejecutar la instrucción *sleep* se realiza un cambio de contexto y no se reanuda el proceso hasta 1 segundo después. Durante este tiempo el SSOO envía al disco todas las páginas del proceso. Indicar cual es el mecanismo para enviar una de estas páginas al disco y, además, indicar dónde se guardan cada una de las páginas del proceso en el disco.

Paginas a enviar:

- Código: Nunca se modifica, solo se lee del ejecutable.
- Datos con V.I.: No se ha modificado, se escribe en swap (por no ser constante).
- Datos sin V.I.: Se ha modificado (valor), se escribe en swap.
- *Heap*: Se ha modificado, se manda a swap.
- Pila del *thread*: Se ha modificado, se manda a swap.
- Pila: se ha modificado, se escribe en swap

Mecanismo de envió de páginas:

- El algoritmo de reemplazo selecciona la página para ser reemplazada.
- Si la página tiene soporte en swap y ha sido modificada se copia en el swap (si tiene soporte en otro fichero y se modificó se copiará en ese fichero).
- Se apunta en la tabla de páginas que dicha página está en el soporte que le corresponda (swap u otro).
- Si está en swap, se apunta en swap que la reserva para dicha página se encuentra ocupada.

Grupo: …… NIA: …… Nombre y apellidos: ……..

Ejercicio 3

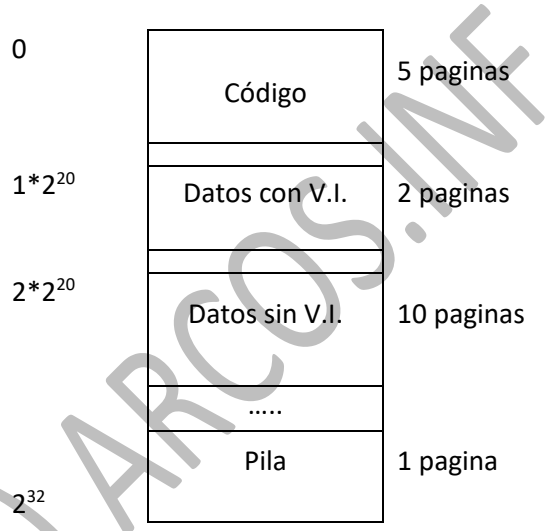
Se tiene un computador con direcciones de 32 bits que usa memoria virtual utilizando la técnica de la segmentación paginada por demanda (las paginas de un proceso solo se cargan en RAM cuando son utilizadas por el proceso). Este sistema usa páginas de 4 KBytes y segmentos de entre 4 KBytes y 64 MBytes. La memoria RAM dispone de 4 GBytes. El sistema dispone de un dispositivo de intercambio (swap) sin preasignación de 8 GBytes.

La tabla de páginas asociada dispone de los bits de página válida, página modificada y página accedida que son utilizados por la MMU. El sistema sigue una política de reemplazo local con una asignación fija de 5 páginas por proceso. El algoritmo de reemplazo es el algoritmo del reloj (o de segunda oportunidad).

Se pide:

a) Indicar cuál es el formato de la dirección virtual y de la dirección física del sistema, así como de la tabla de páginas empleada.

b) El proceso P1 tiene el siguiente mapa de memoria virtual en el momento de ser creado.



Al ejecutar el programa se producen la siguiente secuencia de accesos a memoria:

Nº operación	Región	Página	Tipo acceso
1	Código	1	Ejecución
2	Código	2	Ejecución
3	Datos con V.I.	1	Escritura
4	Datos sin V.I.	1	Escritura
5	Pila	1	Lectura
6	Código	1	Ejecución
7	Datos sin V.I.	2	Lectura
8	Código	3	Ejecución

Grupo: …… NIA: …… Nombre y apellidos: ……..

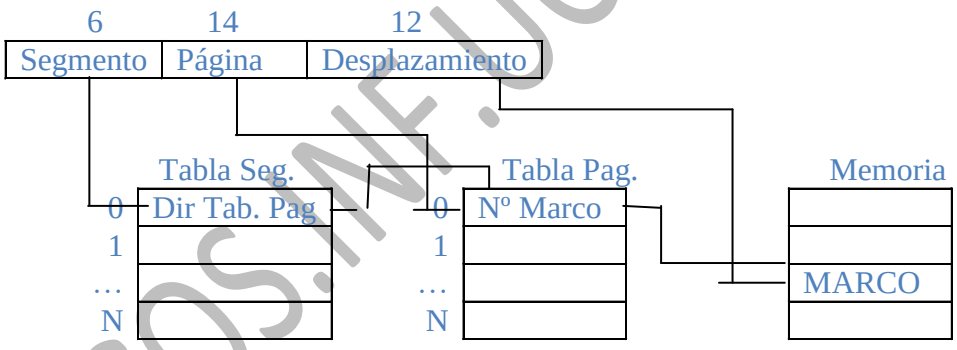
9	Pila	2	Escritura
10	Datos con V.I.	3	Escritura

Se pide, indicar con detalle que tarea realiza el gestor de memoria para cada uno de los accesos de la tabla. Deben incluirse los cambios que se producen en la memoria RAM, los cambios en el dispositivo de almacenamiento secundario, en la tabla de páginas y en el resto de estructuras del sistema operativo involucradas.

Solución

a) Página 4KB➔ 12 bits para desplazamiento dentro de la página
Máximo Tamaño segmento: 64MB ➔ 2^{14} páginas ➔ 14 bits para determinar la página de cada segmento
Direcciones de 32bits → Como uso 12 para desplazamiento y 14 para página me quedan 6 para el segmento

Dirección virtual (32 bits)



Dirección física (32 bits).

Existe una tabla de segmentos por proceso.

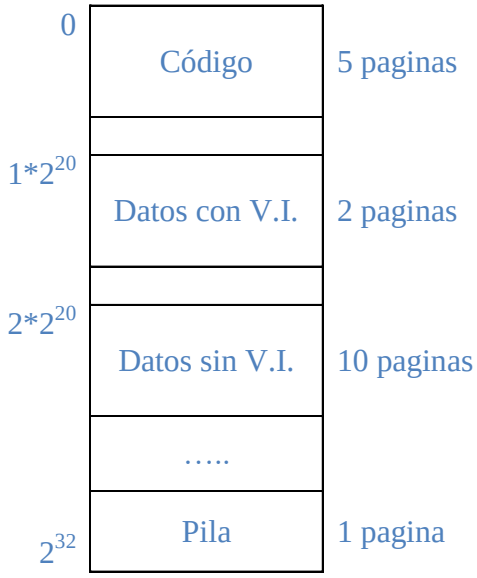
Cada segmento tiene una tabla de páginas.

La tabla de segmentos contiene además el límite del segmento.

Si el total (Página, Desplazamiento) es mayor que el límite provoca un error.

b) Mapa inicial del proceso: En memoria RAM solo está la página inicial de la pila.

Grupo: …… NIA: …… Nombre y apellidos: ……..



Lista de operaciones:

Nº operación	Región	Página	Tipo acceso
1	Código	1	Ejecución
2	Código	2	Ejecución
3	Datos con V.I.	1	Escritura
4	Datos sin V.I.	1	Escritura
5	Pila	1	Lectura
6	Código	1	Ejecución
7	Datos sin V.I.	2	Lectura
8	Código	3	Ejecución
9	Pila	2	Escritura
10	Datos con V.I.	3	Escritura

1. La página no es válida por lo que salta la excepción.
   La página se lee del ejecutable y se copia en RAM.
   Su entrada en la tabla se pone como válida
   Pag. Mem (alg. Reloj): Pila1, Codigo1

2. La página no es válida por lo que salta la excepción.
   La página se lee del ejecutable y se copia en RAM
   Su entrada en la tabla se pone como válida
   Pag. Mem (alg. Reloj): Pila1, Codigo1, Codigo2

3. La página no es válida por lo que salta la excepción.
   La página se lee del ejecutable, se copia en RAM y se escribe
   Su entrada en la tabla se pone como válida y modificada.
   Pag. Mem (alg. Reloj): Pila1, Codigo1, Codigo2, DatosCon1

4. La página no es válida por lo que salta la excepción.
   La página rellena a ceros en RAM y se escribe

Grupo: …… NIA: …… Nombre y apellidos: ……..

Su entrada en la tabla se pone como válida y modificada.
Pag. Mem (alg. Reloj): Pila1, Codigo1, Codigo2, DatosCon1, DatosSin1

5. La página es válida por lo se accede normalmente.
   Su entrada en la tabla activa el bit de accedida.
   Pag. Mem (alg. Reloj): Pila1, Codigo1, Codigo2, DatosCon1, DatosSin1

6. La página es válida por lo se accede normalmente.
   Su entrada en la tabla activa el bit de accedida.
   Pag. Mem (alg. Reloj): Pila1, Codigo1, Codigo2, DatosCon1, DatosSin1

7. La página no es válida por lo que salta la excepción.
Como ya hay 5 páginas en RAM del proceso se expulsa una.
Como Pila1 y Codigo1 tienen el bit accedido a 1 se expulsa (borra) Codigo2 y se ponen el bit de accedido de las páginas Pila1 y Codigo1 otra vez a 0. Ahora la primera página de la lista pasa a ser la siguiente a la reemplazada, es decir DatosCon1
La página rellena a ceros en RAM
Su entrada en la tabla se pone como válida.
Pag. Mem (alg. Reloj): DatosCon1, DatosSin1, Pila1, Codigo1, DatosSin2

8. La página no es válida por lo que salta la excepción.
Como ya hay 5 páginas en RAM del proceso se expulsa una y se pone a la siguiente a la expulsada como primera de la lista
Se expulsa DatosCon1, como ha sido modificada se guarda en SWAP.
La página se lee del ejecutable y se copia en RAM
Su entrada en la tabla se pone como válida.
Pag. Mem (alg. Reloj): DatosSin1, Pila1, Codigo1, DatosSin2, Codigo3

9. La página no es válida por lo que salta la excepción.
La página no pertenece al segmento (que inicialmente sólo tenía 1 página tal y como indica el enunciado) pero como es la última de la pila se incrementa su segmento en una página.
Como ya hay 5 páginas en RAM del proceso se expulsa una.
Se expulsa DatosSin1, como ha sido modificada se guarda en SWAP.
La página se rellena a ceros y se escribe
Su entrada en la tabla se pone como válida y modificada.
Pag. Mem (alg. Reloj): Pila1, Codigo1, DatosSin2, Codigo3, Pila2

10. La página no es válida por lo que salta la excepción.
    La página no pertenece al segmento se mata el proceso.

Grupo: …… NIA: …… Nombre y apellidos: ……..

Ejercicio 4

Se tiene un computador con direcciones de 32 bits que usa memoria virtual utilizando la técnica de la segmentación paginada por demanda. Este sistema usa páginas de 4 KBytes y segmentos de entre 4 KBytes y 4 MBytes. La memoria RAM dispone de 4 GBytes y se dispone de una partición de 8 GBytes utilizada para implementar un volumen de SWAP con preasignación. El sistema implementa la política *Copy-On-Write*.

Se pide:

a) Indicar con detalle como es el proceso de traducción de la dirección virtual 0 de un proceso (donde empieza el código del programa), resaltando como se obtiene la dirección física y cuáles son las estructuras de datos necesarias.

b) Calcular cuál es el total de memoria física (RAM+disco) del sistema que pueden usar conjuntamente todos los procesos. ¿Y cuánto puede usar un solo proceso?

c) Dado el programa de nombre `doblar` con el siguiente código:

```
int factor = 2;
int resultado;
int main (int argc, char *argv[])
{
    int dato;
    dato = atoi(argv[1]);  /*convierte de ASCII a entero*/
    resultado = factor * dato;
    sleep(1);
    fork();
    printf ("El resultado es: %d\n",resultado);
}
```

Indicar con el máximo detalle posible cual es el mapa de memoria del proceso creado al ejecutar en la shell el comando "`doblar 10`" y justo antes de comenzar la 1ª instrucción del programa. Es necesario remarcar al menos:

- Las regiones que lo componen (tamaño, posición en el mapa, permisos, etc.).
- Las paginas que lo componen (cantidad, contenido, etc.).

d) Suponiendo que existe suficiente memoria RAM libre en el computador, indicar la cantidad de memoria física (RAM+disco) reservada durante el proceso de carga e inicio del proceso apuntado en el apartado c) (antes de comenzar la 1ª instrucción).

e) Siguiendo con el proceso del apartado c). En el momento de ejecutar la instrucción `sleep` se realiza un cambio de contexto y no se reanuda el proceso hasta 1 segundo después. Durante este tiempo el SSOO envía al disco todas las páginas del proceso. Indicar cual es el mecanismo para enviar una de estas páginas al disco y, además, indicar donde se guardan cada una de las páginas del proceso en el disco.

Grupo: …… NIA: …… Nombre y apellidos: ……..

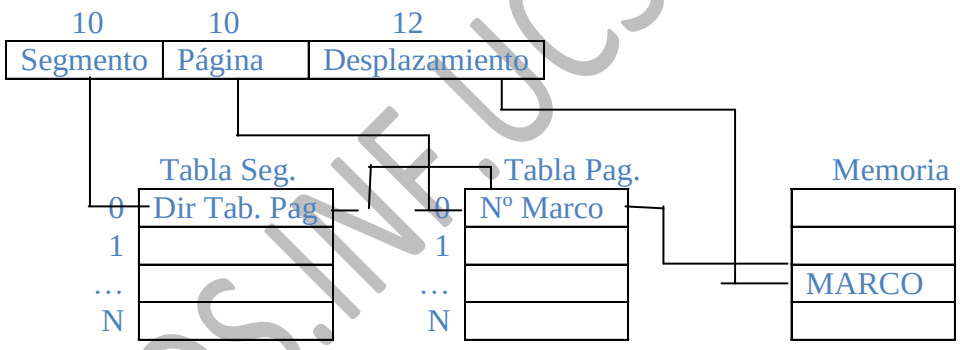
f) Por último indicar los cambios producidos en la gestión de memoria del sistema desde que el proceso del apartado c) reanuda la ejecución al terminar el `sleep` hasta que se ejecuta la instrucción `printf`. Indique que estructuras de datos han sido creadas/modificadas, que paginas/regiones han sido creadas/modificadas/destruidas, que cantidad de memoria física (RAM+disco) ha sido reservada/liberada, etc.

Solución

a)

Página 4KB➔ 12 bits para desplazamiento dentro de la página
Máximo Tamaño segmento: 4MB ➔ 210 páginas ➔ 10 bits para determinar la página de cada segmento
Direcciones de 32bits → Como uso 12 para desplazamiento y 10 para página me quedan 10 para el segmento

Dirección virtual (32 bits)



Dirección física (32 bits).

Existe una tabla de segmentos por proceso.
Cada segmento tiene una tabla de páginas.
La tabla de segmentos contiene además el límite del segmento.
Si el total (Página, Desplazamiento) es mayor que el límite provoca un error.

b) Calcular cual es el total de memoria física (RAM+disco) del sistema que pueden usar conjuntamente todos los procesos. ¿Y cuánto puede usar un solo proceso?
Como hay preasignación todos los procesos deben reservas sus páginas en SWAP, luego se desperdicia tanta SWAP como RAM haya. Así:

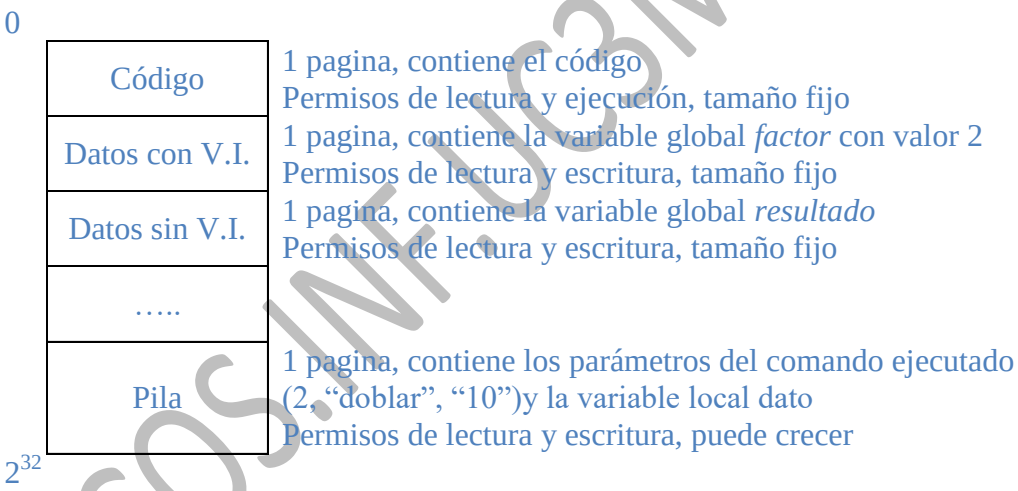
Total memoria = RAM + (SWAP – RAM) = 4GB + (8GB -4GB) = 8 GB

El total de memoria por proceso solo esta limitado por el tamaño de la dirección (cada segmento solo tiene 4MB pero un proceso puede tener innumerables segmentos de ficheros proyectados, memoria compartida, bibliotecas dinámicas, pilas de threads, etc.)

Total memoria por proceso = 2^{32} = 4GB

c) Dado el programa de nombre `doblar` con el siguiente código:
Indicar con el máximo detalle posible cual es el mapa de memoria del proceso creado al ejecutar en la shell el comando "`doblar 10`" y justo antes de comenzar la 1ª instrucción del programa. Es necesario remarcar al menos:

- Las regiones que lo componen (tamaño, posición en el mapa, permisos, etc.).
- Las páginas que lo componen (cantidad, contenido, etc.).



d) Suponiendo que existe suficiente memoria RAM libre en el computador, indicar la cantidad de memoria física (RAM+disco) reservada durante el proceso de carga e inicio del proceso apuntado en el apartado c) (antes de comenzar la 1ª instrucción).
Marcos en RAM = 4 paginas (las del apartado c)

Reserva de paginas en Swap = 3 paginas (todas menos el código que siempre se leerá del ejecutable).

TOTAL = 16KB +12 KB = 28 KB.

e) Siguiendo con el proceso del apartado c). En el momento de ejecutar la instrucción `sleep` se realiza un cambio de contexto y no se reanuda el proceso hasta 1 segundo después. Durante este tiempo el sistema operativo envía al disco todas las páginas del proceso. Indicar cual es el mecanismo para enviar una de estas páginas al disco y, además, indicar donde se guardan cada una de las páginas del proceso en el disco.
Paginas a enviar:

Grupo: …… NIA: …… Nombre y apellidos: ……..

- Código: Nunca se modifica, solo se lee del ejecutable.
- Datos con V.I.: No se ha modificado (*factor*), no se escribe en swap (se lee del ejecutable).
- Datos con V.I.: Se ha modificado (*resultado*), se escribe en swap.
- Pila: se ha modificado (dato, parámetros del comando), se escribe en swap

Mecanismo de envió de páginas.

- El algoritmo de reemplazo selecciona la página para ser reemplazada.
- Si la página tiene soporte en swap y ha sido modificada se copia en el *swap* (si tiene soporte en otro fichero y se modificó se copiará en ese fichero).
- Se apunta en la tabla de páginas que dicha página está en el soporte que le corresponda (*swap* u otro).
- Si está en *swap*, se apunta en *swap* que la reserva para dicha página se encuentra ocupada.

f) Por último indicar los cambios producidos en la gestión de memoria del sistema desde que el proceso del apartado c) reanuda la ejecución al terminar el `sleep` hasta que se ejecuta la instrucción `printf`. Indique que estructuras de datos han sido creadas/modificadas, que paginas/regiones han sido creadas/modificadas/destruidas, que cantidad de memoria física (RAM+disco) ha sido reservada/liberada, etc.
Solo influye la sentencia fork de la siguiente forma:

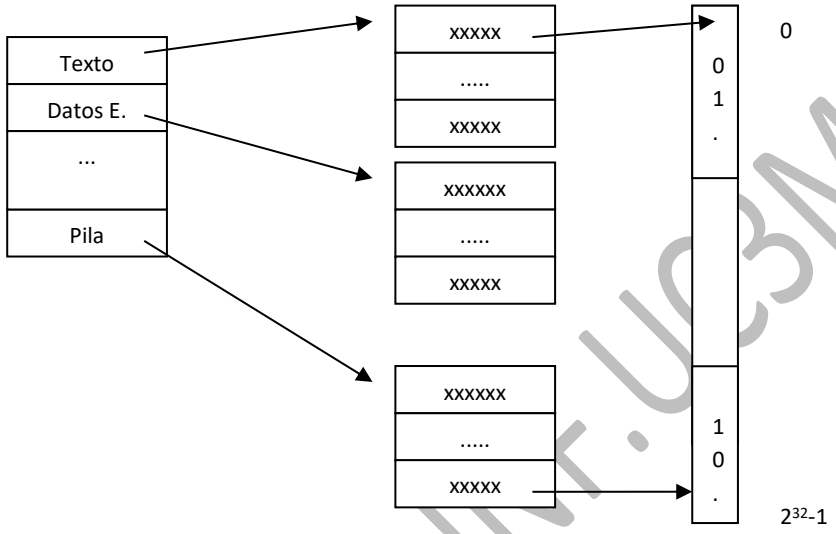
- Se crea un nuevo proceso que creará un nuevo BCP (copiado del padre salvo detalles como el PID), una nueva tabla de segmentos y nuevas tablas de páginas.
- Al disponer de *Copy-On-Write*, las páginas apuntaran a los mismos marcos de memoria que el padre, pero desactivando los permisos de escritura, (cuando uno de los dos escriba se realizaría la copia y se volvería a activar la escritura, como esto no ocurre, no se copia ningún marco).
- El proceso hijo reservara en *swap* espacio para todas sus páginas que lo necesitan (todas salvo el código).

Grupo: …… NIA: …… Nombre y apellidos: ……..

Ejercicio 5

Se dispone de un sistema operativo en un computador con memoria virtual.

La memoria virtual es gestionada por una MMU (*memory management unit*) que emplea un esquema de segmentación paginada como el que se representa en la siguiente figura:



Sobre este computador se implementa el modelo clásico de procesos de UNIX, disponiendo también de hilos (*threads*) a nivel de núcleo. Cada proceso consta de, al menos, cuatro segmentos: texto, datos estáticos, datos dinámicos y pila. El sistema operativo gestiona la memoria virtual según las siguientes políticas: Paginación por demanda y pila inicial en memoria. Adicionalmente soporta copy-on-write. En esta máquina los procesos usan 32 bits de direccionamiento y enteros de 32 bits, siendo el tamaño de página de 4 KB. Sobre este computador se desea ejecutar el siguiente programa:

Grupo: …… NIA: …… Nombre y apellidos: ……..

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include <sys/wait.h>

int    M1[1024] ;
int  * M2;

void *F1 ( void *tp )
{
       int i;
       for (i=0; i<1024; i++)
             M1[i] = 2 ;
             return 0 ;
       }
}

int main ( )
{
       int pid, pid2, status, ret ;
       pthread_t tid ;

       M2 = (int *)malloc(4000) ;
       /******************** A ******************/
       pid = fork() ;
       /******************** B ******************/
       switch (pid)  {
         case  0:  memmove(M2,M1,4096) ;
                   /******************** C ******************/
                   ret = pthread_create(&tid, NULL, F1, NULL);
                   if (ret != 0)  perror("pthread_create") ;
                   /******************** D ******************/
                   break ;
          case -1: perror("fork") ;
                    break ;
          default:
                    do { pid2=wait(&status) ; } while (pid2!=pid) ;
       }
       exit (0) ;

}
```

Grupo: …… NIA: …… Nombre y apellidos: ……..

Considerando que:

- El segmento de texto tiene un tamaño de 2 Kbytes y comienza en la dirección virtual 0.
- El segmento de datos tiene un tamaño de 6 Kbytes y comienza en la dirección virtual 2097152 (0x200000).
- El segmento de pila tiene un tamaño de 4 Kbytes y comienza en la última dirección virtual que permite direccionar este computador y crece hacia direcciones decrecientes de memoria.
- El programa principal comienza en la dirección virtual 0.
- El vector M1 comienza en la posición 0 con respecto al comienzo del segmento de datos.
- Solo se accede a una página del segmento de pila por hilo.
- El tamaño máximo de pila es de 31 páginas más 1 página extra de frontera por seguridad.

Se pide:

1. Indicar el formato de las tablas de segmento y de página, incluyendo los distintos atributos típicos de las tablas de páginas (protección, etc.)
2. Representar de forma gráfica las tablas de página del proceso en el punto marcado con /*A*/ y /*B*/, incluyendo los punteros a las páginas.
3. En el punto marcado con /*C*/, ¿se puede producir una llamada al sistema?. ¿Y una excepción?. ¿Y una interrupción software?. ¿Y una interrupción?. Razone su respuesta, indicando que pasaría en cada caso.
4. Suponiendo que se ejecuta hasta el punto /*D*/ normalmente, represente de forma gráfica las tablas de página en el punto /*D*/ para todos los procesos.

Solución

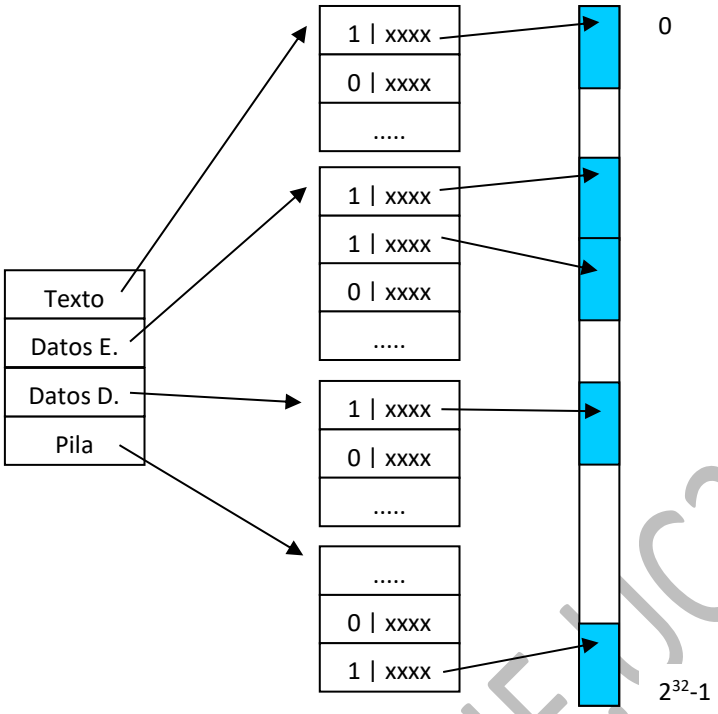
1) La tabla de segmentos dispone de una entrada por segmento, siendo los campos más comunes de esa entrada:
    a. Permisos: de lectura, de escritura y ejecución
    b. Registro base: delimita la posición de la tabla de páginas asociada al segmento
    c. Registro límite: indica el tamaño de la tabla de páginas

   La tabla de páginas dispone de una entrada por página, siendo los campos más comunes de cada página:

    a. Válida: indica si la página asociada es válida o no.
    b. Presente: indica si está presente en memoria principal.
    c. Referenciada: indica que se ha accedido para lectura o escritura esta página.
    d. Modificada: indica que se ha modificado el contenido de la página.
    e. NoCache: indica que no ha de desalojarse (siempre presente en memoria).
    f. Marco de página: es parte de la dirección de la página.

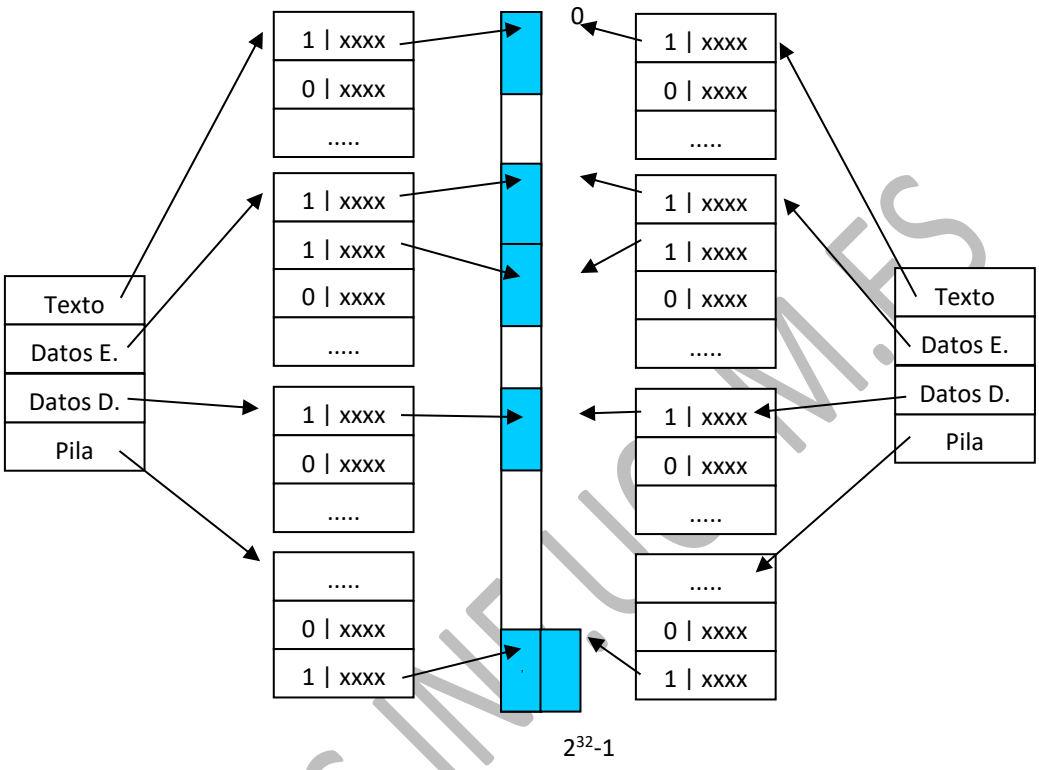
2) La representación de las tablas de página en el punto /*A*/ es:

Grupo: …… NIA: …… Nombre y apellidos: ……..



Grupo: …… NIA: …… Nombre y apellidos: ……..

La representación de las tablas de página en el punto /*B*/ es:



Después del *fork* se tiene dos procesos que comparten las páginas con el indicador de COW (*copy on write*) activado, de manera que cuando se realice el *memmove* (copiar de una zona de memoria a otra) se copiarán realmente las páginas.
Dado que los dos procesos la función *fork* devuelve valores diferentes, en pila (donde está la variable local que almacena el resultado devuelto por el *fork*) se duplicará la página.

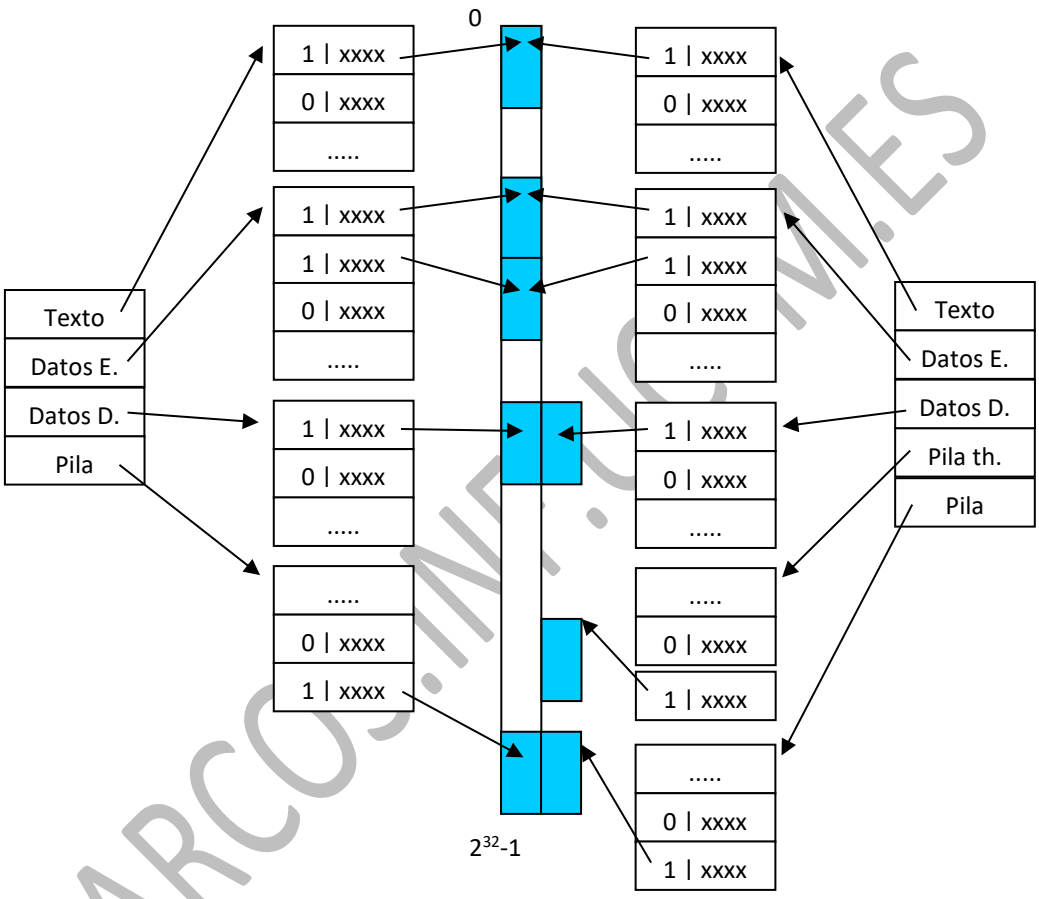
3) En el punto marcado como /*C*/ se puede dar:
   a. Una interrupción, por ejemplo por un dispositivo que precisa atención por parte del sistema operativo
   b. Un excepción, por acceso a posiciones de memoria que no han sido asignadas al proceso (la función malloc reserva 4000 bytes y sin embargo se accede a 4096 bytes).

No se ha de dar:

   c. Una interrupción software puesto que se ejecutan al final de los tratamientos de interrupción y antes de volver a ejecutar un proceso de usuario en CPU.

d. Una llamada al sistema, puesto que *memmove* no produce ninguna llamada al sistema y no se realiza ningún TRAP.

4) En el punto /*D*/ la representación es como sigue:



Como se puede observar, un *thread* comparte los segmentos de código y datos, pero no el de pila. Las páginas de pila pueden reservarse cuando se crea el *thread*.

Grupo: …… NIA: …… Nombre y apellidos: ……..

Ejercicio 6

Sea un computador que dispone de 36 MB de memoria principal y cuyo sistema operativo ocupa 4 MB sin incluir las estructuras necesarias para el gestor de memoria. En este computador se prevé la ejecución de programas con un espacio de direcciones lógico compuesto por tres segmentos: texto, datos y pila, siendo los tamaños medios de estos segmentos: 264 KB, 124 KB y 124 KB, respectivamente.

Se desea implementar un gestor de memoria, siendo los posibles esquemas de gestión a implementar (suponiendo que el hardware puede soportar todos ellos) los siguientes:

1. Asignación continua con particiones dinámicas, tal que cada proceso ocupa una única partición.
2. Paginación simple.
3. Segmentación simple.
4. Memoria virtual paginada.
5. Memoria virtual con segmentación paginada.

Para los sistemas basados en páginas, considere que el tamaño de las páginas es de 4 KB y que cada entrada de la tabla de páginas ocupa 32 bits. En el caso de esquemas con segmentación, el tamaño máximo del segmento es de 16 MB. Por último, suponga que en los modelos con memoria virtual se usa una política de asignación fija que otorga a cada proceso un número fijo de 16 marcos de página.

Se pide analizar para cada uno de estos esquemas los siguientes aspectos:

a. La memoria utilizada por el sistema de gestión de memoria para las estructuras de datos necesarias.
b. La memoria perdida debido al esquema de gestión de memoria utilizado, indicando el motivo de dicha pérdida.
c. El grado de multiprogramación que se puede alcanzar en el sistema con los tamaños medios propuestos para los procesos
d. El tamaño máximo del proceso que se puede ejecutar en cada uno de los distintos esquemas de gestión de memoria.

NOTA: en los casos en que no se disponga de otros datos, y sea aplicable, considérese que se pierde un 25 % de la memoria debido al esquema de gestión utilizado.

Grupo: …… NIA: …… Nombre y apellidos: ……..

Solución

Dado el espacio ocupado por el S.O., quedan 32 MB para la ejecución de los programas y para las estructuras de datos del gestor de memoria.

1. Asignación contigua

En este esquema de gestión de memoria, las particiones son dinámicas tanto en número como en tamaño. Cuando un proceso se trae a memoria se le asigna exactamente la memoria que necesita y no más.

a) En este caso basta con un par de palabras por proceso que indiquen la dirección de memoria donde comienza el proceso (registro base) y la longitud del mismo (registro límite). Con este par de palabras se resuelve tanto el problema de la reubicación como el de la protección.

Aparte de estas dos palabras, el sistema de gestión de memoria necesita conocer en todo momento, qué partes de la memoria están disponibles (huecos). Para ello requerirá alguna estructura como puede ser una lista enlazada.

b) En este esquema de gestión de memoria se produce fragmentación externa. Ésta se presenta cuando en memoria existe suficiente espacio para cargar un nuevo proceso, pero sin embargo éste no es contiguo; la memoria está compuesta por bloques que son demasiado pequeños para contener un proceso.

Según el enunciado, la memoria que se pierde como consecuencia de la fragmentación externa es un 25% de la memoria disponible, es decir, 8 MB.

c) El tamaño medio de un proceso en este computador es de 512 KB. Dado que la cantidad de memoria que en un momento determinado se pierde como consecuencia de la fragmentación externa es de 8 MB, los procesos ocupan en el sistema 24 MB. Por lo tanto el grado de multi-programación será:

n = (24-1024)7512 = 48 procesos.

d) El tamaño máximo del proceso que se puede ejecutar en este computador viene determinado por el tamaño de la memoria física disponible para la ejecución de programas de aplicación, que en este caso es de 32 MB, menos el espacio ocupado por las estructuras consideradas en el apartado a.

2. Paginación simple

Grupo: …… NIA: …… Nombre y apellidos: ……..

Utilizando paginación simple tanto el proceso como la memoria se dividen en un conjunto de particiones fijas y de igual tamaño denominadas páginas y marcos de página respectivamente. Cuando se introduce un proceso en el sistema, se cargan todas sus páginas en la memoria.

a) En este caso se necesita una tabla de páginas por proceso que permita realizar la traducción de direcciones lógicas a físicas, indicando la dirección de memoria física donde se encuentra el marco correspondiente a cada página del proceso. Asimismo se necesita una lista de marcos libres.

b y c) El número medio de páginas que ocupa cada proceso es **512/4 = 128 páginas**. El tamaño mínimo que ocupa la tabla de páginas de cada proceso en memoria es 4*128= 512 bytes (teniendo en cuenta que cada entrada de la tabla de páginas ocupa 4 bytes).

La suma de todas las tablas de páginas más la de marcos libres es de (36MB totales -4MB de SO)*1024/4= **8192** (el número total de marcos disponibles), luego el espacio total requerido para las tablas de páginas es de 8192 *4 Bytes = 32768 KB = **8 páginas de 4KB**. En realidad el gestor de memoria necesitará más espacio, puesto que las tablas de páginas son elementos dinámicos que se crean y destruyen, lo que hace que su gestión requiera más espacio que el mínimo calculado.

La memoria que se pierde en este caso se debe a la fragmentación interna que se puede producir en la última página de cada proceso, y que en media es la mitad de una página, es decir, 2 KB por proceso. Para calcular la memoria que se pierde como consecuencia este tipo fragmentación, es necesario calcular en primer lugar el grado de multiprogramación, que en este caso es:

n=(8192 pags totales - 8 pags usadas para tablas de pags/(512KB por proceso/4KB por pag)= 8184 pags disponibles para procesos /128 pags ocupadas por cada proceso = 63 procesos.

Según esto, la memoria que se pierde en el sistema como consecuencia de la fragmentación interna es 2 KB * 63 = 126 KB, lo que representa menos del 1% de la memoria disponible para la ejecución de programas de usuario.

d) El tamaño máximo del proceso que puede ejecutar en este caso viene determinado por el tamaño de la memoria principal, descontando el espacio necesario para representar la tabla de páginas del proceso, y que en este caso es 8184 páginas.

Por lo tanto el tamaño máximo de un proceso será 8184 páginas *4KB por pag=32736 KB.

3. Segmentación simple

Empleando segmentación simple, el programa se divide en un conjunto de segmentos, no siendo necesario que todos los segmentos tengan la misma longitud ni estén contiguos en memoria. En este caso cada proceso constará de un segmento de texto, uno de datos y otro de

pila. Cuando se trae un proceso a memoria será necesario cargar todos los segmentos del proceso en la misma.

a) El empleo de segmentación resulta muy similar al de particiones dinámicas. Será necesaria una pequeña tabla de segmentos, bastando en este caso con un conjunto de registros base y límite para cada uno de los tres segmentos, es decir, 6 palabras por proceso.

Al igual que en el caso de particiones dinámicas, el sistema de gestión de memoria necesita algún tipo de estructura para conocer la ocupación de la memoria en todo momento, siendo aplicable lo comentado en el apartado 1.

b y c) En este esquema de gestión de memoria se produce al igual que en el apartado 1 fragmentación externa, que en este caso vuelve a ser de 8 MB (el 25% de la memoria disponible para programas de usuario). De igual forma, el grado de multiprogramación que se puede alcanzar es equivalente al del apartado 1 (48 procesos), con la diferencia de que los procesos no se encuentran contiguos en memoria.

Aunque en ambos casos, la fragmentación externa que se produce es la misma, ya que según el enunciado se pierde el 25% de la memoria, esto no es lo que ocurre en situaciones reales. Generalmente cuando se reduce el tamaño del segmento también se reduce la memoria que se pierde como consecuencia de la fragmentación externa. Para comprobar esto, vamos a suponer que el porcentaje que se pierde como consecuencia de la dicha fragmentación es un 50% del tamaño del segmento que se ubica en memoria y que se cumple **la regla de cincuenta por ciento según la cual si el número de segmentos en memoria es n el número de huecos es n/2.**

En el caso de particiones dinámicas, se produce fragmentación externa cuando los huecos existentes en memoria son de tamaño menor a 512 KB (el tamaño medio de un proceso en este computador). Si el tamaño del hueco es un 50% del tamaño de segmento a ubicar (512 KB), entonces se cumple que la cantidad de memoria ocupada por los procesos más la que ocupan los huecos es igual a la memoria total, es decir:

Memora ocupada por procesos (n procesos) + memoria ocupada por huecos no aprovechables (hay n/2 huecos de la mitad de tamaño que los procesos = Memoria total disponible (32MB) ➔ 512KB *n + n/2*(0.5 * 512 KB) = 32*1024 KB -> n = 51 procesos

Cuando se emplea segmentación simple, la fragmentación externa viene determinada por el tamaño del segmento más pequeño, en este caso 124 KB. El número de segmentos es 3n y, por lo tanto, el número de huecos 3n/2. El grado de multiprogramación para este caso será:

512*n + (3n/2) * (124*0.5) = 32 *1024 -> n = 54 procesos.

Por lo que podríamos albergar más procesos en memoria utilizando segmentación simple.

d) El máximo proceso que se puede ejecutar en este computador viene determinado por la memoria física disponible, que es 32 MB, menos el espacio ocupado por las estructuras consideradas en el apartado a.

4. Memoria virtual paginada

En este caso, a diferencia de un esquema de paginación simple, no es necesario que todas las páginas del proceso se encuentren en memoria principal para su ejecución.

a) Se necesita, al igual que en el apartado 2, una tabla de páginas por proceso cuyo tamaño mínimo (ver apartado 2) será de 512 bytes.

b y c) La memoria que se pierde se debe a la fragmentación interna que se produce en la última página del espacio de direcciones virtuales del proceso. Dado que el gestor de memoria otorga a cada proceso un número fijo de 16 marcos de página, la probabilidad de que la última página del proceso se encuentre en memoria principal en un momento determinado es 16/128. Para calcular cantidad de memoria total perdida como consecuencia de esta fragmentación, determinaremos previamente el grado de multiprogramación en el sistema.

El tamaño que ocupa cada proceso en memoria principal viene dado por los 16 marcos de página asignados a cada proceso y por el tamaño de la tabla de páginas cuyo tamaño mínimo es de 512 bytes. Según esto el grado de multiprogramación será:

n * (16 páginas *4KB por página. + 0,5 KB ocupados por la tabla de páginas) = 32 *1024 KB —> n = 508 procesos, un 700% mayor que el obtenido en el apartado 2.

De acuerdo con esto la mínima memoria empleada para las tablas de página será de 508*512 bytes = 254 KB, y la memoria total que se pierde debido a la fragmentación interna será la mitad (2kB) de la última página de cada uno de los 508 procesos pero como sólo tenemos en memoria 16 de las 128 páginas (=512KB) de cada proceso: (16/128)*508*2 KB = 127 KB, prácticamente igual que empleando paginación simple.

d) El tamaño máximo del proceso que se puede ejecutar utilizando memoria virtual paginada vendrá determinado por el menor de:

a) la cantidad de memoria que permita direccionar la arquitectura de este computador;

b) el tamaño reservado para el área de *swap* (más la memoria I principal en su caso).

5. Memoria virtual con segmentación paginada

En este caso el proceso consta de tres segmentos, cada uno de los cuales a su vez se pagina para evitar de esta forma el problema de la fragmentación externa que aparecía en el caso de segmentación simple.

Grupo: …… NIA: …… Nombre y apellidos: ……..

a) Se necesita una estructura para representar cada una de las tres regiones del proceso. Esta estructura contendrá, para cada región: la dirección virtual de comienzo del segmento, la longitud del mismo y la dirección de la tabla de páginas correspondiente al segmento. Por lo tanto se necesitarán **tres palabras por región**, es decir, **nueve palabras por proceso=**9*4 bytes

El mínimo tamaño necesario para cada una de estas tablas es el siguiente: (264Kb /4 KB por página=) 66 entradas para el segmento de texto, es decir, 66* 4bytes = 264 bytes, y (124KB/4KB=) 31 entradas, 31 * 4 = 124 bytes, para los segmentos de datos y pila.

b y c) La memoria que se pierde empleando memoria virtual con segmentación paginada se debe a la fragmentación interna que se produce en la última página del espacio de direcciones virtuales correspondiente a cada segmento, es decir, se puede producir fragmentación interna en tres páginas del proceso (en memoria virtual). De forma análoga al apartado anterior, la probabilidad de que alguna de estas páginas sea la que se almacene en los 16 marcos de memoria que se asignan a cada proceso es (3*16)/128. Para calcular la memoria total perdida como consecuencia de esta fragmentación, calcularemos en primer lugar el grado de multiprogramación.

El número de procesos que puede haber en memoria viene determinado por la cantidad de páginas residentes en memoria, 16 en este caso, y por el tamaño ocupado por las tablas de segmentos y de páginas, cuyo tamaño mínimo en este caso es de 264 + 124 + 124 + (9*4) = 548 bytes. Según esto:

n*(16*4KB por página + 548/1024) = 32*1024 KB-> n = 507 procesos.

La memoria perdida debido a la fragmentación interna en este caso es:

(3*16/128)*507*2 KB = 380 KB.

d) El tamaño máximo del proceso que se puede ejecutar en este caso viene determinado por el tamaño máximo de cada uno de los segmentos que es de 16 MB. Por lo tanto, el tamaño máximo será 16*3 = 48 MB.

Grupo: …… NIA: …… Nombre y apellidos: ……..

Ejercicio 7

Se tiene un ordenador en el que las direcciones virtuales son de 32 bits y posee una gestión de memoria con segmentación paginada. El tamaño de la página es de 1KB y se quiere acceder de forma independiente a cada byte. Se desea tener un mínimo de 200 regiones para cada proceso.

A.- Indicar las partes en las que se dividiría la dirección virtual indicando el número de bits de cada una para maximizar el número de páginas disponibles en cada región.

B.- Si se tiene una memoria física de 1GB. Indicar el formato de la dirección física: partes que la componen y número de bits de cada parte.

C.- Un proceso desea acceder a la dirección virtual $A3B5F3C7_{(16}$; Desglosar esa dirección en sus componentes.

D.- ¿Cuántas tablas de páginas tiene asociadas, como máximo, un segmento de un proceso? ¿Cuántas entradas como máximo puede tener una tabla de páginas? ¿Cuál es el tamaño máximo de un segmento?

Solución

A.-

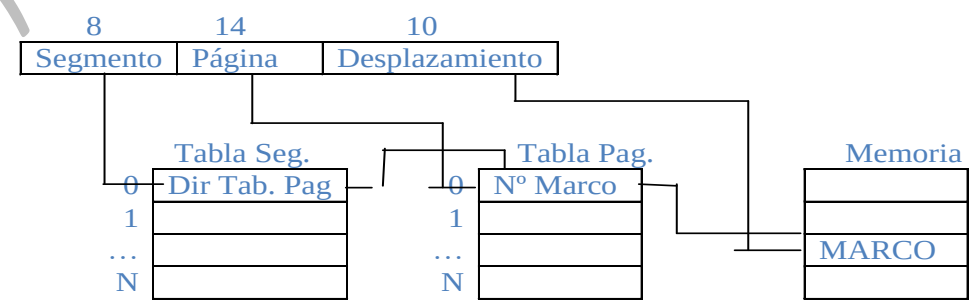
1KB = 2^{10} Bytes luego la parte de la dirección dentro de la página ocupa 10 bits.

Como queremos al menos 200 regiones tenemos que 200 <2^8 y por tanto para las regiones necesitamos al menos 8 bits.

Luego quedan 32-10-8= 14 bits para direccionar el número de página

Dirección virtual:

Dirección virtual (32 bits)



Grupo: …… NIA: …… Nombre y apellidos: ……..

B.- Dirección Física

Dirección física (32 bits). Aunque para direccionar 1GB=2^{30}B sólo se necesitarían 30 (10 para el byte dentro de la página y 20 para el nº del marco de página En nuestro caso los 2 bits más a la izquierda de la dirección física de 32 bits estarán a 0

2+20	10
Direc. del marco de página	Direcc del byte dentro de la página

C.- $A3B5F3C7_{(16}$ = 1010-0011-1011-0101-1111-0011-1100-0111 $_{(2}$

A	3	B	5	F	3	C	7
1010	0011	1011	0101	1111	0011	1100	0111

Los 10 bits menos significativos identifican el byte dentro de la página: $11\text{-}1100\text{-}0111_{(2}$

Los siguientes 14 identifican la página: $1011\text{-}0101\text{-}1111\text{-}00_{(2}$

Los 8 primeros identifican el segmento: $1010\text{-}0011_{(2}$

Dirección virtual desglosada

Segmento	página	Desplazamiento
$1010\text{-}0011_{(2}$	$1011\text{-}0101\text{-}1111\text{-}00_{(2}$	$11\text{-}1100\text{-}0111_{(2}$

D.-

Un segmento de un proceso puede tener asociadas como máximo una tabla de páginas.

Cada tabla de páginas tendrá 2^{14} entradas

El tamaño máximo de un segmento es 2^{24} bytes (16MB)

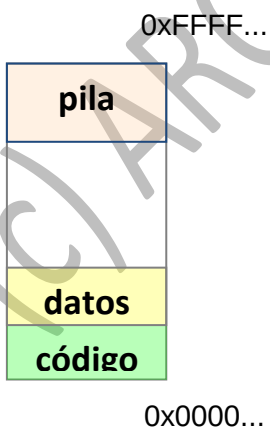
Grupo: …… NIA: …… Nombre y apellidos: ……..

Ejercicio 8

Se dispone del siguiente código fuente de un programa:

```
pthread_t   thid;

void *task1 ( void * arg )
{
    arg = malloc (100*sizeof(int)) ;
    /* punto B */
}

main (int argc, char **argv)
{
          int *pint;
          int   pid, i;

          /* punto A */
          pthread_create(&thid,NULL,task1,(void *)pint) ;
          pthread_join(&thid,NULL) ;

          pid =  fork() ;
          if (0 == pid) {
              free(pint) ;
              /* punto C */
          }
}
```

Si en el punto A de ejecución, la imagen del proceso asociado se representa:



Se pide:

a) Utilizando la representación dada, dibuje la imagen de los procesos en los puntos **B y C** si la gestión de memoria **no** usa COW (*Copy-On-Write*)

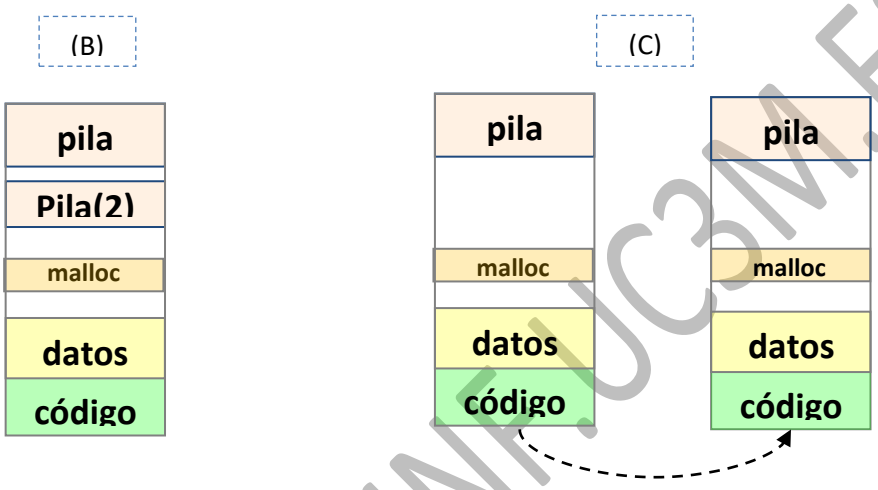
Grupo: …… NIA: …… Nombre y apellidos: ……..

b) ¿Considera que hay algún fallo en el uso de las llamadas de gestión de memoria en el fragmento de programa dado? Razone brevemente su respuesta.

NOTA: recuerde indicar las regiones de memoria que se necesite añadir o quitar para cada uno de los procesos/hilos de ejecución existentes.

Solución

a)



b) Si, que arg sobre la que se hace malloc es una variable local, por lo que cuando se deje de ejecutar la función se perderá el acceso a la zona de memoria pedida.
Posteriormente en el free se podría producir una violación de segmento.

Grupo: …… NIA: …… Nombre y apellidos: ……..

Ejercicio 9

Se dispone del siguiente código fuente de un programa a ejecutar en un sistema tipo UNIX:

```
pthread_t   thid;

void *task1 ( void * arg )
{
      /* punto B */
       pid =  fork() ;
       if (0 == pid) {
              *arg = (int *) malloc (100*sizeof(int)) ;
              /* punto C */
       }
}

main (int argc, char **argv)
{
          int *pint;
          int   pid, i;

          /* punto A */
         pthread_create(&thid,NULL,task1,(void *)&pint) ;
         pthread_join(&thid,NULL) ;
          /* punto D */
 }
```

Si en el punto A de ejecución, la imagen del proceso asociado se representa:

Se pide:

a) Indique en qué región de memoria estarán los elementos **thid**, **pid** y **main**. **(0,25 puntos)**
b) Dibuje la imagen de todos los procesos existentes en el punto **B**.
c) Dibuje la imagen de todos los procesos existentes en el punto **C**.
d) Dibuje la imagen de todos los procesos existentes en el punto **D**.

Para los tres últimos apartados:

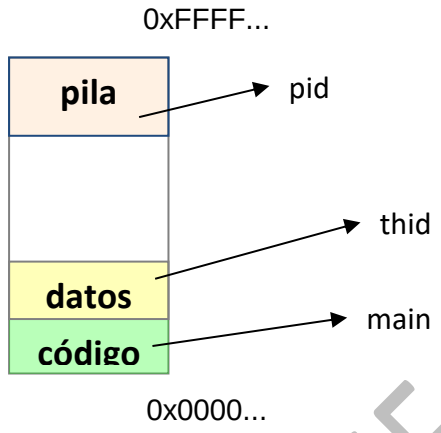
Grupo: …… NIA: …… Nombre y apellidos: ……..

- Utilice la representación dada.
- Indique las regiones de memoria que se necesite añadir o quitar con respecto al anterior.

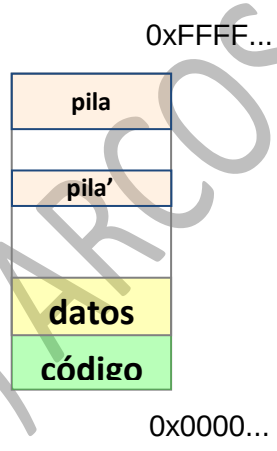
NOTA: En la gestión de memoria no se usa COW (*Copy-On-Write*)

Solución

a) Indique en qué región de memoria estarán los elementos **thid**, **pid** y **main**.

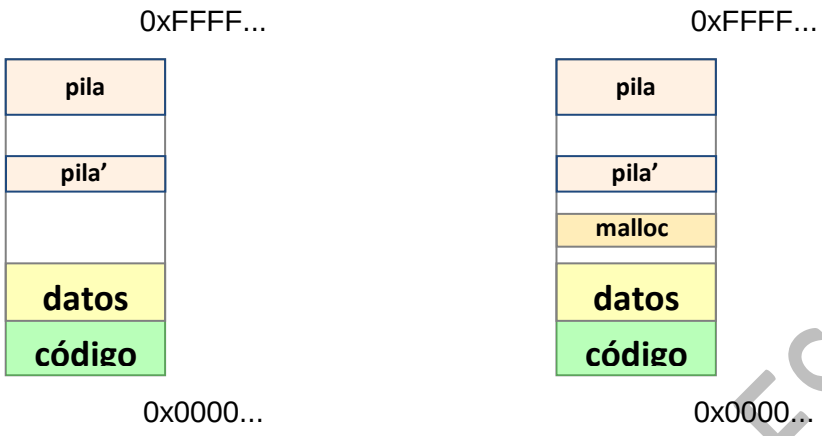


b) En **B** se ha creado un hilo:

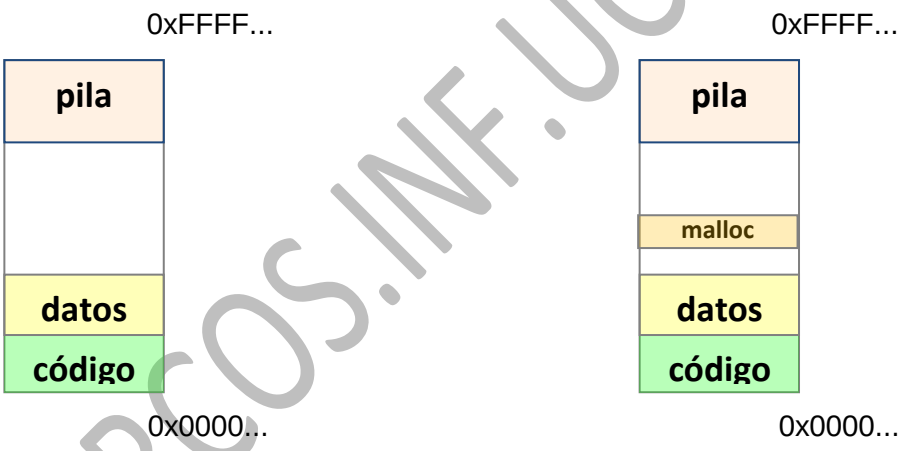


c) En **C** se ha creado un proceso pesado y el hijo ha pedido un malloc:

Grupo: ..... NIA: ..... Nombre y apellidos: .....



d) En el punto **D** finaliza el proceso ligero en ambos procesos pesados:



Grupo: …… NIA: …… Nombre y apellidos: ……..

Ejercicio 10

Un computador proporciona a sus usuarios un espacio de memoria virtual de 2^{32} bytes. Disponemos de una memoria física de 16 MB. El sistema de implantación de memoria virtual es por paginación y el tamaño de página es de 4 KB. Se pide:

a) ¿Cuál es el formato de la dirección virtual? Dibuje las distintas partes.

b) ¿Cuál es el formato de la dirección física? Dibuje las distintas partes.

c) ¿De cuántas páginas podemos disponer en memoria virtual?

d) Un proceso de usuario genera la dirección virtual 11123456, ¿A qué página hace referencia?

e) ¿Cuál es el desplazamiento dentro de esta página?

Solución

a)

Número de página virtual (20 bits)	Desplazamiento (12 bits)

b)

Número de página virtual (12 bits)	Desplazamiento (12 bits)

c)
2^{20} páginas virtuales como máximo

d)
2715 (resultado de la división entera de 11.123.456 / 4.096, es decir, el desplazamiento entre el tamaño de la página en bytes)

e)
2816 (resultado del módulo de 11.123.456 % 4.096).

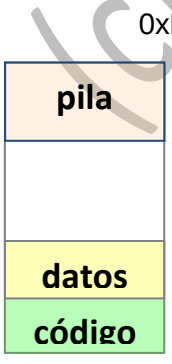
Grupo: …… NIA: …… Nombre y apellidos: ……..

Ejercicio 11

Se dispone del siguiente código fuente de un programa:

```
pthread_t   thid;

void *task1 ( void * arg )
{
    arg = malloc (100*sizeof(int)) ;
    /* punto B */
}

main (int argc, char **argv)
{
        int *pint;
        int   pid, i;

        /* punto A */
        pthread_create(&thid,NULL,task1,(void *)pint) ;
        pthread_join(&thid,NULL) ;

        /* punto C */
        free(pint) ;
        for (i=0; i<2; i++) {
            pid =  fork() ;
            if (0 == pid) {
                pint = (int *) malloc (100*sizeof(int)) ;
            }
        }

        /* punto D */
}
```

Si en el punto A de ejecución, la imagen del proceso asociado se representa:



Se pide:

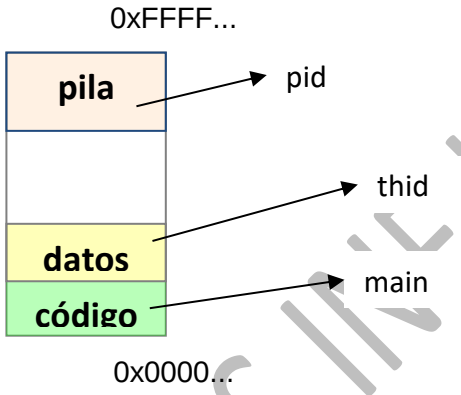
Grupo: …… NIA: …… Nombre y apellidos: ……..

a) Indique en qué región de memoria estarán los elementos **thid**, **pid** y **main**. **(0,5 puntos)**
b) Utilizando la representación dada, dibuje la imagen de los procesos en los puntos **B y C**.
c) Utilizando el mismo tipo de representación, dibuje la imagen de los procesos en el punto **D**.

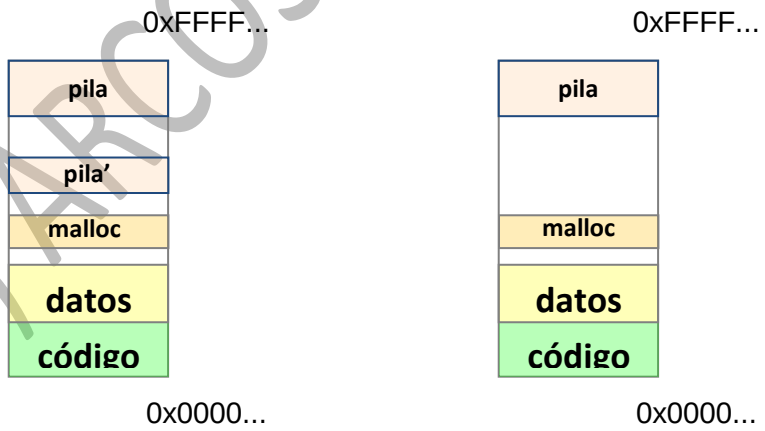
Para los apartados b) y c) indique las regiones de memoria que se necesite añadir o quitar para cada uno de los procesos/hilos de ejecución existentes.
NOTA: En la gestión de memoria no se usa COW (*Copy-On-Write*)

Solución

a)



b)



En b) tenemos un hilo que hace un malloc

En c) el hilo ha terminado y el malloc sigue

c)

```
free(pint) ;
```

- Se libera la sección usada por malloc

```
pid =  fork() ;
if (0 == pid) {
      pint = (int *) malloc (100*sizeof(int)) ;
}
```

- Se crea un proceso hijo y este hace un malloc en la primera iteración

```
pid =  fork() ;
if (0 == pid) {
      pint = (int *) malloc (100*sizeof(int)) ;
}
```

Cada proceso anterior crea otro hijo, y cada hijo hace otro malloc