Universidad Nacional
Autónoma De México
Facultad De Ingeniería
Sistemas Operativos

Tarea 2
Administración de Memoria

Nombre del profesor: Gunnar Wolf

Alumno
Pérez Saldaña Luis Mario

Semestre: 2021-2
Fecha: 3 de julio del 2021

Control de Lectura

En el artículo por Jean-Baptiste Terrazoni, se explora una implementación de alto nivel en el lenguaje C de las funciones de la librería de asignación de memoria dinámica: malloc, free, realloc, calloc.

Se parte exponiendo la necesidad de la memoria dinámica. En C, fuera del esquema de memoria dinámica, se tienen las variables globales y los estáticas que ocupan espacio en la memoria principal del programa, y las variables locales a funciones que se ubican en el stack, o “zona de libres”, del proceso. El problema inherente es que el tamaño de la reservación de memoria debe estar determinado en tiempo de compilación, y de esta forma no hay flexibilidad en el tiempo de vida de las variables. Por lo tanto, en lenguajes donde el manejo en la memoria en tiempo de ejecución es tarea del programador, como en el lenguaje C, deben existir mecanismos para la reservación de memoria de tamaño variable, así como la posibilidad de gestionar la reservación y liberación de memoria a voluntad del programador.

Para la implementación se parte de la interfaz mmap y munmap para asignación de memoria que exponen diferentes sistemas operativos. Un primer comentario inocente podría hacerse sobre las funciones de asignación de memoria: si las llamadas de sistema mmap y munmap existen, entonces debería haber necesidad de malloc y free. Lo que sucede es que malloc, realloc, calloc y free son optimizaciones sobre llamadas al sistema, que suelen ser más costosas cuando no están envueltas en alguna optimización. La forma en que, por ejemplo, malloc y realloc optimizan mmap es que evitan llamadas adicionales al sistema operativo, prealocando un poco de memoria adicional cuando se hace una reservación de un tamaño dado. Esto último es especialmente útil, pues considerando la ejecución de un programa común, se espera que se haga reservación y liberación de memoria frecuentemente.

Se explican las estructuras en memoria cuando se hace una reservación dinámica: los heaps y los bloques. Se tratan con estructuras a nivel implementación y se utilizan apuntadores.

Se dice que un heap, de tamaño fijo, contiene a su vez cierta cantidad de bloques, también de tamaño fijo. Ambos son espacios en memoria que pueden guardar valores y además reservan un espacio pequeño al principio para guardar metadatos. También cada bloque y cada heap tiene referencia para el siguiente y el anterior bloque y heap, respectivamente. Dependiendo de la cantidad de memoria requerida por la llamada a malloc, a un bloque le corresponde cierta cantidad de bytes.

Para la implementación también se considera que se trabaja en una computadora con administración de memoria paginada. El tamaño de un heap se maneja como múltiplo del tamaño de la página en el sistema, para así evitar espacios de memoria sin utilizar.

Los algoritmos para las funciones malloc, realloc, free utilizan llamadas al sistema, también implementadas por el autor en el lenguaje C. Se ubican en el repositorio junto con el resto del código.

El algoritmo para malloc no es complejo; sólo basta tener en consideración los detalles de los bloques y los heaps. Funciona de la siguiente manera:

- Con la referencia global al primer heap, se busca uno por uno si alguno de sus bloques no ocupados (o liberados) tienen espacio suficiente para alojar la memoria solicitada como argumento a malloc.
- Si el primer heap no tiene memoria disponible, se repite lo mismo para el siguiente heap hasta encontrar un bloque con espacio suficiente.
- Si no se halla un heap (por ejemplo, cuando es la primera llamada), o un heap con espacio, se crea un nuevo heap y se reserva el bloque, mediante la llamada a mmap.
- Se colocan valores 0xAA en las direcciones de memoria reservadas, por si existía algo indeterminado anteriormente.

El algoritmo de free tampoco es compilado.

- Se pasa un apuntador a memoria y a partir del primer heap y el primer bloque, se busca la dirección pasada.
- Si el bloque tiene algún otro bloque contiguo también libre, se juntan en uno solo. Hay que matizar que es posible que la memoria quede fragmentada si lo anterior no ocurre.
- Luego, si el bloque liberado es el último del último heap, se hace la llamada a munmap para liberar la memoria, o si es el último heap, se libera todo el heap también con munmap.

El algoritmo de realloc es una aplicación de las funciones malloc y free. Es básicamente una secuencia de las operaciones malloc para el nuevo tamaño, seguido de una copia de los valores memoria de las direcciones anteriores a las nuevas direcciones de la nueva llamada a malloc (memcpy),y luego un free de las direcciones antiguas.

Es un ejercicio agradable y provechoso hacer la implementación en un lenguaje con asignación de memoria pues es posible comprender mejor el funcionamiento de las librerías de memoria, y revisar conceptos de administración de memoria y sistemas operativos.

Bibliografía

- https://medium.com/a-42-journey/how-to-create-your-own-malloc-library-b86feedd39b96
- https://github.com/jterrazz/42-malloc