



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PRÁCTICA TRES

Id del proceso

Alumno:

Rojas Zepeda Luis Eduardo

Profesor:

Velez Saldaña Ulises

2CM6, 18/03/19



Índice

1. Teoría	3
2. Material	4
3. Desarrollo	5
4. Errores	11
5. Bibliografía	12

1. Teoría

Llamada getpid

Es una llamada para el control de procesos que retorna el pid del proceso que la invoco. Esta función no recibe ningún argumento y retorna un entero del tipo pid_t. Para hacer uso de esta llamada se debe incluir en el código los archivos de cabecera:

```
sys/types.h
unistd.h
```

Libreria time

La función localtime convierte el tiempo en formato condensado apuntado por tiempoPtr en el tiempo en formato separado, expresado como el tiempo local.

Sintaxis:

```
struct tm *localtime(const time_t *tiempoPtr);
```

Libreria signal

signal.h es un archivo de cabecera definido en la Biblioteca estándar de C para especificar como un programa maneja señales mientras se ejecuta. Una señal puede reportar un comportamiento excepcional en el programa (tales como la división por cero), o una señal puede reportar algún evento asíncrono fuera del programa (como alguien está pulsando una tecla de atención interactiva en el teclado).

2. Material

Editor de Texto Nano

Solo se instala con la siguiente linea:

```
sudo apt-get install nano
```

Copilador gcc

Se instala con la siguiente linea:

```
sudo apt-get install gcc
```

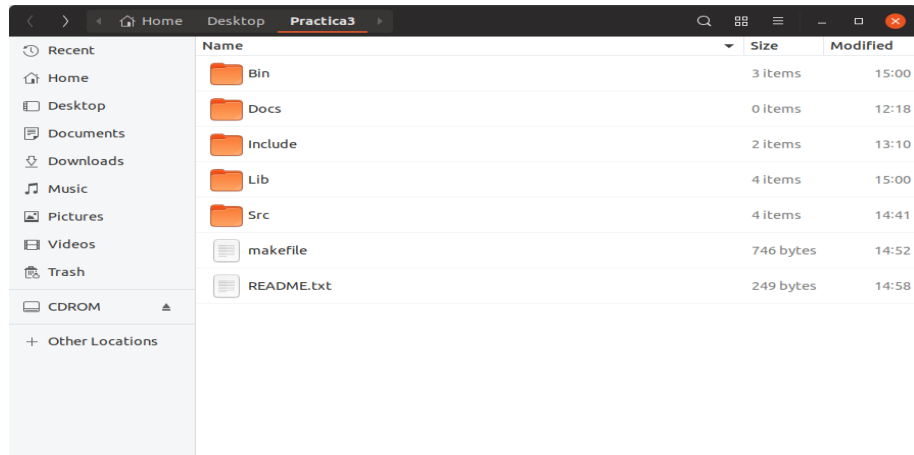
Makefile

Se instala con la siguiente linea:

```
sudo make install
```

3. Desarrollo

Crearemos las siguientes carpetas:



Después se debe generar los siguiente códigos:

Código del programa uno:

```
1  /**
2  programa: Practica 3 programa 3
3  Autor: Rojas Zepeda Luis Eduardo
4  Fecha: 18/03/2019
5  Descripcion: Ciclo durante 30 segundos imprimiendo un mensaje y el proceso del id.
6  |      Esta la posibilidad de interrumpir el programa con CTRL+C mostrando un mensaje.
7  */
8
9  #include <stdio.h>
10 #include <stdlib.h>
11 #include <unistd.h>
12
13 int main(){
14
15     while(1){
16         printf("Aqui estoy\n");
17         printf ( "Id: %d\n", getpid());
18     }
19     return 0;
20 }
```

Código del programa dos:

```

1  /**
2  programa: Practica 3 programa 2
3  Autor: Rojas Zepeda Luis Eduardo
4  Fecha: 18/03/2019
5  Descripcion: Ciclo durante 30 segundos imprimiendo un mensaje y el proceso del id.
6  | | | Esta la posibilidad de interrumpir el programa con CTRL+C mostrando un mensaje.
7  */
8
9  #include <stdio.h>
10 #include <stdlib.h>
11 #include <unistd.h>
12 #include "../Include/tiempo.h"
13
14 int main(){
15
16     double utime0, stime0, wtime0, utime1, stime1, wtime1; //Variables para medición de tiempos
17
18     uswtime(&utime0, &stime0, &wtime0);
19
20     while((wtime1 - wtime0) < 30){
21         printf("Aqui estoy\n");
22         printf ( "Id: %d\n", getpid());
23         uswtime(&utime1, &stime1, &wtime1);
24     }
25
26     printf("Terminacion normal");
27
28     return 0;
29 }

```

Código del programa tres:

```

16 /**
17 *@brief INThandler se manda a llamar cada que se presiona CTRL+C e imprime un mensaje y termina el programa.
18 *@param numero entero para identificar la señal
19 *@return vacio
20 */
21 void INThandler(int);
22
23 int main(){
24
25     double utime0, stime0, wtime0, utime1, stime1, wtime1; //Variables para medición de tiempos
26
27     uswtime(&utime0, &stime0, &wtime0);
28
29     signal(SIGINT, INThandler);
30
31     while((wtime1 - wtime0) < 30 ){
32         printf("Aqui estoy\n");
33         printf("Id: %d\n", getpid());
34         uswtime(&utime1, &stime1, &wtime1);
35     }
36
37     printf("Terminacion normal");
38
39     return 0;
40 }
41
42 void INThandler(int sig)
43 {
44     signal(sig, SIG_IGN);
45     printf("Terminacion por interrupcion del SO\n\n");
46     exit(0);
47 }

```

Los códigos anteriores se guardarán en la carpeta Src con sus respectivos nombres y extensión .c.

Después crearemos un archivo sin extensión llamado Makefile y en dicho archivo escribiremos los siguiente:

Archivo Makefile

```
1 CFLAGS = -c
2
3 progra1.o: Src/Progra1.c
4 $(CC) $(CFLAGS) Src/Progra1.c -o Lib/Progra1.o
5
6 progra1: progra1.o Lib/Progra1.o
7 $(CC) Lib/Progra1.o -o Bin/Progra1
8
9 runProgra1:
10 Bin/Progra1
11
12 progra2.o: Src/Progra2.c Src/tiempo.c Include/tiempo.h
13 $(CC) $(CFLAGS) Src/Progra2.c -o Lib/Progra2.o
14 $(CC) $(CFLAGS) Src/tiempo.c -o Lib/tiempo.o
15
16 progra2: progra2.o Lib/Progra2.o
17 $(CC) Lib/Progra2.o Lib/tiempo.o -o Bin/Progra2
18
19 runProgra2:
20 Bin/Progra2
21
22 progra3.o: Src/Progra3.c Src/tiempo.c Include/tiempo.h
23 $(CC) $(CFLAGS) Src/Progra3.c -o Lib/Progra3.o
24 $(CC) $(CFLAGS) Src/tiempo.c -o Lib/tiempo.o
25
26 progra3: progra3.o Lib/Progra3.o
27 $(CC) Lib/Progra3.o Lib/tiempo.o -o Bin/Progra3
28
29 runProgra3:
30 Bin/Progra3
31
32 clean:
33 -rm -f Bin/*
34 -rm -f Lib/*
```

Y finalmente crearemos el archivo README.txt con el siguiente texto:

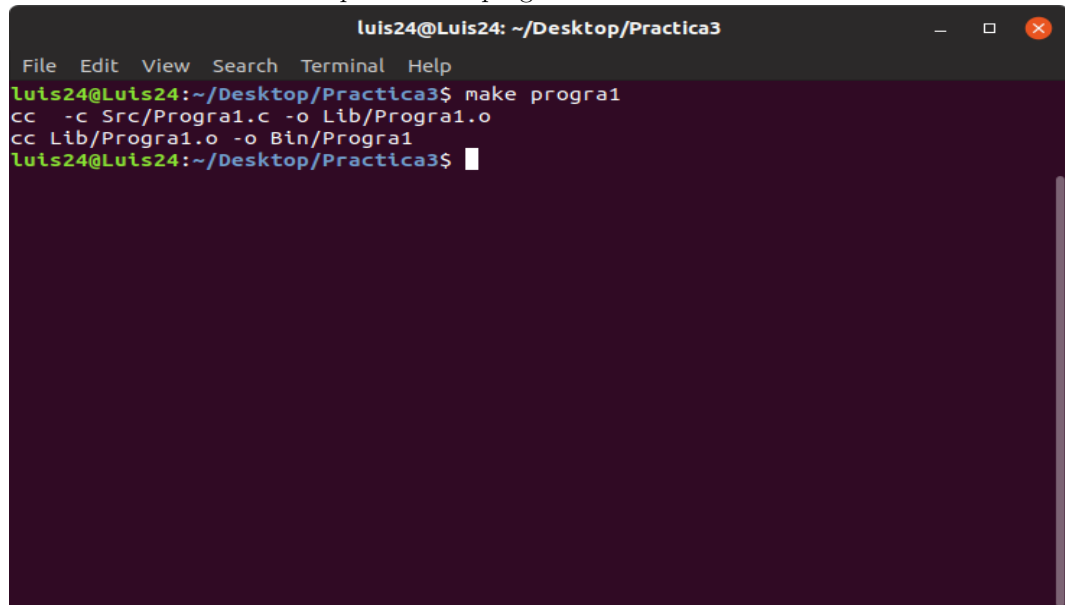
Archivo README

```
1 Compilacion del programa #1:
2   make progra1
3
4 y para correrlo:
5   make runProgra2
6
7 Es analogamente el mismo procedimiento para el programa #2 y #3:
8
9 Compilación:
10  make progra2
11  make progra3
12
13 Ejecución:
14  make runProgra2
15  make runProgra3
```

Este archivo servirá como manual para el usuario para correr los programas de manera más fácil. Este archivo, junto con el makefile, irán en la raíz de la carpeta como se muestra en la primera imagen de esta sección de desarrollo.

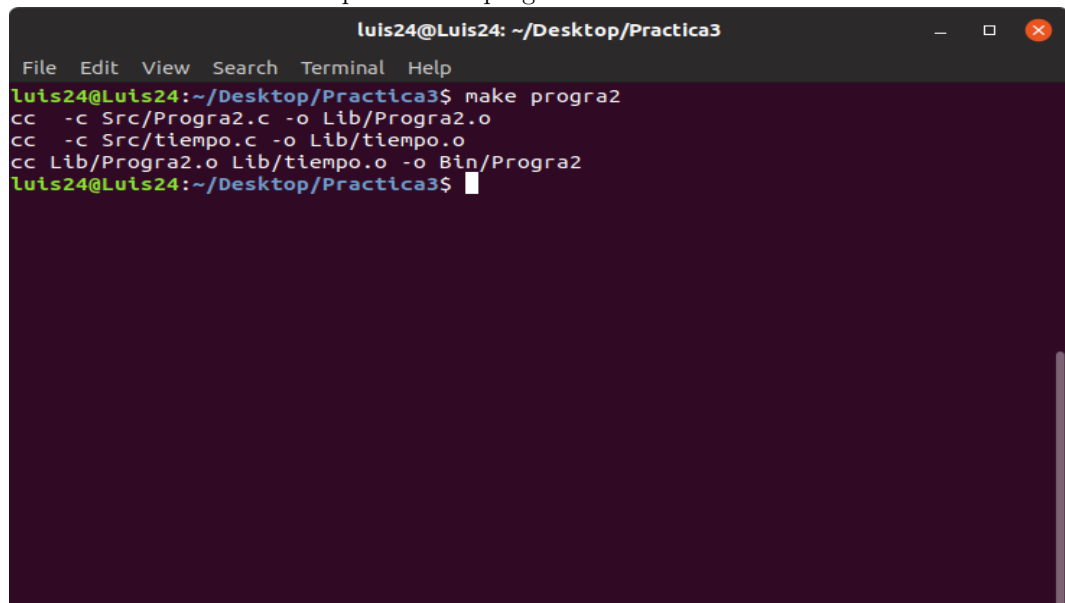
Posteriormente para compilar cada programa se hará de la siguiente forma:

Compilación del programa uno:

A terminal window titled 'luis24@Luis24: ~/Desktop/Practica3' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'make progra1' being executed, which runs two compilation commands: 'cc -c Src/Progra1.c -o Lib/Progra1.o' and 'cc Lib/Progra1.o -o Bin/Progra1'. The prompt returns to 'luis24@Luis24:~/Desktop/Practica3\$' with a cursor.

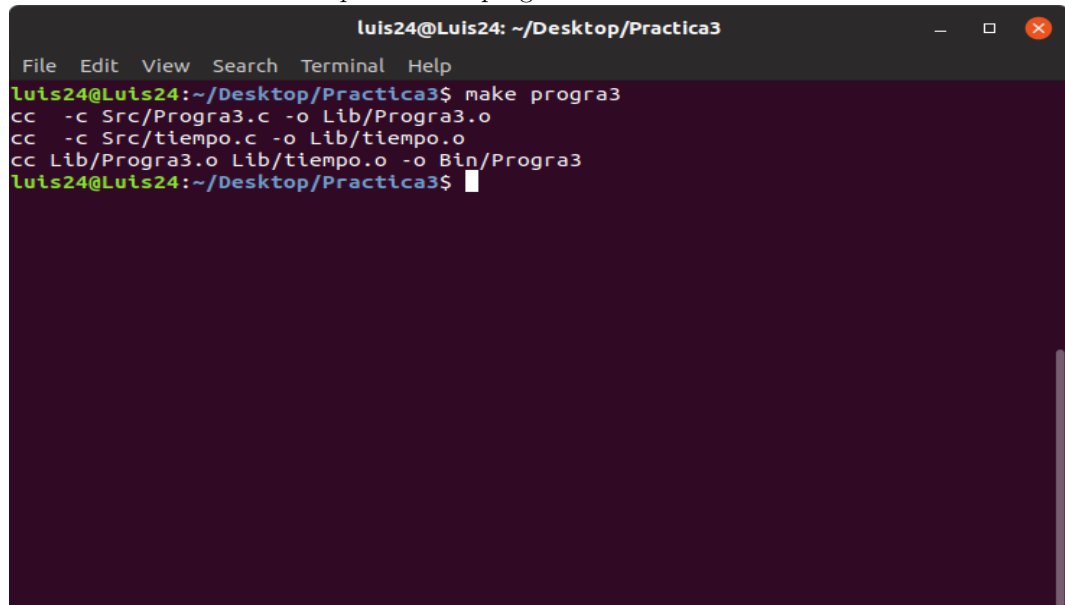
```
luis24@Luis24: ~/Desktop/Practica3
File Edit View Search Terminal Help
luis24@Luis24:~/Desktop/Practica3$ make progra1
cc -c Src/Progra1.c -o Lib/Progra1.o
cc Lib/Progra1.o -o Bin/Progra1
luis24@Luis24:~/Desktop/Practica3$
```

Compilación del programa dos:

A terminal window titled 'luis24@Luis24: ~/Desktop/Practica3' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'make progra2' being executed, which runs three compilation commands: 'cc -c Src/Progra2.c -o Lib/Progra2.o', 'cc -c Src/tiempo.c -o Lib/tiempo.o', and 'cc Lib/Progra2.o Lib/tiempo.o -o Bin/Progra2'. The prompt returns to 'luis24@Luis24:~/Desktop/Practica3\$' with a cursor.

```
luis24@Luis24: ~/Desktop/Practica3
File Edit View Search Terminal Help
luis24@Luis24:~/Desktop/Practica3$ make progra2
cc -c Src/Progra2.c -o Lib/Progra2.o
cc -c Src/tiempo.c -o Lib/tiempo.o
cc Lib/Progra2.o Lib/tiempo.o -o Bin/Progra2
luis24@Luis24:~/Desktop/Practica3$
```


Compilación del programa tres:

A terminal window titled 'luis24@Luis24: ~/Desktop/Practica3' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of 'make progra3', which compiles 'Src/Progra3.c' into 'Lib/Progra3.o', 'Src/tiempo.c' into 'Lib/tiempo.o', and then links them into 'Bin/Progra3'.

```
luis24@Luis24:~/Desktop/Practica3$ make progra3
cc -c Src/Progra3.c -o Lib/Progra3.o
cc -c Src/tiempo.c -o Lib/tiempo.o
cc Lib/Progra3.o Lib/tiempo.o -o Bin/Progra3
luis24@Luis24:~/Desktop/Practica3$
```

Y finalmente se compilara cada uno con los siguientes comandos:

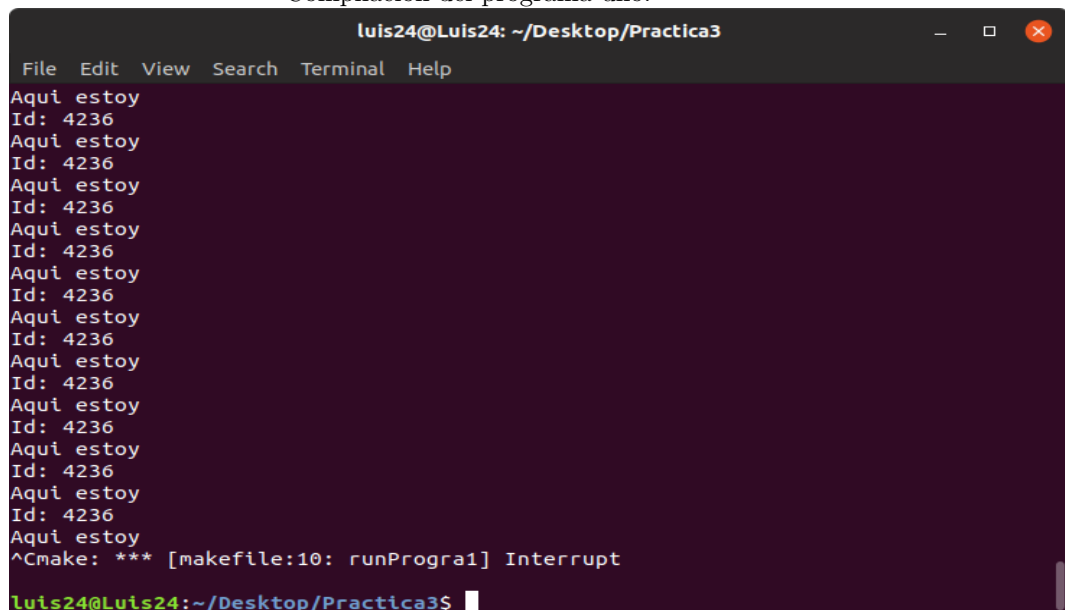
```
make runProgra1
```

```
make runProgra2
```

```
make runProgra3
```

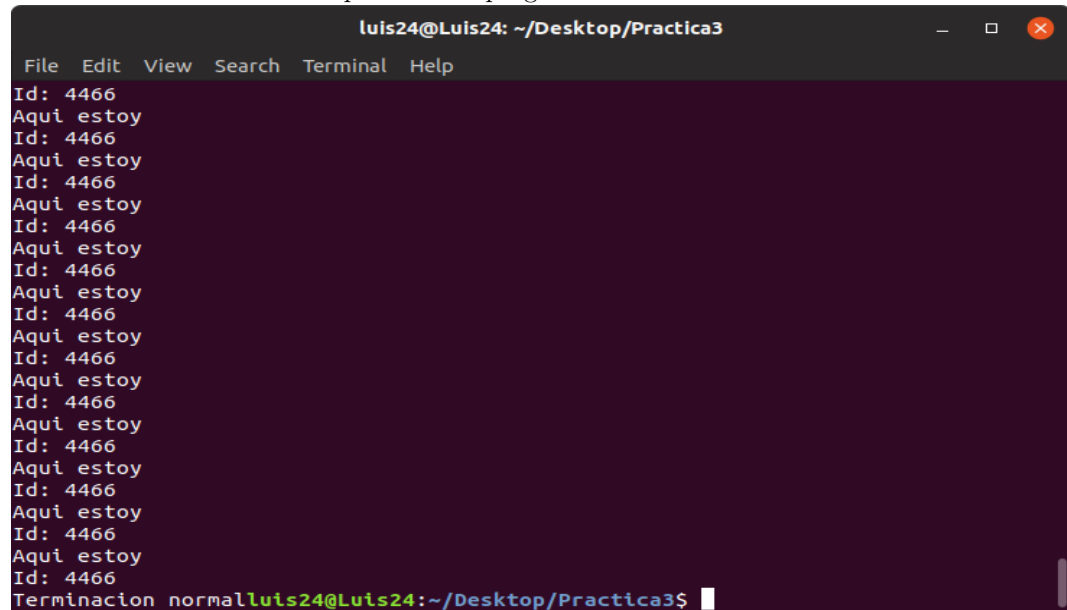
Y se verá de la siguiente forma cada resultado correspondiente:

Compilación del programa uno:

A terminal window titled 'luis24@Luis24: ~/Desktop/Practica3' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of 'make runProgra1', which results in a loop of 'Aqui estoy' and 'Id: 4236' being printed 15 times, followed by an interrupt signal (^C) and the message 'makefile:10: runProgra1] Interrupt'.

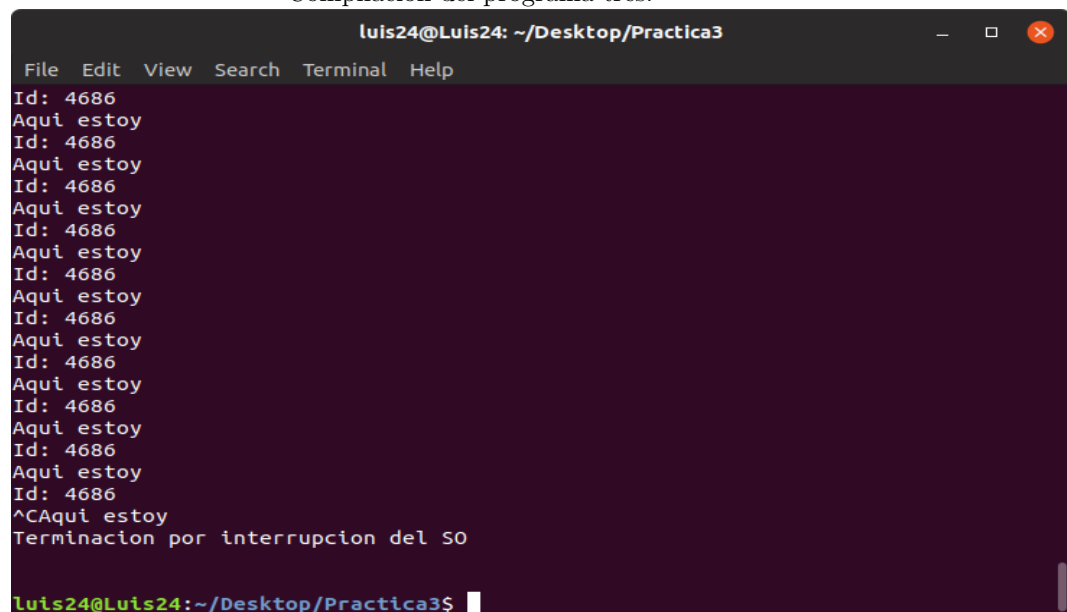
```
luis24@Luis24:~/Desktop/Practica3$ make runProgra1
Aqui estoy
Id: 4236
Aqui estoy
Id: 4236
Aqui estoy
Id: 4236
Aqui estoy
Id: 4236
Aqui estoy
Id: 4236
Aqui estoy
Id: 4236
Aqui estoy
Id: 4236
Aqui estoy
Id: 4236
Aqui estoy
Id: 4236
Aqui estoy
Id: 4236
Aqui estoy
Id: 4236
^Cmake: *** [makefile:10: runProgra1] Interrupt
luis24@Luis24:~/Desktop/Practica3$
```

Compilación del programa dos:



```
luis24@Luis24: ~/Desktop/Practica3
File Edit View Search Terminal Help
Id: 4466
Aqui estoy
Id: 4466
Aqui estoy
Id: 4466
Aqui estoy
Id: 4466
Aqui estoy
Id: 4466
Aqui estoy
Id: 4466
Aqui estoy
Id: 4466
Aqui estoy
Id: 4466
Aqui estoy
Id: 4466
Aqui estoy
Id: 4466
Aqui estoy
Id: 4466
Terminacion normalluis24@Luis24:~/Desktop/Practica3$
```

Compilación del programa tres:



```
luis24@Luis24: ~/Desktop/Practica3
File Edit View Search Terminal Help
Id: 4686
Aqui estoy
Id: 4686
Aqui estoy
Id: 4686
Aqui estoy
Id: 4686
Aqui estoy
Id: 4686
Aqui estoy
Id: 4686
Aqui estoy
Id: 4686
Aqui estoy
Id: 4686
Aqui estoy
Id: 4686
^CAqui estoy
Terminacion por interrupcion del S0
luis24@Luis24:~/Desktop/Practica3$
```

4. Errores

5. Bibliografía

Referencias

- [1] <https://www.howtogeek.com/105413/how-to-compile-and-install-from-source-on-ubuntu/>
- [2] <https://askubuntu.com/questions/271388/how-to-install-gcc-4-8>
- [3] <https://askubuntu.com/questions/271388/how-to-install-gcc-4-8>
- [4] <https://es.wikipedia.org/wiki/Signal.h>
- [5] <http://docs.mis-algoritmos.com/c.funcion.localtime.html>
- [6] <https://sites.google.com/site/sogrupop15/llamada-getpid>