

Asignación de Memoria en un Sistema Real

Primera parte.

Reimplementación de pmap.

Lenguaje de programación: Python 3.7

Comandos adicionales para cargar bibliotecas necesarias:

```
$sudo apt install python-pip
```

```
$sudo pip3 install eel
```

```
$sudo pip3 install wheel
```

```
$sudo pip3 install setuptools
```

Para ésta parte se decidió hacer uso de la información que nos provee Linux en `/proc/${PID}/maps`, ya que nos indica las direcciones en uso para cada proceso, así como los permisos (lectura, escritura, ejecutable, privado o compartido) que contiene cada región, e incluso en la mayoría del tiempo nos indica de qué sección se trata (datos, texto, heap, stack).

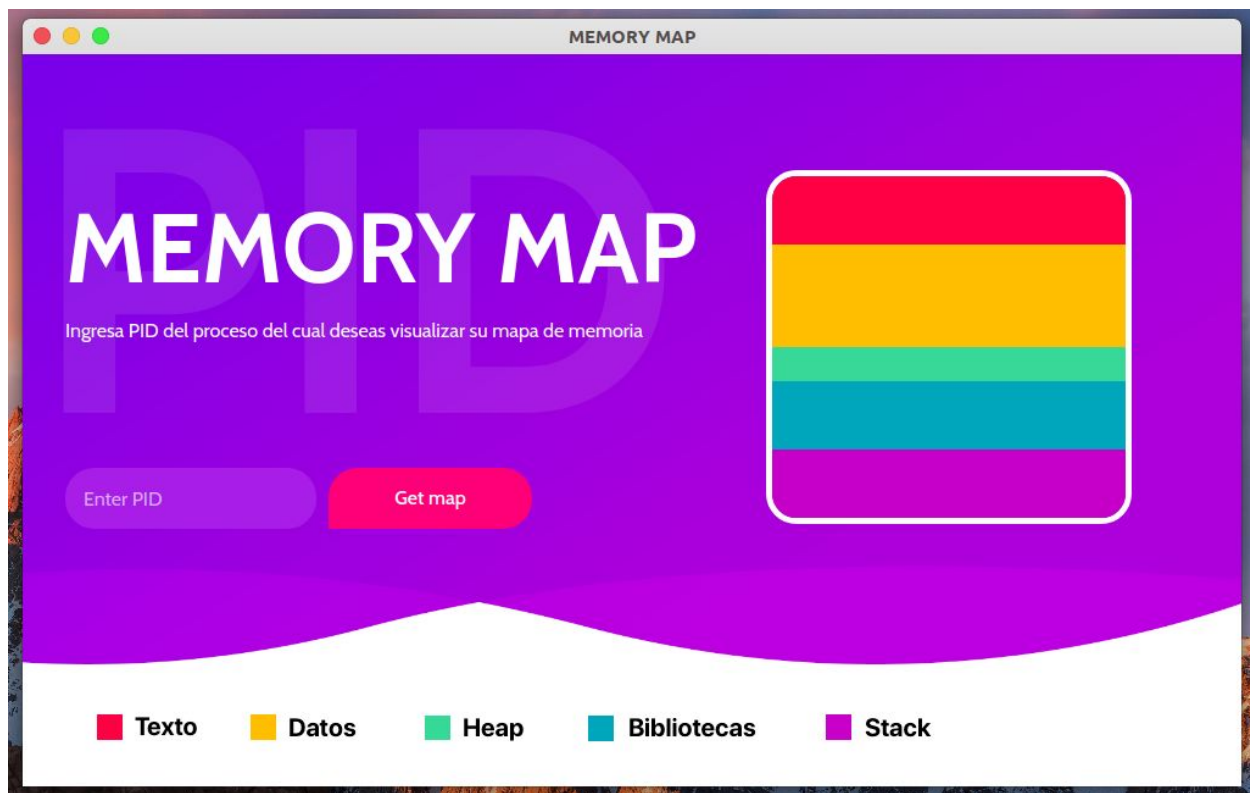
Sin embargo, durante la realización del programa notamos que no siempre se presenta ésta información, particularmente el *heap*, ya que si recordamos es el espacio de memoria que se emplea para la asignación dinámica de memoria durante la ejecución del proceso, esto se debe a que no algunos procesos utilizan técnicas diferentes a las estándares para poder obtener estos espacios de memoria, un ejemplo en donde pudimos notar esto fue con los navegadores web. En éste caso hablamos de Google Chrome que usa `TCMalloc`, la cual es una implementación personalizada de las típicas funciones de `malloc()` en C, ya que es más rápida, y soporta asignación de memoria multi-hilo. De hecho descubrimos que para cada OS, esto cambia.

Por lo tanto hemos tenido que guiarnos, meramente por la teoría haciendo uso de las ubicación de las direcciones, los permisos que tiene, etc.

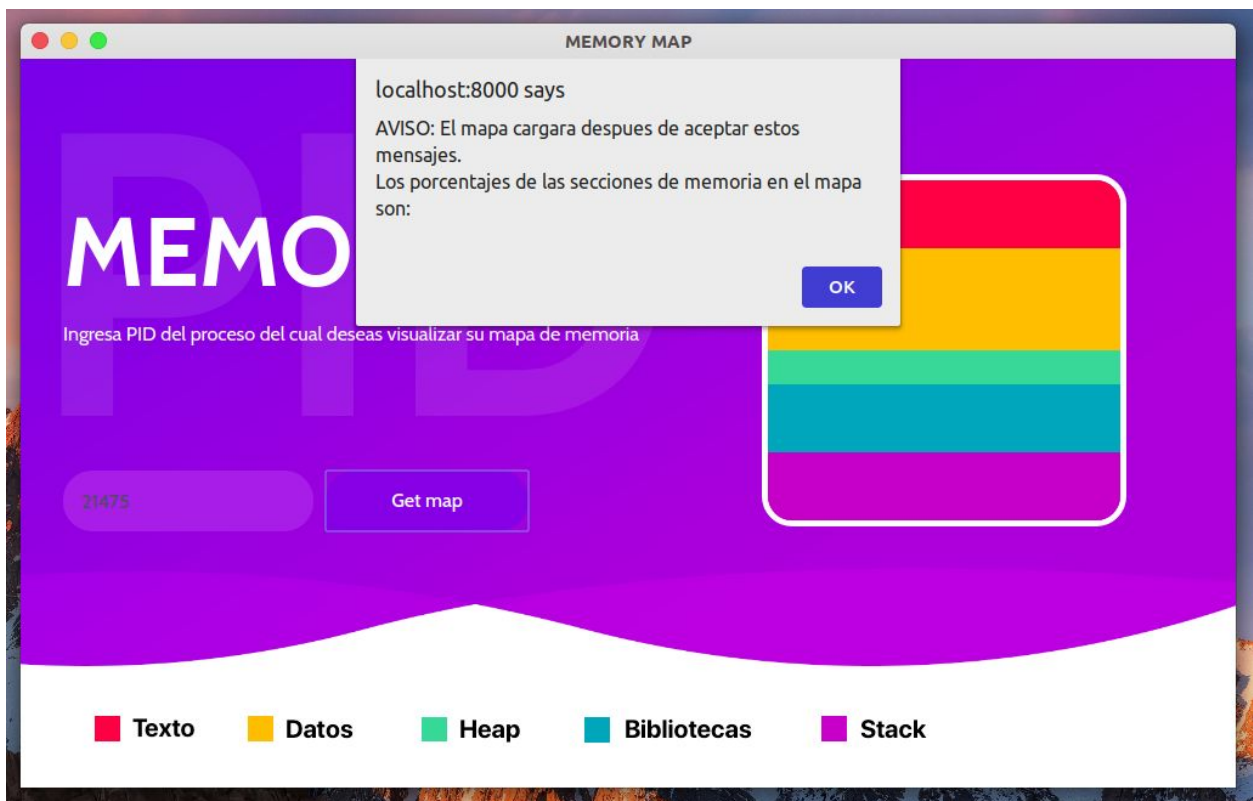
Un ejemplo del mapa de un proceso es el siguiente:

```
carlos@pc-jcah: ~/Desktop/donde_en_la_memoria
File Edit View Search Terminal Help
carlos@pc-jcah:~/Desktop$ python3 proyecto350.py
Dame el numero: 22971
Uso | Tam | Pags | Direcciones | Perm | Ubicacion o uso
---|---|---|---|---|---
Texto | 4 kb | 1 pag | 55f75b57a000-55f75b57b000 | r-xp | /home/carlos/Desktop/donde_en_la_memoria/donde_en_la_memoria
Datos | 4 kb | 1 pag | 55f75b77b000-55f75b77c000 | r--p | /home/carlos/Desktop/donde_en_la_memoria/donde_en_la_memoria
Datos | 4 kb | 1 pag | 55f75b77c000-55f75b77d000 | rw-p | /home/carlos/Desktop/donde_en_la_memoria/donde_en_la_memoria
Heap | 132 kb | 33 pag | 55f75d4f6000-55f75d517000 | rw-p | [heap]
bib->Texto | 1948 kb | 487 pag | 7f9d1d782000-7f9d1d969000 | r-xp | /lib/x86_64-linux-gnu/libc-2.27.so
bib->Datos | 2048 kb | 512 pag | 7f9d1d969000-7f9d1db69000 | --p | /lib/x86_64-linux-gnu/libc-2.27.so
bib->Datos | 16 kb | 4 pag | 7f9d1db69000-7f9d1db6d000 | r--p | /lib/x86_64-linux-gnu/libc-2.27.so
bib->Datos | 8 kb | 2 pag | 7f9d1db6d000-7f9d1db6f000 | rw-p | /lib/x86_64-linux-gnu/libc-2.27.so
Anon | 16 kb | 4 pag | 7f9d1db6f000-7f9d1db73000 | rw-p | [anon]
bib->Texto | 156 kb | 39 pag | 7f9d1db73000-7f9d1db9a000 | r-xp | /lib/x86_64-linux-gnu/ld-2.27.so
Anon | 8 kb | 2 pag | 7f9d1db7c000-7f9d1db7e000 | rw-p | [anon]
bib->Datos | 4 kb | 1 pag | 7f9d1dd9a000-7f9d1dd9b000 | r--p | /lib/x86_64-linux-gnu/ld-2.27.so
bib->Datos | 4 kb | 1 pag | 7f9d1dd9b000-7f9d1dd9c000 | rw-p | /lib/x86_64-linux-gnu/ld-2.27.so
Anon | 4 kb | 1 pag | 7f9d1dd9c000-7f9d1dd9d000 | rw-p | [anon]
Stack | 132 kb | 33 pag | 7fff04654000-7fff04675000 | rw-p | [stack]
???? | 12 kb | 3 pag | 7fff04735000-7fff04738000 | r--p | [vvar]
???? | 4 kb | 1 pag | 7fff04738000-7fff04739000 | r-xp | [vdso]
???? | 4 kb | 1 pag | ffffffff600000-ffffffff601000 | --xp | [vsyscall]
```

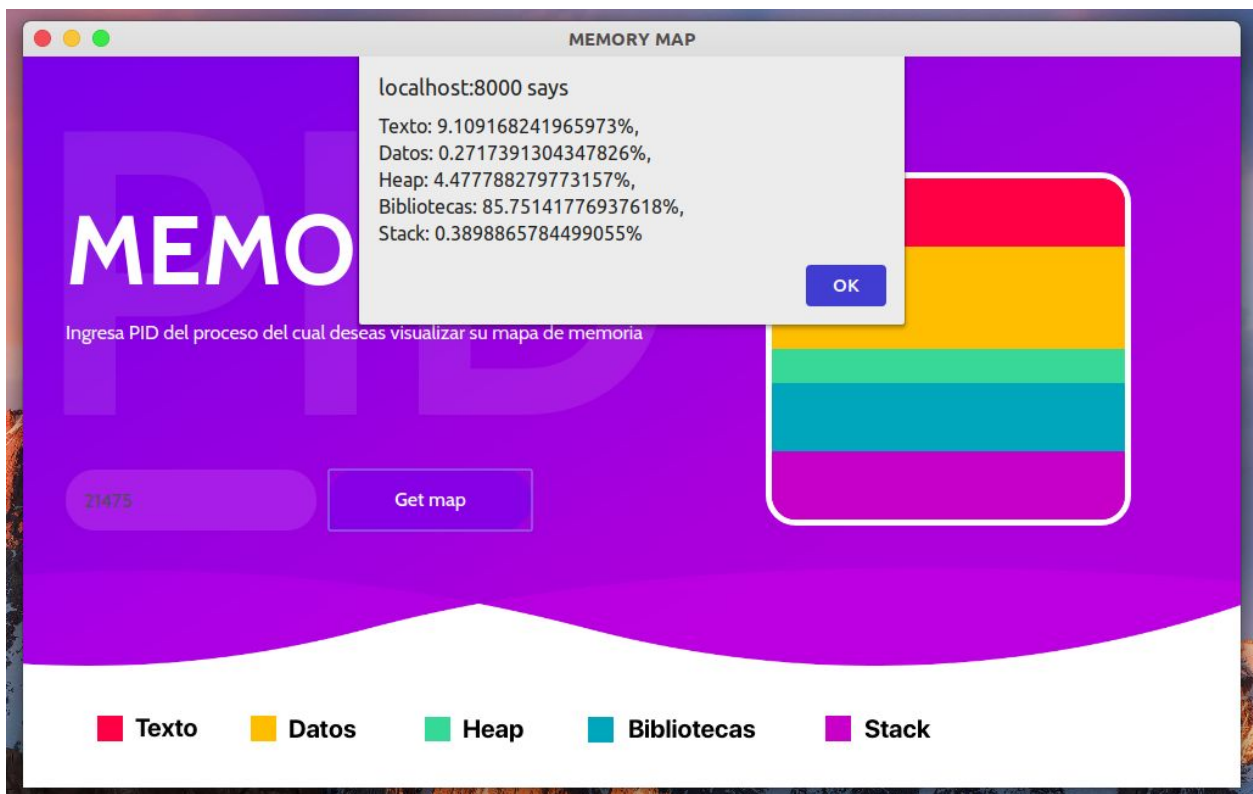
Interfaz grafica:



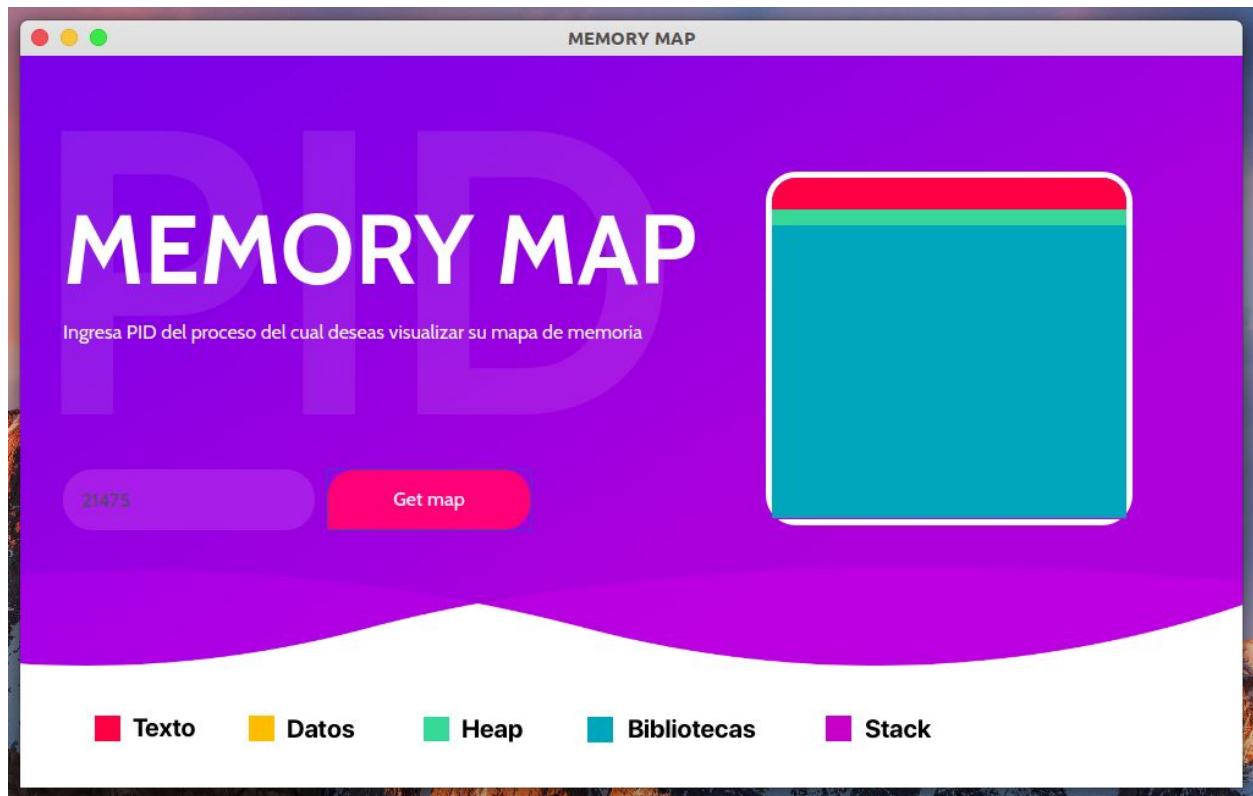
Ingresando un proceso:



Mostrando porcentajes del gráfico que se generará:



Resultado:



Segunda parte.

¿Y realmente se acomoda así?

Acomodo teórico de memoria

Cada proceso tiene sus bloques de memoria mapeados en direcciones de memoria virtual. Esta relación de bloques-direcciones pueden ser consultadas en los sistemas operativos Linux en el archivo `/proc/PID/maps`, reemplazando "PID" con el ID del proceso vivo. Con base en las regiones de memoria ubicadas con ayuda de la reimplementación de `pmap` elaborada para la Primera parte.

Se piensa que la memoria del proceso se constituirá de la siguiente manera: el código ensamblador se ubicará en la región de texto (con permisos de lectura y ejecución), el contenido de las variables globales se encontrarán en la región de datos (con permisos de lectura y escritura), la información que se encuentra en la memoria reservada por las funciones `malloc` y `calloc` se tendría que ubicar en el heap. Y por ultimo, el stack debería contener la pila de llamadas del programa. La información de todas estas variables deberían de ser congruentes con el momento en el que se realizó un volcado de memoria ya que se toma un *Snapshot* al mapa de memoria actual.

El programa realizado para comprobar la distribución de memoria debería de tener la siguiente información en su volcado:

Region	Contenido
Texto	Ensamblador y símbolos
Datos	De las variables: cadena1, tamaño
Heap	De las variables: Cadena3, cadena4, etc
Bibliotecas	-----
Stack	Llamadas a funciones

Comandos y manejo de información

El volcado de memoria de un proceso puede realizarse a partir de la ejecución del comando *gcore* junto con el PID del proceso; el comando generará un core en donde se encontrará el mapa de memoria que se desea analizar, este se deberá abrir con ayuda de GDB el cual es un depurador estándar para el compilador GNU, GDB también nos da la opción de generar el core de un proceso con el comando *generate-core-file*. Además podemos traer el contenido de una región de memoria a un archivo .dat y otras opciones.

Determinación de Datos

Para comenzar, se ejecutó el programa que vamos a analizar, se extrajo su PID y a continuación se ejecutó el programa hecho para la Primera Parte para ubicar en que direcciones de memoria podríamos encontrar la información que buscamos.

Se utilizó el siguiente comando para analizar el proceso con GDB:

```

carlos@pc-jcah: ~/Desktop/donde_en_la_memoria
File Edit View Search Terminal Help
carlos@pc-jcah:~/Desktop/donde_en_la_memoria$ sudo gdb -p 22971
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
Attaching to process 22971
Reading symbols from /home/carlos/Desktop/donde_en_la_memoria/donde_en_la_memoria...
(no debugging symbols found)...done.
Reading symbols from /lib/x86_64-linux-gnu/libc.so.6...Reading symbols from /usr/lib/
debug//lib/x86_64-linux-gnu/libc-2.27.so...done.
done.
Reading symbols from /lib64/ld-linux-x86-64.so.2...Reading symbols from /usr/lib/debu
g//lib/x86_64-linux-gnu/ld-2.27.so...done.
done.
0x00007f9d1d892081 in __GI__libc_read (fd=0, buf=0x55f75d4f6750, nbytes=1024)
at ../sysdeps/unix/sysv/linux/read.c:27
27      ../sysdeps/unix/sysv/linux/read.c: No such file or directory.

```


Y con el siguiente se puede generar el corefile:

```
carlos@pc-jcah: ~/Desktop/donde_en_la_memoria
File Edit View Search Terminal Help
(gdb) generate-core-file
warning: Memory read failed for corefile section, 4096 bytes at 0xffffffff600000.
Saved corefile core.22971
```

Una vez generado el corefile, podemos extraer de él las secciones de memoria que queramos, por ejemplo con las siguientes direcciones de memoria correspondientes a la sección de Datos:

```
carlos@pc-jcah: ~/Desktop/donde_en_la_memoria
File Edit View Search Terminal Help
carlos@pc-jcah:~/Desktop$ python3 proyecto350.py
Dame el numero: 22971

```

Uso	Tam	Pags	Direcciones	Perm	Ubicacion o uso
Texto	4 kb	1 pag	55f75b57a000-55f75b57b000	r-xp	/home/carlos/Desktop/donde_en_la_memoria/donde_en_la_memoria
Datos	4 kb	1 pag	55f75b77b000-55f75b77c000	r--p	/home/carlos/Desktop/donde_en_la_memoria/donde_en_la_memoria
Datos	4 kb	1 pag	55f75b77c000-55f75b77d000	rw-p	/home/carlos/Desktop/donde_en_la_memoria/donde_en_la_memoria
Heap	132 kb	33 pag	55f75d4f6000-55f75d517000	rw-p	[heap]
bib->Texto	1948 kb	487 pag	7f9d1d782000-7f9d1d969000	r-xp	/lib/x86_64-linux-gnu/libc-2.27.so
	2048 kb	512 pag	7f9d1d969000-7f9d1db69000	---	
bib->Datos	16 kb	4 pag	7f9d1db69000-7f9d1db6d000	r--p	/lib/x86_64-linux-gnu/libc-2.27.so
bib->Datos	8 kb	2 pag	7f9d1db6d000-7f9d1db6f000	rw-p	/lib/x86_64-linux-gnu/libc-2.27.so
Anon	16 kb	4 pag	7f9d1db6f000-7f9d1db73000	rw-p	[anon]
bib->Texto	156 kb	39 pag	7f9d1db73000-7f9d1db9a000	r-xp	/lib/x86_64-linux-gnu/ld-2.27.so
Anon	8 kb	2 pag	7f9d1dd7c000-7f9d1dd7e000	rw-p	[anon]
bib->Datos	4 kb	1 pag	7f9d1dd9a000-7f9d1dd9b000	r--p	/lib/x86_64-linux-gnu/ld-2.27.so
bib->Datos	4 kb	1 pag	7f9d1dd9b000-7f9d1dd9c000	rw-p	/lib/x86_64-linux-gnu/ld-2.27.so
Anon	4 kb	1 pag	7f9d1dd9c000-7f9d1dd9d000	rw-p	[anon]
Stack	132 kb	33 pag	7fff04654000-7fff04675000	rw-p	[stack]
????	12 kb	3 pag	7fff04735000-7fff04738000	r--p	[vvar]
????	4 kb	1 pag	7fff04738000-7fff04739000	r-xp	[vdso]
????	4 kb	1 pag	ffffffff600000-ffffffff601000	--xp	[vsyscall]

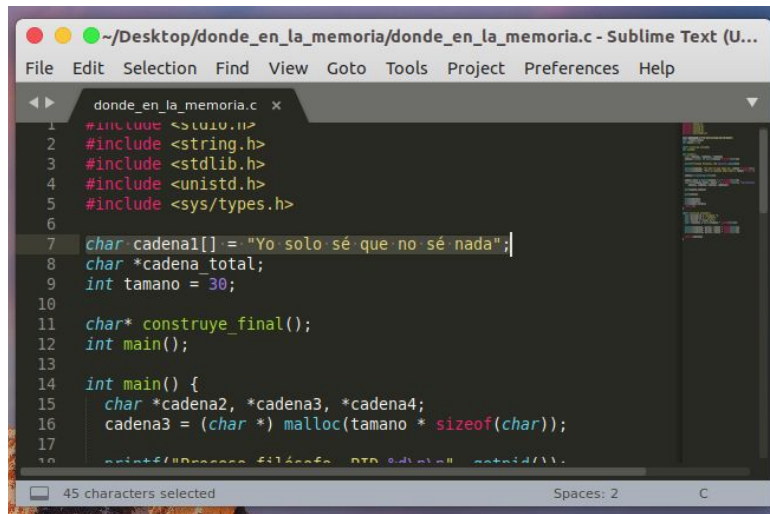
Usando el comando siguiente:

```
carlos@pc-jcah: ~/Desktop/donde_en_la_memoria
File Edit View Search Terminal Help
(gdb) dump memory prueba.dat 0x55f75b77c000 0x55f75b77d000
(gdb)
(gdb)
(gdb) _
```

Obtuvimos:

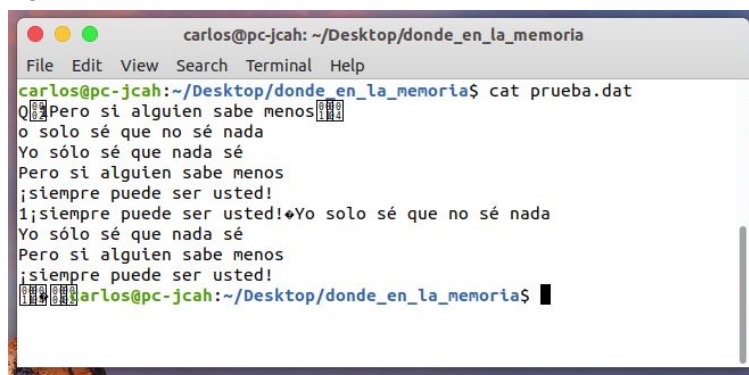
```
carlos@pc-jcah: ~/Desktop/donde_en_la_memoria
File Edit View Search Terminal Help
carlos@pc-jcah:~/Desktop/donde_en_la_memoria$ cat prueba.dat
w[UYo solo sé que no sé nada]f0]Ucarlos@pc-jcah:~/Desktop
/donde_en_la_memoria$
```

Con lo que se puede ver que el contenido de esta sección de memoria, en efecto corresponde con lo que la variable cadena1 debería tener al ser una variable global.



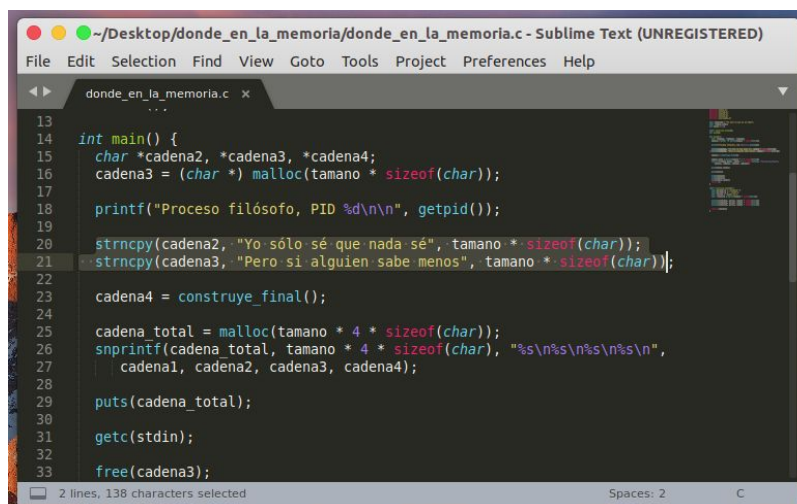
```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6
7 char cadena1[] = "Yo solo sé que no sé nada";
8 char *cadena_total;
9 int tamano = 30;
10
11 char* construye_final();
12 int main();
13
14 int main() {
15     char *cadena2, *cadena3, *cadena4;
16     cadena3 = (char *) malloc(tamano * sizeof(char));
17     printf("Proceso filósofo, PID %d\n", getpid());
```

Volviendo a ejecutar el comando pero con las direcciones de la sección del heap, generó el siguiente resultado:



```
carlos@pc-jcah: ~/Desktop/donde_en_la_memoria
File Edit View Search Terminal Help
carlos@pc-jcah:~/Desktop/donde_en_la_memoria$ cat prueba.dat
Yo solo sé que no sé nada
Pero si alguien sabe menos
o solo sé que no sé nada
Yo sólo sé que nada sé
Pero si alguien sabe menos
¡siempre puede ser usted!
¡siempre puede ser usted!Yo solo sé que no sé nada
Yo sólo sé que nada sé
Pero si alguien sabe menos
¡siempre puede ser usted!
carlos@pc-jcah:~/Desktop/donde_en_la_memoria$
```

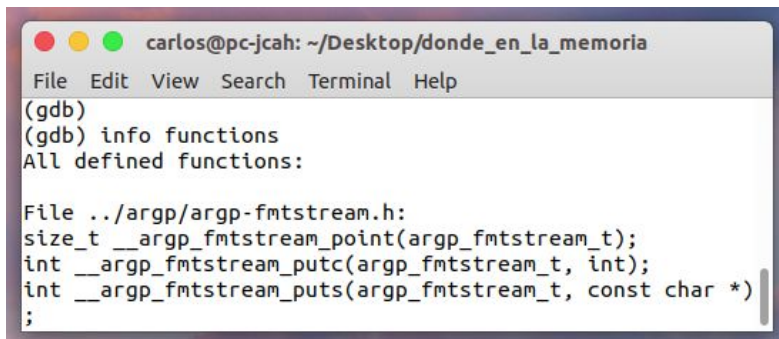
El cual coincide con lo esperado:



```
13
14 int main() {
15     char *cadena2, *cadena3, *cadena4;
16     cadena3 = (char *) malloc(tamano * sizeof(char));
17
18     printf("Proceso filósofo, PID %d\n", getpid());
19
20     strncpy(cadena2, "Yo sólo sé que nada sé", tamano * sizeof(char));
21     strncpy(cadena3, "Pero si alguien sabe menos", tamano * sizeof(char));
22
23     cadena4 = construye_final();
24
25     cadena_total = malloc(tamano * 4 * sizeof(char));
26     snprintf(cadena_total, tamano * 4 * sizeof(char), "%s\n%s\n%s\n%s\n",
27             cadena1, cadena2, cadena3, cadena4);
28
29     puts(cadena_total);
30
31     getc(stdin);
32
33     free(cadena3);
```

Determinación de Símbolos

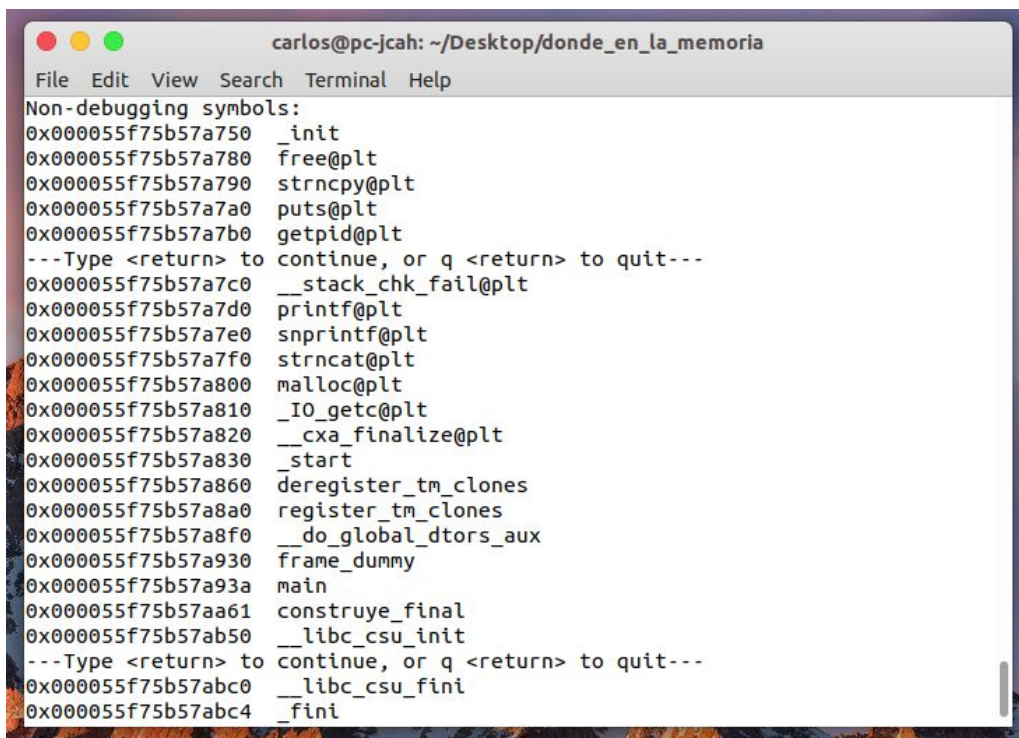
Para determinar en donde se ubican los símbolos, se decidió emplear el siguiente comando en gdb



```
carlos@pc-jcah: ~/Desktop/donde_en_la_memoria
File Edit View Search Terminal Help
(gdb)
(gdb) info functions
All defined functions:

File ../argp/argp-fmtstream.h:
size_t __argp_fmtstream_point(argp_fmtstream_t);
int __argp_fmtstream_putc(argp_fmtstream_t, int);
int __argp_fmtstream_puts(argp_fmtstream_t, const char *)
;
```

En la ultima parte de la información proporcionada por el comando se puede apreciar en que parte de la memoria tenemos los símbolos del programa:



```
carlos@pc-jcah: ~/Desktop/donde_en_la_memoria
File Edit View Search Terminal Help
Non-debugging symbols:
0x000055f75b57a750  _init
0x000055f75b57a780  free@plt
0x000055f75b57a790  strncpy@plt
0x000055f75b57a7a0  puts@plt
0x000055f75b57a7b0  getpid@plt
---Type <return> to continue, or q <return> to quit---
0x000055f75b57a7c0  __stack_chk_fail@plt
0x000055f75b57a7d0  printf@plt
0x000055f75b57a7e0  snprintf@plt
0x000055f75b57a7f0  strncat@plt
0x000055f75b57a800  malloc@plt
0x000055f75b57a810  _IO_getc@plt
0x000055f75b57a820  __cxa_finalize@plt
0x000055f75b57a830  _start
0x000055f75b57a860  deregister_tm_clones
0x000055f75b57a8a0  register_tm_clones
0x000055f75b57a8f0  __do_global_ctors_aux
0x000055f75b57a930  frame_dummy
0x000055f75b57a93a  main
0x000055f75b57aa61  construye_final
0x000055f75b57ab50  __libc_csu_init
---Type <return> to continue, or q <return> to quit---
0x000055f75b57abc0  __libc_csu_fini
0x000055f75b57abc4  _fini
```


Como podemos observar, estos símbolos se encuentran en la región de texto:

```

carlos@pc-jcah: ~/Desktop
File Edit View Search Terminal Help
carlos@pc-jcah:~/Desktop$ python3 proyecto350.py
Dame el numero: 22971

```

Uso	Tam	Pags	Direcciones	Perm	Ubicacion o uso
Texto	4 kb	1 pag	55f75b57a000-55f75b57b000	r-xp	/home/carlos/Desktop/donde_en_la_memoria/donde_en_la_memoria
Datos	4 kb	1 pag	55f75b77b000-55f75b77c000	r--p	/home/carlos/Desktop/donde_en_la_memoria/donde_en_la_memoria
Datos	4 kb	1 pag	55f75b77c000-55f75b77d000	rw-p	/home/carlos/Desktop/donde_en_la_memoria/donde_en_la_memoria
Heap	132 kb	33 pag	55f75d4f6000-55f75d517000	rw-p	[heap]
lib->Texto	1948 kb	487 pag	7f9d1d782000-7f9d1d969000	r-xp	/lib/x86_64-linux-gnu/libc-2.27.so
	2048 kb	512 pag	7f9d1d969000-7f9d1db69000	--p	/lib/x86_64-linux-gnu/libc-2.27.so
lib->Datos	16 kb	4 pag	7f9d1db69000-7f9d1db6d000	r--p	/lib/x86_64-linux-gnu/libc-2.27.so
lib->Datos	8 kb	2 pag	7f9d1db6d000-7f9d1db6f000	rw-p	/lib/x86_64-linux-gnu/libc-2.27.so
Anon	16 kb	4 pag	7f9d1db6f000-7f9d1db73000	rw-p	[anon]
lib->Texto	156 kb	39 pag	7f9d1db73000-7f9d1db9a000	r-xp	/lib/x86_64-linux-gnu/ld-2.27.so
Anon	8 kb	2 pag	7f9d1dd7c000-7f9d1dd7e000	rw-p	[anon]
lib->Datos	4 kb	1 pag	7f9d1dd9a000-7f9d1dd9b000	r--p	/lib/x86_64-linux-gnu/ld-2.27.so
lib->Datos	4 kb	1 pag	7f9d1dd9b000-7f9d1dd9c000	rw-p	/lib/x86_64-linux-gnu/ld-2.27.so
Anon	4 kb	1 pag	7f9d1dd9c000-7f9d1dd9d000	rw-p	[anon]
Stack	132 kb	33 pag	7fff04654000-7fff04675000	rw-p	[stack]
???	12 kb	3 pag	7fff04735000-7fff04738000	r--p	[vvar]
???	4 kb	1 pag	7fff04738000-7fff04739000	r-xp	[vdso]
???	4 kb	1 pag	ffffffffff600000-ffffffffff601000	--xp	[vsyscall]

Como se puede visualizar, algunos símbolos de las funciones se encuentran con un @plt que indica que sus direcciones no se conocen en el momento de ligado, dejando el problema a resolver en tiempo de ejecución. Se debe recalcar que los símbolos que no se conocen, son parte de las bibliotecas del lenguaje C, ya que no se encuentran en el ensamblador del programa.

De la misma forma, con info variables podemos extraer lo siguiente:

```

carlos@pc-jcah: ~/Desktop/donde_en_la_memoria
File Edit View Search Terminal Help
Non-debugging symbols:
0x000055f75b57abd0 __IO_stdin_used
0x000055f75b57ac3c __GNU_EH_FRAME_HDR
0x000055f75b57ada4 __FRAME_END__
0x000055f75b77bd70 __frame_dummy_init_array_entry
0x000055f75b77bd70 __init_array_start
0x000055f75b77bd78 __do_global_dtors_aux_fini_array_entry
0x000055f75b77bd78 __init_array_end
0x000055f75b77bd80 __DYNAMIC
0x000055f75b77bf70 __GLOBAL_OFFSET_TABLE__
0x000055f75b77c000 __data_start
0x000055f75b77c000 data_start
0x000055f75b77c008 __dso_handle
0x000055f75b77c010 cadena1
0x000055f75b77c02c tamano
0x000055f75b77c030 __TMC_END__
0x000055f75b77c030 __bss_start
0x000055f75b77c030 _edata
0x000055f75b77c030 stdin
---Type <return> to continue, or q <return> to quit---
0x000055f75b77c030 stdin@@GLIBC_2.2.5
0x000055f75b77c038 completed
0x000055f75b77c040 cadena_total
0x000055f75b77c048 _end
0x00007f9d1d91ecc7 inmask
0x00007f9d1d91eda0 slashdot
0x00007f9d1d91edc0 codeset_idx
0x00007f9d1d92b9a0 mask
0x00007f9d1d92bb60 nbits
0x00007f9d1d92bba0 nbits

```

```

carlos@pc-jcah: ~/Desktop/donde_en_la_memoria
File Edit View Search Terminal Help
---Type <return> to continue, or q <return> to quit---
0x000055f75b57a9b4 <+122>: mov    0x201672(%rip),%eax    # 0x55f75b77c02c <tamano>
0x000055f75b57a9ba <+128>: shl    $0x2,%eax
0x000055f75b57a9bd <+131>: cltq
0x000055f75b57a9bf <+133>: mov    %rax,%rdi
0x000055f75b57a9c2 <+136>: callq 0x55f75b57a800 <malloc@plt>
0x000055f75b57a9c7 <+141>: mov    %rax,0x201672(%rip)    # 0x55f75b77c040 <cadena_total>
0x000055f75b57a9ce <+148>: mov    0x201658(%rip),%eax    # 0x55f75b77c02c <tamano>
0x000055f75b57a9d4 <+154>: shl    $0x2,%eax
0x000055f75b57a9d7 <+157>: movslq %eax,%rsi
0x000055f75b57a9da <+160>: mov    0x20165f(%rip),%rax    # 0x55f75b77c040 <cadena_total>
0x000055f75b57a9e1 <+167>: mov    -0x18(%rbp),%rcx
0x000055f75b57a9e5 <+171>: mov    -0x10(%rbp),%rdx
0x000055f75b57a9e9 <+175>: sub    $0x8,%rsp
0x000055f75b57a9ed <+179>: pushq  -0x8(%rbp)
0x000055f75b57a9f0 <+182>: mov    %rcx,%r9
0x000055f75b57a9f3 <+185>: mov    %rdx,%r8
0x000055f75b57a9f6 <+188>: lea    0x201613(%rip),%rcx    # 0x55f75b77c010 <cadena1>
0x000055f75b57a9fd <+195>: lea    0x221(%rip),%rdx    # 0x55f75b57ac25
0x000055f75b57aa04 <+202>: mov    %rax,%rdi
0x000055f75b57aa07 <+205>: mov    $0x0,%eax
0x000055f75b57aa0c <+210>: callq  0x55f75b57a7e0 <snprintf@plt>
0x000055f75b57aa11 <+215>: add    $0x10,%rsp
0x000055f75b57aa15 <+219>: mov    0x201624(%rip),%rax    # 0x55f75b77c040 <cadena_total>
0x000055f75b57aa1c <+226>: mov    %rax,%rdi
0x000055f75b57aa1f <+229>: callq  0x55f75b57a7a0 <puts@plt>
0x000055f75b57aa24 <+234>: mov    0x201605(%rip),%rax    # 0x55f75b77c030 <stdin@GLIBC_2.2.5>
0x000055f75b57aa2b <+241>: mov    %rax,%rdi
0x000055f75b57aa2e <+244>: callq  0x55f75b57a810 <_IO_getc@plt>
---Type <return> to continue, or q <return> to quit---

```

vemos a comprobar con ayuda del ensamblador que los símbolos de las variables se encuentran en la región de texto y tienen referencia a la región de datos.

Hemos determinado que efectivamente los datos se encuentran en la región que esperábamos. Aunque están organizados en diferente manera a como se pensaba en un principio, se llegó a la conclusión de que lo visto en clase se cumple en el sistema operativo Linux.