



Instituto Politecnico Nacional

Escuela Superior de Cómputo



Práctica 4

Administrador de procesos en Linux y Windows (2)

Sistemas Operativos

Grupo: 2CM12

Integrantes:

- ⇒ Baldovinos Gutiérrez Kevin
- ⇒ Bocanegra Heziquio Yestlanezi
- ⇒ Castañares Torres Jorge David
- ⇒ Hernández Hernández Rut Esther

Profesor
Jorge Cortes Galicia



Contenido

| | |
|---|----------|
| OBJETIVO | 3 |
| INTRODUCCIÓN TEÓRICA | 3 |
| DESARROLLO EXPERIMENTAL | 4 |
| Proceso en Linux | 8 |
| Proceso Windows Hilos..... | 10 |
| TIEMPO DE EJECUCIÓN EN LINUX PROCESOS POR HILOS..... | 12 |
| Tiempo de ejecución con hilos | 13 |
| WINDOWS | 14 |
| LINUX PROCESOS POR HILOS..... | 15 |
| LINUX..... | 17 |
| COPIA EXACTA DE PROCESOS | 17 |
| 1.-CÓDIGO ECHO EN WINDOWS POR HILOS..... | 20 |
| 2-CODIGO HECHO EN WINDOWS POR COPIA EXACTA DE CÓDIGO..... | 21 |
| CONCLUSIONES..... | 26 |
| REFERENCIAS | 27 |
| Imagen 1 Consola Linux | 5 |
| Imagen 2 Consola Linux desde el hilo | 6 |
| Imagen 3 Consola Windows | 7 |
| Imagen 4 Proceso en linux | 8 |
| Imagen 5 Consola Linux Procesos | 9 |
| Imagen 6 Procesos desde Windows Hilos..... | 10 |

OBJETIVO

El alumno aprende a familiarizarse con el administrador de procesos del sistema operativo Linux y Windows a través de la creación de procesos ligeros (hilos) para el desarrollo de aplicaciones concurrentes sencillas.

INTRODUCCIÓN TEÓRICA

Administrador de procesos.

Un administrador de procesos es un programa de cómputo que se utiliza para proporcionar información sobre los procesos y programas que se están activos en la computadora, esta aplicación se utiliza para detener o comúnmente matar procesos.

Un proceso podría ser una instancia de un programa en ejecución. A los procesos frecuentemente se les refiere como tareas. El contexto de un programa que está en ejecución es lo que se llama un proceso. Linux, es un sistema operativo multitarea y multiusuario. Esto quiere decir que múltiples procesos pueden operar simultáneamente sin interferirse unos con los otros. Cada proceso tiene la "ilusión" que es el único proceso en el sistema y que tiene acceso exclusivo a todos los servicios del sistema operativo.

Programas y procesos son entidades distintas, múltiples instancias de un programa pueden ejecutarse simultáneamente. Cada instancia es un proceso separado. Por ejemplo, si usuarios desde equipos diferentes, ejecutan el mismo programa al mismo tiempo, habría tantas instancias del mismo programa, es decir, procesos distintos.

Cada proceso que se inicia es referenciado con un número de identificación único conocido como Process ID PID, que es siempre un entero positivo. Prácticamente todo lo que se está ejecutando en el sistema en cualquier momento es un proceso, incluyendo el shell, el ambiente gráfico que puede tener múltiples procesos, etc. La excepción a lo anterior es el kernel en sí, el cual es un conjunto de rutinas que residen en memoria y a los cuales los procesos a través de llamadas al sistema pueden tener acceso

DESARROLLO EXPERIMENTAL

1. A través de la ayuda en línea que proporciona Linux, investigue el funcionamiento de las funciones: `pthread_create()`, `pthread_join()`, `pthread_self()`, `pthread_exit()`, `scandir()`, `stat()`. Explique los argumentos y retorno de cada función.

- **`pthread_create()`**

La función `pthread_create ()` inicia un nuevo hilo en la llamada proceso. El nuevo hilo comienza la ejecución invocando `start_routine ()`; `arg` se pasa como el único argumento de `start_routine ()`.

- **`pthread_join()`**

La función `pthread_join ()` espera a que termine el hilo especificado por `hilo`. Si ese hilo ya ha terminado, `pthread_join ()` regresa inmediatamente. El hilo especificado por `hilo` debe poder unirse.

- **`pthread_self()`**

La función `pthread_self ()` devuelve el ID del hilo de llamada. Este es el mismo valor que se devuelve en `* hilo` en la llamada `pthread_create (3)` que creó este hilo.

- **`pthread_exit()`**

La función `pthread_exit ()` termina el hilo de llamada y devuelve un valor a través de `retval` que (si el hilo se puede unir) está disponible para otro hilo en el mismo proceso que llama a `pthread_join (3)`.

- **`scandir()`**

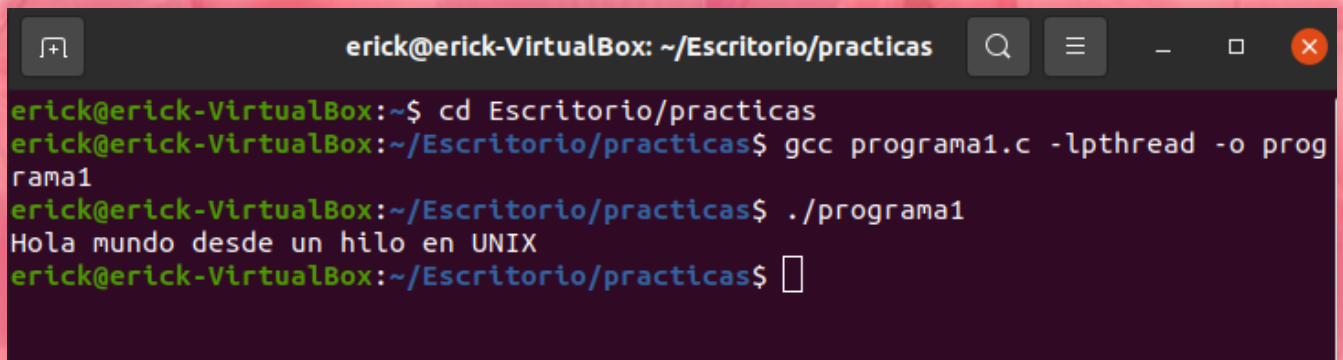
La función `scandir ()` escanea el directorio `dirp`, llamando a `filter ()` en cada entrada del directorio.

- **`stat()`**

Muestra el archivo o el estado del sistema de archivos.

2. Capture, compile y ejecute el programa de creación de un nuevo hilo en Linux. Observe su funcionamiento.

```
#include <stdio.h>
#include <pthread.h>
void *hilo(void *arg);
int main(void)
{
    pthread_t id_hilo;
    pthread_create(&id_hilo, NULL, (void*)hilo, NULL);
    pthread_join(id_hilo, NULL);
    return 0;
}
void *hilo(void *arg)
{
    printf("Hola mundo desde un hilo en UNIX\n");
    return NULL;
}
```



```
erick@erick-VirtualBox: ~/Escritorio/practicas
erick@erick-VirtualBox:~$ cd Escritorio/practicas
erick@erick-VirtualBox:~/Escritorio/practicas$ gcc programa1.c -lpthread -o programa1
erick@erick-VirtualBox:~/Escritorio/practicas$ ./programa1
Hola mundo desde un hilo en UNIX
erick@erick-VirtualBox:~/Escritorio/practicas$
```

Imagen 1 Consola Linux

3. Capture, compile y ejecute el siguiente programa de creación de hilos en Linux. Observe su funcionamiento.

```
#include <stdio.h>
#include <pthread.h>
void *hilo(void *arg)
int main(void)
{
    pthread_t id_hilo;
    char* mensaje="Hola a todos desde el hilo";
    int devolucion_hilo;
    pthread_create(&id_hilo,NULL,hilo,(void*)mensaje);
    pthread_join(id_hilo,(void*)&devolucion_hilo);
    printf("valor = %i\n",devolucion_hilo)
    return 0;
}

void *hilo(void *arg)
{
    char* men;
    int resultado_hilo=0;
    men=(char*)arg;
    printf("%s\n",men);
    resultado_hilo=100;
    pthread_exit((void*)resultado_hilo);
}
```

```
erick@erick-VirtualBox:~/Escritorio/practicas$ ./programa2
Hola a todos desde el hilo
valor=100
erick@erick-VirtualBox:~/Escritorio/practicas$
```

Imagen 2 Consola Linux desde el hilo

4. Capture, compile y ejecute el siguiente programa de creación de hilos en Windows. Observe su funcionamiento.

```
#include <windows.h>
#include <stdio.h>
DWORD WINAPI funcionHilo(LPVOID lpParam);
typedef struct Informacion info;
struct Informacion
{
    int val_1;
    int val_2;
};

int main(void)
{
    DWORD idHilo;           /* Identificador del Hilo */
    HANDLE manHilo;         /* Manejador del Hilo */
    info argumentos;
    argumentos.val_1=10;
    argumentos.val_2=100;

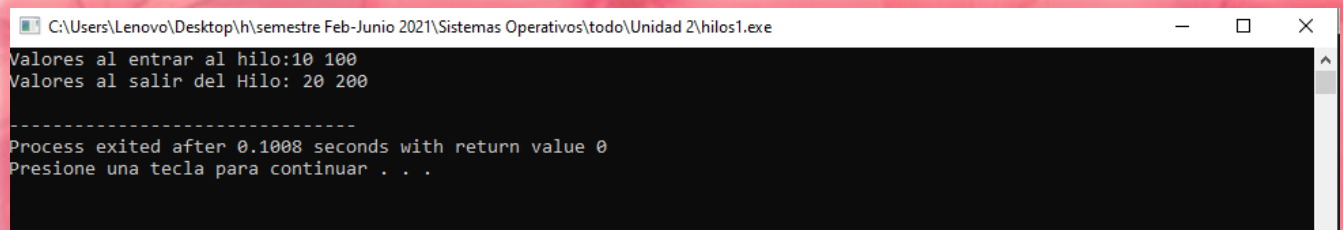
    // Creacion del hilo
    manHilo=CreateThread(NULL, 0, funcionHilo, &argumentos, 0, &idHilo);

    // Espera la finalización del hilo
    WaitForSingleObject(manHilo, INFINITE);

    printf("Valores al salir del Hilo: %i %i\n", argumentos.val_1, argumentos.val_2);

    // Cierre del manejador del hilo creado
    CloseHandle(manHilo);
    return 0;
}

DWORD WINAPI funcionHilo(LPVOID lpParam)
{
    info *datos=(info *)lpParam;
    printf("Valores al entrar al Hilo: %i %i\n", datos->val_1, datos->val_2);
    datos->val_1*=2;
    datos->val_2*=2;
    return 0;
}
```



The screenshot shows a Windows command prompt window titled "C:\Users\Lenovo\Desktop\h\semestre Feb-Junio 2021\Sistemas Operativos\todo\Unidad 2\hilos1.exe". The output of the program is displayed in the console:

```
Valores al entrar al hilo:10 100
Valores al salir del Hilo: 20 200

-----
Process exited after 0.1008 seconds with return value 0
Presione una tecla para continuar . . .
```

Imagen 3 Consola Windows

- 5.- Programe una aplicación (tanto en Linux como en Windows), que cree un proceso hijo a partir de un proceso padre, el hijo creado a su vez creará 15 hilos. A su vez cada uno de los 15 hilos creará 10 hilos más. A su vez cada uno de los 10 hilos creará 5 hilos más. Cada uno de los hilos creados imprimirá en pantalla “Práctica 2” si se trata de un hilo terminal o los identificadores de los hilos creados si se trata de un proceso o hilo padre.

Proceso en Linux

```
1#include<stdio.h>
2#include<pthread.h>
3
4void*hilo(void *arg);
5
6int main(void){
7    pthread_t id_hilo;
8    int v=4;
9    pthread_create(&id_hilo, NULL,hilo, &v);
10   pthread_join(id_hilo, NULL);
11   return 0;
12}
13
14void *hilo(void *arg){
15    pthread_t id_hilo;
16    int i, val=((int*) arg);
17    val-=1;
18    if(val ==3){
19        for(int i=0; i<15; i++){
20            pthread_create(&id_hilo, NULL,hilo, &val);
21            pthread_join(id_hilo, NULL);
22            printf("Practica2 hijo del hijo Principal Proceso 1\n\n");
23        }
24    }
25    if(val == 2){
26        for(int i=0; i<10; i++){
27            pthread_create(&id_hilo, NULL,hilo, &val);
28            pthread_join(id_hilo, NULL);
29            printf("Practica2 hijo 1 Proceso 2\n");
30        }
31    }
32
33    if(val == 1){
34        for(int i=0; i<5; i++){
35            printf("3 Practica2 hijo de 2 Proceso terminal\n");
36        }
37    }
38    return NULL;
39 }
40 }
```

Imagen 4 Proceso en linux


```
erick@erick-VirtualBox:~$ cd Escritorio/practicas
erick@erick-VirtualBox:~/Escritorio/practicas$ gcc programa6.c -lpthread -o programa6
erick@erick-VirtualBox:~/Escritorio/practicas$ ./programa6
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
Practica 2 Hijo 1 Proceso 2
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
Practica 2 Hijo 1 Proceso 2
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
Practica 2 Hijo 1 Proceso 2
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
Practica 2 Hijo 1 Proceso 2
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
Practica 2 Hijo 1 Proceso 2
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
3 Practica2 hijo de 2 Proceso terminal
```

Imagen 5 Consola Linux Procesos

```
1  #include <windows.h>
2  #include <stdio.h>
3  DWORD WINAPI quinceHilos(LPVOID lpParam1);
4  DWORD WINAPI diezHilos(LPVOID lpParam2);
5  DWORD WINAPI cincoHilos(LPVOID lpParam3);
6  int main(){
7      DWORD idHilo1;
8      HANDLE manHilo1;
9      printf("idHilo: %i\n",GetCurrentThreadId());
10     for(int i=0;i<15;i++){
11         manHilo1=CreateThread(NULL,0,quinceHilos,NULL,0,&idHilo1);
12         WaitForSingleObject(manHilo1,INFINITE);
13         CloseHandle(manHilo1);
14     }
15     return 0;
16 }
17 DWORD WINAPI quinceHilos(LPVOID lpParam1){
18     DWORD idHilo2;
19     HANDLE manHilo2;
20     printf("\tidHilo: %i\n",GetCurrentThreadId());
21     for(int i=0;i<10;i++){
22         manHilo2=CreateThread(NULL,0,diezHilos,NULL,0,&idHilo2);
23         WaitForSingleObject(manHilo2,INFINITE);
24         CloseHandle(manHilo2);
25     }
26     return 0;
27 }
28 DWORD WINAPI diezHilos(LPVOID lpParam2){
29     DWORD idHilo3;
30     HANDLE manHilo3;
31     printf("\t\tidHilo: %i\n",GetCurrentThreadId());
32     for(int i=0;i<5;i++){
33         manHilo3=CreateThread(NULL,0,cincoHilos,NULL,0,&idHilo3);
34         WaitForSingleObject(manHilo3,INFINITE);
35         CloseHandle(manHilo3);
36     }
37     return 0;
38 }
39 DWORD WINAPI cincoHilos(LPVOID lpParam3){
40     printf("\t\t\tPractica 2 \n");
41     public int __cdecl printf (const char * __restrict __Format, ...)
42 }
```

Imagen 6 Procesos desde Windows Hilos


```
C:\Users\Lenovo\Desktop\h\semestre Feb-Junio 2021\Sistemas Operativos\todo\Unidad 2\hilos.exe
idHilo: 2832
  idHilo: 1084
    idHilo: 7988
      Practica 2
      Practica 2
      Practica 2
      Practica 2
      Practica 2
    idHilo: 7016
      Practica 2
      Practica 2
      Practica 2
      Practica 2
      Practica 2
  idHilo: 6564
    Practica 2
    Practica 2
    Practica 2
    Practica 2
    Practica 2
  idHilo: 5056
    Practica 2
    Practica 2
    Practica 2
    Practica 2
    Practica 2
  idHilo: 2652
    Practica 2
    Practica 2
    Practica 2
```

6. Programe la misma aplicación del punto 5 de la práctica 3 pero utilizando hilos (tanto en Linux como en Windows) en vez de procesos. Compare ambos programas (el creado en la práctica 1 y el creado en esta práctica) y dé sus observaciones tanto de funcionamiento como de los tiempos de ejecución resultantes.

TIEMPO DE EJECUCIÓN EN LINUX PROCESOS POR HILOS

```
erick@erick-VirtualBox:~$ cd Escritorio
erick@erick-VirtualBox:~/Escritorio$ ./programa_hilos
erick@erick-VirtualBox:~/Escritorio$
```

Resultado de la Suma

```
2 4 6 8 10 12 14
14 12 10 8 6 4 2
6 4 2 8 10 12 14
14 12 10 2 4 6 8
12 6 10 4 8 2 14
14 8 10 2 4 6 12
2 4 6 8 10 12 14
```

Resultado de la Resta

```
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
```

Resultado de la multiplicacion

```
1 4 9 16 25 36 49
49 36 25 16 9 4 1
9 4 1 16 25 36 49
49 36 25 1 4 9 16
36 9 25 4 16 1 49
49 16 25 1 4 9 36
1 4 9 16 25 36 49
```

Resultado de la transpuesta

```
1 2 3 4 5 6 7
7 6 5 4 3 2 1
3 2 1 4 5 6 7
7 6 5 1 2 3 4
6 3 5 2 4 1 7
7 4 5 1 2 3 6
1 2 3 4 5 6 7
```

```
1 2 3 4 5 6 7
7 6 5 4 3 2 1
3 2 1 4 5 6 7
7 6 5 1 2 3 4
6 3 5 2 4 1 7
7 4 5 1 2 3 6
1 2 3 4 5 6 7
```


Tiempo de ejecución con hilos

```
erick@erick-VirtualBox:~$ cd Escritorio
erick@erick-VirtualBox:~/Escritorio$ ./programa
erick@erick-VirtualBox:~/Escritorio$
```

Resultado de la Suma

```
2 4 6 8 10 12 14
14 12 10 8 6 4 2
6 4 2 8 10 12 14
14 12 10 2 4 6 8
12 6 10 4 8 2 14
14 8 10 2 4 6 12
2 4 6 8 10 12 14
```

Resultado de la Resta

```
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
```

Resultado de la multiplicacion

```
1 4 9 16 25 36 49
49 36 25 16 9 4 1
9 4 1 16 25 36 49
49 36 25 1 4 9 16
36 9 25 4 16 1 49
49 16 25 1 4 9 36
1 4 9 16 25 36 49
```

Resultado de la transpuesta

```
1 2 3 4 5 6 7
7 6 5 4 3 2 1
3 2 1 4 5 6 7
7 6 5 1 2 3 4
6 3 5 2 4 1 7
7 4 5 1 2 3 6
1 2 3 4 5 6 7
```

```
1 2 3 4 5 6 7
7 6 5 4 3 2 1
3 2 1 4 5 6 7
7 6 5 1 2 3 4
6 3 5 2 4 1 7
7 4 5 1 2 3 6
1 2 3 4 5 6 7
```

WINDOWS

Tiempo de ejecución en Windows usando hilos

```
C:\Users\germt\Desktop\so>gcc PW6.c -o p6
```

```
C:\Users\germt\Desktop\so>p6.exe
```

```
Resultado del 1a Suma
```

```
2 4 6 8 10 12 14
14 12 10 8 6 4 2
6 4 2 8 10 12 14
14 12 10 2 4 6 8
12 6 10 4 8 2 14
14 8 10 2 4 6 12
2 4 6 8 10 12 14
```

```
Resultado del 1a Resta
```

```
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
```

```
Resultado del 1a multiplicacion
```

```
1 4 9 16 25 36 49
49 36 25 16 9 4 1
9 4 1 16 25 36 49
49 36 25 1 4 9 16
36 9 25 4 16 1 49
49 16 25 1 4 9 36
1 4 9 16 25 36 49
```

```
Resultado del 1a Transpuesta
```

```
1 7 3 7 6 7 1
2 6 2 6 3 4 2
3 5 1 5 5 5 3
4 4 4 1 2 1 4
5 3 5 2 4 2 5
6 2 6 3 1 3 6
7 1 7 4 7 6 7
```

```
1 7 3 7 6 7 1
2 6 2 6 3 4 2
3 5 1 5 5 5 3
4 4 4 1 2 1 4
5 3 5 2 4 2 5
6 2 6 3 1 3 6
7 1 7 4 7 6 7
```

```
Tiempo de ejecución: 0.028000 Segundos
C:\Users\germt\Desktop\so>
```

```
7 1 7 4 7 6 7
```

```
Tiempo de ejecución: 0.028000 Segundos
```

```
C:\Users\germt\Desktop\so>
```

Tiempo de ejecución usando copia exacta de código en Windows

```
=== Suma ===
```

```
11 9 8 7 6 5 4 3 2 1
1 11 8 7 6 5 4 3 2 1
1 2 11 7 6 5 4 3 2 1
1 2 3 11 6 5 4 3 2 1
1 2 3 4 11 5 4 3 2 1
1 2 3 4 5 11 4 3 2 1
1 2 3 4 5 6 11 3 2 1
1 2 3 4 5 6 7 11 2 1
1 2 3 4 5 6 7 8 11 1
1 2 3 4 5 6 7 8 9 11
```

```
=== Resta ===
```

```
-9 -9 -8 -7 -6 -5 -4 -3 -2 -1
1 -7 -8 -7 -6 -5 -4 -3 -2 -1
1 2 -5 -7 -6 -5 -4 -3 -2 -1
1 2 3 -3 -6 -5 -4 -3 -2 -1
1 2 3 4 -1 -5 -4 -3 -2 -1
1 2 3 4 5 1 -4 -3 -2 -1
1 2 3 4 5 6 3 -3 -2 -1
1 2 3 4 5 6 7 5 -2 -1
1 2 3 4 5 6 7 8 7 -1
1 2 3 4 5 6 7 8 9 9
```

```
=== Multiplicacion ===
```

```
=== Traspuesta ===
```

```
1 1 1 1 1 1 1 1 1 1
0 2 2 2 2 2 2 2 2 2
0 0 3 3 3 3 3 3 3 3
0 0 0 4 4 4 4 4 4 4
0 0 0 0 5 5 5 5 5 5
0 0 0 0 0 6 6 6 6 6
0 0 0 0 0 0 7 7 7 7
0 0 0 0 0 0 0 8 8 8
0 0 0 0 0 0 0 0 9 9
0 0 0 0 0 0 0 0 0 10
```

```
----El tiempo de ejecucion fue de: 0.171000
```

```
-----
Process exited after 0.4354 seconds with return value 0
Presione una tecla para continuar . . .
```


LINUX PROCESOS POR HILOS

```
1 #include<stdio.h>
2 #include<pthread.h>
3 #include<unistd.h>
4 #include<stdlib.h>
5 #include<sys/wait.h>
6 #include<fcntl.h>
7 #include<string.h>
8
9 void *hilo(void *arg);
10 int File1;
11 int matriz1[7][7] = {{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,2,1,4,5,6,7},{7,6,5,1,2,3,4},{6,3,5,2,4,1,7},{7,4,5,1,2,3,6},{1,2,3,4,5,6,7}};
12 int matriz2[7][7] = {{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,2,1,4,5,6,7},{7,6,5,1,2,3,4},{6,3,5,2,4,1,7},{7,4,5,1,2,3,6},{1,2,3,4,5,6,7}};
13
14 int asn[7][7],cont=0, l, j,k,i;
15 char ll[100];
16
17 ssize_t nr_bytes;
18
19 int main(void){
20     pthread_t id_hilo;
21     int v;
22     for(i=0; i<5; i++){
23         v=i;
24         pthread_create(&id_hilo, NULL,hilo, &v);
25         pthread_join(id_hilo, NULL);
26     }
27     return 0;
28 }
29
30 void *hilo(void *arg){
31     pthread_t id_hilo;
32     int i, val=((int*) arg);
33     char answer[400];
34
35     if(val == 0){
36         for(i=0; i<400; i++){
37             answer[i]=0;
38         }
39         for(k=0; k<7; k++){
40             for(j=0; j<7; j++){
41                 l = matriz2[k][j] + matriz1[k][j];
42                 sprintf(ll, "%d",l);
43                 strcat(answer, ll);
44                 strcat(answer, " ");
45
46                 if(l>=0 && l<10){cont++;}
47                 if(l>=10 && l<100){cont+=2;}
48                 if(j !=6){cont++;}
49             }
50             strcat(answer, "\n"); cont+=2;
51         }
52         File1 = open("Suma.txt",O_CREAT|O_WRONLY,0700);
53         write(File1,answer,cont);
54         close(File1);
55     }
56     if(val == 1){
57         for(i=0; i<400; i++){
58             answer[i]=0;
59         }
60         for(k = 0; k<7; k++){
61             for(j=0; j<7;j++){
62                 l=matriz2[k][j] - matriz1[k][j];
63
64                 sprintf(ll, "%d", l);
65                 strcat(answer, ll);
66                 strcat(answer, " ");
67
68                 if(l>=0 && l<10){cont++;}
69                 if(l>=10 && l<100){cont+=2;}
70                 if(l<0 && l>-10){cont+=2;}
71                 if(l<=-10 && l>-100){cont+=3;}
72                 if(j != 6){cont++;}
73             }
74             strcat(answer, "\n"); cont+=2;
```

```

75     }
76     File1 = open("Resta.txt",O_CREAT|O_WRONLY,0700);
77     write(File1,answer,cont);
78     close(File1);
79 }
80 if(val == 2){
81     for(i=0;i<400;i++){
82         answer[i]=0;
83     }
84
85     for(k=0; k<7; k++){
86         for(j=0; j<7; j++){
87             l = matriz2[k][j] * matriz1[k][j];
88             sprintf(ll, "%d",l);
89             strcat(answer, ll);
90             strcat(answer, " ");
91
92             if(l>=0 && l<10){cont++;}
93             if(l>=10 && l<100){cont+=2;}
94             if(l<0 && l>-10){cont+=2;}
95             if(l<=-10 && l>-100){cont+=3;}
96             if(j != 6){cont++;}
97         }
98         strcat(answer, "\n"); cont+=2;
99     }
100
101     File1 = open("Multiplicacion.txt",O_CREAT|O_WRONLY,0700);
102     write(File1,answer,cont);
103     close(File1);
104 }
105 if(val == 3){
106     for(i=0;i<400;i++){
107         answer[i]=0;
108     }
109     for(k=0; k<7; k++){
110         for(j=0; j<7; j++){
111             l = matriz2[k][j];
112             sprintf(ll, "%d",l);
113             strcat(answer, ll);
114             strcat(answer, " ");
115
116             if(l>=0 && l<10){cont++;}
117             if(l>=10 && l<100){cont+=2;}
118             if(l<0 && l>-10){cont+=2;}
119             if(l<=-10 && l>-100){cont+=3;}
120             if(j != 6){cont++;}
121         }
122         strcat(answer, "\n"); cont+=2;
123     }
124     strcat(answer, "\n"); cont+=2;
125     strcat(answer, "\n");
126
127     for(k=0; k<7; k++){
128         for(j=0; j<7; j++){
129             l = matriz1[k][j];
130
131             sprintf(ll, "%d",l);
132             strcat(answer, ll);
133             strcat(answer, " ");
134
135             if(l>=0 && l<10){cont++;}
136             if(l>=10 && l<100){cont+=2;}
137             if(l<0 && l>-10){cont+=2;}
138             if(l<=-10 && l>-100){cont+=3;}
139             if(j != 6){cont++;}
140         }
141         strcat(answer, "\n"); cont+=2;
142     }
143
144     File1 = open("Transpuesta.txt",O_CREAT|O_WRONLY,0700);
145     write(File1,answer,cont);
146     close(File1);
147 }
148 if(val == 4){
149     for(i=0;i<400;i++){

```



```

149     for(i=0;i<400;i++){
150         answer[i]=0;
151     }
152     printf("\nResultado de la Suma \n\n");
153     File1 = open("Suma.txt",O_RDONLY);
154     nr_bytes = read(File1, answer,200);
155     printf("%s", answer);
156     close(File1);
157
158     printf("\nResultado de la Resta \n\n");
159     File1 = open("Resta.txt",O_RDONLY);
160     nr_bytes = read(File1, answer,200);
161     printf("%s", answer);
162     close(File1);
163
164     printf("\nResultado de la multiplicacion \n\n");
165     File1 = open("Multiplicacion.txt",O_RDONLY);
166     nr_bytes = read(File1, answer,200);
167     printf("%s", answer);
168     close(File1);
169
170     printf("\nResultado de la transpuesta \n\n");
171     File1 = open("Transpuesta.txt",O_RDONLY);
172     nr_bytes = read(File1, answer,400);
173     printf("%s", answer);
174     close(File1);
175     }
176     return NULL;
177 }

```

LINUX

COPIA EXACTA DE PROCESOS

```

1 #include<stdio.h>
2 #include<unistd.h>
3 #include<stdlib.h>
4 #include<sys/wait.h>
5 #include<fcntl.h>
6 #include<string.h>
7
8 int main(void){
9     int File1;
10    int i,j,k,l;
11    int matriz1[7][7] = {{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,2,1,4,5,6,7},{7,6,5,1,2,3,4},{6,3,5,2,4,1,7},{7,4,5,1,2,3,6},{1,2,3,4,5,6,7}};
12    int matriz2[7][7] = {{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,2,1,4,5,6,7},{7,6,5,1,2,3,4},{6,3,5,2,4,1,7},{7,4,5,1,2,3,6},{1,2,3,4,5,6,7}};
13    int asn[7][7],cont=0, l;
14    char ll[100];
15    char answer[400];
16    ssize_t nr_bytes;
17
18    for(i=0; i<5; i++){
19        id = fork();
20        wait(NULL);
21        for(k=0;k<400;k++){
22            answer[i]=0;
23        }
24        if(id == 0){
25
26            //printf("Proceso padre:)
27            if(id == 0) {
28                for(k=0;k<7;k++){
29                    for(j=0;j<7;j++){
30                        l = matriz2[k][j] + matriz1[k][j];
31                        sprintf(ll, "%d",l);
32                        strcat(answer, ll);
33                        strcat(answer, " ");
34
35                        if(l>=0 && l<10){cont++;}
36                        if(l>=10 && l<100){cont+=2;}
37

```

C Anchura del tabulador: 10 Ln 132, Col 9 INS

```

38         if(j !=6){cont++;}
39     }
40     strcat(answer, "\n"); cont+=2;
41 }
42 File1 = open("Suma.txt",O_CREAT|O_WRONLY,0700);
43 write(File1,answer,cont);
44 close(File1);
45 }
46
47 if(i == 1){
48     for(k = 0; k<7; k++){
49         for(j=0; j<7;j++){
50             l=matriz2[k][j] - matriz1[k][j];
51
52             sprintf(ll, "%d", l);
53             strcat(answer, ll);
54             strcat(answer, " ");
55
56             if(l>=0 && l<10){cont++;}
57             if(l>=10 && l<100){cont+=2;}
58             if(l<0 && l>-10){cont+=2;}
59             if(l<=-10 && l>-100){cont+=3;}
60             if(j != 6){cont++;}
61         }
62         strcat(answer, "\n"); cont+=2;
63     }
64     File1 = open("Resta.txt",O_CREAT|O_WRONLY,0700);
65     write(File1,answer,cont);
66     close(File1);
67 }
68
69 if( i == 2){
70     for(k = 0; k<7; k++){
71         for(j=0; j<7;j++){
72             l=matriz2[k][j] * matriz1[k][j];
73
74             sprintf(ll, "%d", l);
75             strcat(answer, ll);
76             strcat(answer, " ");
77
78             if(l>=0 && l<10){cont++;}
79             if(l>=10 && l<100){cont+=2;}
80             if(l<0 && l>-10){cont+=2;}
81             if(l<=-10 && l>-100){cont+=3;}
82             if(j != 6){cont++;}
83         }
84         strcat(answer, "\n"); cont+=2;
85     }
86     File1 = open("Multiplicacion.txt",O_CREAT|O_WRONLY,0700);
87     write(File1,answer,cont);
88     close(File1);
89 }
90
91 if(i == 3){
92     for(k = 0; k<7; k++){
93         for(j=0; j<7;j++){
94             l=matriz2[j][k];
95
96             sprintf(ll, "%d", l);
97             strcat(answer, ll);
98             strcat(answer, " ");
99
100             if(l>=0 && l<10){cont++;}
101             if(l>=10 && l<100){cont+=2;}
102             if(l<0 && l>-10){cont+=2;}
103             if(l<=-10 && l>-100){cont+=3;}
104             if(j != 6){cont++;}
105         }
106         strcat(answer, "\n"); cont+=2;
107     }
108     strcat(answer, "\n"); cont+=2;
109     strcat(answer, "\n");
110
111     for(k = 0; k<7; k++){
112         for(j=0; j<7;j++){

```



```

112         l= matriz1[j][k];
113
114         sprintf(ll, "%d", l);
115         strcat(answer, ll);
116         strcat(answer, " ");
117
118         if(l>=0 && l<10){cont++;}
119         if(l>=10 && l<100){cont+=2;}
120         if(l<0 && l>=-10){cont+=2;}
121         if(l<=-10 && l>=-100){cont+=3;}
122         if(j != 6){cont++;}
123     }
124     strcat(answer, "\n"); cont+=2;
125 }
126 File1 = open("Transpuesta.txt",O_CREAT|O_WRONLY,0700);
127 write(File1,answer,cont);
128 close(File1);
129 }
130
131 if(i== 4){
132
133     printf("\nResultado de la Suma \n\n");
134     File1 = open("Suma.txt",O_RDONLY);
135     nr_bytes = read(File1, answer,200);
136     printf("%s", answer);
137     close(File1);
138
139     printf("\nResultado de la Resta \n\n");
140     File1 = open("Resta.txt",O_RDONLY);
141     nr_bytes = read(File1, answer,200);
142     printf("%s", answer);
143     close(File1);
144
145     printf("\nResultado de la multiplicacion \n\n");
146     File1 = open("Multiplicacion.txt",O_RDONLY);
147     nr_bytes = read(File1, answer,200);
148     printf("%s", answer);
149     close(File1);
150
151     printf("\nResultado de la transpuesta \n\n");
152     File1 = open("Transpuesta.txt",O_RDONLY);
153     nr_bytes = read(File1, answer,400);
154     printf("%s", answer);
155     close(File1);
156 }
157 break;
158 }
159 }

```

1.-CÓDIGO ECHO EN WINDOWS POR HILOS

```
#include <windows.h>
#include <stdio.h>
#include <time.h>

DWORD WINAPI funcionHilo(LPVOID lpParam);

typedef struct Informacion info;

int matriz1[7][7] = {{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,2,1,4,5,6,7},{7,6,5,1,2,3,4},{6,3,5,2,4,1,7},{7,4,5,1,2,3,6},{1,2,3,4,5,6,7}};
int matriz2[7][7] = {{1,2,3,4,5,6,7},{7,6,5,4,3,2,1},{3,2,1,4,5,6,7},{7,6,5,1,2,3,4},{6,3,5,2,4,1,7},{7,4,5,1,2,3,6},{1,2,3,4,5,6,7}};
int cont=0, l, j,k,i;

int main(void){
    clock_t tiempo_inicio, tiempo_final;
    double segundos;
    tiempo_inicio = clock();
    DWORD idHilo;
    HANDLE manHilo;
    int val;
    for (i=0; i<5; i++){
        val = i;
        manHilo=CreateThread(MULTI, 0, funcionHilo, &val, 0, &idHilo);
        WaitForSingleObject(manHilo, INFINITE);
    }
    CloseHandle(manHilo);

    tiempo_final = clock();
    segundos = (double)(tiempo_final - tiempo_inicio) / CLOCKS_PER_SEC;
    printf("Tiempo de ejecucion: %f Segundos",segundos);
    return 0;
}

DWORD WINAPI funcionHilo(void* arg){
    DWORD idHilo;
    HANDLE manHilo;
    int var = *((int*)arg);
    char imp;
    if(var == 0){
        FILE *s;
        s=fopen("suma.txt","w");
        for(k = 0; k<7; k++){
            for(j=0; j<7;j++){
                l = matriz2[k][j] + matriz1[k][j];
                fprintf(s,"%d ", l);
            }
            fprintf(s,"\n");
        }
        fclose(s);
    }
    if(var == 1){
        FILE *s;
        s=fopen("resta.txt","w");
        for(k = 0; k<7; k++){
            for(j=0; j<7;j++){
                l = matriz2[k][j] - matriz1[k][j];
                fprintf(s,"%d ", l);
            }
            fprintf(s,"\n");
        }
        fclose(s);
    }
    if(var == 2){
        FILE *s;
        s=fopen("mult.txt","w");
        for(k = 0; k<7; k++){
            for(j=0; j<7;j++){
                l = matriz2[k][j] * matriz1[k][j];
                fprintf(s,"%d ", l);
            }
            fprintf(s,"\n");
        }
        fclose(s);
    }
    if(var == 3){
        FILE *s;
        s=fopen("tra.txt","w");
        for(k = 0; k<7; k++){
```

```
        for(j=0; j<7;j++){
            l = matriz2[k][j] * matriz1[k][j];
            fprintf(s,"%d ", l);
        }
        fprintf(s,"\n");
    }
    if(var == 3){
        FILE *s;
        s=fopen("tra.txt","w");
        for(k = 0; k<7; k++){
```



```

for(k = 0; k<7; k++){
    for(j=0; j<7; j++){
        l = matriz2[j][k];
        fprintf(s, "%d ", l);
    }
    fprintf(s, "\n");
}
fprintf(s, "\n\n");
for(k = 0; k<7; k++){
    for(j=0; j<7; j++){
        l = matriz1[j][k];
        fprintf(s, "%d ", l);
    }
    fprintf(s, "\n");
}

fclose(s);
}

if(var == 4){
    FILE *s;
    printf("\nResultado del la Suma \n\n");
    s=fopen("suma.txt", "r+t");
    while(!feof(s) == 0){
        imp = fgetc(s);
        printf("%c", imp);
    }
    fclose(s);
    printf("\nResultado del la Resta\n\n");
    s=fopen("resta.txt", "r+t");
    while(!feof(s) == 0){
        imp = fgetc(s);
        printf("%c", imp);
    }
    fclose(s);
    printf("\nResultado del la multiplicacion\n\n");
    s=fopen("mult.txt", "r+t");
    s=fopen("mult.txt", "r+t");
    while(!feof(s) == 0){
        imp = fgetc(s);
        printf("%c", imp);
    }
    fclose(s);
    printf("\nResultado del la Transpuesta\n\n");
    s=fopen("tra.txt", "r+t");
    while(!feof(s) == 0){
        imp = fgetc(s);
        printf("%c", imp);
    }
    fclose(s);
}
return 0;
}

```

2-CODIGO HECHO EN WINDOWS POR COPIA EXACTA DE CÓDIGO

```

1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  #include <time.h>
5
6  float mat1[40][40]={1,0,0,0,0,0,0,0,0,0},
7                      1,2,0,0,0,0,0,0,0,0},
8                      1,2,3,0,0,0,0,0,0,0},
9                      1,2,3,4,0,0,0,0,0,0},
10                     1,2,3,4,5,0,0,0,0,0},
11                     1,2,3,4,5,6,0,0,0,0},
12                     1,2,3,4,5,6,7,0,0,0},
13                     1,2,3,4,5,6,7,8,0,0},
14                     1,2,3,4,5,6,7,8,9,0},
15                     1,2,3,4,5,6,7,8,9,10}};
16
17  float mat2[40][40]={10,9,8,7,6,5,4,3,2,1},
18                     0,9,8,7,6,5,4,3,2,1},
19                     0,0,8,7,6,5,4,3,2,1},
20                     0,0,0,7,6,5,4,3,2,1},
21                     0,0,0,0,6,5,4,3,2,1},
22                     0,0,0,0,0,5,4,3,2,1},
23                     0,0,0,0,0,0,4,3,2,1},

```

```

25         {0,0,0,0,0,0,0,0,0,2,1},
26         {0,0,0,0,0,0,0,0,0,0,1}};
27 int mat1t[10][10]={};
28 int mat2t[10][10]={};
29 int mat1inv[10][10]={};
30 int mat2inv[10][10]={};
31 int multi[10][10]={};
32
33 char archivo[10000];
34
35
36 void InverseOfMatrix(float matrix[][40], int order,int opc){
37
38     float temp;
39     for (int i = 0; i < order; i++) {
40         for (int j = 0; j < 2 * order; j++) {
41             if (j == (i + order))
42                 matrix[i][j] = 1;
43         }
44     }
45
46     for (int i = order - 1; i > 0; i--) {
47         if (matrix[i - 1][0] < matrix[i][0])
48             for (int j = 0; j < 2 * order; j++) {
49                 temp = matrix[i][j];
50                 matrix[i][j] = matrix[i - 1][j];
51                 matrix[i - 1][j] = temp;
52             }
53     }
54
55     for (int i = 0; i < order; i++) {
56         for (int j = 0; j < 2 * order; j++) {
57             if (j != i) {
58                 temp = matrix[j][i] / matrix[i][i];
59                 for (int k = 0; k < 2 * order; k++) {
60                     matrix[j][k] -= matrix[i][k] * temp;
61                 }
62             }
63         }
64     }
65
66     for (int i = 0; i < order; i++) {
67
68         temp = matrix[i][i];
69         for (int j = 0; j < 2 * order; j++) {
70

```



```

71         matrix[i][j] = matrix[i][j] / temp;
72     }
73 }
74
75 for (int i = 0; i < order; i++) {
76     for (int j = order; j < 2 * order; j++) {
77         if(opc==0)
78             mat1inv[i][j-order]=matrix[i][j];
79         else
80             mat2inv[i][j-order]=matrix[i][j];
81     }
82 }
83
84 return;
85 }
86
87 void leer(char entrada[]){
88     char *secuencia;
89     FILE *ptrs;
90     ptrs= fopen(entrada,"r");
91     if(ptrs==NULL) {
92         printf("No hay datos");
93         exit(1);
94     }else{
95         while (fgets((char*)&archivo, sizeof(archivo), ptrs)) {
96             printf("%s",archivo);
97         }
98         fclose(ptrs);
99     }
100 }
101
102 int main(void){
103     double total_time;
104     clock_t start, end;
105     start = clock();
106     FILE* fichero;
107     fichero = fopen("suma.txt", "w");
108     for(int i=0; i<10; i++) {
109         for(int j=0; j<10; j++) {
110             //printf("%d ",(int)(mat1[i][j]+mat2[i][j]));
111             fprintf(fichero, "%d ", (int)(mat1[i][j]+mat2[i][j]));
112         }
113         fprintf(fichero, "\n");
114     }
115     fclose(fichero);

```

```

116
117 fichero = fopen("resta.txt", "w");
118 for(int i=0; i<10; i++) {
119     for(int j=0; j<10; j++) {
120         //printf("%d ", (int)(mat1[i][j]-mat2[i][j]));
121         fprintf(fichero, "%d ", (int)(mat1[i][j]-mat2[i][j]));
122     }
123     fprintf(fichero, "\n");
124 }
125 fclose(fichero);
126
127 fichero = fopen("multi.txt", "w");
128
129 int i, j, k;
130 for (i = 0; i < 10; i++) {
131     for (j = 0; j < 10; j++) {
132         multi[i][j] = 0;
133         for (k = 0; k < 10; k++)
134             multi[i][j] += mat1[i][k]*mat2[k][j];
135     }
136 }
137
138 for(int i=0; i<10; i++) {
139     for(int i=0; i<10; i++) {
140         for(int j=0; j<10; j++) {
141             fprintf(fichero, "%d ", (int)(multi[i][j]));
142         }
143         fprintf(fichero, "\n");
144     }
145
146     for(int i=0; i<10; i++) {
147         for(int j=0; j<10; j++) {
148             mat1t[i][j] = mat1[j][i];
149             mat2t[i][j] = mat2[j][i];
150         }
151     }
152
153     fichero = fopen("traspuesta.txt", "w");
154     for(int i=0; i<10; i++) {
155         for(int j=0; j<10; j++) {
156             fprintf(fichero, "%d ", (int)(mat1t[i][j]));
157         }
158         fprintf(fichero, "\n");
159     }
160     fprintf(fichero, "\n");
161     for(int i=0; i<10; i++) {

```



```
178 for(int i=0; i<10; i++) {
179     for(int j=0; j<10; j++) {
180         fprintf(fichero, "%.3f ", (float)(mat2inv[i][j]));
181     }
182     fprintf(fichero, "\n");
183 }
184 fclose(fichero);
185
186 printf("\n=== Suma ===\n");
187 leer("suma.txt");
188 printf("\n=== Resta ===\n");
189 leer("resta.txt");
190 printf("\n=== Multiplicacion ===\n");
191 leer("multi.txt");
192 printf("\n=== Traspuesta ===\n");
193 leer("traspuesta.txt");
194 printf("\n=== Inversa ===\n");
195 leer("inversa.txt");
196 end = clock();
197 total_time = ((double) (end - start)) / CLK_TCK;
198 printf("\n\t---El tiempo de ejecucion fue de: %f\n", total_time);
199 return 0;
200 }
```


CONCLUSIONES.

Con esta práctica hemos aprendido y analizado como es que los sistemas operativos se encuentran estructurados, en esta etapa, pudimos analizar correctamente el desarrollo de procesos y la comprensión de los mismos, al igual que estudiar los hilos de ejecución de estos, a través de los cuales, el sistema operativo delega las funciones y se vuelve multifuncional.

De igual manera aprendimos el uso correcto tanto de monitores como de semáforos y cómo es que actúan dentro del sistema operativo para poder seguir funciones específicas que son de vital importancia dentro de este.

Nos percatamos de que a pesar de que Windows y Linux son dos sistemas operativos diferentes, sus funciones son parecidas, y que al trabajar de manera similar nos podemos dar una referencia o una idea de cómo es que actúan los elementos que analizamos en las prácticas en ambos sistemas operativos.

Se analizó la forma en que el sistema operativo es capaz de gestionar las interrupciones, se encarga de controlar los accesos al procesador, verificar el estatus de un proceso y determinar su ejecución de acuerdo con el nivel de importancia de los mismos.

Además, al observar los resultados obtenidos podemos concluir que el tiempo de ejecución es similar a la hora de implementar copia exacta de código y el uso de hilos siendo la diferencia de tiempo causada por la implementación de código que tiene cada sistema y la forma en la que funcionan los procesos de cada una

REFERENCIAS

EcuRed. (s. f.). Administrador de procesos - EcuRed.
https://www.ecured.cu/Administrador_de_procesos

Gestión de Procesos en Linux y Windows. (s. f.). josecarreres. Recuperado 15 de noviembre de 2020, de <https://josecarreres.wordpress.com/2015/11/22/gestion-de-procesos-en-linux-y-windows/>