Trabajo Práctico 1 - Grupo 7

(72.11) Sistemas Operativos

Integrantes

- Martina Schvartz Tallone - 62560
- Josefina Míguez - 63132

Para realizar el test del memory manager dentro del kernel se deben seguir los siguientes pasos:

1. Correr el contenedor y posicionarse en el directorio `root`.
2. Posicionarse en la carpeta `Toolchain` y correr en la terminal del contenedor `make clean all`.
3. Dentro de `root` otra vez, correr make clean all.
4. Por fuera del contenedor correr `./run.sh` sin ningún argumento extra.

La idea es que el test se corra dentro del main del kernel, previo al paso a userspace. Se utilizaron dos funciones extras a modo de feedback, un print antes de correr la función `test_mm` y el siguiente print dentro de la misma.

Decidimos hacer una lista que incluya los nodos ocupados y libres, para tener un recuento de toda la memoria. Como cada nodo incluye un header con info, hay parte de esa memoria libre que va a ser destinada a guardar el header. Por ende de los 64k que nosotros designamos como usable, lo que realmente se estaría utilizando:

```
64k - (MAX_BLOCKS * sizeof(Header))
```

El resultado de esa cuenta es 63488. Esto se debe a que en el peor de los casos, la lista se particiona 128 veces y se desperdician `MAX_BLOCKS * sizeof(Header)` bytes de memoria.

Para poder realizar el test por fuera del entorno de nuestro sistema operativo se deben seguir los siguientes pasos:

1. Correr el contenedor y posicionarse en el directorio `root`.
2. Compilar el archivo fuente dentro de la terminal del contenedor con el comando `gcc -Wall -g mm_tester.c ./Kernel/memoryManager.c ./Kernel/test_util.c -o test.`
3. Correr el ejecutable por fuera del contenedor pero dentro de la carpeta del trabajo con el comando `./test`.

Como el archivo existe por fuera del entorno del trabajo, se simuló una memoria libre a través del malloc original. Por ende cuando se corre el `test_mm` nunca se llega a liberar esa memoria simulada. Esto nos genera entonces un posible memory leak.

Al correr con el flag `-fsanitize=address`, nos figura que hay un overflow, pero no podemos saber la fuente ya que nuestras funciones están hechas para correr dentro del SO. De igual manera, dentro del kernel se testeó la función reiteradas veces y funciona según lo esperado.