SISTEMAS OPERATIVOS I

Práctica 5: Mapa de memoria de un proceso

Objetivo

Obtener y analizar información sobre el mapa de memoria del proceso.

Información útil para la práctica

El mapa de memoria de un proceso se puede ver en el archivo */proc/PID/maps*, donde PID es el identificador del proceso (recuerda que se puede obtener con la función *getpid()*, o localizarlo desde la terminal con el comando *ps*). El mapa de memoria puede cambiar durante la ejecución de un proceso, ya que de forma dinámica se pueden crear nuevas regiones de memoria. Se recomienda visualizar el mapa de memoria en varios momentos durante la ejecución del proceso para comprobar cómo va cambiando. Se puede usar *scanf()* para mantener el proceso activo hasta que se introduce un dato, y así tener tiempo para visualizar el mapa de memoria. Recuerda que la actualización no necesariemante ocurrer de manera inmediata.

Información adicional para la práctica

En el campus virtual tienes disponible un documento con "*material adicional para la práctica 5: el mapa de memoria*".

Enunciado

Escribe un programa en C que cree y utilice varios tipos de variables, llamadas al sistema de reserva dinámica memoria, funciones, e hilos, y estudia el mapa de memoria del proceso. Se recomienda examinar el mapa de memoria antes y después de cada uno de los pasos siguientes para comprobar donde se almacenan las variables y funciones, y como cambia durante la ejecución del proceso. Los programas deben mostrar de manera clara en la consola las direcciones de memoria de las diferentes variables y funciones.

Implementa los siguientes apartados para estudiar el mapa de memoria de un proceso e identifica las distintas regiones del mapa. El **ENTREGABLE** de la práctica *constará de los códigos* y de un *breve documento en pdf* (con un máximo de diez páginas) donde se resuma el **mapa de memoria con comentarios** sobre los 6 apartados del ejercicio.

1. Analiza un proceso que incluya algunas variables globales, un array global, algunas variables locales y un array local. Muestra en hexadecimal la dirección de los punteros a esas variables y de la función *main*. Identifica las regiones del mapa donde se encuentran el main y todas las variables y arrays.

2. Analiza un proceso con dos funciones *f1* y *f2*. Cada una de ellas recibe como parámetro un número entero y realiza la siguiente tarea: define una variable entera local y muestra en el terminal la dirección del parámetro recibido y de la variable local. Identifica donde se encuentran los códigos de las funciones, los parámetros y las variables locales de la función en el mapa de memoria.

3. Analiza un proceso que incluya llamadas al sistema *malloc* y *free* (comprueba que las llamadas han tenido éxito). Examina el mapa de memoria antes y después de la ejecución de *malloc*, e

identifica la zona de memoria dinámica que has creado para diferentes tamaños de la reserva. Comprueba si con *free* se librea toda la zona reservada por el *malloc*. Haz la misma prueba si luego se ejecuta otro *malloc*. Ten en cuenta los consejos comentados más adelante para completar el informe.

4. Crea un proceso hijo que incluya un *malloc* y a continuación cambie su imagen con *execv* a un código cualquiera creado por ti. Compara los mapas de memoria del proceso padre y el proceso hijo antes y después del *malloc* y comprueba como el cambio de imagen afecta al mapa de memoria del proceso hijo.

5. Comprueba lo que ocurre cuando un proceso, con una función matemática como *sin*, se compila para que enlace la librería matemática de forma estática y cuando lo hace de forma dinámica. Consulta el manual del compilador para ver qué significa esto y cómo hacerlo. Compara los mapas de memoria y su tamaño.

6. Crea un programa con las siguientes características:

   - Una variable global y una variable local de tipo entero.
   - Dos hilos que reciben como parámetro la variable local del hilo principal.
   - Cada hilo define una nueva variable local y la actualiza con el producto de la variable global y el parámetro recibido por el hilo.

   (a) Muestra en pantalla las direcciones de la variable global y las variables locales en el hilo principal y, por otro lado, de la variable global, los parámetros recibidos y las variables locales definidas en cada uno de los dos hilos creados. Haz también una llamada a malloc en los dos hilos.

   (b) Localiza todas estas direcciones en el mapa de direcciones del proceso, e identifica las regiones en las que se encuentran.

   (c) Identifica donde están las pilas y los montículos de memoria dinámica de todos los hilos.

Consejos para la realización del ejercicio 3

La utilización de llamadas al sistema de reserva dinámica de memoria y su influencia sobre el mapa de memoria del proceso merece un poco más de atención. Debes fijarte en los siguientes aspectos:

- Comprueba que se crean zonas diferentes cuando se reserva poca o mucha memoria con *malloc*.
- Comprueba si al hacer dos llamadas a *malloc* se crean zonas de memoria separadas para cada *malloc*.
- Comprueba que ocurre al hacer una reserva de memoria con la llamada al sistema *alloca* frente a la utilización de *malloc*.
- Comprueba que ocurre si se reserva memoria con *malloc* y posteriormente se hace un *realloc*. Ajusta el tamaño del *realloc* para comprobar que la zona de memoria reservada puede crecer.
- Finalmente, usa la función *malloc* para reservar espacio para 1000 *doubles*. Aplica la función *sizeof* al puntero devuelto por *malloc*. ¿Qué es lo que proporciona? Comprueba que sucede si se cambia la dirección del puntero devuelto por *malloc* (por ejemplo, incrementándolo) antes de llamar a *free*.