

# **SISTEMAS OPERATIVOS I**

# **INFORME SOBRE MAPA**

# **DE MEMORIA**

***Por:***

- ***Matheus Muiños Kruschewsky***
- ***Alejandro Alonso Muiños***

# INTRODUCCIÓN

El objetivo de esta práctica es realizar un análisis exhaustivo del mapa de memoria de un proceso. Para conseguirlo, hemos desarrollado un programa en C que simula seis situaciones clave de gestión de memoria, como son el manejo de variables estáticas y locales, la asignación dinámica, el uso de hilos, la vinculación de librerías y la creación de procesos hijos. La metodología principal ha sido examinar el archivo `/proc/PID/maps` en diversos momentos de la ejecución de cada apartado. Esto nos ha permitido identificar y delimitar las regiones fundamentales y entender cómo el sistema operativo gestiona la memoria para cada proceso.

## EJERCICIO 1

-Resultados:

```
math@el19athena:/mnt/c/Users/matho/Downloads/PS_Sistemas_Operativos_1
$ ./a1
PID del proceso principal: 1331
Selecciona el apartado a ejecutar (1-6):
1. Apartado 1: Variables globales, locales y array 3D
2. Apartado 2: Funciones f1 y f2
3. Apartado 3: Uso de malloc y free
4. Apartado 4: Proceso hijo con malloc y memcpy
5. Apartado 5: Librería matemática estática vs dinámica
6. Apartado 6: Hilos y variables locales
Elige una opción: 1
Dirección de la variable global global: 0x577123791040
Dirección de la variable global global2: 0x577123791060
Dirección de la variable global global3: 0x577123791080
Dirección de la variable global global_array: 0x5771237910a0
1
Dirección de la variable local: 0x7ffcb6636e04
Dirección de la variable local2: 0x7ffcb6636f00
Dirección de la variable local3: 0x7ffcb6636f40
Dirección del array 3D local_array: 0x7ffcb6636f80
Dirección de la función main: 0x57712378e499
1
Dirección del elemento del array 3D [0][0][0]: 0x7ffcb6636f20
Dirección del elemento del array 3D [0][0][1]: 0x7ffcb6636f24
Dirección del elemento del array 3D [0][0][2]: 0x7ffcb6636f28
Dirección del elemento del array 3D [0][1][0]: 0x7ffcb6636f48
Dirección del elemento del array 3D [0][1][1]: 0x7ffcb6636f4c
Dirección del elemento del array 3D [0][1][2]: 0x7ffcb6636f50
Dirección del elemento del array 3D [0][2][0]: 0x7ffcb6636f70
Dirección del elemento del array 3D [0][2][1]: 0x7ffcb6636f74
Dirección del elemento del array 3D [0][2][2]: 0x7ffcb6636f78
2
Dirección del elemento del array 3D [1][0][0]: 0x7ffcb6637000
Dirección del elemento del array 3D [1][0][1]: 0x7ffcb6637004
Dirección del elemento del array 3D [1][0][2]: 0x7ffcb6637008
Dirección del elemento del array 3D [1][1][0]: 0x7ffcb6637020
Dirección del elemento del array 3D [1][1][1]: 0x7ffcb6637024
math@el19athena:/mnt/c/Users/matho/Downloads/PS_Sistemas_Operativos_1$ cat /proc/1331/maps
57712378d000-577123790000 r-p 00000000 00:03 3896224743875040 /mnt/c/Users/matho/Down
loads/PS_Sistemas_Operativos_1/a1
57712378d000-57712378f000 r-xp 00000000 00:03 3896224743875040 /mnt/c/Users/matho/Down
loads/PS_Sistemas_Operativos_1/a1
57712378f000-577123790000 r-p 00000000 00:03 3896224743875040 /mnt/c/Users/matho/Down
loads/PS_Sistemas_Operativos_1/a1
577123790000-577123791000 r-p 00000000 00:03 3896224743875040 /mnt/c/Users/matho/Down
loads/PS_Sistemas_Operativos_1/a1
577123791000-577123792000 rw-p 00000000 00:03 3896224743875040 /mnt/c/Users/matho/Down
loads/PS_Sistemas_Operativos_1/a1
577130600000-57713060c000 rw-p 00000000 00:00 0 [heap]
7f93d50f0000-7f93d5420000 r-p 00000000 00:30 49707 /usr/lib/x86_64-linux-g
nu/libc.so.6
7f93d5420000-7f93d55b0000 r-xp 00000000 00:30 49707 /usr/lib/x86_64-linux-g
nu/libc.so.6
7f93d55b0000-7f93d55f0000 r-p 00100000 00:30 49707 /usr/lib/x86_64-linux-g
nu/libc.so.6
7f93d55f0000-7f93d5603000 r-p 001f0000 00:30 49707 /usr/lib/x86_64-linux-g
nu/libc.so.6
7f93d5603000-7f93d5605000 rw-p 00000000 00:30 49707 /usr/lib/x86_64-linux-g
nu/libc.so.6
7f93d5605000-7f93d5612000 rw-p 00000000 00:00 0
7f93d5612000-7f93d56d0000 rw-p 00000000 00:00 0
7f93d56d0000-7f93d56e0000 rw-p 00000000 00:00 0
7f93d56e0000-7f93d56e0000 r-p 00000000 00:30 49597 /usr/lib/x86_64-linux-g
nu/ld-linux-x86-64.so.2
7f93d56e0000-7f93d5711000 r-xp 00000000 00:30 49597 /usr/lib/x86_64-linux-g
nu/ld-linux-x86-64.so.2
7f93d5711000-7f93d5715000 r-p 00000000 00:30 49597 /usr/lib/x86_64-linux-g
nu/ld-linux-x86-64.so.2
7f93d5715000-7f93d5717000 rw-p 00000000 00:30 49597 /usr/lib/x86_64-linux-g
nu/ld-linux-x86-64.so.2
7ffc66130000-7ffc6613a000 rw-p 00000000 00:00 0 [stack]
7ffc6613a000-7ffc6613b000 r-p 00000000 00:00 0 [vvar]
```

- Al ejecutar el código y buscar el mapa de memoria del proceso creado para su ejecución, se puede ver la dirección de memoria de la función main (0x57712378e499). Cuando buscamos en el mapa de memoria el intervalo donde se encuentra esta dirección, vemos que está en el intervalo marcado como 57712378d000–57712378f000 r-xp. Este marco de permiso r-xp (lectura y ejecución, sin escritura) está asociado al fichero ejecutable del programa y se corresponde con la región de memoria conocida como el segmento TEXT. Esto confirma que el código de la función main y, en general, todas las instrucciones del programa residen en esta área fija del proceso.

- Cuando buscamos las variables globales del programa, vemos que presentan direcciones en el rango 0x577123791040–0x577123791060. Estas direcciones se sitúan dentro de una zona del mapa de memoria marcada con permisos rw-p (577123791000-577123792000). El permiso de lectura y escritura indica que se trata del segmento de datos estáticos. Esto confirma que todas las variables globales se almacenan en la región de datos globales del proceso, donde permanecen accesibles durante toda la vida de la ejecución.

- Para las variables locales, estas presentan direcciones del tipo 0x7ffcb6636e04, 0x7ffcb6636f0 y valores similares. Al compararlas con el mapa de memoria, se verifica que caen dentro de un intervalo identificado como [stack]. Esta región corresponde al segmento de pila del proceso, utilizado para almacenar variables automáticas, parámetros de funciones y direcciones de retorno. El hecho de que las variables locales estén en esta zona confirma que su duración es automática (solo existen mientras la función está activa).

- Para estudiar el orden de almacenamiento del array tridimensional, se imprimieron las direcciones de cada uno de sus elementos. Las direcciones muestran que los elementos  $[i][j][k]$ ,  $[i][j][k+1]$  y  $[i][j][k+2]$  ocupan posiciones contiguas en memoria, con incrementos constantes en la dirección, mientras que los elementos  $[i][j+1][0]$  aparecen a continuación de terminar la secuencia del último índice. Este comportamiento demuestra que el índice más interno (k) es el que varía más rápido, seguido por el índice j y finalmente por i. Dicho patrón coincide con el esquema de almacenamiento row-major utilizado por el lenguaje C para matrices multidimensionales, en el cual el array 3D se representa linealmente en memoria siguiendo el orden natural de los índices desde el más interno al más externo.

-Resultados:

```
variable local para f1 : b7ffff029424  
parametro de f2 n: b7ffff29d3c3  
variable local para f2 y: b7ffff029424  
Pausa: Presiona ENTER para finalizar y examinar el mapa (TEXT y STACK).  
  
pid del proceso: 32474  
Selecciona el apartado a ejecutar (1-6):  
1. Apartado 1: Variables locales, locales y array 3D  
2. Apartado 2: Funciones f1 y f2  
3. Apartado 3: Uso de malloc y free  
4. Apartado 4: Proceso hijo con malloc y execv  
5. Apartado 5: Librería matemática estática vs dinámica  
6. Apartado 6: Hilos y variables locales  
Elije una opción 2:  
parametro de f1 n: b7ffff4db8bc  
variable local para f1 x: b7ffff4db8c4  
parametro de f2 ir: b7ffff4db8c4  
variable local para f2 y: b7ffff4db8c4  
Pausa: Presiona ENTER para finalizar y examinar el mapa (TEXT y STACK).  
  
#alexonso@portatil-alex:/SOL/Memorias -/b  
PID del proceso: 32474  
Selecciona el apartado a ejecutar (1-6):  
1. Apartado 1: Variables locales, locales y array 3D  
2. Apartado 2: Funciones f1 y f2  
3. Apartado 3: Uso de malloc y free  
4. Apartado 4: Proceso hijo con malloc y execv  
5. Apartado 5: Librería matemática estática vs dinámica  
6. Apartado 6: Hilos y variables locales  
Elije una opción 2:  
parametro de f1 n: b7ffff663b34  
variable local para f1 x: b7ffff663b35  
Dirección de la función f1: 0x607fc32fa3b9  
parametro de f2 ir: b7ffff663b34  
variable local para f2 y: b7ffff663b35  
Dirección de la función f2: 0x607fc32fa3be  
Pausa:Introduzca un caracter y pulse ENTER para finalizar.
```

-Durante la ejecución, el programa muestra que la dirección del parámetro n en f1 y en f2 es 0x7fff06635b4c. Al comparar esta dirección con el mapa de memoria del proceso, se comprueba que cae dentro del intervalo marcado como [stack]. Esta región corresponde al segmento de pila, utilizado para almacenar los marcos de activación de las funciones. En consecuencia, los parámetros enteros pasados a f1 y f2 se almacenan en la pila cuando dichas funciones están en ejecución.

Observación sobre la coincidencia de direcciones:

Es importante destacar que el hecho de que  $f_1$  y  $f_2$  muestren exactamente las mismas direcciones para sus parámetros y sus variables locales no constituye un error, sino que es el comportamiento esperado.

Sin embargo, las funciones f1 y f2 residen en el segmento TEXT del programa. Este segmento es estático y de solo lectura (r-xp). La dirección de inicio de cada función es fija y se determina en el momento de la carga del programa, por lo que nunca se libera ni se reutiliza. A diferencia de la pila, que es un espacio de trabajo temporal, el código es una parte permanente del proceso.

**-Mapa de memoria después del primer malloc (400 bytes):**

[illegible]

**-Mapa de memoria después del free(p).**

```

#1
fin del proceso principal; no
selección el algoritmo a ejecutar (1-6);
    Ajustar 1: Variables globales, locales y array 30
    Ajustar 2: Funciones 1 y 2
    Ajustar 3: Uso de mallas y vect
    Ajustar 4: Procesos hipo-muertos y muros
    Ajustar 5: Librería matemática estática vs dinámica
    Ajustar 6: Mallas y variables locales
    Elige una opción a
        por mallas de un byte, dirección = %cccccc%
        malla (MTR para mirar el tipo de memoria DESPUS del primer malla...
            free(a)
        malla (MTR para mirar el tipo de memoria DESPUS del free())...
    ]

```

En esta captura se observa que la región [heap] sigue apareciendo exactamente igual que en la captura anterior. Aunque ya hemos llamado a free(p), el tamaño de la heap no cambia y tampoco desaparece del mapa de memoria. Esto se debe a que free no libera la zona completa de la heap en el sistema operativo, sino que simplemente marca internamente el bloque como “libre” para que el propio programa pueda reutilizarlo más adelante. Por eso, desde el punto de vista del mapa de memoria del proceso, no hay ningún cambio visible, aunque el free se haya ejecutado correctamente.

## -Mapa de memoria después del segundo malloc (4000 bytes).

```
mathe@G319Matheus:/mnt/c/Users/mathe/Downloads/PS.Sistemas_Operativos_1$ ./
el
PID del proceso principal: 1060
Selección el apartado a ejecutar (1-6):
1. Apartado 1: Variables globales, locales y array 3D
2. Apartado 2: Funciones f1 y f2
3. Apartado 3: Uso de malloc y free
4. Apartado 4: Proceso hijo con malloc y execv
5. Apartado 5: Librería matemática estática vs dinámica
6. Apartado 6: Hilos y variables locales
Elige una opción: 3
Primer malloc de 400 bytes. Dirección p = 0x5caf2260c60
Pulsa ENTER para mirar el mapa de memoria DESPUES del primer malloc...
free(p) ejecutado.
Pulsa ENTER para mirar el mapa de memoria DESPUES del free(p)...
Segundo malloc de 4000 bytes. Dirección q = 0x5caf2260c60
Pulsa ENTER para mirar el mapa de memoria DESPUES del segundo malloc...
[]

mathe@G319Matheus:/mnt/c/Users/mathe/Downloads/PS.Sistemas_Operativos_1$ cat /proc/1060/maps
5caf226000-5caf226000 r--p 00000000 00:43 59189/4510998654 /mnt/c/Users/mathe/
Downloads/PS.Sistemas_Operativos_1/eli
5cc4a6d000-5cc4a6d000 r-xp 00001000 00:43 59189/4510998654 /mnt/c/Users/mathe/
Downloads/PS.Sistemas_Operativos_1/eli
5cc4a6d000-5cc4a6f000 r--p 00002000 00:43 59189/4510998654 /mnt/c/Users/mathe/
Downloads/PS.Sistemas_Operativos_1/eli
5cc4a6f000-5cc4a70000 r--p 00003000 00:43 59189/4510998654 /mnt/c/Users/mathe/
Downloads/PS.Sistemas_Operativos_1/eli
5cc4a70000-5cc4a71000 r--p 00004000 00:43 59189/4510998654 /mnt/c/Users/mathe/
Downloads/PS.Sistemas_Operativos_1/eli
5cc4f2200000-5cc4f2201000 rw-p 00000000 00:00 0 /heap
7f0ad2c00000-7f0ad2c20000 r--p 00000000 00:30 49707 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7f0ad2c20000-7f0ad2d00000 r-xp 00002000 00:30 49707 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7f0ad2d00000-7f0ad2dfff000 r--p 00100000 00:30 49707 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7f0ad2dfff000-7f0ad2e03000 r--p 001fe000 00:30 49707 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7f0ad2e03000-7f0ad2e05000 rw-p 00202000 00:30 49707 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7f0ad2e05000-7f0ad2e12000 rw-p 00000000 00:00 0 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7f0ad2e12000-7f0ad2f00000 rw-p 00000000 00:00 0 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7f0ad2f12000-7f0ad2f15000 r--p 00000000 00:30 0 7f0ad2f14000-7f0ad2f15000 r--p 00000000 00:30 49597 /usr/lib/x86_64-lin
ux-gpu/ld-linux-x86-64.so.2
7f0ad2f15000-7f0ad2f40000 r--p 00001000 00:30 49597 /usr/lib/x86_64-lin
ux-gpu/ld-linux-x86-64.so.2
7f0ad2f40000-7f0ad2f4a000 r--p 00002000 00:30 49597 /usr/lib/x86_64-lin
ux-gpu/ld-linux-x86-64.so.2
7f0ad2f4a000-7f0ad2f4b000 r--p 00036000 00:30 49597 /usr/lib/x86_64-lin
ux-gpu/ld-linux-x86-64.so.2
7f0ad2f4b000-7f0ad2f4c000 rw-p 00038000 00:30 49597 /usr/lib/x86_64-lin
ux-gpu/ld-linux-x86-64.so.2
7f0ad2f4c000-7f0ad2f4d000 r--p 00000000 00:00 0 [stack]
7f0ad2f4d000-7f0ad2f4e000 r--p 00000000 00:00 0 [vvar]
7f0ad2f4e000-7f0ad2f4f000 r-xp 00000000 00:00 0 [vdso]
```

Después de reservar un segundo bloque de memoria más grande (4000 bytes), el puntero recibido también corresponde a una dirección que pertenece a la región [heap]. En el mapa de memoria se puede ver que la heap continúa existiendo como antes, sin desaparecer ni dividirse. Además, el tamaño de la región tampoco cambia de forma visible, lo cual indica que la heap ya disponía de suficiente espacio interno para atender esta nueva reserva.

## -Mapa de memoria después del tercer malloc (10MB).

```
5. Apartado 3: Librería matemática estática vs dinámica
6. Apartado 6: Hilos y variables locales
Elige una opción: 3
Primer malloc de 400 bytes. Dirección p = 0x5caf2260c60
Pulsa ENTER para mirar el mapa de memoria DESPUES del primer malloc...
free(p) ejecutado. El heap se mantiene.
Mirar el cat /proc/PID/maps y después pulsa ENTER para mirar el mapa de memoria DESPUES del free(p)...
alexalansoportall@alex:~/S01/Memoria$ gcc ENTREGABLE1.c -o b
alexalansoportall@alex:~/S01/Memoria$ ./b
PID del proceso: 33583
Selección el apartado a ejecutar (1-6):
1. Apartado 1: Variables globales, locales y array 3D
2. Apartado 2: Funciones f1 y f2
3. Apartado 3: Uso de malloc y free
4. Apartado 4: Proceso hijo con malloc y execv
5. Apartado 5: Librería matemática estática vs dinámica
6. Apartado 6: Hilos y variables locales
Elige una opción: 3
Primer malloc de 400 bytes. Dirección p = 0x5caf2260c60
Mirar el cat /proc/PID/maps y después pulsa ENTER para mirar el mapa de memoria DESPUES del primer malloc...
Pulsa antes del free
5
Después del free.
Mirar el cat /proc/PID/maps y después pulsa ENTER para mirar el mapa de memoria DESPUES del free(p)...
Segundo malloc de 4000 bytes. Dirección q = 0x5caf2260c60
Mirar el cat /proc/PID/maps y después pulsa ENTER para mirar el mapa de memoria DESPUES del segundo malloc...
5
Después del free.
Segundo malloc de 10485760 bytes. Dirección q = 0x7fb2d21ff010
Mirar el cat /proc/PID/maps y después pulsa ENTER para mirar el mapa de memoria DESPUES del segundo malloc...

alexalansoportall@alex:~/S01/Memoria$ cat /proc/33583/maps
509241e000-509241e1000 r--p 00000000 103:06 3882296 /home/alexalanso/S01
/Memoria/D
509241e1000-509241e2000 r-xp 00001000 103:06 3882296 /home/alexalanso/S01
/Memoria/D
509241e2000-509241e3000 r--p 00002000 103:06 3882296 /home/alexalanso/S01
/Memoria/D
509241e3000-509241e4000 r--p 00003000 103:06 3882296 /home/alexalanso/S01
/Memoria/D
509241e4000-509241e5000 rw-p 00004000 103:06 3882296 /home/alexalanso/S01
/Memoria/D
509241e5000-509241e6000 r--p 00005000 103:06 3882296 /home/alexalanso/S01
/Memoria/D
7fb2d21ff000-7fb2d2c00000 rw-p 00000000 00:00 0 /heap
7fb2d2c00000-7fb2d2c20000 r--p 00000000 00:00 0 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7fb2d2c20000-7fb2d2d00000 r-xp 00002000 103:06 168663 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7fb2d2d00000-7fb2d2dfff000 r--p 00100000 103:06 168663 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7fb2d2dfff000-7fb2d2e03000 r--p 001fe000 103:06 168663 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7fb2d2e03000-7fb2d2e05000 rw-p 00202000 103:06 168663 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7fb2d2e05000-7fb2d2e12000 rw-p 00000000 00:00 0 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7fb2d2e12000-7fb2d2f00000 rw-p 00000000 00:00 0 /usr/lib/x86_64-lin
ux-gnu/libc.so.6
7fb2d2f12000-7fb2d2f15000 r--p 00000000 00:30 0 7fb2d2f14000-7fb2d2f15000 r--p 00000000 00:30 168660 /usr/lib/x86_64-lin
ux-gpu/ld-linux-x86-64.so.2
7fb2d2f15000-7fb2d2f40000 r--p 00001000 103:06 168660 /usr/lib/x86_64-lin
ux-gpu/ld-linux-x86-64.so.2
7fb2d2f40000-7fb2d2f4a000 r--p 00002000 103:06 168660 /usr/lib/x86_64-lin
ux-gpu/ld-linux-x86-64.so.2
7fb2d2f4a000-7fb2d2f4b000 r--p 00036000 103:06 168660 /usr/lib/x86_64-lin
ux-gpu/ld-linux-x86-64.so.2
7fb2d2f4b000-7fb2d2f4c000 rw-p 00038000 103:06 168660 /usr/lib/x86_64-lin
ux-gpu/ld-linux-x86-64.so.2
7fb2d2f4c000-7fb2d2f4d000 r--p 00000000 00:00 0 [stack]
7fb2d2f4d000-7fb2d2f4e000 r--p 00000000 00:00 0 [vsyscall]
```

En esta captura podemos ver que la dirección devuelta por el malloc no pertenece a la región [heap] ya existente. Esto se debe a que la memoria reservada (10 MB) supera el umbral interno de asignación de la



biblioteca C, lo que obliga a utilizar la llamada al sistema `mmap()` en lugar del gestor del Heap tradicional. Por lo tanto, se crea una nueva región de memoria anónima en el mapa, identificada únicamente por sus permisos `rw-p` y el rango de direcciones.

**-Mapa de memoria después de liberar los 10 MB del último malloc.**

```

Mirar el ctrl/prncp/IDmap y después pulsa ENTER para mirar el mapa de memoria
DESPUES del primer malloc...

free() ejecutado. El map se mantiene.
Mirar el ctrl/prncp/IDmap y después pulsa ENTER para mirar el mapa de memoria
DESPUES del free()...

# alex@leonsoportil-alex:/501/Memorias gcc ENTREGABLE1.c -o b
alex@leonsoportil-alex:/501/Memorias /D
PID del proceso: 3293
Selección el apartado a ejecutar (1-6):
1. Aparente 3: Variables globales, locales y array 3D
2. Aparente 2: Funciones f1 y f2
3. Aparente 4: Uso de malloc y free
4. Aparente 4: Proceso M1 y M2 malloc y execv
5. Aparente 3: Librería matemática estática vs dinámica
6. Aparente 5: Hilos y variables locales
Elige una opción: 3
Primer malloc de 400 bytes. Dirección = 0x5c926c39960
Mirar el ctrl/prncp/IDmap y después pulsa ENTER para mirar el mapa de memoria
DESPUES del primer malloc...
Pausa: Antes del free
Después del free.
Mirar el ctrl/prncp/IDmap y después pulsa ENTER para mirar el mapa de memoria
DESPUES del free()...

Segundo malloc de 4000 bytes. Dirección = 0x5c926c39960
Mirar el ctrl/prncp/IDmap y después pulsa ENTER para mirar el mapa de memoria
DESPUES del segundo malloc...
Pausa: Antes del free
Después del free.

Segundo malloc de 10485760 bytes. Dirección = 0x7fb2e1ff80
Mirar el ctrl/prncp/IDmap y después pulsa ENTER para mirar el mapa de memoria
DESPUES del segundo malloc...
Pausa: Antes del free
Después del free.

# alex@leonsoportil-alex:/501/Memorias
alex@leonsoportil-alex:/501/Memorias gcc /opt/cgcc3533/mps
50124d1e000-5024d1e1000 r-x 00000000 103.06 3802286
(Memoria)D
5024d1e1000-5024d1e2000 r-x 00000000 103.06 3802286
(Memoria)D
5024d1e2000-5024d1e3000 r-x 00000000 103.06 3802286
(Memoria)D
5024d1e3000-5024d1e4000 r-x 00000000 103.06 3802286
(Memoria)D
5024d1e4000-5024d1e5000 r-x 00000000 103.06 3802286
(Memoria)D
5025c399000-5025c39a000 r-x 00000000 00.00 0
(heap)
7fb2e1ff800-7fb2e1ff82000 r-x 00000000 103.06 168660
x-mmu/libc.so.6
7fb2e1ff82000-7fb2e1ff83000 r-x 00000000 103.06 168663
x-mmu/libc.so.6
7fb2e1ff83000-7fb2e1ff8ff00 r-x 00100000 103.06 168663
x-mmu/libc.so.6
7fb2e1ff8ff00-7fb2e1ff90000 r-x 00100000 103.06 168663
x-mmu/libc.so.6
7fb2e1ff90000-7fb2e1ff92000 r-x 00200000 103.06 168663
x-mmu/libc.so.6
7fb2e1ff92000-7fb2e1ff93000 r-x 00000000 00.00 0
7fb2e1ff93000-7fb2e1ff94000 r-x 00000000 00.00 0
7fb2e1ff94000-7fb2e1ff95000 r-x 00000000 00.00 0
7fb2e1ff95000-7fb2e1ff96000 r-x 00000000 00.00 0
7fb2e1ff96000-7fb2e1ff97000 r-x 00000000 00.00 0
7fb2e1ff97000-7fb2e1ff98000 r-x 00000000 00.00 0
7fb2e1ff98000-7fb2e1ff99000 r-x 00000000 00.00 0
7fb2e1ff99000-7fb2e1ff9a000 r-x 00000000 00.00 0
7fb2e1ff9a000-7fb2e1ff9b000 r-x 00000000 00.00 0
7fb2e1ff9b000-7fb2e1ff9c000 r-x 00000000 00.00 0
7fb2e1ff9c000-7fb2e1ff9d000 r-x 00000000 00.00 0
7fb2e1ff9d000-7fb2e1ff9e000 r-x 00000000 00.00 0
7fb2e1ff9e000-7fb2e1ff9f000 r-x 00000000 00.00 0
7fb2e1ff9f000-7fb2e1ffa0000 r-x 00000000 00.00 0
7fb2e1ffa0000-7fb2e1ffa1000 r-x 00000000 00.00 0
7fb2e1ffa1000-7fb2e1ffa2000 r-x 00000000 00.00 0
7fb2e1ffa2000-7fb2e1ffa3000 r-x 00000000 00.00 0
7fb2e1ffa3000-7fb2e1ffa4000 r-x 00000000 00.00 0
7fb2e1ffa4000-7fb2e1ffa5000 r-x 00000000 00.00 0
7fb2e1ffa5000-7fb2e1ffa6000 r-x 00000000 00.00 0
7fb2e1ffa6000-7fb2e1ffa7000 r-x 00000000 00.00 0
7fb2e1ffa7000-7fb2e1ffa8000 r-x 00000000 00.00 0
7fb2e1ffa8000-7fb2e1ffa9000 r-x 00000000 00.00 0
7fb2e1ffa9000-7fb2e1ffa0000 r-x 00000000 00.00 0
7fb2e1ffa0000-7fb2e1ffa1000 r-x 00000000 00.00 0
7fb2e1ffa1000-7fb2e1ffa2000 r-x 00000000 00.00 0
7fb2e1ffa2000-7fb2e1ffa3000 r-x 00000000 00.00 0
7fb2e1ffa3000-7fb2e1ffa4000 r-x 00000000 00.00 0
7fb2e1ffa4000-7fb2e1ffa5000 r-x 00000000 00.00 0
7fb2e1ffa5000-7fb2e1ffa6000 r-x 00000000 00.00 0
7fb2e1ffa6000-7fb2e1ffa7000 r-x 00000000 00.00 0
7fb2e1ffa7000-7fb2e1ffa8000 r-x 00000000 00.00 0
7fb2e1ffa8000-7fb2e1ffa9000 r-x 00000000 00.00 0
7fb2e1ffa9000-7fb2e1ffa0000 r-x 00000000 00.00 0
7fb2e1ffa0000-7fb2e1ffa1000 r-x 00000000 00.00 0
7fb2e1ffa1000-7fb2e1ffa2000 r-x 00000000 00.00 0
7fb2e1ffa2000-7fb2e1ffa3000 r-x 00000000 00.00 0
7fb2e1ffa3000-7fb2e1ffa4000 r-x 00000000 00.00 0
7fb2e1ffa4000-7fb2e1ffa5000 r-x 00000000 00.00 0
7fb2e1ffa5000-7fb2e1ffa6000 r-x 00000000 00.00 0
7fb2e1ffa6000-7fb2e1ffa7000 r-x 00000000 00.00 0
7fb2e1ffa7000-7fb2e1ffa8000 r-x 00000000 00.00 0
7fb2e1ffa8000-7fb2e1ffa9000 r-x 00000000 00.00 0
7fb2e1ffa9000-7fb2e1ffa0000 r-x 00000000 00.00 0
7fb2e1ffa0000-7fb2e1ffa1000 r-x 00000000 00.00 0
7fb2e1ffa1000-7fb2e1ffa2000 r-x 00000000 00.00 0
7fb2e1ffa2000-7fb2e1ffa3000 r-x 00000000 00.00 0
7fb2e1ffa3000-7fb2e1ffa4000 r-x 00000000 00.00 0
7fb2e1ffa4000-7fb2e1ffa5000 r-x 00000000 00.00 0
7fb2e1ffa5000-7fb2e1ffa6000 r-x 00000000 00.00 0
7fb2e1ffa6000-7fb2e1ffa7000 r-x 00000000 00.00 0
7fb2e1ffa7000-7fb2e1ffa8000 r-x 00000000 00.00 0
7fb2e1ffa8000-7fb2e1ffa9000 r-x 00000000 00.00 0
7fb2e1ffa9000-7fb2e1ffa0000 r-x 00000000 00.00 0
7fb2e1ffa0000-7fb2e1ffa1000 r-x 00000000 00.00 0
7fb2e1ffa1000-7fb2e1ffa2000 r-x 00000000 00.00 0
7fb2e1ffa2000-7fb2e1ffa3000 r-x 00000000 00.00 0
7fb2e1ffa3000-7fb2e1ffa4000 r-x 00000000 00.00 0
7fb2e1ffa4000-7fb2e1ffa5000 r-x 00000000 00.00 0
7fb2e1ffa5000-7fb2e1ffa6000 r-x 00000000 00.00 0
7fb2e1ffa6000-7fb2e1ffa7000 r-x 00000000 00.00 0
7fb2e1ffa7000-7fb2e1ffa8000 r-x 00000000 00.00 0
7fb2e1ffa8000-7fb2e1ffa9000 r-x 00000000 00.00 0
7fb2e1ffa9000-7fb2e1ffa0000 r-x 00000000 00.00 0
7fb2e1ffa0000-7fb2e1ffa1000 r-x 00000000 00.00 0
7fb2e1ffa1000-7fb2e1ffa2000 r-x 00000000 00.00 0
7fb2e1ffa2000-7fb2e1ffa3000 r-x 00000000 00.00 0
7fb2e1ffa3000-7fb2e1ffa4000 r-x 00000000 00.00 0
7fb2e1ffa4000-7fb2e1ffa5000 r-x 00000000 00.00 0
7fb2e1ffa5000-7fb2e1ffa6000 r-x 00000000 00.00 0
7fb2e1ffa6000-7fb2e1ffa7000 r-x 00000000 00.00 0
7fb2e1ffa7000-7fb
```

Al ejecutar `free()` sobre el bloque grande (10 MB), observamos que la región de memoria anónima que se había creado para esta reserva desaparece completamente del mapa de memoria. Esto sucede porque, dado que el bloque fue asignado inicialmente con la llamada al sistema `mmap()`, la función `free()` del gestor de la biblioteca C invoca a `munmap()`. Esta acción es la que desvincula el rango de direcciones virtuales de la memoria física y devuelve inmediatamente la RAM al sistema operativo. Por lo tanto, se confirma que las asignaciones grandes no se gestionan dentro del Heap tradicional, sino como mapeos temporales de memoria.

## EJERCICIO 4

-Resultados:

**-Después del fork:**

```
alexalonso@portatil-alex:~/SOI/Memorias$ cat /proc/19933/maps
5c728975d000-5c728975e000 r--p 000000000 103:06 3804484      /home/alexalonso/SOI/Memoria/b
56728975c000-56728975d000 r-xp 000010000 103:06 3804484      /home/alexalonso/SOI/Memoria/b
56728975d000-56728975e000 r--p 000020000 103:06 3804484      /home/alexalonso/SOI/Memoria/b
56728975f000-56728975g000 r--p 000030000 103:06 3804484      /home/alexalonso/SOI/Memoria/b
5672a1b6c000-5672a1b2f000 rw-p 000000000 00:00 0              [heap]
7c84ec609000-7c84ec28000 r--p 000000000 103:06 168663      /usr/lib/...
7c84ec28000-7c84ec70000 r-xp 000020000 103:06 168663      /usr/lib/...
x86_64-linux-gnu/libc.so.6
7c84ec70000-7c84ec7ff000 r--p 0010b0000 103:06 168663      /usr/lib/...
x86_64-linux-gnu/libc.so.6
7c84ec7ff000-7c84ec803000 r--p 001fe0000 103:06 168663      /usr/lib/...
x86_64-linux-gnu/libc.so.6
7c84ec803000-7c84ec805000 rw-p 002020000 103:06 168663      /usr/lib/...
x86_64-linux-gnu/libc.so.6
7c84ec805000-7c84ec812000 rw-p 000000000 00:00 0
7c84ec9fb000-7c84ec9fe000 rw-p 000000000 00:00 0
7c84eca0e000-7c84eca10000 r--p 000000000 00:00 0
7c84eca10000-7c84eca12000 r--p 000000000 00:00 0
7c84eca12000-7c84eca14000 r--p 000000000 00:00 0
[vsar]
[vsar_vcl
]
7c84eca14000-7c84eca16000 r-xp 000000000 00:00 0
7c84eca16000-7c84eca17000 r-p 000000000 103:06 168660
x86_64-linux-gnu/ld-linux-x86_64.so.2
7c84eca17000-7c84eca18000 r-xp 000010000 103:06 168660
x86_64-linux-gnu/ld-linux-x86_64.so.2
7c84eca12000-7c84eca1c000 r--p 00002c000 103:06 168606
x86_64-linux-gnu/ld-linux-x86_64.so.2
7c84eca1c000-7c84eca1e000 r-p 000036000 103:06 168660
x86_64-linux-gnu/ld-linux-x86_64.so.2
7c84eca1e000-7c84eca1f000 r-p 000038000 103:06 168660
x86_64-linux-gnu/ld-linux-x86_64.so.2
7ffef47da5000-7ffef47c7000 rw-p 000000000 00:00 0
[stack]
ffffffffffff60000-ffffffffffff61000 -xp 000000000 00:00 0
[vsyscall
```

```

# alexon@osportati1: alex - /$01/Memoria$ cat /proc/19944/maps
56728975b000-56728975c000 r--p 00000000 103:06 3804484      /home/ale
xalono/$01/Memoria/b
56728975c000-56728975d000 r-xp 00001000 103:06 3804484      /home/ale
xalono/$01/Memoria/b
56728975d000-56728975e000 r--p 00002000 103:06 3804484      /home/ale
xalono/$01/Memoria/b
56728975e000-56728975f000 r--p 00002000 103:06 3804484      /home/ale
xalono/$01/Memoria/b
56728975f000-567289760000 rw-p 00003000 103:06 3804484      /home/ale
xalono/$01/Memoria/b
5672a1b0e000-5672a1b2f000 rw-p 00000000 00:00 0      [heap]
x86_64-linux-gnu/7c84ac280000 r--p 00000000 103:06 168663      /usr/lib/
x86_64-linux-gnu/libc.so.6
7c84ac628000-7c84ac7b0000 r-xp 00028000 103:06 168663      /usr/lib/
x86_64-linux-gnu/libc.so.6
7c84ac7b0000-7c84ac7ff000 r-p 00010000 103:06 168663      /usr/lib/
x86_64-linux-gnu/libc.so.6
7c84ac7ff000-7c84ac803000 r--p 0001fe00 103:06 168663      /usr/lib/
x86_64-linux-gnu/libc.so.6
7c84ac803000-7c84ac805000 rw-p 00020200 103:06 168663      /usr/lib/
x86_64-linux-gnu/libc.so.6
7c84ac805000-7c84ac812000 rw-p 00000000 00:00 0
7c84ac9f5000-7c84ac9fe000 rw-p 00000000 00:00 0
7c84ac9fe000-7c84aca19000 rw-p 00000000 00:00 0
7c84aca10000-7c84aca12000 r--p 00000000 00:00 0      [var]
7c84aca12000-7c84aca1c000 r--p 00000000 00:00 0      [vvar_vcl
ock]
7c84aca14000-7c84aca16000 r-xp 00000000 00:00 0
7c84aca16000-7c84aca17000 r-p 00000000 103:06 168660      /usr/lib/
x86_64-linux-gnu/libd-linux-x86_64.so.2
7c84aca17000-7c84aca24000 r-xp 00010000 103:06 168660      /usr/lib/
x86_64-linux-gnu/libd-linux-x86_64.so.2
7c84aca20000-7c84aca4c000 r--p 0002c000 103:06 168660      /usr/lib/
x86_64-linux-gnu/libd-linux-x86_64.so.2
7c84aca4c000-7c84aca4e000 r--p 00036000 103:06 168660      /usr/lib/
x86_64-linux-gnu/libd-linux-x86_64.so.2
7c84aca4e000-7c84aca50000 rw-p 00030000 103:06 168660      /usr/lib/
x86_64-linux-gnu/libd-linux-x86_64.so.2
7ffe47da5000-7ffe47dc7000 rw-p 00000000 00:00 0      [stack]
ffffffffffff0000-ffffffffffff01000 --xp 00000000 00:00 0      [vsyscall]

```

**-Después del cambio de imagen:**

```
alexalonso@portatil-alex:~/SOI/Memoria$ gcc ENTREGABLE1.c -o b
alexalonso@portatil-alex:~/SOI/Memoria$ ./b
PID del proceso: 19644
Selecciona el apartado a ejecutar (1-6):
1. Apartado 1: Variables globales, locales y array 3D
2. Apartado 2: Funciones f1 y f2
3. Apartado 3: Uso de malloc y free
4. Apartado 4: Proceso hijo con malloc y execv
5. Apartado 5: Librería matemática estática vs dinámica
6. Apartado 6: Hilos y variables locales
Elige una opción: 4
Hijo: PID=19636, PPID=19644
Hijo: Pausa antes de malloc
Hijo: Pausa después de malloc
Direccion del malloc: 0x7ec8153ff018
5
Hola mundo
Pausa para mirar el mapa de memoria después del cambio de imagen
5
alexalonso@portatil-alex:~/SOI/Memoria$ gcc ENTREGABLE1.c -o b
alexalonso@portatil-alex:~/SOI/Memoria$ ./b
PID del proceso: 19933
Selecciona el apartado a ejecutar (1-6):
1. Apartado 1: Variables globales, locales y array 3D
2. Apartado 2: Funciones f1 y f2
3. Apartado 3: Uso de malloc y free
4. Apartado 4: Proceso hijo con malloc y execv
5. Apartado 5: Librería matemática estática vs dinámica
6. Apartado 6: Hilos y variables locales
Elige una opción: 4
Hijo: PID=19944, PPID=19933
Hijo: Pausa antes de malloc
Hijo: Pausa después de malloc
Direccion del malloc: 0x7c84ebff018
5
Hola mundo
Pausa para mirar el mapa de memoria después del cambio de imagen
alexalonso@portatil-alex:~/SOI/Memoria$ cat /proc/19944/maps
5ac00451000-5ac00450000 r--p 00000000 103:06 3817246 /home/ale
xalonso/SOI/Memoria/programa
5ac00454000-5ac00450000 r-xp 00001000 103:06 3817246 /home/ale
xalonso/SOI/Memoria/programa
5ac00458000-5ac0045f000 r--p 00002000 103:06 3817246 /home/ale
xalonso/SOI/Memoria/programa
5ac0045f000-5ac00550000 r--p 00002000 103:06 3817246 /home/ale
xalonso/SOI/Memoria/programa
5ac218f0000-5ac21910000 rw-p 00000000 00:00 0 [heap]
70b9c360000-70b9c320000 r--p 00000000 103:06 168663 /usr/lib/
x86_64-linux-gnu/libc.so.6
70b9c320000-70b9c370000 r-xp 00028000 103:06 168663 /usr/lib/
x86_64-linux-gnu/libc.so.6
70b9c370000-70b9c37ff000 r--p 001b0000 103:06 168663 /usr/lib/
x86_64-linux-gnu/libc.so.6
70b9c37ff000-70b9c303000 r--p 001fe000 103:06 168663 /usr/lib/
x86_64-linux-gnu/libc.so.6
70b9c303000-70b9c303000 rw-p 00202000 103:06 168663 /usr/lib/
x86_64-linux-gnu/libc.so.6
70b9c303000-70b9c3812000 rw-p 00000000 00:00 0 [vdso]
70b9c381000-70b9c352000 r--p 00000000 103:06 168660 /usr/lib/
x86_64-linux-gnu/ld-linux-x86_64.so.2
70b9c352000-70b9c370000 r-xp 00001000 103:06 168660 /usr/lib/
x86_64-linux-gnu/ld-linux-x86_64.so.2
70b9c370000-70b9c3825000 r--p 0002c000 103:06 168660 /usr/lib/
x86_64-linux-gnu/ld-linux-x86_64.so.2
70b9c3825000-70b9c3837000 r--p 00030000 103:06 168660 /usr/lib/
x86_64-linux-gnu/ld-linux-x86_64.so.2
70b9c3837000-70b9c3820000 rw-p 00030000 103:06 168660 /usr/lib/
7fffd09625000-7fffd09647000 rw-p 00000000 00:00 0 [stack]
fffffffff60000-fffffffff601000 --xp 00000000 00:00 0 [vsyscall]
```

Al ejecutar `execv` o `execclp`, se produce una destrucción y reconstrucción completa del espacio de direcciones del proceso hijo, reemplazando el mapa de memoria original. Esto significa que todas las regiones previas heredadas del padre, incluyendo el código (TEXT), los datos estáticos (DATA/BSS) y cualquier memoria dinámica asignada previamente con `malloc` o `mmap` (incluso si era grande), son eliminadas. El mapa de memoria resultante solo contendrá las regiones esenciales (TEXT, DATA, STACK) del nuevo programa cargado, logrando que el hijo ejecute un código totalmente nuevo sin rastro de su predecesor.

## EJERCICIO 5

-Resultados:

-Enlazamos de forma dinámica y estática:

Mapa de memoria (-lm)

```
alexalonso@portatil-alex:~/SOI/Memoria$ cat /proc/21493/maps
5a9cdebcd000-5a9cdebcd7000 r--p 00000000 103:06 3804484 /home/ale
xalonso/SOI/Memoria/b
5a9cdebcd7000-5a9cdebcd8000 r-xp 00001000 103:06 3804484 /home/ale
xalonso/SOI/Memoria/b
5a9cdebcd8000-5a9cdebcd9000 r--p 00002000 103:06 3804484 /home/ale
xalonso/SOI/Memoria/b
5a9cdebcd9000-5a9cdebcdca000 r--p 00002000 103:06 3804484 /home/ale
xalonso/SOI/Memoria/b
5a9cdebcdca000-5a9cdebcdcb000 rw-p 00003000 103:06 3804484 /home/ale
xalonso/SOI/Memoria/b
5a9db4ecb000-5a9db4ecb000 rw-p 00000000 00:00 0 [heap]
70754b400000-70754b428000 r--p 00000000 103:06 168663 /usr/lib/
x86_64-linux-gnu/libc.so.6
70754b428000-70754b5b0000 r-xp 00028000 103:06 168663 /usr/lib/
x86_64-linux-gnu/libc.so.6
70754b5b0000-70754b5ff000 r--p 001b0000 103:06 168663 /usr/lib/
x86_64-linux-gnu/libc.so.6
70754b5ff000-70754b603000 r--p 001fe000 103:06 168663 /usr/lib/
x86_64-linux-gnu/libc.so.6
70754b603000-70754b605000 rw-p 00202000 103:06 168663 /usr/lib/
x86_64-linux-gnu/libc.so.6
70754b605000-70754b612000 rw-p 00000000 00:00 0 [vdso]
70754b612000-70754b654000 rw-p 00000000 00:00 0 [vdso]
70754b654000-70754b666000 r--p 00000000 00:00 0 [vdso]
70754b666000-70754b668000 r--p 00000000 00:00 0 [vdso]
70754b668000-70754b66a000 r--p 00000000 00:00 0 [vdso]
70754b66a000-70754b66c000 r-xp 00000000 00:00 0 [vdso]
x86_64-linux-gnu/ld-linux-x86_64.so.2
70754b66c000-70754b698000 r-xp 00001000 103:06 168660 /usr/lib/
x86_64-linux-gnu/ld-linux-x86_64.so.2
70754b698000-70754b6a2000 r--p 0002c000 103:06 168660 /usr/lib/
x86_64-linux-gnu/ld-linux-x86_64.so.2
70754b6a2000-70754b6a4000 r--p 00030000 103:06 168660 /usr/lib/
x86_64-linux-gnu/ld-linux-x86_64.so.2
70754b6a4000-70754b6a6000 rw-p 00030000 103:06 168660 /usr/lib/
x86_64-linux-gnu/ld-linux-x86_64.so.2
7ffdb096000-7ffdb096000 r--p 00000000 00:00 0 [stack]
fffffffff60000-fffffffff601000 --xp 00000000 00:00 0 [vsyscall]
```

Mapa de memoria (-static -lm)

```
alexalonso@portatil-alex:~/SOI/Memoria$ cat /proc/21822/maps
00400000-00401000 r--p 00000000 103:06 3802286 /home/ale
xalonso/SOI/Memoria/b
00401000-00403000 r-xp 00001000 103:06 3802286 /home/ale
xalonso/SOI/Memoria/b
00403000-0040e000 r--p 000b3000 103:06 3802286 /home/ale
xalonso/SOI/Memoria/b
0040e000-004e3000 r--p 000dd000 103:06 3802286 /home/ale
xalonso/SOI/Memoria/b
004e3000-004e5000 rw-p 000e2000 103:06 3802286 /home/ale
xalonso/SOI/Memoria/b
004e5000-004eb000 rw-p 00000000 00:00 0 [heap]
10b07000-10b29000 rw-p 00000000 00:00 0 [vvar]
71d178379000-71d17837b000 r--p 00000000 00:00 0 [vvar_vcl
ock]
71d17837b000-71d17837d000 r--p 00000000 00:00 0 [vdso]
7ffdd12216000-7ffdd12238000 rw-p 00000000 00:00 0 [stack]
fffffffff60000-fffffffff601000 --xp 00000000 00:00 0 [vsyscall]
```



## EJERCICIO 6

**- Creamos los hilos, imprimimos direcciones de memoria y observamos el mapa:**

[illegible]

**-Parámetros y variables locales de cada hilo:**

Observamos que para cada hilo, parámetro y variable local pertenecen a la misma región de memoria.

Estas se ubican en las pilas privadas de cada hijo. Aparecen como bloques de memoria anónima que, aunque se encuentran en rangos de direcciones distintos entre sí y separadas del [stack], garantizan el aislamiento del contexto de ejecución de cada hilo. Estas regiones dinámicas y separadas se asignan con mmap, lo que explica por qué sus direcciones son muy diferentes a la pila principal y a las pilas de los otros hilos, y por qué parecen estar en la misma zona que el heap.

#### **-Memoria dinámica creada por malloc:**

Observamos que las dos direcciones de memoria creadas por el malloc pertenecen a la misma región: 77e5a15fc000-77e5a29fe000 rw-p.

Se ubican en el heap, que es una región de memoria compartida por todos los hilos del proceso, etiquetada como [heap] y con permisos rw-p. Las llamadas a malloc realizadas por el hilo principal y por los hilos secundarios asignan memoria desde este mismo espacio, resultando en que todas las asignaciones dinámicas se encuentren en el mismo rango de direcciones virtuales. Debido a que el HEAP es un recurso compartido, la biblioteca C debe implementar la gestión de malloc de forma segura para hilos para prevenir conflictos de asignación concurrentes.