Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

Part 3: Memory management

Seminar 7

Memory map of a Linux process

fSO

DISCA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Goals and bibliography

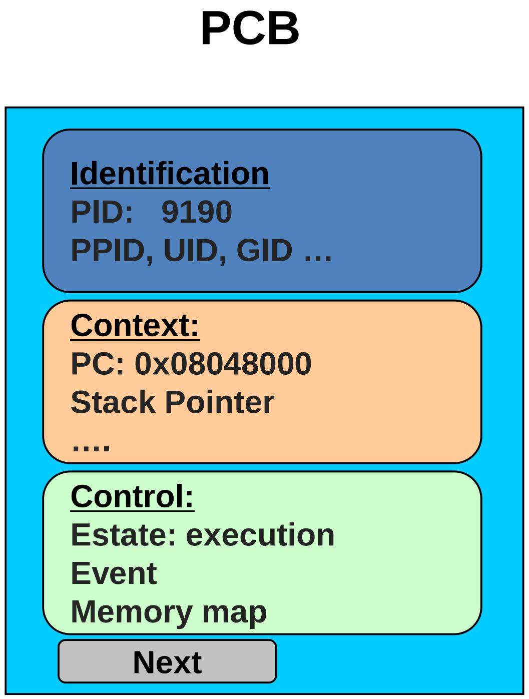
- Goals
  - To understand the **process memory map concept**
  - To know the features of **process memory map in Linux**
  - To know the technique used to **map files into memory**
  - To be aware about the advantages and disadvantages of using **static and dynamic libraries**

- Bibliography
  - Carretero, chapter 5

Content

Introduction

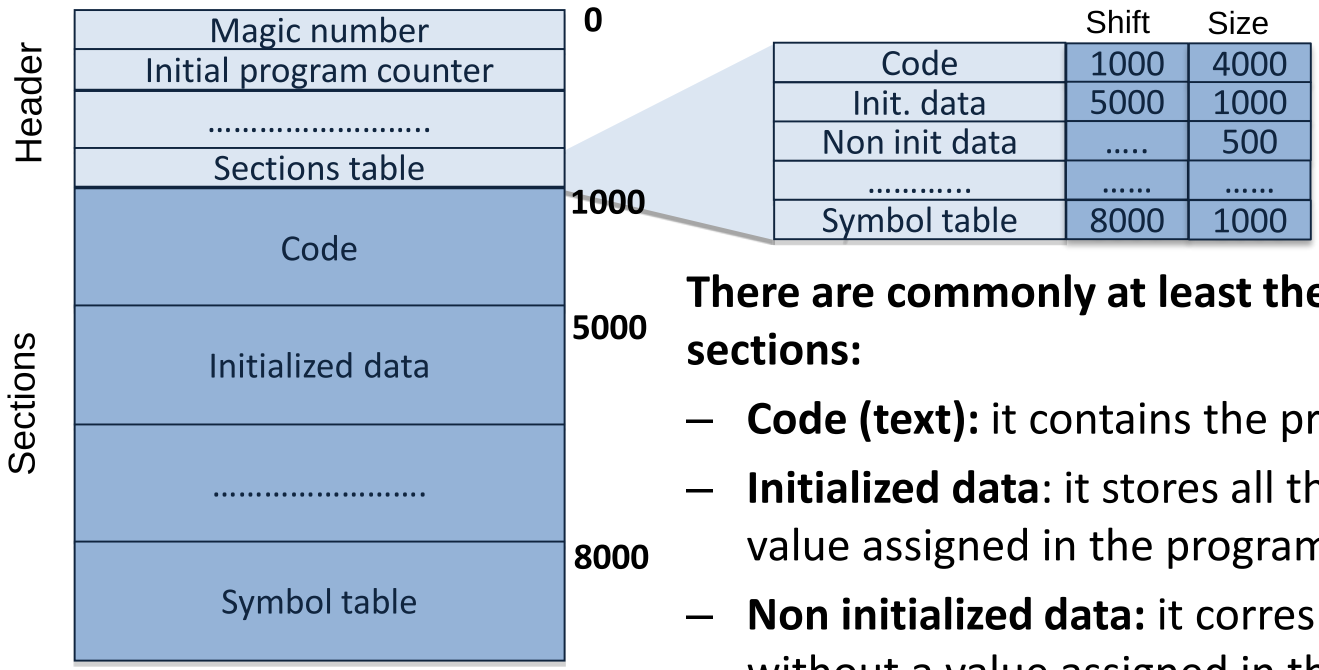
- **Process memory map**
  - The OS manages the memory map of every process during its lifetime
  - Memory map is a process **attribute** -> it is included in its PCB
  - It contains information about a process memory regions:
    - Code, data, stack, etc



- The **initial memory map** of a process is strongly linked to its executable file
- Nowadays OSs offer a **dynamic memory model** that gives support to process memory regions allocation:
  - **New regions could be created** to allocate dynamic process entities: stack, mapped files, dynamic memory, etc.
  - **Unused regions could be removed**

Introduction

- **Simplified format of an executable file**
  - After compiling and linking an executable file is generated that contains the program machine code ready for execution
  - An executable file is structured into a header and a sections set
    - **Header**: it contains control information that allows reading into the remaining executable file content
    - **Sections:** Executable file content is organized in sections



	Shift	Size
Code	1000	4000
Init. data	5000	1000
Non init data	…..	500
…………	……	……
Symbol table	8000	1000

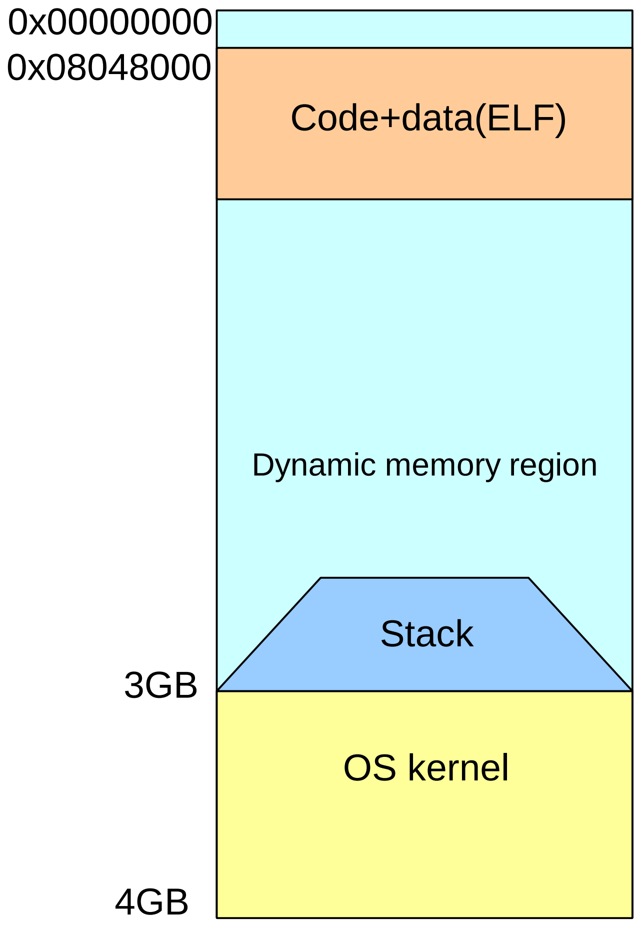
There are commonly at least the following three sections:

- **Code (text):** it contains the program code
- **Initialized data**: it stores all the variables with a value assigned in the program text
- **Non initialized data:** it corresponds to variables without a value assigned in the program text

Content

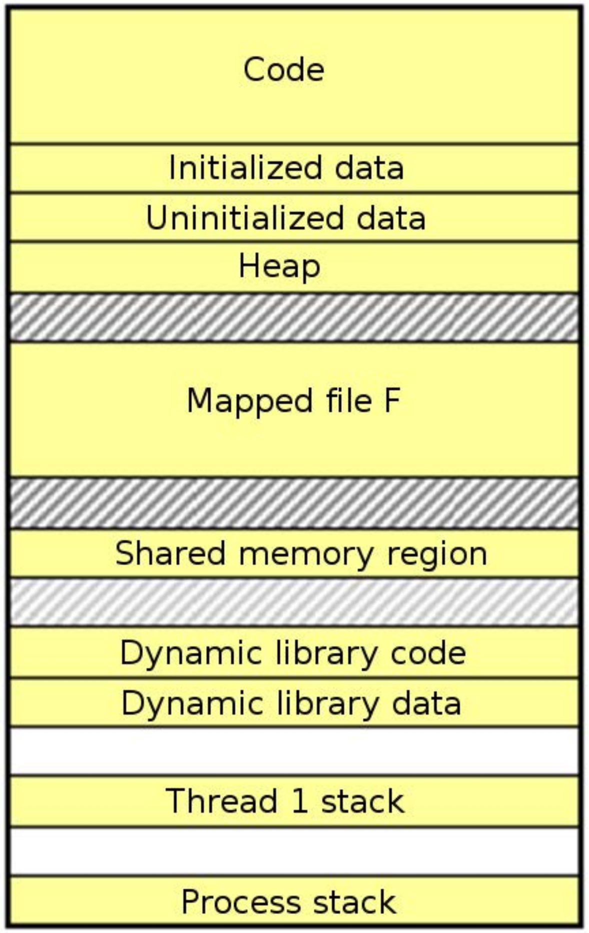
Memory map of a Linux process

- Linux 2.6.x versions on 32 bits architecture:
  - Logical space is 4GB.
  - 1st GByte: Code + initialized data begin at address 0x08048000 (ELF format).
  - 3rd GByte:The stack starts in the 3rd GB upper side and it grows downwards.
  - 4th GByte: The upper GB is reserved to the OS
  - The remaining space is available for data (uninitialized + heap) and dynamic linked libraries



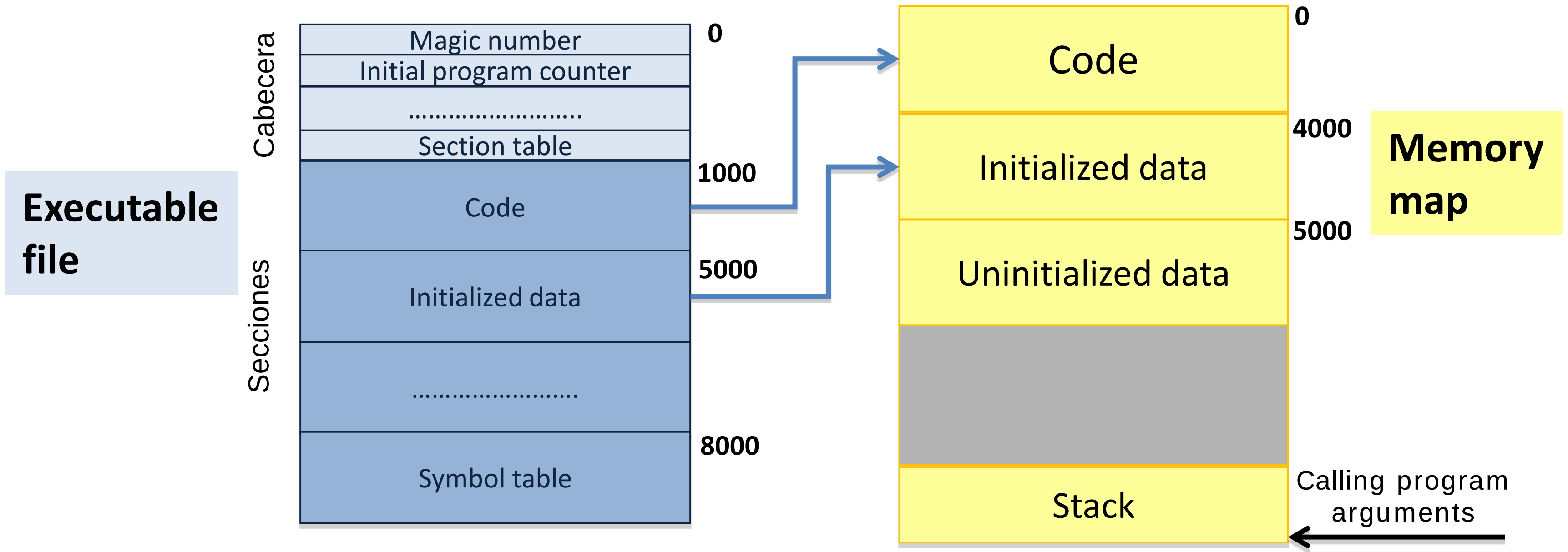
Memory map of a Linux process

- **Memory maps are region made**
  - A region has a certain type of information associated
    - Code, initialized data, uninitialized data, mapped files and dynamic linked libraries
  - A region is a **contiguous memory chunk** featured by the address inside the process memory map where it begins and its size
  - Region features:
    - **Support:** where the region information is stored
      - File support: region information is stored in a file
      - Without support: region without initial content
    - **Share type**:
      - Private (p) : content is only accessible by the owner process
      - Shared: content is accessible by several processes
    - **Protection**: region access type allowed
      - Read, write and/or execution
    - **Size**: fixed or variable



Memory map of a Linux process

- Program execution starts building a process memory map from the executable file
- **Every executable file section** becomes **an initial map region**
  - **Code (text):** shared region, read and execution, fixed size, executable file support
  - **Initialized data**: private region (every process needs a private copy), read and write, fixed size, executable file support
  - **Uninitialized data**: private region, read and write, fixed size, without support (some compilers/languages initialize it to zero)
  - **Stack:** private region, read and write, variable size, without support. It grows towards lower addresses. When execution starts it only contains calling program arguments



Memory map of a Linux process

- Process memory map is **dynamic** -> along process lifetime some regions can be created like:
  - **Heap**
    - Dynamic memory support (i.e. pointers)
    - Private, read and write, variable size, without support (initialized to zero)
    - It grows towards upper addresses
  - **Mapped files**
    - When a file is mapped into memory a new region is created
    - Variable size, file support
    - Protection and sharing specified in the mapping
  - **Shared memory**
    - Region that supports interprocess communication
    - Shared, variable size, without support (initialized to zero)
    - Protection specified in the program
  - **Thread stacks**
    - Every thread stack has its own region
    - Same features as process stack

Memory map of a Linux process

- **Two visualization methods of a process memory map:**

– Viewing the process maps file:

$cat /proc/PID/maps

» Logic address range
» Permissions
» Shift from the beginning of the executable file
» Device
» Node-i
» Mapped file name

– Executing the **shell command**
$pmap PID

» Logic base address
» Size
» Permissions
» Mapped file name

Nota: Replacing PID by variable **$$** we refer to the **process** in **execution** like:

```
$ pmap $$
$ cat /proc/$$/maps
```

Memory map of a Linux process

32 bit architecture

```
pblanes@pblanes-desktop:~$ pmap $$
1608:   /bin/bash
00140000     32K r-x--  /lib/tls/i686/cmov/libnss_nis-2.11.1.so
00148000      4K r----  /lib/tls/i686/cmov/libnss_nis-2.11.1.so
00149000      4K rw---  /lib/tls/i686/cmov/libnss_nis-2.11.1.so
00266000      8K r-x--  /lib/tls/i686/cmov/libdl-2.11.1.so
00268000      4K r----  /lib/tls/i686/cmov/libdl-2.11.1.so
00269000      4K rw---  /lib/tls/i686/cmov/libdl-2.11.1.so
00319000   1356K r-x--  /lib/tls/i686/cmov/libc-2.11.1.so
0046c000      4K -----  /lib/tls/i686/cmov/libc-2.11.1.so
0046d000      8K r----  /lib/tls/i686/cmov/libc-2.11.1.so
0046f000      4K rw---  /lib/tls/i686/cmov/libc-2.11.1.so
00470000     12K rw---    [ anon ]
005b0000      4K r-x--    [ anon ]
007f7000    208K r-x--  /lib/libncurses.so.5.7
0082b000      4K -----  /lib/libncurses.so.5.7
0082c000      8K r----  /lib/libncurses.so.5.7
0082e000      4K rw---  /lib/libncurses.so.5.7
00cbc000    108K r-x--  /lib/ld-2.11.1.so
00cd7000      4K r----  /lib/ld-2.11.1.so
00cd8000      4K rw---  /lib/ld-2.11.1.so
00cf7000     24K r-x--  /lib/tls/i686/cmov/libnss_compat-2.11.1.so
00cfd000      4K r----  /lib/tls/i686/cmov/libnss_compat-2.11.1.so
00cfe000      4K rw---  /lib/tls/i686/cmov/libnss_compat-2.11.1.so
00f56000     40K r-x--  /lib/tls/i686/cmov/libnss_files-2.11.1.so
00f60000      4K r----  /lib/tls/i686/cmov/libnss_files-2.11.1.so
00f61000      4K rw---  /lib/tls/i686/cmov/libnss_files-2.11.1.so
00fc1000     76K r-x--  /lib/tls/i686/cmov/libnsl-2.11.1.so
00fd4000      4K r----  /lib/tls/i686/cmov/libnsl-2.11.1.so
00fd5000      4K rw---  /lib/tls/i686/cmov/libnsl-2.11.1.so
00fd6000      8K rw---    [ anon ]
08048000    780K r-x--  /bin/bash
0810b000      4K r----  /bin/bash
0810c000     20K rw---  /bin/bash
08111000     20K rw---    [ anon ]
0876e000   1368K rw---    [ anon ]
b75ef000    156K r----  /usr/share/locale-langpack/es/LC_MESSAGES/bash.mo
b7616000    252K r----  /usr/lib/locale/es_ES.utf8/LC_CTYPE
b7655000      4K r----  /usr/lib/locale/es_ES.utf8/LC_NUMERIC
b7656000      4K r----  /usr/lib/locale/es_ES.utf8/LC_TIME
b7657000   1144K r----  /usr/lib/locale/es_ES.utf8/LC_COLLATE
b7775000      8K rw---    [ anon ]
b7777000      4K r----  /usr/lib/locale/es_ES.utf8/LC_MONETARY
b7778000      4K r----  /usr/lib/locale/es_ES.utf8/LC_MESSAGES/SYS_LC_MESSAGES
b7779000      4K r----  /usr/lib/locale/es_ES.utf8/LC_PAPER
b777a000      4K r----  /usr/lib/locale/es_ES.utf8/LC_NAME
b777b000      4K r----  /usr/lib/locale/es_ES.utf8/LC_ADDRESS
b777c000      4K r----  /usr/lib/locale/es_ES.utf8/LC_TELEPHONE
b777d000      4K r----  /usr/lib/locale/es_ES.utf8/LC_MEASUREMENT
b777e000     28K r--s-  /usr/lib/gconv/gconv-modules.cache
b7785000      4K r----  /usr/lib/locale/es_ES.utf8/LC_IDENTIFICATION
b7786000      8K rw---    [ anon ]
bf95a000     84K rw---    [ stack ]
 total     5868K
```

Memory map of a Linux process

64 bit architecture

```
pblanes@shell-sisop:~$ pmap -d $$
29916:   -bash
Address                  Kbytes Mode  Offset           Device    Mapping
0000000000400000     760 r-x-- 0000000000000000 008:00002 bash
00000000006bd000      40 rw--- 00000000000bd000 008:00002 bash
00000000006c7000    2616 rw--- 00000000006c7000 000:00000   [ anon ]
00007fa6e6728000      40 r-x-- 0000000000000000 008:00002 libnss_files-2.7.so
00007fa6e6732000    2048 ----- 000000000000a000 008:00002 libnss_files-2.7.so
00007fa6e6932000       8 rw--- 000000000000a000 008:00002 libnss_files-2.7.so
00007fa6e6934000      40 r-x-- 0000000000000000 008:00002 libnss_nis-2.7.so
00007fa6e693e000    2044 ----- 000000000000a000 008:00002 libnss_nis-2.7.so
00007fa6e6b3d000       8 rw--- 0000000000009000 008:00002 libnss_nis-2.7.so
00007fa6e6b3f000      88 r-x-- 0000000000000000 008:00002 libnsl-2.7.so
00007fa6e6b55000    2044 ----- 0000000000016000 008:00002 libnsl-2.7.so
00007fa6e6d54000       8 rw--- 0000000000015000 008:00002 libnsl-2.7.so
00007fa6e6d56000       8 rw--- 00007fa6e6d56000 000:00000   [ anon ]
00007fa6e6d58000      32 r-x-- 0000000000000000 008:00002 libnss_compat-2.7.so
00007fa6e6d60000    2044 ----- 0000000000008000 008:00002 libnss_compat-2.7.so
00007fa6e6f5f000       8 rw--- 0000000000007000 008:00002 libnss_compat-2.7.so
00007fa6e6f61000    1376 r-x-- 0000000000000000 008:00002 libc-2.7.so
00007fa6e70b9000    2048 ----- 0000000000158000 008:00002 libc-2.7.so
00007fa6e72b9000      12 r---- 0000000000158000 008:00002 libc-2.7.so
00007fa6e72bc000       8 rw--- 000000000015b000 008:00002 libc-2.7.so
00007fa6e72be000      20 rw--- 00007fa6e72be000 000:00000   [ anon ]
00007fa6e72c3000       8 r-x-- 0000000000000000 008:00002 libdl-2.7.so
00007fa6e72c5000    2048 ----- 0000000000002000 008:00002 libdl-2.7.so
00007fa6e74c5000       8 rw--- 0000000000002000 008:00002 libdl-2.7.so
00007fa6e74c7000     220 r-x-- 0000000000000000 008:00002 libncurses.so.5.6
00007fa6e74fe000    2044 ----- 0000000000037000 008:00002 libncurses.so.5.6
00007fa6e76fd000      20 rw--- 0000000000036000 008:00002 libncurses.so.5.6
00007fa6e7702000     116 r-x-- 0000000000000000 008:00002 ld-2.7.so
00007fa6e77da000      60 r---- 0000000000000000 008:00002 bash.mo
00007fa6e77e9000     252 r---- 0000000000000000 008:00002 LC_CTYPE
00007fa6e7828000     900 r---- 0000000000000000 008:00002 LC_COLLATE
00007fa6e7909000       8 rw--- 00007fa6e7909000 000:00000   [ anon ]
00007fa6e790b000       4 r---- 0000000000000000 008:00002 LC_NUMERIC
00007fa6e790c000       4 r---- 0000000000000000 008:00002 LC_TIME
00007fa6e790d000       4 r---- 0000000000000000 008:00002 LC_MONETARY
00007fa6e790e000       4 r---- 0000000000000000 008:00002 SYS_LC_MESSAGES
00007fa6e790f000       4 r---- 0000000000000000 008:00002 LC_PAPER
00007fa6e7910000       4 r---- 0000000000000000 008:00002 LC_NAME
00007fa6e7911000       4 r---- 0000000000000000 008:00002 LC_ADDRESS
00007fa6e7912000       4 r---- 0000000000000000 008:00002 LC_TELEPHONE
00007fa6e7913000       4 r---- 0000000000000000 008:00002 LC_MEASUREMENT
00007fa6e7914000      28 r--s- 0000000000000000 008:00002 gconv-
         modules.cache
00007fa6e791b000       4 r---- 0000000000000000 008:00002
         LC_IDENTIFICATION
00007fa6e791c000      12 rw--- 00007fa6e791c000 000:00000   [ anon ]
00007fa6e791f000       8 rw--- 000000000001d000 008:00002 ld-2.7.so
00007fff8bc0f000      84 rw--- 00007ffffffe9000 000:00000   [ stack ]
00007fff8bd1d000       8 r-x-- 00007fff8bd1d000 000:00000   [ anon ]
ffffffffff600000       4 r-x-- 0000000000000000 000:00000   [ anon ]
mapped: 21168K    writeable/private: 2864K    shared: 28K
gandreu@shell-sisop:~$
```

Memory map of a Linux process

- **File /proc/PID/maps**
  - It contains actual memory regions associated to process PID and their access permissions
  - Maps file format:

Address	perm	shift	device	node-i	path
08048000-08056000	r-xp	00000000	03:0c	64593	/usr/sbin/gpm
08056000-08058000	rw-p	0000d000	03:0c	64593	/usr/sbin/gpm
08058000-0805b000	rwxp	00000000	00:00	0	
40000000-40013000	r-xp	00000000	03:0c	4165	/lib/ld-2.2.4.so
40013000-40015000	rw-p	00012000	03:0c	4165	/lib/ld-2.2.4.so
4001f000-40135000	r-xp	00000000	03:0c	45494	/lib/libc-2.2.4.so
40135000-4013e000	rw-p	00115000	03:0c	45494	/lib/libc-2.2.4.so
4013e000-40142000	rw-p	00000000	00:00	0	
bf f f f 000-c0000000	rwxp	00000000	00:00	0	

Address: Logical address ranges for process regions

Node-i: device node-i, 0 means no node-i

Device: device id (major number: minor number)

Permissions:
r = read
w = write
x = execute
s = shared
p = private (copy on write)

Shift: shift inside the supporting file

Memory map of a Linux process

32 bit architecture

```
pblanes$ cat /proc/$$/maps
00140000-00148000 r-xp 00000000 08:01 266235     /lib/tls/i686/cmov/libnss_nis-2.11.1.so
00148000-00149000 r--p 00007000 08:01 266235     /lib/tls/i686/cmov/libnss_nis-2.11.1.so
00149000-0014a000 rw-p 00008000 08:01 266235     /lib/tls/i686/cmov/libnss_nis-2.11.1.so
00266000-00268000 r-xp 00000000 08:01 266220     /lib/tls/i686/cmov/libdl-2.11.1.so
00268000-00269000 r--p 00001000 08:01 266220     /lib/tls/i686/cmov/libdl-2.11.1.so
00269000-0026a000 rw-p 00002000 08:01 266220     /lib/tls/i686/cmov/libdl-2.11.1.so
00319000-0046c000 r-xp 00000000 08:01 266214     /lib/tls/i686/cmov/libc-2.11.1.so
0046c000-0046d000 ---p 00153000 08:01 266214     /lib/tls/i686/cmov/libc-2.11.1.so
0046d000-0046f000 r--p 00153000 08:01 266214     /lib/tls/i686/cmov/libc-2.11.1.so
0046f000-00470000 rw-p 00155000 08:01 266214     /lib/tls/i686/cmov/libc-2.11.1.so
00470000-00473000 rw-p 00000000 00:00 0
005b0000-005b1000 r-xp 00000000 00:00 0          [vdso]
007f7000-0082b000 r-xp 00000000 08:01 261740     /lib/libncurses.so.5.7
0082b000-0082c000 ---p 00034000 08:01 261740     /lib/libncurses.so.5.7
0082c000-0082e000 r--p 00034000 08:01 261740     /lib/libncurses.so.5.7
0082e000-0082f000 rw-p 00036000 08:01 261740     /lib/libncurses.so.5.7
00cbc000-00cd7000 r-xp 00000000 08:01 261663     /lib/ld-2.11.1.so
00cd7000-00cd8000 r--p 0001a000 08:01 261663     /lib/ld-2.11.1.so
00cd8000-00cd9000 rw-p 0001b000 08:01 261663     /lib/ld-2.11.1.so
```

Memory map of a Linux process

64 bit architecture

gandreu$ cat /proc/$$/maps

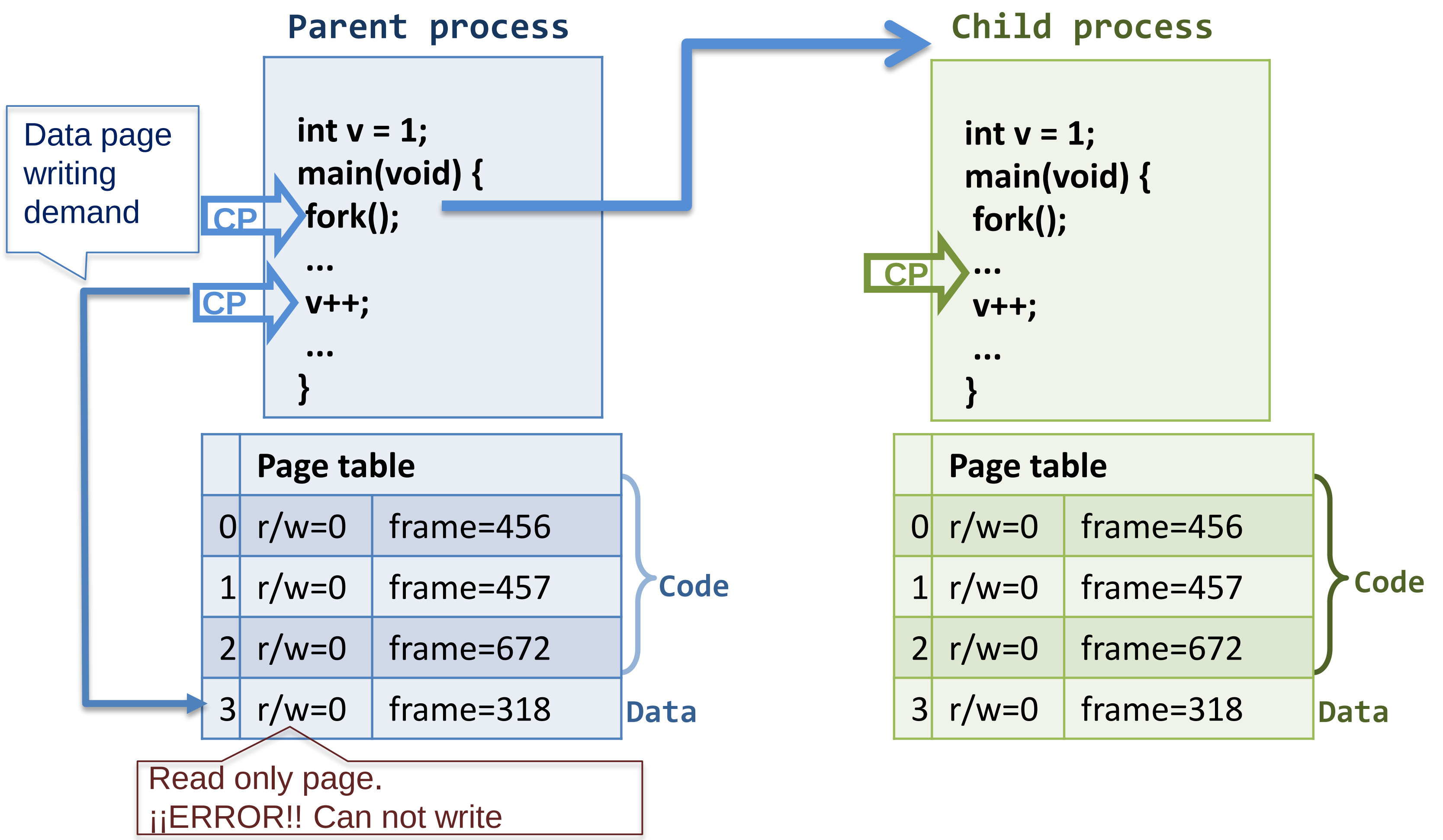
```
00400000-004be000 r-xp 00000000 08:02 65607                /bin/bash
006bd000-006c7000 rw-p 000bd000 08:02 65607                /bin/bash
006c7000-00955000 rw-p 006c7000 00:00 0                    [heap]
7fa6e6728000-7fa6e6732000 r-xp 00000000 08:02 81942        /lib/libnss_files-2.7.so
7fa6e6732000-7fa6e6932000 ---p 0000a000 08:02 81942        /lib/libnss_files-2.7.so
7fa6e6932000-7fa6e6934000 rw-p 0000a000 08:02 81942        /lib/libnss_files-2.7.so
7fa6e6934000-7fa6e693e000 r-xp 00000000 08:02 81944        /lib/libnss_nis-2.7.so
7fa6e693e000-7fa6e6b3d000 ---p 0000a000 08:02 81944        /lib/libnss_nis-2.7.so
7fa6e6b3d000-7fa6e6b3f000 rw-p 00009000 08:02 81944        /lib/libnss_nis-2.7.so
7fa6e6b3f000-7fa6e6b55000 r-xp 00000000 08:02 81939        /lib/libnsl-2.7.so
7fa6e6b55000-7fa6e6d54000 ---p 00016000 08:02 81939        /lib/libnsl-2.7.so
7fa6e6d54000-7fa6e6d56000 rw-p 00015000 08:02 81939        /lib/libnsl-2.7.so
7fa6e6d56000-7fa6e6d58000 rw-p 7fa6e6d56000 00:00 0
7fa6e6d58000-7fa6e6d60000 r-xp 00000000 08:02 81940        /lib/libnss_compat-2.7.so
7fa6e6d60000-7fa6e6f5f000 ---p 00008000 08:02 81940        /lib/libnss_compat-2.7.so
7fa6e6f5f000-7fa6e6f61000 rw-p 00007000 08:02 81940        /lib/libnss_compat-2.7.so
7fa6e6f61000-7fa6e70b9000 r-xp 00000000 08:02 81930        /lib/libc-2.7.so
7fa6e70b9000-7fa6e72b9000 ---p 00158000 08:02 81930        /lib/libc-2.7.so
7fa6e72b9000-7fa6e72bc000 r--p 00158000 08:02 81930        /lib/libc-2.7.so
7fa6e72bc000-7fa6e72be000 rw-p 0015b000 08:02 81930        /lib/libc-2.7.so
7fa6e72be000-7fa6e72c3000 rw-p 7fa6e72be000 00:00 0
7fa6e72c3000-7fa6e72c5000 r-xp 00000000 08:02 81936        /lib/libdl-2.7.so
7fa6e72c5000-7fa6e74c5000 ---p 00002000 08:02 81936        /lib/libdl-2.7.so
7fa6e74c5000-7fa6e74c7000 rw-p 00002000 08:02 81936        /lib/libdl-2.7.so
7fa6e74c7000-7fa6e74fe000 r-xp 00000000 08:02 82217        /lib/libncurses.so.5.6
7fa6e74fe000-7fa6e76fd000 ---p 00037000 08:02 82217        /lib/libncurses.so.5.6
7fa6e76fd000-7fa6e7702000 rw-p 00036000 08:02 82217        /lib/libncurses.so.5.6
7fa6e7702000-7fa6e771f000 r-xp 00000000 08:02 81927        /lib/ld-2.7.so
7fa6e77da000-7fa6e77e9000 r--p 00000000 08:02 271736       /usr/share/locale-langpack/es/LC_MESSAGES/bash.mo
7fa6e77e9000-7fa6e7828000 r--p 00000000 08:02 439402       /usr/lib/locale/es_ES.utf8/LC_CTYPE
7fa6e7828000-7fa6e7909000 r--p 00000000 08:02 439411       /usr/lib/locale/es_ES.utf8/LC_COLLATE
7fa6e7909000-7fa6e790b000 rw-p 7fa6e7909000 00:00 0
7fa6e790b000-7fa6e790c000 r--p 00000000 08:02 439403       /usr/lib/locale/es_ES.utf8/LC_NUMERIC
7fa6e790c000-7fa6e790d000 r--p 00000000 08:02 28628        /usr/lib/locale/es_ES.utf8/LC_TIME
7fa6e790d000-7fa6e790e000 r--p 00000000 08:02 28629        /usr/lib/locale/es_ES.utf8/LC_MONETARY
7fa6e790e000-7fa6e790f000 r--p 00000000 08:02 21591
          /usr/lib/locale/es_ES.utf8/LC_MESSAGES/SYS_LC_MESSAGES
7fa6e790f000-7fa6e7910000 r--p 00000000 08:02 439406       /usr/lib/locale/es_ES.utf8/LC_PAPER
7fa6e7910000-7fa6e7911000 r--p 00000000 08:02 439410       /usr/lib/locale/es_ES.utf8/LC_NAME
7fa6e7911000-7fa6e7912000 r--p 00000000 08:02 28631        /usr/lib/locale/es_ES.utf8/LC_ADDRESS
7fa6e7912000-7fa6e7913000 r--p 00000000 08:02 28633        /usr/lib/locale/es_ES.utf8/LC_TELEPHONE
7fa6e7913000-7fa6e7914000 r--p 00000000 08:02 439407       /usr/lib/locale/es_ES.utf8/LC_MEASUREMENT
7fa6e7914000-7fa6e791b000 r--s 00000000 08:02 446759       /usr/lib/gconv/gconv-modules.cache
7fa6e791b000-7fa6e791c000 r--p 00000000 08:02 28635        /usr/lib/locale/es_ES.utf8/LC_IDENTIFICATION
7fa6e791c000-7fa6e791f000 rw-p 7fa6e791c000 00:00 0
7fa6e791f000-7fa6e7921000 rw-p 0001d000 08:02 81927        /lib/ld-2.7.so
7fff8bc0f000-7fff8bc24000 rw-p 7ffffffe9000 00:00 0        [stack]
7fff8bd1d000-7fff8bd1f000 r-xp 7fff8bd1d000 00:00 0        [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0    [vsyscall]
gandreu@shell-sisop:~$
```

Memory map of a Linux process

- **Copy-on-Write**
  - Linux technique to **efficiently perform copies of memory pages** (it saves memory and time)
  - When a process creates a new one, parent and child share data and stack pages in memory
    - All shared pages are marked as “read-only”
    - A write access attempt to these pages makes the MMU to send a page access failure interrupt, then:
      - The kernel does a copy of the troubling page to the process that wants to write it
        - » If there are more than two processes, the remaining processes continue being unable to write
        - » If it remains only one process using the page it will be able to write changing previously its descriptor bit
      - The interrupted instruction is restarted
  - **Advantage:** Time and space required to copy unused pages are avoided

Memory map of a Linux process

- ***fork()* call "Copy-on-Write"**
  - **Remember**: when a process creates a new one, parent and child share data and stack pages in memory
    - All shared pages are marked as "read-only"
    - An write access attempt to these pages make the MMU to send a page access failure interrupt, then what happens is the following:



Parent process

```
int v = 1;
main(void) {
 fork();
 ...
 v++;
 ...
}
```

	Page table	
0	r/w=0	frame=456
1	r/w=0	frame=457
2	r/w=0	frame=672
3	r/w=0	frame=318

Child process

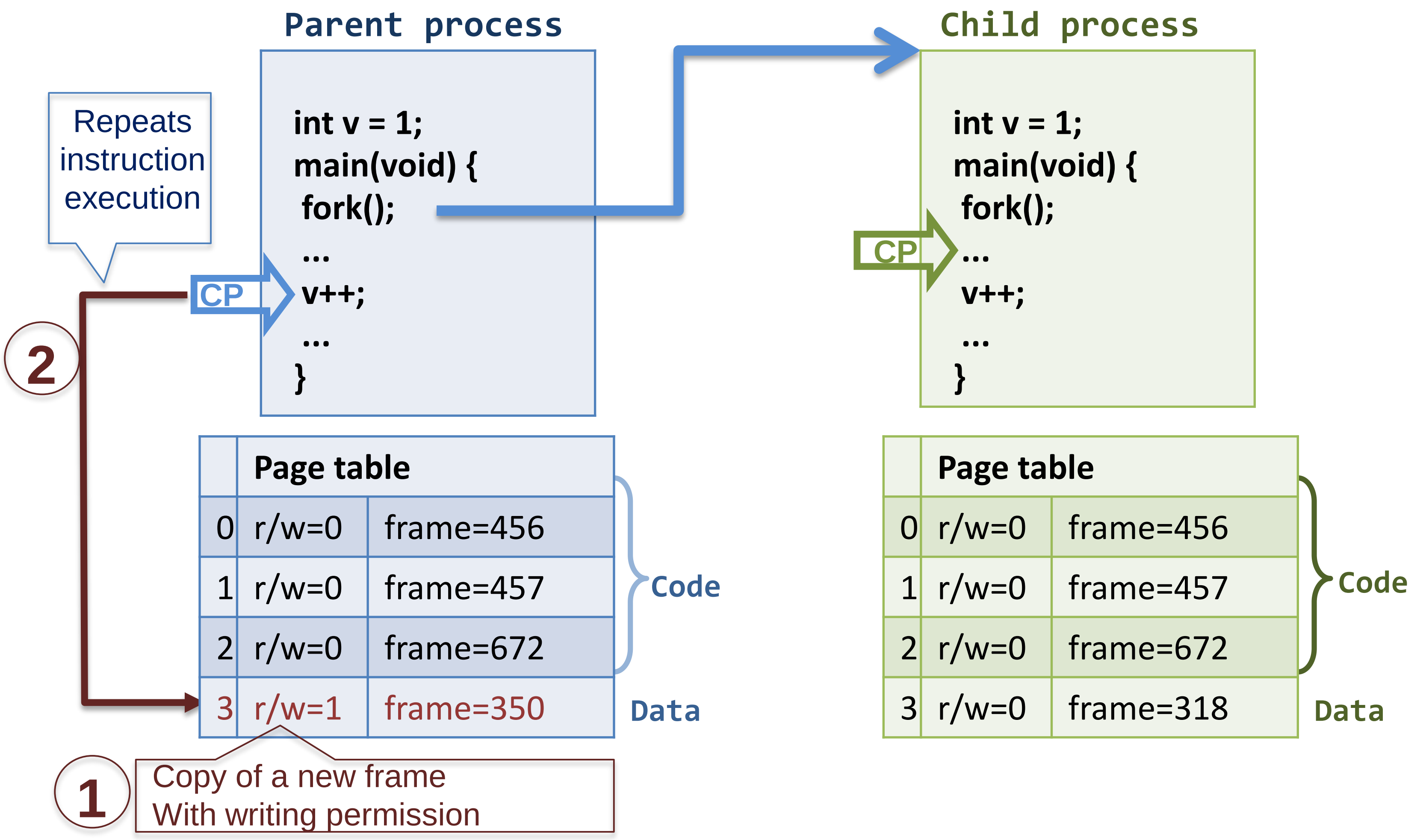
```
int v = 1;
main(void) {
 fork();
 ...
 v++;
 ...
}
```

	Page table	
0	r/w=0	frame=456
1	r/w=0	frame=457
2	r/w=0	frame=672
3	r/w=0	frame=318

Read only page.
¡¡ERROR!! Can not write

Memory map of a Linux process

- ***fork()* call "Copy-on-Write"**
  - **Remember**: when a process creates a new one, parent and child share data and stack pages in memory
    - All shared pages are marked as "read-only"
    - An write access attempt to these pages make the MMU to send a page access failure interrupt, then what happens is the following:



```
int v = 1;
main(void) {
 fork();
 ...
 v++;
 ...
}
```

Parent process page table:

	Page table	
0	r/w=0	frame=456
1	r/w=0	frame=457
2	r/w=0	frame=672
3	r/w=1	frame=350

Child process page table:

	Page table	
0	r/w=0	frame=456
1	r/w=0	frame=457
2	r/w=0	frame=672
3	r/w=0	frame=318

Content

Memory mapped files

- **Memory mapped file**
  - A file (whole or part) is included inside a process memory map

```
POSIX: mmap()
Win32: CreateFileMapping()
```

  - Advantages:
    - **File access time improved**, once mapped into memory access time is set by memory speed instead of hard disk
    - **Intermediate copies avoided** the OS transfers data directly between mapped file memory region and file
  - It is an alternative file access method instead of using I/O calls *read* and *write*
    - The file becomes an array of byte

Memory mapped files

POSIX call **mmap** creates a new region in the process memory map and some of its properties can be set, like sharing and permissions

```
caddr_t mmap (caddr_t addr, size_t length, int protec,
              int indicator, int fd, offt_t shift)
```

- **addr:** memory address for file mapping. If 0 the OS decides. mmap always returns the mapping address used
- **fd:** file descriptor for the file to map (it must be opened)
- **shift** and **length** define the region to map the file, it is from shift to shift+lenght-1
- **protec**: PROT_READ, PROT_WRITE, PROT_EXEC and combinations (i.e. PROT_WRITE | PROT_EXEC)
- **indicator:** MAP_SHARED, MAP_PRIVATE

munmap: removes a previous whole or part file mapping

Memory mapped files

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
```

count_char.c It maps a file in memory and counts the number of apperances of a given character. The character to count is specified as the first parameter and the file to map as the second paramenter.

```
$ gcc count_char.c –o count_char
$ ./count_char c count_char.c
```

```
int main(int  argc, char *argv[] ) {
    int i,fd,count;
     char *p,*org;
     struct stat bstat;
     char ch;

     ch = argv[1][0];
    fd = open(argv[2], O_RDONLY); /* File open */
    fstat(fd, &bstat); /* Gets file length*/
    /* File mapping */
    org = mmap((caddr_t) 0, bstat.st_size, PROT_READ,MAP_SHARED, fd, 0);
    close(fd); /* File close */
    /* Access loop */
    p = org;
    count = 0;
    for (i=0; i<bstat.st_size; i++)
      if (*p++ == ch) count++;
    /* Remove mapping */
    munmap(org, bstat.st_size);
    printf("%d\n", count);
}
```

Memory mapped files

- Memory map before file mapping

```
pblanes$ pmap 21914:
21914     ./count_char c count_char.c
08048000      4K r-x--  /home/naomac/fso/count_char
08049000      4K rw---  /home/naomac/fso/count_char
b7de1000      4K rw---    [ anon ]
b7de2000   1316K r-x--  /lib/tls/i686/cmov/libc-2.7.so
b7f2b000      4K r----  /lib/tls/i686/cmov/libc-2.7.so
b7f2c000      8K rw---  /lib/tls/i686/cmov/libc-2.7.so
b7f2e000     12K rw---    [ anon ]
b7f42000     12K rw---    [ anon ]
b7f45000      4K r-x--    [ anon ]
b7f46000    104K r-x--  /lib/ld-2.7.so
b7f60000      8K rw---  /lib/ld-2.7.so
bf986000     84K rw---    [ stack ]
 total     1564K
```

- Memory map after file mapping

```
pblanes$ pmap 21914
21914:   ./count_char c count_char.c
08048000      4K r-x--  /home/naomac/fso/count_char
08049000      4K rw---  /home/naomac/fso/count_char
b7de1000      4K rw---    [ anon ]
b7de2000   1316K r-x--  /lib/tls/i686/cmov/libc-2.7.so
b7f2b000      4K r----  /lib/tls/i686/cmov/libc-2.7.so
b7f2c000      8K rw---  /lib/tls/i686/cmov/libc-2.7.so
b7f2e000     12K rw---    [ anon ]
b7f41000      4K r--s-  /home/naomac/fso/count_char.c
b7f42000     12K rw---    [ anon ]
b7f45000      4K r-x--    [ anon ]
b7f46000    104K r-x--  /lib/ld-2.7.so
b7f60000      8K rw---  /lib/ld-2.7.so
bf986000     84K rw---    [ stack ]
 total     1568K
```

Mapped file count_char.c

Memory mapped files

*“**map.c**”* maps a file into memory and it shows the process memory map before and after mapping the file which name is specified in the first program parameter

```
$ gcc map.c –o map
$ map map.c
```

```c
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>

void error (char * message) {
  perror(message);
  exit(EXIT_FAILURE);
}

void build_command(char command[80]) {
  // Build command to show memory map
  sprintf(command,"cat /proc/%d/maps",getpid());

}
```

```c
int main (int argc,char *argv[])
{
  int fd;
  void *map;
  struct stat statbuf;
  char path_maps[80];

  // Open the file to map
  if (argc!=2) {
    puts("Usage: map FileName \n");
    exit(EXIT_FAILURE) ;
  }
  if ((fd=open(argv[1],O_RDONLY))<0)
    error("Open file failure (open) \n");

  // Get file length
  fstat(fd, &statbuf);
  // fstat dumps its information to statbuf

  // SHOW MAP
  printf(" PROCESS MEMORY MAP /proc/%d/maps \n", getpid());
  build_command(path_maps);
  system(path_maps); // Command execution system call
```

continues…

Memory mapped files

“***map.c***” continuing…

```
   // Map input file
   if ((map = mmap(0,statbuf.st_size,PROT_READ,MAP_SHARED,fd,0)) == MAP_FAILED)
     error(“Mapping failure(mmap)");

   close(fd); // Close file

   // SHOW MAP
   printf ("\n\n MEMORY MAPPED FILE \n");

   system(path_maps); // Command execution system call

   munmap(map, statbuf.st_size); // Remove mapping

   printf ("\n\n MEMORY MAPPING MAP REMOVED \n");

   system(path_maps);

   exit(EXIT_SUCCESS);

 } /* main end */
```

System call to execute the previously defined command:
`cat /proc/%d/maps`
It shows this process “maps” file

Content

Dynamic linking libraries

- **Programming libraries**
  - Binary (no text) files that contain functions code
  - There are two way of linking programs with libraries:
    - **Static linking**: the executable file includes all library functions code
      - Program code = Own program code + Library functions code
      - **.lib files on Windows, .a files on UNIX/Linux**
    - **Dynamic linking:** the executable file contains references to library functions that it uses and the memory region required to store them
      - Library functions are loaded in memory on demand mapping them in the corresponding process memory map region. This is done by the library loader program (i.e. **ld** in Linux) by means of ***dlopen*** call
      - **Windows:** `.dll` files
      - **UNIX/Linux**: `.so` files

Dynamic linking libraries

- **Static linking**
  - Disadvantages
    - Generally big executable files
    - Library functions code replicated in many executables in the file system and in memory during execution
    - A library update requires rebuilding programs
  - Advantages
    - Executable files are self contained
- **Dynamic linking**
  - Advantages
    - Smaller executable size that saves disk and memory space
    - No library functions code replication
      - Processes share library code in memory
      - Library updates don’t require rebuilding and several library version can coexists
  - Disadvantages
    - Executable files rely in library files
    - Dynamic linking introduces execution time overhead

Dynamic linking libraries

- `ejemplo1` executable size with static and dynamic linking of math library
  - Static linking

```
pblanes$  gcc ejemplo1.c -static -o ejemplo1 -lm
pblanes$  ls –l
total 660
-rwxr-xr-x 1 pblanes disca-upvnet 670227 2011-10-20 15:54 ejemplo1
-rw-r--r--   1 pblanes disca-upvnet     905 2011-10-20 13:30 ejemplo1.c
```

  - Dynamic linking (gcc default)

```
pblanes$  gcc ejemplo1.c  -o ejemplo1 -lm
pblanes$  ls –l
total 16
-rwxr-xr-x 1 pblanes disca-upvnet 10301  2011-10-20 15:56 ejemplo1
-rw-r--r–   1  pblanes disca-upvnet   905 2011-10-20 13:30 ejemplo1.c
```