Ejecutable y Mapa de memoria de un proceso – EJEMPLO TRANSPARENCIA 8

El ejemplo 1 de la transparencia 8 del tema de procesos se ha modificado para sacar en pantalla las direcciones de la única función que aparece en el programa (`main()`), las direcciones de las variables globales (iniciadas y no iniciadas), las direcciones de las variables locales (iniciadas y no iniciadas), así como el mapa de memoria del programa al ejecutarse (del proceso) en dos instantes: al comienzo y después de la primera reserva dinámica de memoria en ejecución con la función `malloc()` de C. El mapa de memoria se obtiene accediendo al archivo `/proc/PID/maps`[1].

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int imprimir = 1;
char *nomprog;
char mapa[80];

int main(int argc, char *argv[])
{
   int i;
   char *ptr;
   nomprog = argv[0];

   printf("DIRECCIONES DE funciones y variables globales y locales --- ¿LÓGICAS O FÍSICAS?\n\n");
   printf("Dirección de función main: %p \n", main);
   printf("Dirección de variable global iniciada imprimir:   %p\n", &imprimir);
   printf("Dirección de variable global NO iniciada nomprog:   %p\n", &nomprog);
   printf("Dirección de parámetro argv[0]:   %p\n", nomprog);
   printf("Dirección de variable local no iniciada i:   %p\n", &i);
   printf("/************************************************/\n\n\n");

   printf("MAPA DE MEMORIA DEL PROCESO ANTES DE malloc() -OBTENIDA accediendo a /proc/%d/maps \n",
          getpid());
   sprintf(mapa, "cat /proc/%d/maps", getpid());
   fflush(stdout);
   system(mapa);
   printf(" ======================================  \n\n\n");


   printf("número de argumentos = %d\n", argc);
   printf("Nombre del programa: %s\n",
          nomprog);

   for(i = 1; i < argc; i++)
   {
       ptr = malloc(strlen(argv[i]) + 1);
       strcpy(ptr, argv[i]);
       if (imprimir) printf("%s\n", ptr);
       if (i == 1) {
         printf("Dirección DEL VECTOR DINÁMICO --- ¿LÓGICA O FÍSICA?\n\n");
         printf("Dirección de la cadena reservada dinámicamente ptr: %p\n", ptr);
         printf("/************************************************/\n\n\n");


         printf("MAPA DE MEMORIA DEL PROCESO DESPUÉS DE malloc() -accediendo a /proc/%d/maps \n",
                  getpid());
         sprintf(mapa, "cat /proc/%d/maps", getpid());
         fflush(stdout);
         system(mapa);
         printf("  ======================================  \n\n\n");
       }
       free(ptr);
   }

   return 0;
}
```

Tras compilar el programa anterior y ejecutarlo de la siguiente forma:

```
$ ./mapaproc –p prueba.txt
```

---

[1] `/proc` es un "pseudo-sistema de archivos" que se monta automáticamente cuando arranca Linux.

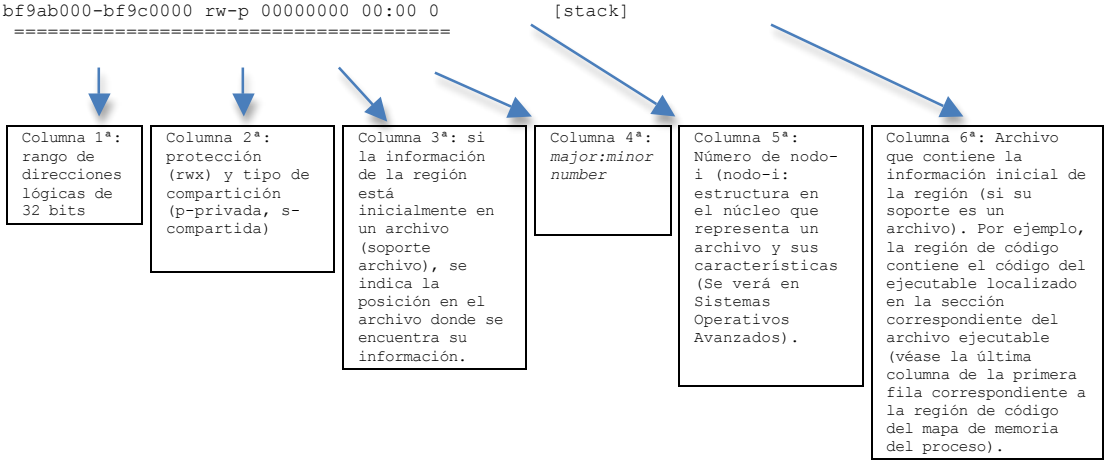
se obtiene en pantalla la información que se muestra abajo[2]. Como puede observarse, se imprimen las direcciones de funciones y variables y los mapas en dos instantes distintos. Se demuestra que el mapa de memoria del proceso es dinámico puesto que va variando a lo largo de la ejecución del programa. En concreto, al ejecutar el programa, se ha mostrado en pantalla el mapa de memoria inicial y el mapa de memoria después de la primera reserva de memoria dinámica (con la función `malloc()`) , y ambos mapas son distintos.

Cada línea del mapa de memoria mostrado en pantalla representa una región del mapa de memoria del proceso. Se ha añadido en el estado inicial del mapa de memoria ciertas explicaciones sobre las propiedades de cada región asociada al mapa. Para ello se han utilizando flechas y anotaciones enmarcadas.

```
DIRECCIONES DE funciones y variables globales y locales --- ¿LÓGICAS O FÍSICAS?

Dirección de función main: 0x80485e4
Dirección de variable global inicializada imprimir:   0x804a038
Dirección de variable global NO inicializada nomprog:   0x804a060
Dirección de parámetro argv[0]:   0xbf9bf637
Dirección de variable local no inicializada i:   0xbf9bdbac
/***********************************************/


MAPA DE MEMORIA DEL PROCESO ANTES DE malloc() -OBTENIDA accediendo a /proc/1845/maps
08048000-08049000 r-xp 00000000 08:01 8151836    /home/labssoo/Escritorio/mapaproc
08049000-0804a000 r--p 00000000 08:01 8151836    /home/labssoo/Escritorio/mapaproc
0804a000-0804b000 rw-p 00001000 08:01 8151836    /home/labssoo/Escritorio/mapaproc
b76e9000-b76ea000 rw-p 00000000 00:00 0
b76ea000-b783d000 r-xp 00000000 08:01 688132     /lib/tls/i686/cmov/libc-2.11.1.so
b783d000-b783e000 ---p 00153000 08:01 688132     /lib/tls/i686/cmov/libc-2.11.1.so
b783e000-b7840000 r--p 00153000 08:01 688132     /lib/tls/i686/cmov/libc-2.11.1.so
b7840000-b7841000 rw-p 00155000 08:01 688132     /lib/tls/i686/cmov/libc-2.11.1.so
b7841000-b7844000 rw-p 00000000 00:00 0
b7854000-b7857000 rw-p 00000000 00:00 0
b7857000-b7858000 r-xp 00000000 00:00 0          [vdso]
b7858000-b7873000 r-xp 00000000 08:01 4292609    /lib/ld-2.11.1.so
b7873000-b7874000 r--p 0001a000 08:01 4292609    /lib/ld-2.11.1.so
b7874000-b7875000 rw-p 0001b000 08:01 4292609    /lib/ld-2.11.1.so
bf9ab000-bf9c0000 rw-p 00000000 00:00 0          [stack]
 ======================================
```



La primera línea del estado del mapa de memoria mostrado anteriormente corresponde a la región de código del mapa de memoria (permisos de lectura y ejecución, pero no escritura –véase la columna 2ª). Se puede observar que a esta región pertenece la dirección de la función `main()` que se imprime en pantalla. La fila 5 (correspondiente a código de una biblioteca dinámica) también tiene esos permisos. La región de la 2ª fila corresponde a datos iniciados de sólo lectura (constantes). La 3ª fila corresponde a la región de variables globales iniciadas, que van a existir durante toda la vida del proceso (permisos de r y w, pero no x). Sus valores iniciales están en la sección correspondiente del ejecutable (ver su última columna). A la región anterior pertenece la dirección de la variable `imprimir` que sale en pantalla. Las filas 5, 6, 7, 8, 12, 13 y 14 corresponden a regiones cuyos archivos soporte son `.so` (códigos y datos de bibliotecas dinámicas requeridas por el proceso). La fila 15 corresponde a la pila del proceso. La pila sirve, entre otras funciones, para almacenar las variables locales, parámetros y direcciones de retorno de llamadas a funciones. Por ello, a esta región pertenece la dirección de la variable local `i` y la dirección de `argv[0]` (direcciones que se muestran en pantalla). La fila 11 corresponde a código del sistema operativo, concretamente a una biblioteca compartida, fuera del ámbito de este documento y del curso.

[2] Los argumentos concretos que se pasan al archivo ejecutable (`./mapaproc`) en la línea de órdenes pueden ser cualesquiera. Su significado en este ejemplo es irrelevante.

```
número de argumentos = 3
Nombre del programa: ./mapaproc
-p
DIRECCIONES DEL VECTOR DINÁMICO --- ¿LÓGICAS O FÍSICAS?

Dirección de la cadena reservada dinámicamente ptr: 0x8f31008
/***********************************************/


MAPA DE MEMORIA DEL PROCESO DESPUÉS DE malloc() -OBTENIDA accediendo a /proc/1845/maps
08048000-08049000 r-xp 00000000 08:01 8151836    /home/labssoo/Escritorio/mapaproc
08049000-0804a000 r--p 00000000 08:01 8151836    /home/labssoo/Escritorio/mapaproc
0804a000-0804b000 rw-p 00001000 08:01 8151836    /home/labssoo/Escritorio/mapaproc
08f31000-08f52000 rw-p 00000000 00:00 0          [heap]
b76e9000-b76ea000 rw-p 00000000 00:00 0
b76ea000-b783d000 r-xp 00000000 08:01 688132     /lib/tls/i686/cmov/libc-2.11.1.so
b783d000-b783e000 ---p 00153000 08:01 688132     /lib/tls/i686/cmov/libc-2.11.1.so
b783e000-b7840000 r--p 00153000 08:01 688132     /lib/tls/i686/cmov/libc-2.11.1.so
b7840000-b7841000 rw-p 00155000 08:01 688132     /lib/tls/i686/cmov/libc-2.11.1.so
b7841000-b7844000 rw-p 00000000 00:00 0
b7854000-b7857000 rw-p 00000000 00:00 0
b7857000-b7858000 r-xp 00000000 00:00 0          [vdso]
b7858000-b7873000 r-xp 00000000 08:01 4292609    /lib/ld-2.11.1.so
b7873000-b7874000 r--p 0001a000 08:01 4292609    /lib/ld-2.11.1.so
b7874000-b7875000 rw-p 0001b000 08:01 4292609    /lib/ld-2.11.1.so
bf9ab000-bf9c0000 rw-p 00000000 00:00 0          [stack]
  ========================================


prueba.txt
```

En el estado del mapa de memoria que se saca por pantalla después de la primera reserva de memoria dinámica realizada dentro del bucle `for`, se puede comprobar que la diferencia es que ha aparecido una nueva región, *heap*, sin soporte, es decir, su información inicial no procede de ningún archivo y con protección de r y w, pero no x (ver segunda columna). Además, es una región privada. A esta región pertenece la dirección vector dinámico `ptr` que, la primera vez que se utiliza para el segundo argumento pasado por la línea de órdenes ("–p"), corresponde a la dirección de comienzo de la cadena `"-p"` copiada con la función `strcpy()`.