

**Escuela Superior de Cómputo  
Instituto Politécnico Nacional**

**Sistemas Operativos**

**Reporte Práctica 3**

---

Alumno:

González Barrios Alan Ernesto

2CM7

March 21, 2019

March 21, 2019

## 1 Índice

Explicación de la teoría detras de la práctica	3
Explicación de la lógica de pogramación y sáldas obtenidas	3
Listado del código, sáldas a consola y pantallas de resultado	4
Errores y problemas encontrados y como fueron resueltos	9
Bibliografía	9

## 2 Explicación de la teoría detras de la práctica

Una señal es un "aviso" que puede enviar un proceso a otro proceso. El sistema operativo unix se encarga de que el proceso que recibe la señal la trate inmediatamente. De hecho, termina la línea de código que esté ejecutando y salta a la función de tratamiento de señales adecuada. Cuando termina de ejecutar esa función de tratamiento de señales, continua con la ejecución en la línea de código donde lo había dibujado.

El sistema operativo envía señales a los procesos en determinadas circunstancias. Por ejemplo, si en el programa que se está ejecutando en una shell nosotros apretamos Ctrl-C, se está enviando una señal de terminación al proceso. Este la trata inmediatamente y sale. Si nuestro programa intenta acceder a una memoria no válida (por ejemplo, accediendo al contenido de un puntero a NULL), el sistema operativo detecta esta circunstancia y le envía una señal de terminación inmediata, con lo que el programa "se cae".

## 3 Explicación de la lógica de programación y las salidas obtenidas

Se tenia que crear un programa en C el cual consistia en que este tenia que imprimir de manera infinita dentro de un bucle la cadena de que decia: "Aqui estoy" pero como se comento en las clases previas de sistemas operativos acerca del arbol de procesos cada proceso tiene un padre y estoy estan organizados de manera organizada dentro de este arbol, con el objetivo de poder vizulaizar el numero de identificacion del proceso que se le asigno a este.

Para poder vizualizarlo en el comando printf se le manda como parametro de impresion la funcion getpid() el cual obtiene el process id del respectivo proceso. Una vez habiando terminado esto se tenia que modificar el programa para que cuando pasaran 30 segundos se tenia que terminar el programa escribiendo en pantalla la cadena "Terminacion Normal". Para poder lograr este cometido se tuvo que implementar el uso de la funcion alarm() y a este se le tenia que asignar el parametro de 30 segundos y como funciona esta funcion es que al parametro en segundos que se le mande este lo ira decrementado y para cuando este acabe manda de regreso una señal SIGALRM la cual se asocio con con la funcion terminar bucle mediante la funcion signal.

Para el siguiente paso consistia en que cuando detectara un Ctl+C para ello se tuvo que volver a utilizar la funcion signal dado que cuando se presiona un Ctl+c, se genera una señal SIGINT y esta se asocio con otra funcion que se implemento en el momento llamada SO (de sistema operativo) la cual tiene la funcion de detectar esta señal y manda a imprimir en pantalla una cadena que mencione que hubo una interrupcion del SO y se cierre el programa.

## 4 Listado del código, salidas a consola y pantallas de resultado

El siguiente código corresponde al programa desarrollado en C, el cual tiene como función imprimir de manera infinita una cadena con su id correspondiente.

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <signal.h>
4
5 int main ()
6 {
7     /* se crea el bucle infinito mientras el valor del entero bucle sea
8        igual a 1
9        while (1)
10    /* mientras se esta en el bucle se estara imprimiendo en pantalla
11       el texto aqui estoy con su respectivo numero de proceso
12    printf("Process id (%d): Aqui estoy\n", getpid());
13    return 0;
14 }
```

Listing 1: Programa id de procesos y detección de señales del SO

El siguiente código es una modificación del código previo en la parte de que se le agrego una alarma la cual servirá de temporizador y para cuando haya transcurrido el tiempo este acabará el programa

```
1
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <signal.h>
5 /* se declaran los prototipos de las funciones que se ocuparan
6 void terminar_bucle();
7 /* declaracion de variable global para el bucle de impresion
8 int bucle=1;
9 /* inicia el main
10 int main ()
11 {
12     int bucle=1;
13     /* se asocia la se al que se genera cuando la funcion alarm
14        termina de decrementar los 30 segundos
15     signal (SIGALRM, terminar_bucle);
16     /* se inicializa la funcion alarm con parametro de 30 segundos el
17        cual servira de temporizador
18     alarm(30);
19     /* se crea el bucle infinito mientras el valor del entero bucle sea
20        igual a 1
21     while (bucle==1)
22     /* mientras se esta en el bucle se estara imprimiendo en pantalla
23        el texto aqui estoy con su respectivo numero de proceso
24     printf("Process id (%d): Aqui estoy\n", getpid());
25     return 0;
26 }
```

```

25 //! se declara la funcion a la cual esta asociada la se al de
    alarm cuando pasen los 30 segundos
26 void terminar_bucle() {
27 //! se desactiva la deteccion de la se al alarm
28     signal (SIGALRM, SIG_IGN);
29 //! se cambia el valor del entero bucle a 0 para salir del bucle
    infinito en el main esto cuando ya terminaron de pasar los 30
    segundos
30     bucle=0;
31 //! se imprime la cadena para mostrar que pasaron los 30 seg
32     printf ("Terminaci n normal\n");
33 //! se sale del programa
34     exit(0);
35 }

```

Listing 2: Programa id de procesos y deteccion de señales del SO

El siguiente codigo corresponde al programa desarrollado en C, el cual cuando detecta una señal de interrupcion del S.O. (Ctl+c) termina el programa.

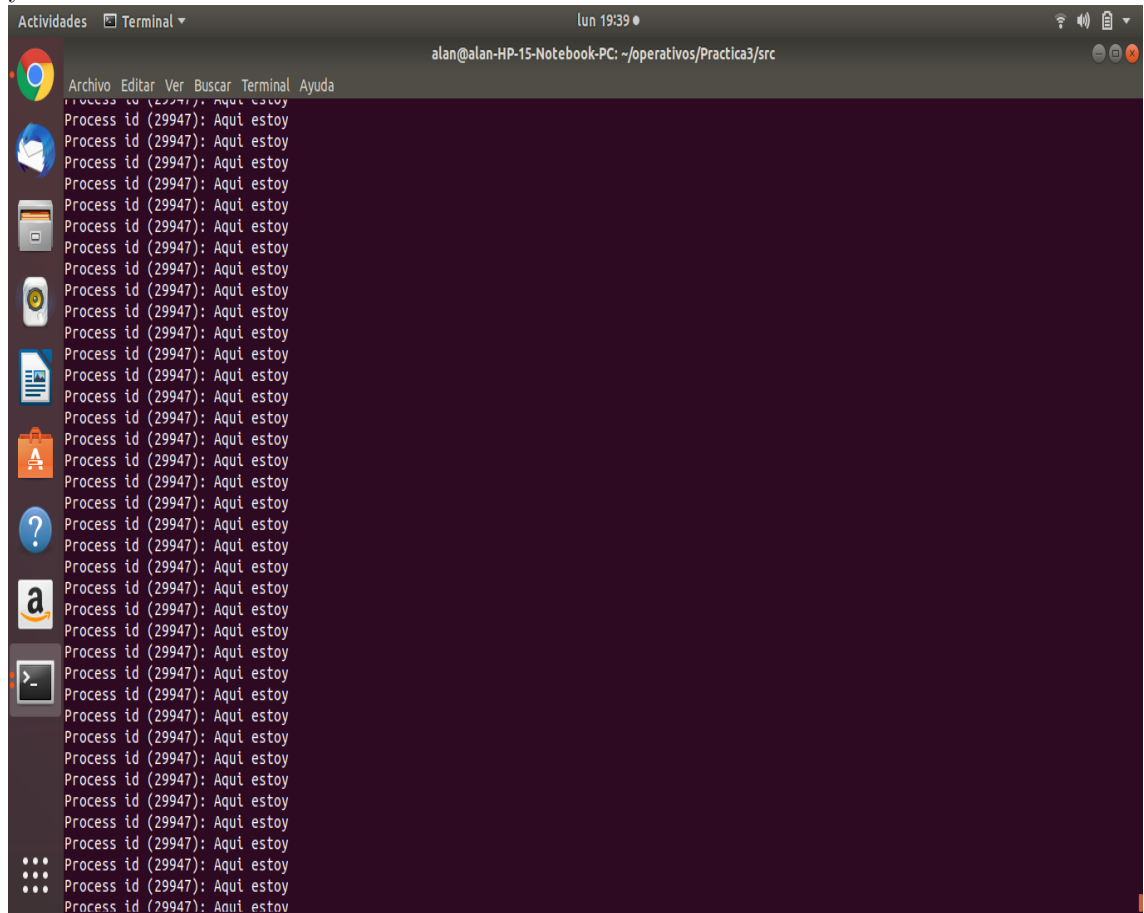
```

1
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <signal.h>
5 //! se declaran los prototipos de las funciones que se ocuparan
6 void SO();
7 //! inicia el main
8 int main ()
9 {
10 //! se asocia la se al que se genera cuando se presiona Ctl+C con
    la funcion SO
11     signal (SIGINT, SO);
12 //! se crea el bucle infinito mientras el valor del entero bucle sea
    igual a 1
13     while (1)
14 //! mientras se esta en el bucle se estara imprimiendo en pantalla
    el texto aqui estoy con su respectivo numero de proceso
15     printf("Process id (%d): Aqui estoy\n", getpid());
16 //! se desactiva la deteccion de la se al de Ctl+c
17     signal (SIGINT, SIG_IGN);
18     return 0;
19 }
20
21
22 //! se inicializa la funcion SO que esta asociada a la se al del
    Ctl+C
23 void SO()
24 {
25 //! se desactiva la se al de Ctl+C
26     signal (SIGINT, SIG_IGN);
27 //! una vez que se detecto el Ctl+C se imprimira la cadena
    interrupcion
28     printf ("Terminaci n por interrupci on del S.O.\n");
29 //! termina el programa
30     exit(0);
31 }

```

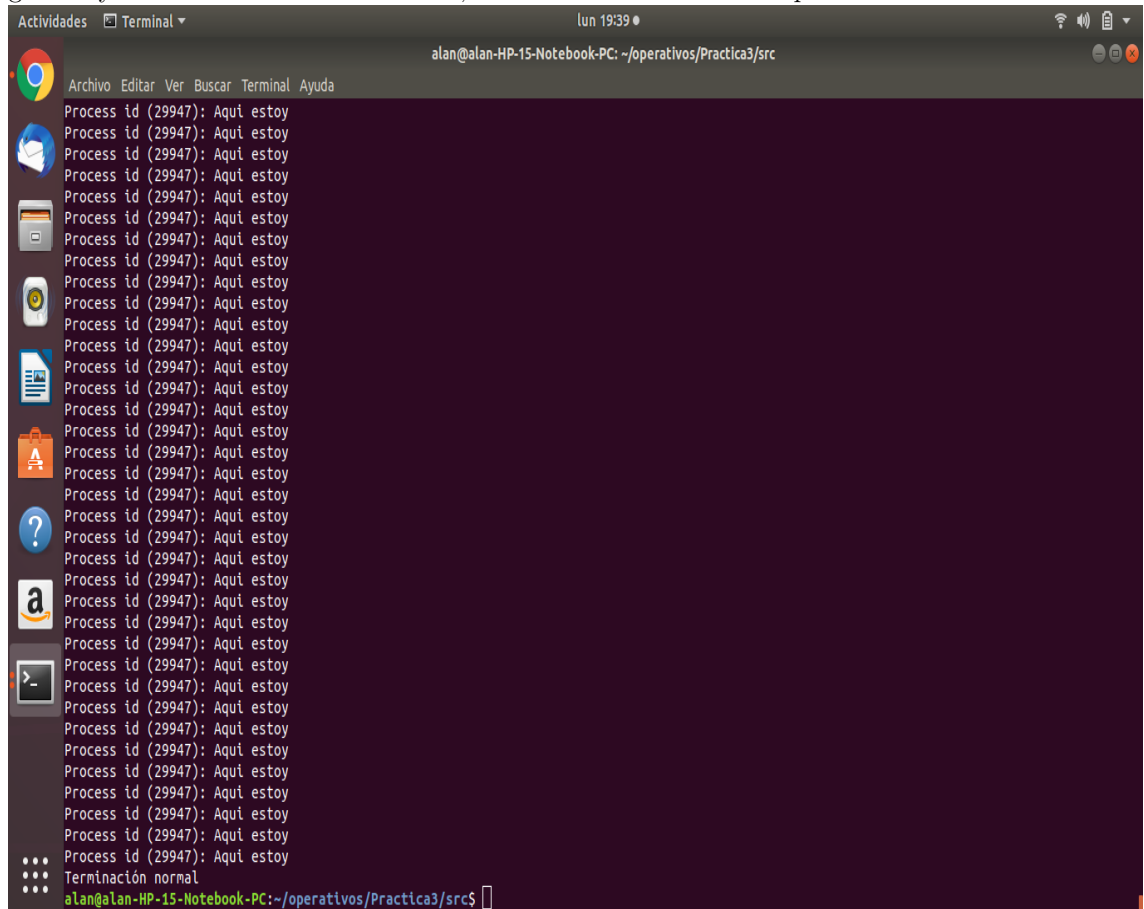
Listing 3: Programa id de procesos y deteccion de señales del SO

Impresion de pantalla de la salida del programa cuando el bucle es infinito y se muestra en consola n cantidad de cadenas



The screenshot shows a Linux terminal window titled "Terminal" with a menu bar containing "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The window's title bar includes the text "alan@alan-HP-15-Notebook-PC: ~/operativos/Practica3/src" and system icons for network, volume, and battery. The terminal displays a continuous stream of text: "Process id (29947): Aqui estoy". On the left side of the terminal, a vertical dock contains icons for various applications, including Google Chrome, a mail client, a file manager, a terminal, a calculator, a web browser, a help icon, and an Amazon logo. The terminal's output is truncated at the bottom, showing "Process id (29947): Aqui estov".

Impresion de pantalla de la salida del programa con temporizador de 30 segundos y han terminado de transcurrir, mostrando la cadena correspondiente



The screenshot shows a Linux desktop environment with a terminal window open. The terminal title bar indicates the user is 'alan' on a machine named 'alan-HP-15-Notebook-PC', and the current directory is '~/operativos/Practica3/src'. The terminal output consists of 25 lines, each displaying 'Process id (29947): Aqui estoy'. After the 25th line, the text 'Terminación normal' is printed. The prompt 'alan@alan-HP-15-Notebook-PC:~/operativos/Practica3/src\$' is visible at the bottom of the terminal. The desktop background is dark purple, and the left sidebar contains various application icons including Chrome, a mail client, a file manager, and others.

```
alan@alan-HP-15-Notebook-PC: ~/operativos/Practica3/src
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Process id (29947): Aqui estoy
Terminación normal
alan@alan-HP-15-Notebook-PC:~/operativos/Practica3/src$
```

A screenshot of a Linux terminal window. The title bar at the top shows 'Actividades' and 'Terminal'. The terminal text shows a user 'alan' at a machine 'alan-HP-15-Notebook-PC' in the directory '~/operativos/Practica3/src'. The output is a continuous stream of 'Process id (30118): Aqui estoy' messages. The window's left sidebar contains various application icons, and the top status bar shows system icons for network, volume, and battery.

1	Programa id de procesos y deteccion de señales del SO . . . . .	4
2	Programa id de procesos y deteccion de señales del SO . . . . .	4
3	Programa id de procesos y deteccion de señales del SO . . . . .	5



## 5 Errores y problemas encontrados y como fueron resueltos

Realmente no se encontraron errores, lo unico que si hubo duda fue en que forma era la correcta para el temporizador, con ello me refiero a si era mejor implementarlo con la biblioteca `time.h` o con las alarmas esta bien, de cualquier modo el resultado hubiera sido el mismo.

## 6 Bibliografía

Apuntes en clase

<http://www.chuidiang.org/clinux/senhales/senhales.php>

<https://diegolog.wordpress.com/2010/05/14/capturar-sigint-senal-de-interrupcion-ctrlc-cc/>