

# Universidad de Guadalajara

---

## Ingeniería Informática



### Hands-on 5: Process & ProcessImage

**SEMINARIO DE SOLUCION DE PROBLEMAS DE USO, ADAPTACION, EXPLOTACION  
DE SISTEMAS OPERATIVOS - 103857 - D03**

**Alumno: Ramirez Gomez Kevin Ramses**

**Código: 218218194**

**Profesor: Jose Antonio Aviña Mendez**

**Fecha: 07/11/2024**



# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De  
Sistemas Operativos - 103857 - D03

## Índice

Desarrollo.....	3
Información de un proceso .....	3
¿Cómo a través de un programa obtener información de un Proceso?.....	6
Imagen de un proceso.....	6
¿Cómo a través de un programa obtener información sobre la Imagen de un Proceso? .....	10



## DESARROLLO

### INFORMACIÓN DE UN PROCESO

Primero tenemos que ejecutar el siguiente comando para tener el compilador de C presente en el sistema: `sudo apt install gcc`.

```
KevinRamirezGomez@MaquinaUbuntu: ~  
[sudo] password for KevinRamirezGomez:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  binutils binutils-common binutils-x86-64-linux-gnu gcc-13  
  gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu libasan8 libbinutils libbct1-0  
  libctf-nobfd0 libctf0 libgcc-13-dev libgprofng0 libhwasean0 libitm1 liblsan0  
  libquadmath0 libstdc++11 libtsan2 libubsan1  
Suggested packages:  
  binutils-doc gprofng-gui gcc-multilib make autoconf automake libtool flex  
  bison gcc-doc gcc-13-multilib gcc-13-doc gcc-13-locales gdb-x86-64-linux-gnu  
The following NEW packages will be installed:  
  binutils binutils-common binutils-x86-64-linux-gnu gcc gcc-13  
  gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu libasan8 libbinutils libbct1-0  
  libctf-nobfd0 libctf0 libgcc-13-dev libgprofng0 libhwasean0 libitm1 liblsan0  
  libquadmath0 libstdc++11 libtsan2 libubsan1  
0 upgraded, 21 newly installed, 0 to remove and 71 not upgraded.  
Need to get 39.5 MB of archives.  
After this operation, 144 MB of additional disk space will be used.  
Do you want to continue? [Y/n] Y  
Get:1 http://mx.archive.ubuntu.com/ubuntu noble/main amd64 binutils-common amd64  
2.42-4ubuntu2 [239 kB]
```

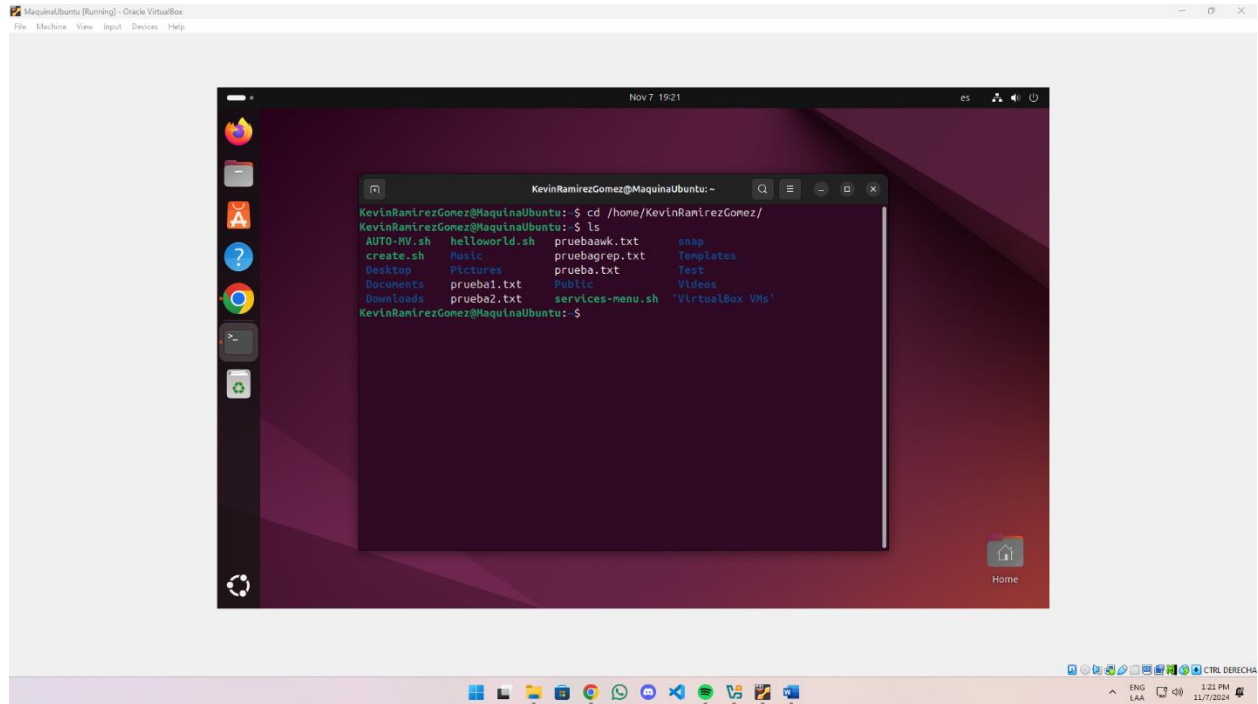


# Universidad de Guadalajara

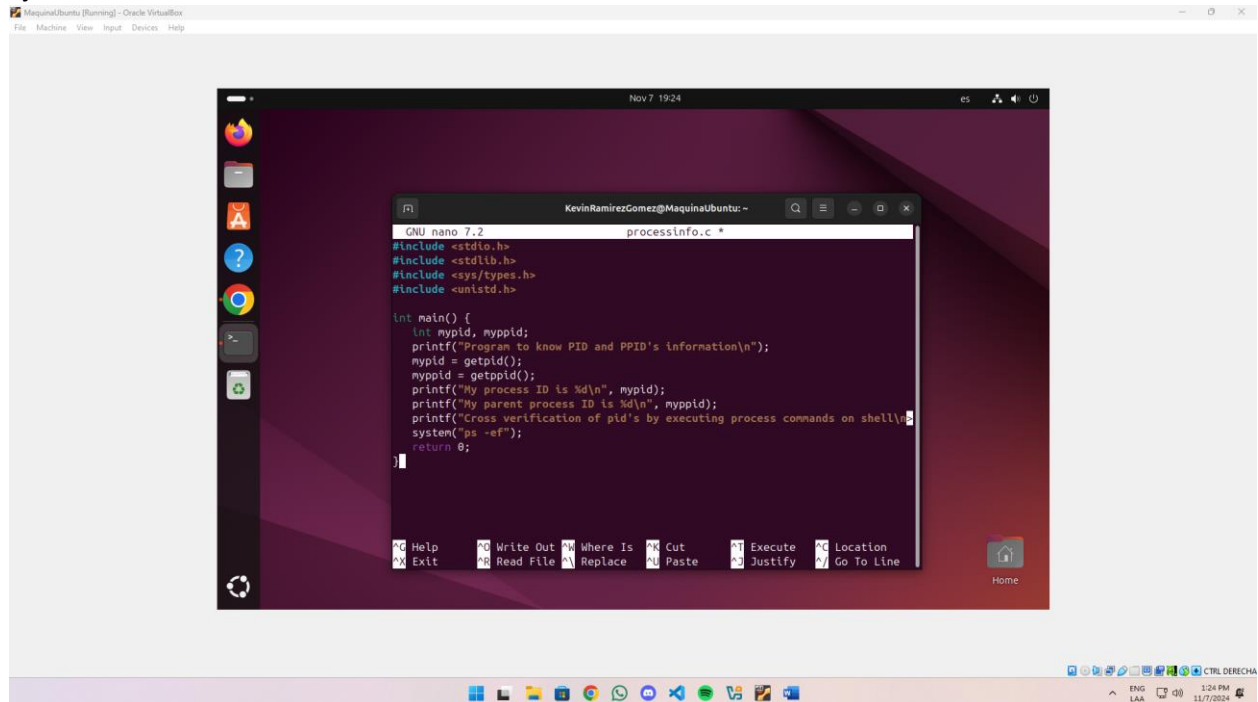
Centro Universitario de Ciencias Exactas e Ingenierías

Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De  
Sistemas Operativos - 103857 - D03

Ya que termine de instalar el compilador luego procederemos a moverlos a la carpeta donde hemos estado trabajando los hand-on anteriores.



Luego procederemos a crear el archivo con el programa en c para poder convertirlo en un ejecutable.





# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De Sistemas Operativos - 103857 - D03

Ahora convertiremos el archivo en un archivo ejecutable para el compilador, con el siguiente comando: gcc processinfo.c -o processinfo

```
KevinRamirezGomez@MaquinaUbuntu: ~  
KevinRamirezGomez@MaquinaUbuntu: $ cd /home/KevinRamirezGomez/  
KevinRamirezGomez@MaquinaUbuntu: $ ls  
AUTO-MV.sh  helloworld.sh  pruebaawk.txt  snap  
create.sh   Music          pruebaagrep.txt Templates  
Desktop     Pictures       prueba.txt     Test  
Documents   prueba1.txt    Public         Videos  
Downloads   prueba2.txt    services-menu.sh 'VirtualBox VMs'  
KevinRamirezGomez@MaquinaUbuntu: $ nano processinfo.c  
KevinRamirezGomez@MaquinaUbuntu: $ gcc processinfo.c -o processinfo  
KevinRamirezGomez@MaquinaUbuntu: $ ls  
AUTO-MV.sh  Music          pruebaawk.txt  Templates  
create.sh   Pictures       pruebaagrep.txt Test  
Desktop     processinfo    prueba.txt     Videos  
Documents   processinfo.c  Public         'VirtualBox VMs'  
Downloads   prueba1.txt    services-menu.sh  
helloworld.sh prueba2.txt    snap  
KevinRamirezGomez@MaquinaUbuntu: $
```

Ahora ejecutamos el programa con el comando ./processinfo.

```
KevinRamirezGomez@MaquinaUbuntu: ~  
KevinRamirezGomez@MaquinaUbuntu: $ ./processinfo  
Program to know PID and PPID's information  
My process ID is 7143  
My parent process ID is 7020  
Cross verification of pid's by executing process commands on shell  
UID      PID      PPID      C  STIME  TTY      TIME  CMD  
root      1         0  0  18:22  ?        00:00:01 /sbin/init splash  
root      2         0  0  18:22  ?        00:00:00 [kthreadd]  
root      3         2  0  18:22  ?        00:00:00 [pool_workqueue_release]  
root      4         2  0  18:22  ?        00:00:00 [worker/R-rcu_p]  
root      5         2  0  18:22  ?        00:00:00 [worker/R-rcu_p]  
root      6         2  0  18:22  ?        00:00:00 [worker/R-slub_]  
root      7         2  0  18:22  ?        00:00:00 [worker/R-netns]  
root     10         2  0  18:22  ?        00:00:00 [worker/0:0H-events_highpri]  
root     12         2  0  18:22  ?        00:00:00 [worker/R-mm_pe]  
root     13         2  0  18:22  ?        00:00:00 [rcu_tasks_kthread]  
root     14         2  0  18:22  ?        00:00:00 [rcu_tasks_rude_kthread]  
root     15         2  0  18:22  ?        00:00:00 [rcu_tasks_trace_kthread]  
root     16         2  0  18:22  ?        00:00:00 [ksftirqd/0]  
root     17         2  0  18:22  ?        00:00:00 [rcu_preempt]  
root     18         2  0  18:22  ?        00:00:00 [migration/0]  
root     19         2  0  18:22  ?        00:00:00 [idle_inject/0]  
root     20         2  0  18:22  ?        00:00:00 [cpuhp/0]  
root     21         2  0  18:22  ?        00:00:00 [cpuhp/1]  
root     22         2  0  18:22  ?        00:00:00 [idle_inject/1]  
root     23         2  0  18:22  ?        00:00:00 [migration/1]  
root     24         2  0  18:22  ?        00:00:05 [ksftirqd/1]  
root     26         2  0  18:22  ?        00:00:00 [worker/1:0H-events_highpri]  
root     27         2  0  18:22  ?        00:00:00 [cpuhp/2]  
root     28         2  0  18:22  ?        00:00:00 [idle_inject/2]  
root     29         2  0  18:22  ?        00:00:00 [migration/2]  
root     30         2  0  18:22  ?        00:00:00 [ksftirqd/2]  
root     31         2  0  18:22  ?        00:00:00 [ksftirqd/2]
```



# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

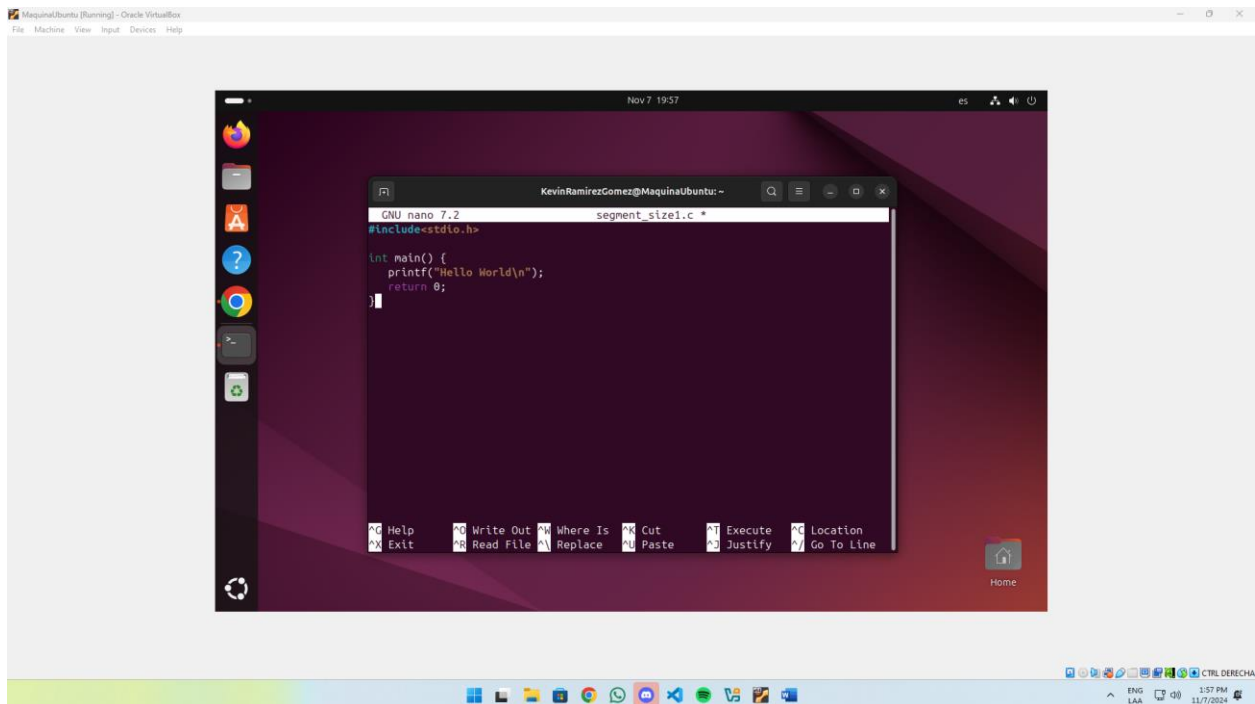
Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De Sistemas Operativos - 103857 - D03

## ¿Cómo a través de un programa obtener información de un Proceso?

Los procesos en un entorno Linux permite entender de manera profunda cómo los programas interactúan con el sistema operativo. Utilizar funciones como getpid() y getppid() ayuda a identificar el proceso actual y su proceso padre, mientras que comandos como system("ps -ef") permiten verificar y gestionar los procesos en ejecución. Esta parte de la actividad me sirvió para comprender y trabajar con la información de procesos desde un programa en C ofreciendo una perspectiva profunda sobre cómo los programas interactúan con el sistema operativo.

## IMAGEN DE UN PROCESO

Ahora para esta parte de la actividad crearemos cinco programas en c, donde estos nos muestran los diferentes segmentos que suele contener una imagen de proceso.



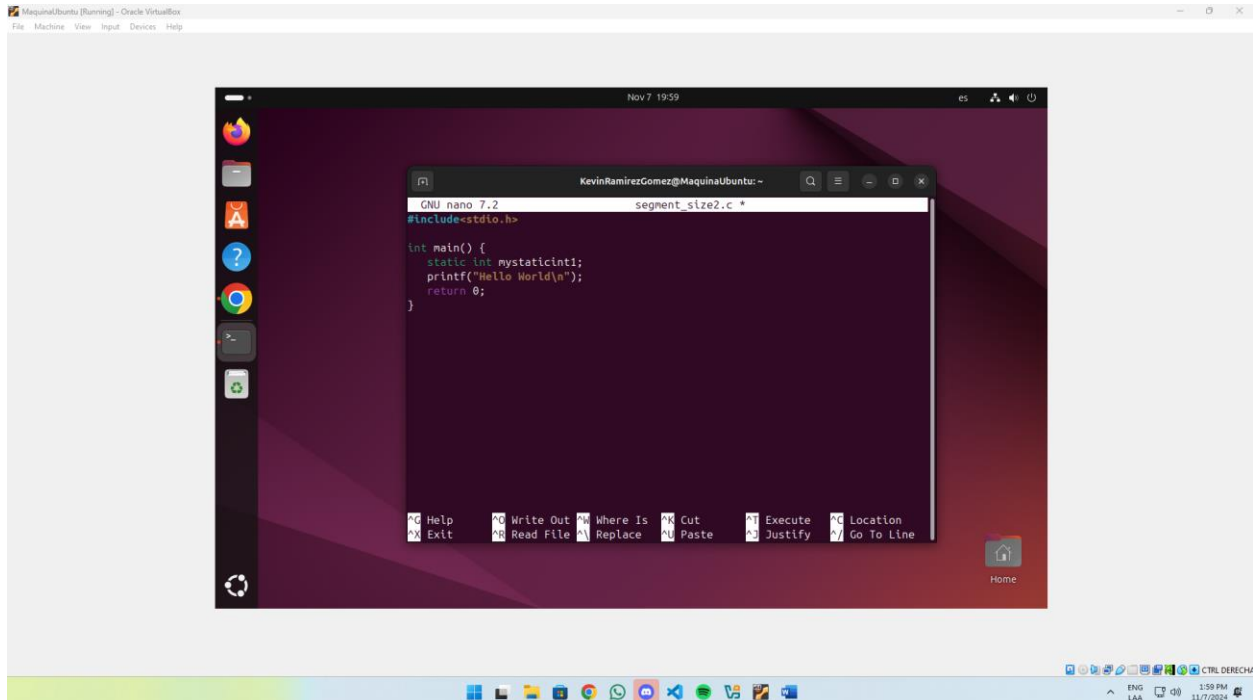
Este programa o tiene variables estáticas ni globales, por lo que el segmento de datos y BSS no se ven afectados.



# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

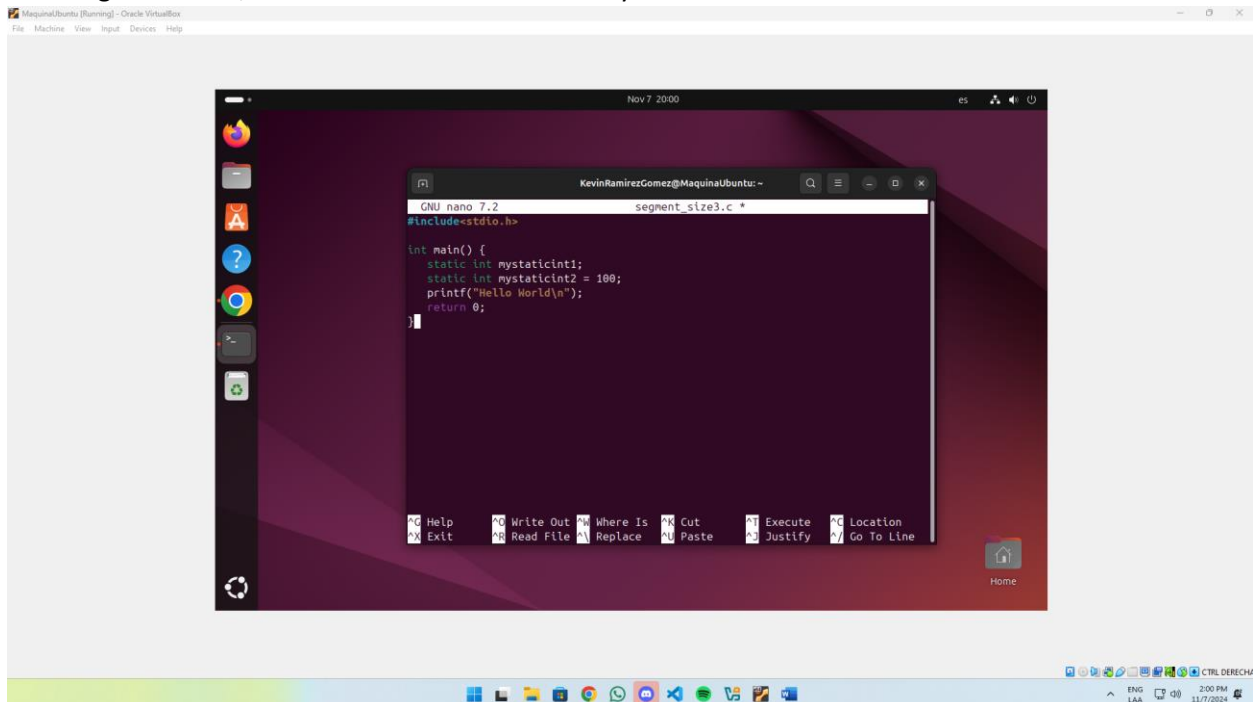
Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De Sistemas Operativos - 103857 - D03



```
GNU nano 7.2 segment_size2.c *
#include<stdio.h>

int main() {
    static int mystaticint1;
    printf("Hello World\n");
    return 0;
}
```

Este codigo introduce una variable estática no inicializada, static int mystaticint1. Esta variable se aloja en el segmento BSS, aumentando su tamaño en 4 bytes.



```
GNU nano 7.2 segment_size3.c *
#include<stdio.h>

int main() {
    static int mystaticint1;
    static int mystaticint2 = 100;
    printf("Hello World\n");
    return 0;
}
```

Este codigo añade una variable estática inicializada static int mystaticint2 = 100. Esta variable se coloca



# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De  
Sistemas Operativos - 103857 - D03

en el segmento de datos, que aumenta en 4 bytes, mientras que mystaticint1 sigue en el BSS.

```
GNU nano 7.2 segment_size4.c *
#include<stdio.h>

int myglobalint1 = 500;
int main() {
    static int mystaticint1;
    static int mystaticint2 = 100;
    printf("Hello World\n");
    return 0;
}

File Name to Write: segment_size4.c
^C Help      ^M DOS Format  ^M Append      ^M Backup File
^C Cancel    ^M Mac Format  ^M Prepend     ^M Browse
```

Este codigo declara una variable global inicializada int myglobalint1 = 500. Esta variable incrementa el tamaño del segmento de datos en 4 bytes, además de las variables estáticas.

```
GNU nano 7.2 segment_size5.c *
#include<stdio.h>

int myglobalint1 = 500;
int myglobalint2;
int main() {
    static int mystaticint1;
    static int mystaticint2 = 100;
    printf("Hello World\n");
    return 0;
}
```

Este código incluye una variable global no inicializada int myglobalint2. Esta variable se aloja en el





# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De  
Sistemas Operativos - 103857 - D03

segmento BSS, aumentando su tamaño en 4 bytes. Además, las variables myglobalint1 y mystaticint2 ocupan espacio en el segmento de datos.

Ahora procederemos a convertirlos en archivos ejecutables.

```
KevinRamirezGomez@MaquinaUbuntu: ~  
Desktop      processinfo  prueba.txt    Videos  
Documents    processinfo.c Public        'VirtualBox VMs'  
Downloads    prueba1.txt  services-menu.sh  
helloworld.sh prueba2.txt  snap  
KevinRamirezGomez@MaquinaUbuntu: $ nano segment_size1.c  
KevinRamirezGomez@MaquinaUbuntu: $ nano segment_size2.c  
KevinRamirezGomez@MaquinaUbuntu: $ nano segment_size3.c  
KevinRamirezGomez@MaquinaUbuntu: $ nano segment_size4.c  
KevinRamirezGomez@MaquinaUbuntu: $ nano segment_size5.c  
KevinRamirezGomez@MaquinaUbuntu: $ gcc segment_size1.c -o segment_size1  
KevinRamirezGomez@MaquinaUbuntu: $ gcc segment_size2.c -o segment_size2  
KevinRamirezGomez@MaquinaUbuntu: $ gcc segment_size3.c -o segment_size3  
KevinRamirezGomez@MaquinaUbuntu: $ gcc segment_size4.c -o segment_size4  
KevinRamirezGomez@MaquinaUbuntu: $ gcc segment_size5.c -o segment_size5  
KevinRamirezGomez@MaquinaUbuntu: $ ls  
AUTO-MV.sh  processinfo  segment_size1  segment_size5  
create.sh   processinfo.c segment_size1.c segment_size5.c  
Desktop     prueba1.txt  segment_size2  services-menu.sh  
Downloads   prueba2.txt  segment_size2.c snap  
helloworld.sh pruebaawk.txt segment_size3  templates  
helloworld.sh pruebaagrep.txt segment_size3.c text  
Music       prueba.txt   segment_size4  Videos  
Pictures    Public       segment_size4.c 'VirtualBox VMs'  
KevinRamirezGomez@MaquinaUbuntu: $
```



# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De Sistemas Operativos - 103857 - D03

Ahora podemos observar como el segmento BSS y DATA cambian según las variables estaticas o globales, inicializadas/no inicializadas en el programa.

```
KevinRamirezGomez@MaquinaUbuntu:~$ nano segment_size5.c
KevinRamirezGomez@MaquinaUbuntu:~$ gcc segment_size1.c -o segment_size1
KevinRamirezGomez@MaquinaUbuntu:~$ gcc segment_size2.c -o segment_size2
KevinRamirezGomez@MaquinaUbuntu:~$ gcc segment_size3.c -o segment_size3
KevinRamirezGomez@MaquinaUbuntu:~$ gcc segment_size4.c -o segment_size4
KevinRamirezGomez@MaquinaUbuntu:~$ gcc segment_size5.c -o segment_size5
KevinRamirezGomez@MaquinaUbuntu:~$ ls
AUTO-NV.sh  processinfo  segment_size1  segment_size5
create.sh   processinfo.c  segment_size1.c  segment_size5.c
Desktop     prueba1.txt   segment_size2   services-menu.sh
Documents   prueba2.txt   segment_size2.c  snap
Downloads   pruebaawk.txt segment_size3     Templates
helloworld.sh pruebaagrep.txt segment_size3.c   Test
Music        prueba.txt    segment_size4     Videos
Pictures     Public        segment_size4.c   'VirtualBox VMs'
KevinRamirezGomez@MaquinaUbuntu:~$ size segment_size1 segment_size2 segment_size3 segment_size4 segment_size5
3 segment_size4 segment_size5
text  data  bss  dec  hex filename
1374  680   8  1982  7be segment_size1
1374  680   8  1982  7be segment_size2
1374  684  12  1990  7c6 segment_size3
1374  688   8  1998  7c6 segment_size4
1374  688  16  1998  7ce segment_size5
KevinRamirezGomez@MaquinaUbuntu:~$
```

## ¿Cómo a través de un programa obtener información sobre la Imagen de un Proceso?

Al analizar cómo las variables estáticas y globales afectan la segmentación de memoria usando herramientas como size. Al agregar diferentes tipos de variables en los codigos, los segmentos BSS y DATA cambiarán. El comando size nos permite ver cómo se distribuyen los tamaños de estos segmentos y cómo varían según el programa compilado, dándonos los siguientes resultados: text: instrucciones ejecutables, data: variables inicializadas, bss: variables no inicializadas, dec: Tamaño total en decimal, hex: Tamaño total en hexadecimal y filename: Nombre del archivo ejecutable.