Sistemas Operativos – 4º Semestre – GII & GMI

Tercer Parcial. Gestión de Memoria. 18 de mayo de 2015

Sea un sistema Linux con páginas de 2 KiB, palabras y direcciones de 32 bits, 3 GiB de RAM y un disco de 1 TiB. El fichero /home/pepe/datos.bin tiene un tamaño de 1MiB y está relleno con los enteros en binario 0, 1, 2, 3, 4, 5, etc.

```
#include <stdio.h>
#include <pthread.h>
int fdin, fdout, a, *p, *q; ...    //hay más declaraciones
void* fun1 (void* arg)
{...}
void* fun2 (void* arg)
{...}
int main(void) {
   pthread_t fun1th, fun2th;
   struct stat mistat;
   fdin = open("/home/pepe/datos.bin",O_RDWR);
   fstat(fdin, &mistat);
   p = mmap(0,mistat.st_size,PROT_READ|PROT_WRITE,MAP_PRIVATE,fdin,0);
   close(fdin);
   pthread_create(&fun1th, NULL, fun1, NULL);
   pthread_create(&fun2th, NULL, fun2, NULL);
   pthread_join(fun1th, NULL);
   pthread_join(fun2th, NULL);
   q = (int *) *p;
   *q = 4;
   a = *q;
   munmap(p, mistat.st_size);
   return 0;
}
```

Se pide:

***a) (2 ptos)** Razonar si los dos threads llegan a compartir la región creada por el mmap.*

Todo proceso pesado se crea (mediante un `fork`) con un único flujo de ejecución inicial, que denominamos hilo principal y cuya pila es la pila del proceso. Cualquier otro hilo (proceso ligero o *thread*) creado (mediante un `pthread_create`) desde el proceso, pasa a formar parte del proceso que lo crea. Cada nuevo hilo precisa un espacio de memoria (normalmente tomado del heap) para ser usado como su pila, pero todos los hilos del proceso (incluido el principal) son una unidad desde el punto de vista del Gestor de Memoria del Sistema Operativo. La imagen de memoria de cada proceso pesado concreto es una única para todo el proceso, independientemente de cuantos hilos tenga dicho proceso. De hecho, la tabla de páginas que da soporte a la imagen de memoria de tal proceso es única y se mantiene en la MMU mientras se planifique cualquiera de los hilos del proceso. Es por esto que el cambio de contexto entre los hilos de un proceso es mucho más rápido que entre hilos de procesos distintos.

Dicho esto, es evidente que todos los hilos del proceso comparten la misma imagen de memoria y que cualquier alteración que un hilo haga en dicha imagen la ven los demás. Por lo tanto la zona proyectada en este ejercicio es compartida por los tres hilos del proceso. El flag `MAP_PRIVATE` tan sólo implica que los cambios que se realicen en esta región proyectada no sean permanentes en el fichero ni sean visibles desde otros procesos.

b) (2 ptos) *Indicar razonadamente el máximo tamaño que podría tener la imagen de memoria de un proceso.*

La imagen de memoria de un proceso es el conjunto de regiones de memoria asignadas a un proceso. Esta imagen está soportada sobre el mapa de memoria del sistema. Si el sistema sólo dispone de memoria real el mapa de memoria estará soportado sólo sobre la memoria principal, pero si el sistema dispone de memoria virtual podrá estar soportado además sobre parte del almacenamiento secundario: área de intercambio (*swap*) y ficheros proyectados.

El tamaño máximo del mapa de memoria es el rango de direcciones que el procesador puede producir. En un sistema de 32 bits, como el del ejercicio, pueden direccionarse 2^32 bytes (no bits, ni palabras ni mucho menos páginas). Usualmente una parte fija de este mapa se reserva para el contener el propio núcleo (*kernel*) del Sistema Operativo más el espacio para datos que éste pueda necesitar. Esta parte del mapa estará marcada para residir permanentemente en memoria principal y así se indica en las tablas de páginas de todos los procesos. De esta manera, el cambio de modo usuario a modo kernel y viceversa (cada vez que se llame al sistema) será rápido. Por supuesto, este rango del mapa de memoria para la parte residente del sistema operativo no estará disponible para los procesos y por lo tanto limitará el tamaño máximo de la imagen de memoria de los procesos. En este ejercicio supondremos que el núcleo precisa 1 GiB residente.

Si el sistema no dispusiera de memoria virtual, la imagen de memoria de todos los procesos debería convivir en el espacio de memoria principal restante, esto es, 2 GiB. Este sería el límite máximo.

Pero si disponemos de memoria virtual con espacio virtual separado por proceso, el tamaño máximo de la imagen de memoria de un proceso serían 3 GiB, que estarían soportados sobre los 2GiB de marcos de página disponibles más el espacio de disco destinado a área de intercambio y ficheros proyectados.

c) (2 ptos) *Suponiendo que los threads no modifican ni p ni la región del mmap, indicar el valor que tendrá la variable* ***a*** *antes del munmap.*

Con el primer argumento de la llamada `mmap` a 0 se le pide al sistema que escoja el rango de direcciones que considere más conveniente para proyectar el archivo. Así `p` valdrá una dirección a partir de la cuál se encontrará proyectado el contenido del archivo, con los valores 0, 1, 2, etc. como enteros de 32 bits.

La sentencia "`q = (int *) *p;`" toma el valor apuntado por `p` (el 0), lo interpreta como puntero a entero, y se lo asigna a `q`. Por lo tanto `q` es un puntero a entero de valor 0. En otras palabras, `q` vale `NULL`.

La sentencia "`*q = 4;`" pretende asignar un valor 4 en la dirección apuntada por `q`, pero como `q` vale `NULL`, eso implica dereferenciar un puntero nulo y eso no se puede hacer y para ello la dirección 0 está protegida, para que salte un `SIGSEGV`.

En este momento el proceso moriría y la siguiente sentencia "`a = *q;`"no llegaría a ejecutarse.

d) (4 ptos) *Teniendo en cuenta los siguientes tamaños y que se utiliza montaje dinámico al invocar el procedimiento, rellenar la tabla adjunta con la información de la imagen de memoria del proceso antes de que terminen los dos threads.* ***Utilizar una línea por cada región****, rellenando* ***todos los campos*** *de la tabla.*

text	**data**	**bss**	**filename**
57653 = 56 KiB + 309 B	273 B	374	Miprogr.o
1719061 = 1678 KiB +789 B	11508 = 11 KiB + 244 B	11316 = 11 KiB + 52 B	libc.so
92630 = 90 KiB + 470 B	868 B	8352 = 8KiB + 160 B	libpthread.so

Consideraciones generales:

- La Dirección inicial y el Tamaño deben ser valores múltiplos del tamaño de la página (2 KiB).
- La Dirección final indica la última dirección con contenido de la región según lo especificado en el enunciado.
- La presentación de las regiones ha sido según se han ido creado durante la ejecución del proceso. Las pilas aparecen en las posiciones más altas de la imagen de memoria del proceso.
- En este ejercicio no ha sido necesario crear una región específica para heap, pero también se considera válido crearla sin que lo demande el proceso o agruparla junto con las regiones de datos del proceso.

Región	**Dirección inicial**	**Dirección final**	**Tamaño**	**Características**	**Origen del contenido de la región**
SSOO	0	9.437.184	9 MiB, 4.608 páginas	---	---
...	...	...	...	...	...
Código Miprogr.o	10.485.760 (10 MiB)	10.543.412	58 KiB, 29 páginas	R-X, compartida, tamaño fijo	Ejecutable en el sistema de ficheros
DVI Miprogr.o	10.545.152 (10 MiB + 58 KiB)	10.545.424	2 KiB, 1 página	RW-, privada, tamaño fijo	Ejecutable en el sistema de ficheros
DSVI Miprogr.o	10.547.200 (10 MiB + 60 KiB)	10.547.573	2 KiB, 1 página	RW-, privada, tamaño variable (heap se incluye aquí; si no, sería de tamaño fijo)	Rellenar con ceros el marco de página asignado
...	...	...	...	...	...
Pila Migprogr.o	18.874.368 (18 MiB)	18.771.969	100 KiB, 50 páginas	RW-, privada, tamaño variable	Rellenar con ceros el marco de página asignado
...	...	...	...	...	...
Código libc.so	10.549.248 (10 MiB + 62 KiB)	12.268.308	1680 KiB, 840 páginas	R-X, compartida, tamaño fijo	Biblioteca en el sistema de ficheros
DVI libc.so	12.269.568 (11 MiB + 718 KiB)	12.268.308	12 KiB, 6 páginas	RW-, privada, tamaño fijo	Biblioteca en el sistema de ficheros
DSVI libc.so	12.281.856 (11 MiB + 730 KiB)	12.293.171	12 KiB, 6 páginas	RW-, privada, tamaño fijo	Rellenar con ceros el marco de página asignado
mmap	12.294.144 (11 MiB + 742 KiB)	13.342.719	1 MiB, 512 páginas	RW-, privada, tamaño fijo	Fichero /home/pepe/datos.bin en el sistema de ficheros
Código libpthread.so	13.342.720 (12 MiB + 742 KiB)	13.435.349	92 KiB, 46 páginas	R-X, compartida, tamaño fijo	Biblioteca en el sistema de ficheros
DVI libpthread.so	13.436.928 (12 MiB + 834 KiB)	13.437.795	2 KiB, 1 página	RW-, privada, tamaño fijo	Biblioteca en el sistema de ficheros
DSVI libpthread.so	13.438.976 (12 MiB + 836 KiB)	13.447.327	10 KiB, 5 páginas	RW-, privada, tamaño fijo	Rellenar con ceros el marco de página asignado
...	...	...	...	...	...
Pila fun1	16.777.216 (16 MiB)	16.674.817	100 KiB, 50 páginas	RW-, privada, tamaño variable	Rellenar con ceros el marco de página asignado
Pila fun2	15.728.640 (15 MiB)	15.626.241	100 KiB, 50 páginas	RW-, privada, tamaño variable	Rellenar con ceros el marco de página asignado