

Práctica 5 - Mapa de memoria.

ADRIÁN QUIROGA LINARES, IZAN MONTERROSO CERNADAS

INTRODUCCIÓN

En el presente documento se analizará el fichero *maps* de diferentes procesos, cuyos espacios de direcciones han sido sometidos a diversos cambios o manipulaciones. Este análisis tiene como objetivo comprender cómo el sistema operativo gestiona las diferentes regiones de memoria asociadas a un proceso y cómo estas cambian ante ciertas operaciones, como la reserva dinámica de memoria o el manejo del montón (*heap*).

Se hará especial énfasis en los siguientes aspectos:

- La forma en que el sistema operativo organiza las regiones de memoria virtual.
- La asignación y liberación de memoria en el montón mediante operaciones como `malloc` y `free`.
- La interacción entre las pilas de los hilos y otras regiones de memoria en un contexto multihilo.

Para facilitar la comprensión, las diferentes regiones de memoria se identificarán con colores específicos en las imágenes presentadas: el segmento de texto en **naranja**, la pila en **verde**, y el *stack* (pila de cada hilo) en **rojo**.

Este análisis busca proporcionar una visión clara y práctica de los mecanismos de asignación y gestión de memoria en los sistemas modernos, fundamentales para el diseño y optimización de programas eficientes.

I. EJERCICIO 1

El primer ejercicio consistía en analizar qué lugar de el espacio de direcciones le corresponde a variables de distinta índole (globales, no inicializadas, locales...). A continuación se comenta lo observado:

- Variables globales: En el código proporcionado todas las variables globales han sido inicializadas por lo que se almacenan en el segmento de datos (marcado en azul en la siguiente imagen). Sin embargo, si no son inicializadas se almacenarían en el BSS (*Block Started by Symbol*) una región dedicada a variables estáticas y globales no inicializadas.
- Variables locales: Se declaran e inicializan distintas variables locales y todas ellas se guardan en la pila mas esto no depende de que hayan sido inicializadas, siempre se almacenan en la misma región de memoria.
- Funciones: La función main de el código se encuentra en el segmento de texto, lugar dedicado al almacenamiento del código del programa en ejecución.

```

64b3250f0000-64b3250f0000 r--p 00000000 103:04 1258475
64b3250f0000-64b3250f1000 r--xp 00001000 103:04 1258475
64b3250f1000-64b3250f2000 r--p 00002000 103:04 1258475
64b3250f2000-64b3250f3000 r--p 00002000 103:04 1258475
64b3250f3000-64b3250f4000 rw-p 00003000 103:04 1258475
64b3250f4000-64b3250f45000 rw-p 00003000 00:00 0 [heap]
7f3d81e00000-7f3d81e28000 r--p 00000000 103:04 5782528
7f3d81e28000-7f3d81e40000 r--xp 00000000 103:04 5782528
7f3d81ff0000-7f3d82003000 r--p 001fe000 103:04 5782528
7f3d82003000-7f3d82005000 r--p 00202000 103:04 5782528
7f3d82005000-7f3d820120000 rw-p 00000000 00:00 0
7f3d8218a000-7f3d8218d000 rw-p 00000000 00:00 0
7f3d821a0000-7f3d821a2000 rw-p 00000000 00:00 0
7f3d821a2000-7f3d821a3000 r--p 00000000 103:04 5782340
7f3d821a3000-7f3d821a4000 r--xp 00001000 103:04 5782340
7f3d821c0000-7f3d821d0000 r--p 00002000 103:04 5782340
7f3d821d0000-7f3d821d9000 rw-p 00036000 103:04 5782340
7f3d821d9000-7f3d821d9000 rw-p 00038000 103:04 5782340
[fc]74968000-7fc74969000 rw-p 00000000 00:00 0 [stack]
[fc]74970000-7fc74972000 r--p 00000000 00:00 0
7f3d821e0000-7f3d821e0000 r--p 00000000 00:00 0
7f3d821f0000-7f3d821f0000 r--p 00000000 00:00 0
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0 [vsyscall]

```

```

> ./Ejercicio1
Dirección de var1: 0x64b3250f3020
Dirección de var2: 0x64b3250f3024
Dirección de var3: 0x64b3250f3028
Dirección del array: 0x64b3250f3040
Dirección de varlocal1: 0x7ffc74984ed0
Dirección de varlocal2: 0x7ffc74984ecf
Dirección de arraylocal: 0x7ffc74984ee0
Variable no inicializada: 0x7ffc74984ed4
Dirección de la función main: 0x64b3250f0189

```

II. EJERCICIO 2

El segundo ejercicio consistía en un análisis más detallado del uso de memoria de la pila o *stack* mediante las llamadas a distintas funciones.

- Funciones: Al igual que en el Ejercicio 1 el código correspondiente se encuentra en el segmento de texto con permisos *read* y *execute*.
- Argumentos y variables de las funciones: Tanto los argumentos como las variables ocupan las mismas posiciones en la pila pero esto no es sorprendente pues al ejecutarse de forma secuencial el espacio de memoria liberado por *funcion1* es utilizado por *funcion2*. Además, como ambas usan los mismos tipos y cantidad de datos, coinciden no solo las posiciones de inicio en la pila sino que también las de cada variable.

```

617608a10000-617608a17000 r--p 00000000 103:04 1258475
617608a17000-617608a18000 r--xp 00001000 103:04 1258475
[fc]700818000-7f0818001000 r--p 00000000 103:04 1258475
7f0818001000-7f0818001a000 r--p 00002000 103:04 1258475
7f0818001a000-7f0818001b000 rw-p 00003000 103:04 1258475
7f0818001b000-7f0818001b000 rw-p 00003000 00:00 0 [heap]
7f0818002000-7f08180028000 r--p 00000000 103:04 5782528
7f08180028000-7f08180030000 r--xp 00002000 103:04 5782528
7f08180030000-7f08180032000 r--p 0001b000 103:04 5782528
7f08180032000-7f08180034000 r--p 0001fe000 103:04 5782528
7f08180034000-7f08180036000 rw-p 000202000 103:04 5782528
7f08180036000-7f08180038000 rw-p 000202000 00:00 0
7f08180038000-7f0818003a000 rw-p 00000000 00:00 0
7f0818003a000-7f0818003b000 rw-p 00000000 00:00 0
7f0818003b000-7f0818003c000 r--p 00000000 103:04 5782340
7f0818003c000-7f0818003d000 r--xp 00001000 103:04 5782340
7f0818003d000-7f0818003e000 r--p 00002000 103:04 5782340
7f0818003e000-7f0818003f000 r--p 00003000 103:04 5782340
7f0818003f000-7f08180040000 r--p 00003000 00:00 0 [heap]
7f08180040000-7f08180042000 r--p 00000000 103:04 5782340
7f08180042000-7f08180043000 r--xp 00002000 103:04 5782340
7f08180043000-7f08180044000 r--p 00003000 103:04 5782340
7f08180044000-7f08180045000 r--p 00003000 00:00 0
7fd15630000-7fd156510000 rw-p 00000000 00:00 0 [stack]
[fc]70120000-7f0120001000 r--p 00000000 00:00 0
7fd156ff000-7fd15701000 r--xp 00000000 00:00 0
7fd156ff000-7fd15701000 r--p 00000000 00:00 0
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0 [vsyscall]

```

```

> ./Ejercicio2
Pid: 30617
Dir mem arg funcion1: 0x7ffd1564dd2c
Dir mem vLocal funcion1: 0x7ffd1564dd34
Dir mem arg funcion2: 0x7ffd1564dd2c
Dir mem vLocal funcion2: 0x7ffd1564dd34
Dir memme funcion1: 0x6176d8a171a9
Dir memme funcion2: 0x6176d8a1721b

```

III. EJERCICIO 3

Este apartado se basa en estudiar las funciones malloc y free. Las cuales reservan y liberán memoria dinámica respectivamente.

- Varios *malloc*: La ejecución de varios *malloc* no tiene porque influir el tamaño del montón a pesar de ser llamadas distintas. Solo tendrá lugar un cambio si se quiere reservar más espacio del que se tiene disponible en dicho momento. En este caso se reservamos memoria para 5 bytes dos veces.

```

5f0108735000-5f0108736000 r--p 00000000 103:04 12584616
5f0108736000-5f0108737000 r--p 00001000 103:04 12584616
5f0108737000-5f0108738000 r--p 00002000 103:04 12584616
5f0108738000-5f0108739000 r--p 00002000 103:04 12584616
5f0108739000-5f010873a000 rw-p 00003000 103:04 12584616
5f0109a08000-5f0109a09000 rw-p 00000000 00:00 0 [heap]
0f083750000-7fd37528000 r--p 00000000 103:04 5782528
7fd37528000-7fd37529000 r--p 00001000 103:04 5782528
7fd37529000-7fd3752ff000 r--p 001b0000 103:04 5782528 [página de memoria]
7fd3752ff000-7fd3a7c03000 r--p 001fe000 103:04 5782528
7fd3a7c03000-7fd3a7c05000 r--p 00202000 103:04 5782528 [página de memoria]
7fd3a7c05000-7fd3a7c12000 rw-p 00000000 00:00 0 [pila]
7fd3a7c40000-7fd3a7c50000 rw-p 00000000 00:00 0
7fd3a7c63000-7fd3a7c65000 r--p 00000000 00:00 0
7fd3a7c65000-7fd3a7c66000 r--p 00000000 103:04 5782340
7fd3a7c66000-7fd3a7c91000 r--p 00001000 103:04 5782340
7fd3a7c91000-7fd3a7c92000 r--p 00002000 103:04 5782340
7fd3a7c92000-7fd3a7c93000 r--p 00003000 103:04 5782340
7fd3a7c93000-7fd3a7c97000 rw-p 00038000 103:04 5782340 [memoria]
7fffaad01000-7fffaad0b5000 r--p 00000000 00:00 0 [stack]
7fffaad0b5000-7fffaad07000 r--p 00000000 00:00 0 [var]
7fffff7600000-ffffffffffff601000 --xp 00000000 00:00 0 [vdso]
ffffffffff60000-ffffffffffff601000 --xp 00000000 00:00 0 [vsyscall]

```

- Uso de *realloc*: Usamos la función *realloc* para comprobar como crece la memoria reservada para el montón así como para ver como se crean diferentes regiones de memoria dentro de él.

```

5f0108735000-5f0108736000 r--p 00000000 103:04 12584616
5f0108736000-5f0108737000 r--p 00001000 103:04 12584616
5f0108737000-5f0108738000 r--p 00002000 103:04 12584616
5f0108738000-5f0108739000 r--p 00002000 103:04 12584616
5f0108739000-5f010873a000 rw-p 00003000 103:04 12584616
5f0109a08000-5f0109a09000 rw-p 00000000 00:00 0 [heap]
7fd3a760000-7fd3a7801000 rw-p 00000000 00:00 0
7fd3a760000-7fd3a7801000 r--p 00000000 103:04 5782528
7fd3a7728000-7fd3a77b0000 r--p 00028000 103:04 5782528
7fd3a77b0000-7fd3a77c0000 r--p 001b0000 103:04 5782528
7fd3a77c0000-7fd3a7c3000 r--p 001fe000 103:04 5782528
7fd3a7c3000-7fd3a7c4000 rw-p 00020000 103:04 5782528 [página de memoria]
7fd3a7c40000-7fd3a7c50000 rw-p 00000000 00:00 0
7fd3a7c63000-7fd3a7c65000 r--p 00000000 00:00 0
7fd3a7c65000-7fd3a7c66000 r--p 00000000 103:04 5782340
7fd3a7c66000-7fd3a7c71000 r--p 00001000 103:04 5782340
7fd3a7c71000-7fd3a7c72000 r--p 00002000 103:04 5782340
7fd3a7c72000-7fd3a7c73000 r--p 00003000 103:04 5782340
7fd3a7c73000-7fd3a7c74000 rw-p 00038000 103:04 5782340 [memoria]
7fffaad01000-7fffaad0b5000 r--p 00000000 00:00 0 [stack]
7fffaad0b5000-7fffaad07000 r--p 00000000 00:00 0 [var]
7fffff7600000-ffffffffffff601000 --xp 00000000 00:00 0 [vdso]
ffffffffff60000-ffffffffffff601000 --xp 00000000 00:00 0 [vsyscall]

```

- Uso de *alloc*: Esta función permite asignar memoria de forma dinámica en la pila, ajustando así el rango de direcciones del segmento pero no incrementando el tamaño de

la pila, y al contrario que con *malloc* no es necesario hacer un free (es automático cuando se libera la función que invoca *alloca*). Su principal desventaja es el riesgo de desbordamiento.

```

5f0108735000-5f0108736000 r--p 00000000 103:04 12584616 /home/adriang1/Escritorio/SOI/Practicas/Ejercicio3
5f0108734000-5f0108737000 r--p 00000000 103:04 12584616 /home/adriang1/Escritorio/SOI/Practicas/Ejercicio3
5f0108737000-5f0108738000 r--p 00002000 103:04 12584616 [memoria]
5f0108738000-5f0108739000 r--p 00002000 103:04 12584616 /home/adriang1/Escritorio/SOI/Practicas/Ejercicio3
5f0108739000-5f010873a000 rw-p 00003000 103:04 12584616 [stack]
5f0109ab8000-5f0109a9d000 rw-p 00000000 00:00 0 [heap]
7fd37a0000-7fd3a728000 r--p 00000000 103:04 5782528 [pequeña de memoria]
7fd37a28000-7fd3a7b0000 r--p 000028000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd37b0000-7fd3a7bf000 r--p 001b0000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd37bf000-7fd3a7c03000 r--p 001b0000 103:04 5782528 [pila en la pila...]
7fd37c03000-7fd3a7c05000 rw-p 00202000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd37c05000-7fd3a7c12000 rw-p 00000000 00:00 0 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd37c40000-7fd3a7c50000 rw-p 00000000 00:00 0
7fd3a7c63000-7fd3a7c65000 rw-p 00000000 00:00 0
7fd3a7c65000-7fd3a7c66000 r--p 00000000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd3a7c66000-7fd3a7c71000 r--p 00001000 103:04 5782340 [memoria]
7fd3a7c71000-7fd3a7c71000 r--p 00002000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd3a7c71000-7fd3a7c71d000 r--p 00034000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd3a7c71d000-7fd3a7c74f000 r--p 00033000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7fffaa29f000-7fffaad3a000 rw-p 00000000 00:00 0 [stack]
7fffaad001000-7fffaad05000 r--p 00000000 00:00 0 [vvar]
7fffaad05000-7fffaad77000 r--p 00000000 00:00 0 [vdso]
ffffffffffff600000-ffffffffffff601000 --xp 00000000 00:00 0 [vsyscall]

```

```

> ./Ejercicio3
Presiona enter para realizar la primera reserva pequeña de memoria
75440fb0ec000-75440fb0e000 r--p 00000000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
Memoria reservada con malloc en la dirección: 0x62c20a7b3ac0 /usr/lib/x86_64-linux-gnu/libc.so.6
Presiona Enter para continuar realizar la segunda reserva pequeña de memoria.../usr/x86_64-linux-gnu/libc.so.6
75440fc22000-75440fc24000 r--p 00038000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
Memoria reservada con malloc en la dirección: 0x62c20a7b3ae0 /usr/lib/x86_64-linux-gnu/libc.so.6
Presiona Enter para continuar y reservar memoria con alloca en la pila...[stack]
7ffdbaed4000-7ffdbaed8000 r--p 00000000 00:00 0 [vvar]
[1] 9558 segmentation fault (core dumped) ./Ejercicio3 [vdso]

```

- Uso de *free*: *free* libera toda la memoria reservada por el *malloc* y borra el segmento del montón correspondiente si solo lo ocupaban dichos datos. Si dos *mallocs* conviven en el mismo segmento del *heap* aunque uno haga un *free* no se borra el segmento del archivo *maps*.

IV. EJERCICIO 4

En este ejercicio se observó como el hijo copia el espacio de direcciones del padre así como las direcciones virtuales de las distintas regiones de memoria (1^a imagen) pero una vez realizado el *fork* ambos se vuelven independientes. Por ejemplo el *malloc* realizado en el código del hijo le afecta exclusivamente a él, creando una diferencia entre ambos mapas de memoria (2^a imagen).

```

634b20376000-634b2037f000 r--p 00000000 103:04 12584823 /home/adriang1/Escritorio/SOI/Practicas/Ejercicio4
634b2037f000-634b20380000 r--p 00001000 103:04 12584823 /home/adriang1/Escritorio/SOI/Practicas/Ejercicio4
634b20380000-634b20381000 r--p 00002000 103:04 12584823 /home/adriang1/Escritorio/SOI/Practicas/Ejercicio4
634b20381000-634b20382000 r--p 00003000 103:04 12584823 /home/adriang1/Escritorio/SOI/Practicas/Ejercicio4
634b20382000-634b20383000 rw-p 00003000 103:04 12584823 /home/adriang1/Escritorio/SOI/Practicas/Ejercicio4
634b21280000-634b21d49000 rw-p 00000000 00:00 0 [heap]
7fd413000000-7fd4132c0000 r--p 00000000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd413028000-7fd4131b0000 r--p 00028000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd4131b0000-7fd4131ff000 r--p 001b0000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd4131ff000-7fd4132b3000 r--p 001fe000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd4132b3000-7fd4132b5000 rw-p 00202000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd4132b5000-7fd4132c2000 rw-p 00000000 00:00 0
7fd4132c2000-7fd4132c8000 rw-p 00000000 00:00 0
7fd4132c8000-7fd4132c9000 rw-p 00000000 00:00 0
7fd4132c9000-7fd4132cb000 rw-p 00001000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd4132cb000-7fd4132d000 r--p 0002c000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd4132d000-7fd4132d7000 r--p 00033000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd4132d7000-7fd4132d9000 r--p 00036000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7fd4132d9000-7fd4132dc000 rw-p 00038000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7ffca52a7000-7ffca52c8000 rw-p 00000000 00:00 0 [stack]
7ffca53a5000-7ffca53a9000 r--p 00000000 00:00 0 [vvar]
7ffca53a9000-7ffca53ab000 r--p 00000000 00:00 0 [vdso]
ffffffffffff600000-ffffffffffff601000 --xp 00000000 00:00 0 [vsyscall]

```

```

0x4ab2037e000-634b2937f000 r--p 00000000 103:04 12584823 /home/adrianql/Escritorio/SOI/Practica5/Ejercicio4
0x4ab2037f000-634b29380000 r--p 00001000 103:04 12584823 /home/adrianql/Escritorio/SOI/Practica5/Ejercicio4
634b20380000-634b20381000 r--p 00002000 103:04 12584823 /home/adrianql/Escritorio/SOI/Practica5/Ejercicio4
634b20381000-634b20382000 r--p 00002000 103:04 12584823 /home/adrianql/Escritorio/SOI/Practica5/Ejercicio4
634b20382000-634b20383000 rw-p 00003000 103:04 12584823 /home/adrianql/Escritorio/SOI/Practica5/Ejercicio4
634b20383000-634b20384000 [heap] [0]
0x4ab20384000-634b20385000 rw-p 00000000 00:00 0
0x7413208000-7413208000 r--p 00002000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7b7413208000-7b7413208000 r--p 00028000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7b74131ff000-7b74131ff000 r--p 00100000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7b74131ff000-7b7413203000 r--p 0011e000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7b7413203000-7b7413205000 rw-p 00202000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7b7413205000-7b7413212000 rw-p 00000000 00:00 0
7b7413279000-7b741327c000 rw-p 00000000 00:00 0
7b741327c000-7b741327c000 r--p 00000000 00:00 0
7b741327c000-7b741327c000 r--p 00000000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7b741327c000-7b741327c000 r--p 00001000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7b74132d0000-7b74132c7000 r--p 0002c000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7b74132c7000-7b74132c9000 r--p 00036000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7b74132c9000-7b74132cb000 rw-p 00038000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7ffca52a7000-7ffca52c8000 rw-p 00000000 00:00 0 [stack]
7ffca53a5000-7ffca53a9000 r--p 00000000 00:00 0 [vvar]
7ffca53a9000-7ffca53ab000 r--p 00000000 00:00 0 [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0 [syscall]

```

La segunda parte del ejercicio consistía en observar los cambios producidos por un cambio de imagen usando las funciones exec (en este caso *execv*). La principal diferencia despues de esta ejecución es el reestructuramiento del espacio de direcciones y la asignación de nuevas direcciones virtuales.

```

0x4edcfd1f000-64edcfcd20000 r--p 00000000 103:04 12584785 /home/adrianql/Escritorio/SOI/Practica5/Ejercicio4.2
64edcfcd20000-64edcfcd20000 r--p 00001000 103:04 12584785 /home/adrianql/Escritorio/SOI/Practica5/Ejercicio4.2
64edcfcd21000-64edcfcd22000 r--p 00002000 103:04 12584785 /home/adrianql/Escritorio/SOI/Practica5/Ejercicio4.2
64edcfcd22000-64edcfcd30000 r--p 00002000 103:04 12584785 /home/adrianql/Escritorio/SOI/Practica5/Ejercicio4.2
64edcfcd30000-64edcfcd40000 r--p 00003000 103:04 12584785 /home/adrianql/Escritorio/SOI/Practica5/Ejercicio4.2
64edcfcd40000-64edcfcd40000 rw-p 00003000 103:04 12584785 /home/adrianql/Escritorio/SOI/Practica5/Ejercicio4.2
64edd18c0000-64eddd15d0000 rw-p 00000000 00:00 0 [heap]
7b7405400000-7b7405428000 r--p 00000000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7b7405428000-7b74055b0000 r--p 00028000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7b74055b0000-7b74055f0000 r--p 001b0000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7b74055f0000-7b7405603000 r--p 001fe000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7b7405603000-7b7405605000 r--p 00202000 103:04 5782528 /usr/lib/x86_64-linux-gnu/libc.so.6
7b7405605000-7b7405612000 rw-p 00000000 00:00 0
7b7405704000-7b7405720000 rw-p 00000000 00:00 0
7b7405720000-7b7405720000 r--p 00000000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7b7405723000-7b7405746000 r--p 00001000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7b7405746000-7b7405758000 r--p 00027000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7b7405758000-7b740575a000 r--p 00036000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7b740575a000-7b740575c000 rw-p 00038000 103:04 5782340 /usr/lib/x86_64-linux-gnu/libc.so.6
7ffca52ab44000-7ffca52ab5000 rw-p 00000000 00:00 0 [stack]
7ffca52ab5000-7ffca52ab7000 r--p 00000000 00:00 0 [vvar]
7ffca52ab7000-7ffca52ab7000 --xp 00000000 00:00 0 [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0 [syscall]

```

La tercera parte del ejercicio consistía en analizar el puntero devuelto por malloc. Al aplicarle un *sizeof* nos devuelve el tamaño del tipo de dato al que apunta y si lo incrementamos (aritmética de punteros) se desplaza dicha cantidad de bytes en la región de memoria.

V. EJERCICIO 5

En este ejercicio evaluamos las diferencias entre enlazar una librería de forma estática o de forma dinámica aparte de su impacto en la memoria y en el tamaño del fichero ejecutable.

- Enlace estático: El mapa de memoria resultante con este método es realmente pequeño, contando únicamente con los segmentos de memoria imprescindibles como son: el montón, la pila, el segmento de texto...Además la ejecución se realiza en un tiempo menor respecto al enlace dinámico. A cambio el ejecutable tiene un tamaño desorbitado para la sencillez del programa .

```

00400000-00401000 r--p 00000000 103:04 12584790
00401000-004a2000 r--xp 00001000 103:04 12584790
004a2000-004cb000 r--p 000a2000 103:04 12584790
004cb000-004d0000 r--p 000ca000 103:04 12584790
004d0000-004d2000 rw-p 000cf000 103:04 12584790
004d2000-004d3000 rw-p 00000000 00:00 0
01e23000-01e45000 rw-p 00000000 00:00 0
7ffe0bde1000-7ffe0be02000 rw-p 00000000 00:00 0
7ffe0b788000-7ffe0bf8c000 r--p 00000000 00:00 0
7ffe0bf8c000-7ffe0bf8e000 r--xp 00000000 00:00 0
fffffffff600000-ffffffffffff601000 --xp 00000000 00:00 0

```

[home/adrianql/Escritorio/SOI/Practica5/programa_estatico
[home/adrianql/Escritorio/SOI/Practica5/programa_estatico
[home/adrianql/Escritorio/SOI/Practica5/programa_estatico
[home/adrianql/Escritorio/SOI/Practica5/programa_estatico
[home/adrianql/Escritorio/SOI/Practica5/programa_estatico
[home/adrianql/Escritorio/SOI/Practica5/programa_estatico
[home/adrianql/Escritorio/SOI/Practica5/programa_estatico
[heap]
[stack]
[var]
[vdso]
[vsyscall]

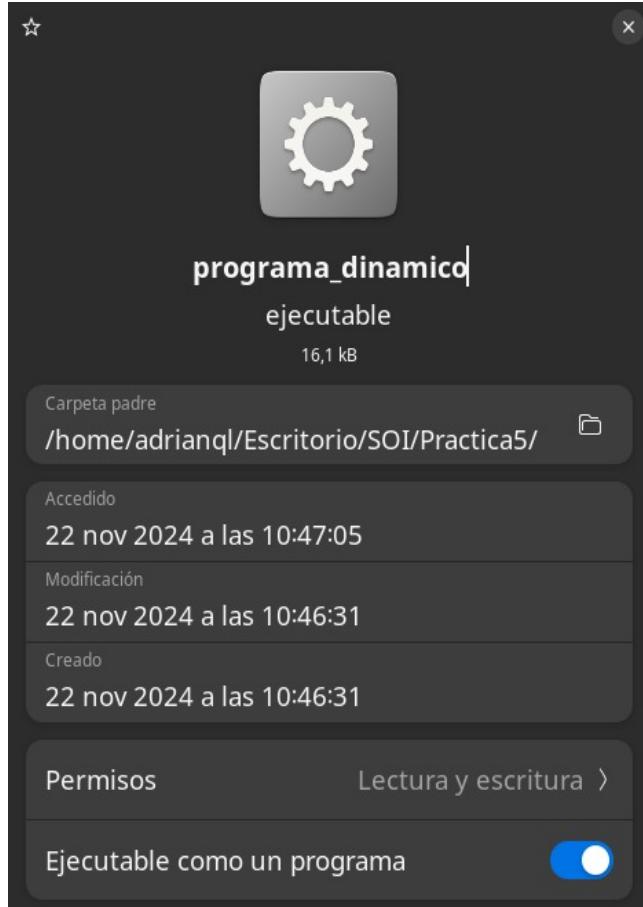


- Enlace dinámico: El mapa de memoria resultante es mucho más amplio y cuenta con numerosas regiones respaldadas por ficheros .so propios de las bibliotecas dinámicas. Con todo, la ventaja principal es la creación de un ejecutable aproximadamente sesenta veces más pequeño que con un enlace estático, a cambio se tiene que acceder a otros ficheros en tiempo de ejecución lo que podría llegar a complicar la portabilidad y eficiencia del programa.

```

5b08abf62000-5bb9abf63000 r--p 00000000 103:04 12584684
5b08abf63000-5bb9abf64000 r--p 00001000 103:04 12584684
5b08abf64000-5bb9abf65000 r--p 00002000 103:04 12584684
5b08abf65000-5bb9abf66000 r--p 00003000 103:04 12584684
5b08abf66000-5bb9abf67000 rw-p 00003000 103:04 12584684
5b08ada84000-5bb9adaaa5000 rw-p 00000000 00:00 0
[heap]
79550a000000-79550a628000 r--p 00000000 103:04 5782528
79550a628000-79550a700000 r--p 00028000 103:04 5782528
79550a700000-79550a7f0000 r--p 00106000 103:04 5782528
79550a7ff000-79550a803000 r--p 001fe000 103:04 5782528
79550a803000-79550a805000 rw-p 00200000 103:04 5782528
79550a805000-79550a806000 rw-p 00000000 00:00 0
79550a806000-79550a807000 rw-p 00000000 00:00 0
79550a807000-79550a812000 r--p 00000000 103:04 5782871
79550a812000-79550a817000 r--p 00010000 103:04 5782871
79550a817000-79550a822000 r--p 0001f000 103:04 5782871
79550a822000-79550a840000 r--p 0000e7000 103:04 5782871
79550a840000-79550a845000 rw-p 0000e8000 103:04 5782871
79550a845000-79550a846000 rw-p 00000000 00:00 0
79550a846000-79550a849000 r--p 00000000 103:04 5782340
79550baa9d000-79550baa7000 r--p 00001000 103:04 5782340
79550baa7000-79550baa7000 r--p 00001000 103:04 5782340
7fc373660000-7ffc37387000 rw-p 00000000 00:00 0
7fc373af000-7ffc373b3000 r--p 00000000 00:00 0
7fc373b3000-7ffc373b5000 r--p 00000000 00:00 0
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0
[stack]
[vvar]
[vdsos]
[vsystcall]

```



VI. EJERCICIO 6

En el último ejercicio de la práctica se debía analizar cómo se comparte el espacio de direcciones entre varios hilos y cómo se manejan sus distintas pilas y montones. Después de la creación de cada hilo se genera una nueva región de memoria en el maps que se corresponderá a su pila (marcadas en verde oscuro para el hilo1 y en azul claro para el hilo2). Todas ellas en el mismo espacio de direcciones pero inaccesibles en un principio por otros hilos (salvo que se conozca la dirección de memoria). A continuación, cada hilo ejecuta un *malloc* dando lugar a una nueva región de memoria dedicada al *heap* para cada uno (marcadas en rojo para el hilo1 y

en violeta para el hilo2).

```
> ./Ejercicio6
Hilo principal: 0 80 0 - 5601 sigsus 15:17 pts/1 00:00:00 2
  Dirección de la variable global (b): 0x619c97fd3010
  Dirección de la variable local (a): 0x7ffffdb147b34 00:00:00 2

Hilo 136382351673024: 0 - 6230 do_sel 15:17 pts/1 00:00:00 2
  Dirección de la variable global (b): 0x619c97fd3010
  Dirección del parámetro recibido (param): 0x7ffffdb147b34 00:00:00 2
  Dirección de la variable local (local_var): 0x7c09fd9ffea8
  Dirección de la memoria dinámica (dynamic_var): 0x7c09f8000b70

Hilo 136382341187264: 0 - 4301 hrtime 15:17 pts/1 00:00:00 2
  Dirección de la variable global (b): 0x619c97fd3010
  Dirección del parámetro recibido (param): 0x7ffffdb147b34 00:00:00 2
  Dirección de la variable local (local_var): 0x7c09fcfffea8
  Dirección de la memoria dinámica (dynamic_var): 0x7c09f0000b70
```