****

**Práctica 5. Administrador de procesos en Windows y Linux (2)**

**Grupo: 2CM9**

**Equipo:**

* **Guerra Vargas Irving Cristóbal Colón**
* **Reyes Valenzuela Alejandro**
* **Dávila Méndez Juan Gabriel**

**Competencia.**

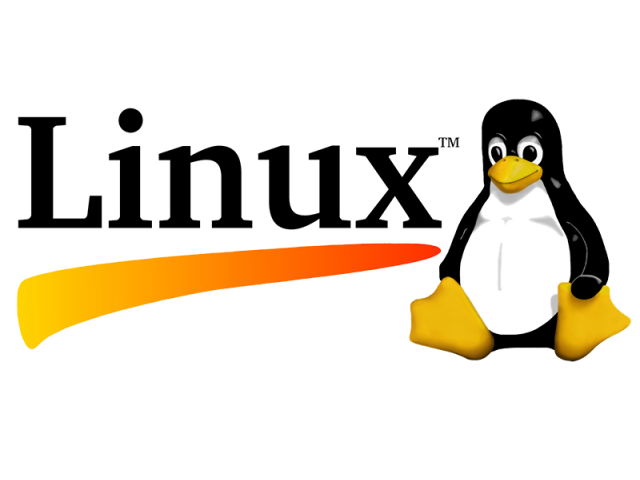
* **El alumno aprende a familiarizarse con el administrador de procesos del sistema operativo Windows y Linux a través de la creación de nuevos procesos por copia exacta de código y/o por sustitución de código para el desarrollo de aplicaciones concurrentes sencillas.**

****

**Desarrollo.**

1. A través de la ayuda en línea que proporciona Linux, investigue el funcionamiento de las funciones: **pthread\_create(), pthread\_join(), pyhread\_self(), pthread\_exit(), scandir(), stat().** Explique los argumentos y retorno de cada función.

|  |  |  |  |
| --- | --- | --- | --- |
| FUNCIÓN | DESCRIPCIÓN | ARGUMENTOS | RETORNO |
| pthread\_create() | Crea un nuevo hilo.  La función pthread\_create () inicia un nuevo hilo en la llamada proceso.  El nuevo subproceso comienza la ejecución invocando start\_routine (); arg se pasa como el único argumento de start\_routine (). | Sintaxis:  **#include <pthread.h>**   **int pthread\_create(pthread\_t \****thread***, const pthread\_attr\_t \****attr***,**  **void \*(\****start\_routine***) (void \*), void \****arg***);**  **DESCRIPCIÓN DE ARGUMENTOS:**  **thread:** Es un apuntador a una variable phtread\_t en la que se almacenará el identificador del hilo creado.  **attr:** Apunta a una estructura pthread\_arttr\_t cuyos contenidos se utilizan en el momento de creación del hilo para determinar los nuevos atributos del nuevo hilo. Si attr es NULL, el hilo se creará con atributos predeterminados.  **start\_routine:** Es un puntero a una función que recibe la dirección de memoria de la función que debe ejecutar el hilo.  **arg:** Apuntador en el que se almacenará la dirección de memoria de aquello que se quiere pasar como argumento a la función hilo, si no se quiere pasar algún valor se puede colocar NULL. | En caso de éxito, pthread\_create () devuelve 0.  Por error, devuelve un número que indica un error específico y los contenidos de \* thread no son definidos.  **ERRORES:**  **Egain:** Recursos insuficientes para crear otro hilo.  **Einval:** Configuración inválida en attr.  **EPERM:** No se permite establecer la política de programación y los parámetros especificados en attr. |
| pthread\_join() | Espera por el término de un hilo.  La función pthread\_join () espera el hilo especificado por thread para terminar. Si ese hilo ya ha terminado, entonces pthread\_join () regresa inmediatamente.  El hilo especificado por hilo debe poder unirse. | Sintaxis:  **#include <pthread.h>**  **int pthread\_join(pthread\_t** *thread***, void \*\****retval***);**  **DESCRIPCIÓN DE ARGUMENTOS:**  **thread:** Es un apuntador a una variable phtread\_t en la que se almacenará el identificador del hilo por el que se esperará su término.  **retval:** Si el hilo en cuestión es cancelado, retval toma el valor PTHREAD\_CANCELED. Si no estamos interesados en el valor retornado, poner NULL. | En caso de éxito, pthread\_join () devuelve 0.  Por error, devuelve un número específico de error.  **ERRORES:**  **Edeadlk:**  Se detectó un punto muerto (por ejemplo, dos hilos intentaron unirse el uno al otro); o el hilo especifica la llamada hilo.  **Einval:**  -El hilo no es un hilo que se puede unir.  -Otro hilo ya está esperando para unirse a este hilo.  **Esrch:**  No se pudo encontrar ningún hilo con el ID hilo. |
| pthread\_self() | Ayuda a obtener ID del hilo que ha hecho la llamada.  Este es el mismo valor que se devuelve en \* thread en  pthread\_create() llamada que creó este hilo. | Sintaxis:  **#include <pthread.h>**   **pthread\_t pthread\_self(void);** | Esta función siempre tiene éxito, devolviendo el ID del hilo que la llamó. |
| pthread\_exit() | La función pthread\_exit () finaliza el hilo de llamada y hace que el valor value\_ptr esté disponible para cualquier unión exitosa con el hilo de terminación.  La función termina el hilo sin terminar el proceso. | Sintaxis:  **void pthread\_exit(void \**value\_ptr*);**  **DESCRIPCIÓN DE ARGUMENTOS:**  **value\_ptr:** Queda disponible para otros hilos al llamar pthread\_join; value\_ptr debe existir después del término del hilo. | La función pthread\_exit () no puede regresar a su llamador. |
| scandir() | La función **scandir()** rastrea el directorio *dirp*, llamando filter**()** en cada entrada de directorio. Las entradas para las que filter**()** devuelve un valor distinto de cero se almacenan en cadenas (strings) reservadas vía **malloc().** | Sintaxis:  **int scandir(const char \****dirp***, struct dirent \*\*\****namelist***,**  **int (\****filter***)(const struct dirent \*),**  **int (\****compar***)(const struct dirent \*\*, const struct dirent \*\*));** | La función **scandir()** devuelve el número de entradas de directorio seleccionadas, o -1 si hubo algún error.  **ERRORES:**  **Enoent:** La ruta en dirp no existe.  **Enomem:** Memoria insuficiente para completar la operación.  **Enotdir:** La ruta en dirp no es un directorio. |
| stat() | **stat()** examina el fichero al que apunta pathname y llena statbuf.  Muestra el estado del archivo o del sistema de archivos. | Sintaxis:  **#include <sys/types.h>**  **#include <sys/stat.h>**  **#include <unistd.h>**   **int stat(const char \****pathname***, struct stat \****statbuf***);**  **DESCRIPCIÓN DE ARGUMENTOS:**  **pathname:** Apuntador al archivo sobre el que se espera recuperar información  **statbuf:** Apunta a una estructura donde se devuelven los resultados de la llamada. | En caso de éxito, se devuelve cero.  En caso de error, se devuelve -1 y errno se  establece apropiadamente. |

****

**Ejemplo:**

**#include <stdio.h>**

**#include <pthread.h>**

**void\* hilo(void \*arg) {**

**printf("Hola mundo desde un hilo en UNIX \n");**

**return 0;**

**}**

**int**

**main (void) {**

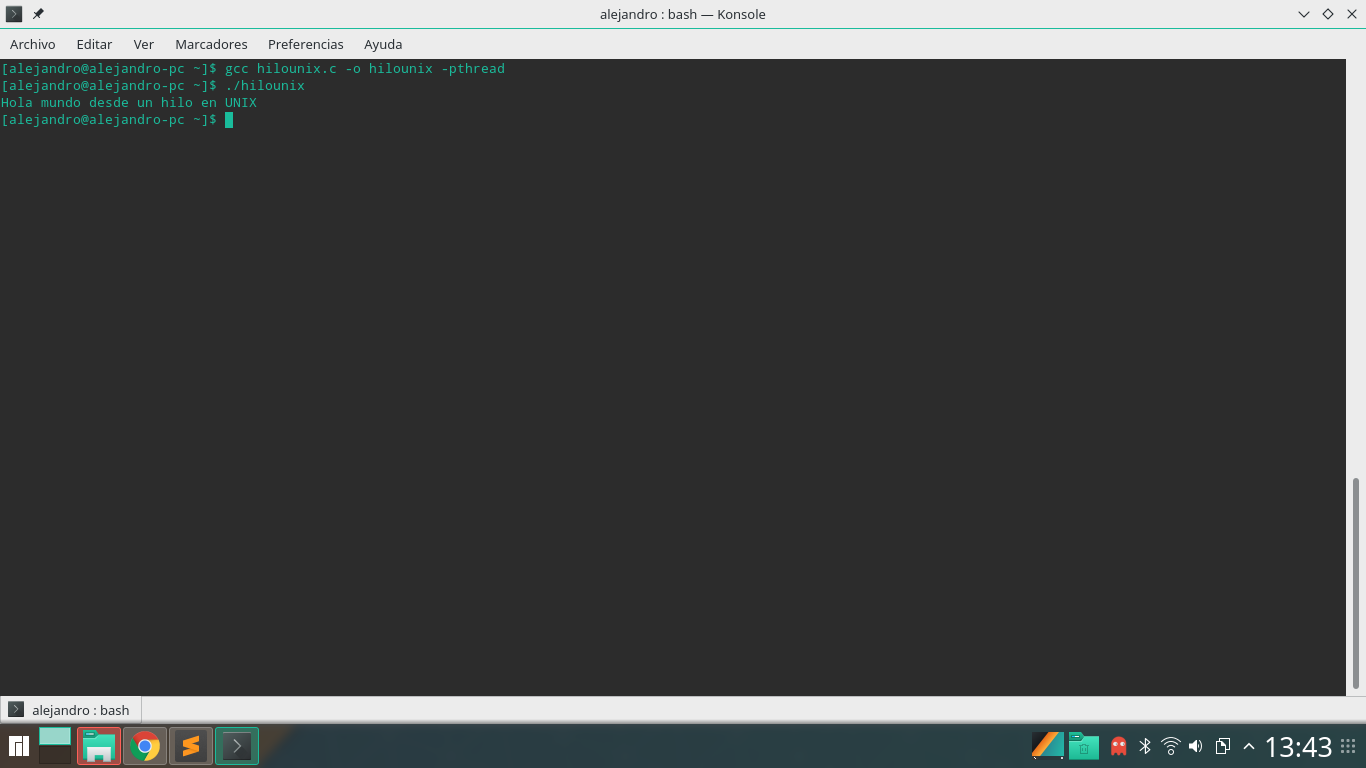
**pthread\_t id\_hilo;**

**pthread\_create(&id\_hilo, 0, (void \*) hilo, 0);**

**pthread\_join(id\_hilo, 0);**

**return 0;**

**}**

****

**#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 = %d \n", devolucion\_hilo);**

**return 0;**

**}**

**void\* hilo(void \*arg) {**

**char \*men = (char \*) arg;**

**int resultado\_hilo=0;**

**printf("%s\n", men);**

**resultado\_hilo = 100;**

**pthread\_exit((void \*) resultado\_hilo);**

**}**

**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) [...]**

**Linux:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <pthread.h>**

**#include <unistd.h>**

**#include <sys/syscall.h>**

**#include <sys/wait.h>**

**#include <sys/types.h>**

**#define firstLevelThread 15**

**#define secondLevelThread 10**

**#define thirdLevelThread 5**

**void \*hiloprincipal(void \*data);**

**void \*hiloSecundario(void \*data);**

**void \*hiloTerciario(void \*data);**

**int ObtenerIDHilo(void );**

**struct datos {**

**int iii;**

**}; //Estructuras para pasar datos entre proceso e hilos**

**struct datos2 {**

**int ii;**

**int hiloPadre;**

**};**

**struct datos3 {**

**int iiii;**

**int hiloPadre;**

**};**

**int main()**

**{**

**printf("PID PADRE: %d\n",getpid());**

**if(fork()==0) //Crear proceso hijo con fork**

**{**

**printf("PID HIJO: %d\n",getpid());**

**pthread\_t thread[firstLevelThread]; //Handlers de los primeros 15 hilos**

**int hilonum[firstLevelThread];**

**struct datos \*datos = malloc(sizeof(struct datos));**

**for (int i=0;i<firstLevelThread;i++)**

**{**

**datos->iii=i; //Pasa el numero de hilo que esta en ejecucion**

**hilonum[i]=pthread\_create(&(thread[i]),NULL, hiloprincipal, datos);**

**if (hilonum[i])**

**printf("No fue posible crear un hilo.\n");**

**pthread\_join(thread[i], NULL); //Espera a que todos los hilos acaben**

**}**

**}**

**wait(0);**

**exit(0);**

**}**

**void \*hiloprincipal(void \*data)**

**{**

**struct datos \*datos = data;**

**printf("[NIVEL 1] HiloN#: %d, Thread id: %d, Padre: %d\n",(datos->iii)+1,ObtenerIDHilo(),getpid());**

**pthread\_t thread2[secondLevelThread]; //Handlers de los primeros 10 hilos**

**int hilonum2[secondLevelThread];**

**struct datos2 \*datos2 = malloc(sizeof(struct datos2));**

**for(int i=0;i<secondLevelThread;i++)**

**{**

**datos2->ii=i; //Num. de hilo**

**datos2->hiloPadre=ObtenerIDHilo(); //Se pasa el id del hilo padre**

**hilonum2[i]=pthread\_create(&(thread2[i]),NULL, hiloSecundario, datos2);**

**if(hilonum2[i])**

**printf("No fue posible crear un hilo. \n");**

**pthread\_join(thread2[i], NULL); //Espera a que todos los hilos acaben**

**}**

**pthread\_exit(NULL); //Sale del hilo**

**}**

**void \*hiloSecundario(void \*data)**

**{**

**struct datos2 \*datos2 = data;**

**printf(" |--[NIVEL 2] HiloN#: %d, Thread id: %d, HiloPadre: %d\n",(datos2->ii)+1,ObtenerIDHilo(),(datos2->hiloPadre));**

**pthread\_t thread3[thirdLevelThread]; //Handlers de los primeros 5 hilos**

**int hilonum3[thirdLevelThread];**

**struct datos3 \*datos3 = malloc(sizeof(struct datos3));**

**for(int i=0;i<thirdLevelThread;i++)**

**{**

**datos3->iiii=i;**

**datos3->hiloPadre=ObtenerIDHilo();**

**hilonum3[i]=pthread\_create(&(thread3[i]),NULL, hiloTerciario, datos3);**

**if(hilonum3[i])**

**printf("No fue posible crear un hilo. \n");**

**pthread\_join(thread3[i], NULL); //Espera a que todos los hilos acaben**

**}**

**pthread\_exit(NULL); //Sale del hilo**

**}**

**void \*hiloTerciario(void \*data)**

**{**

**struct datos3 \*datos3 = data;**

**if((datos3->iiii)==4)**

**printf(" |--[NIVEL 3] PRACTICA 5 (Thread id: %d)\n",ObtenerIDHilo());**

**else**

**printf(" |--[NIVEL 3] HiloN#: %d, Thread id: %d, HiloPadre: %d\n",(datos3->iiii)+1,ObtenerIDHilo(),(datos3->hiloPadre));**

**pthread\_exit(NULL); //Sale del hilo**

**}**

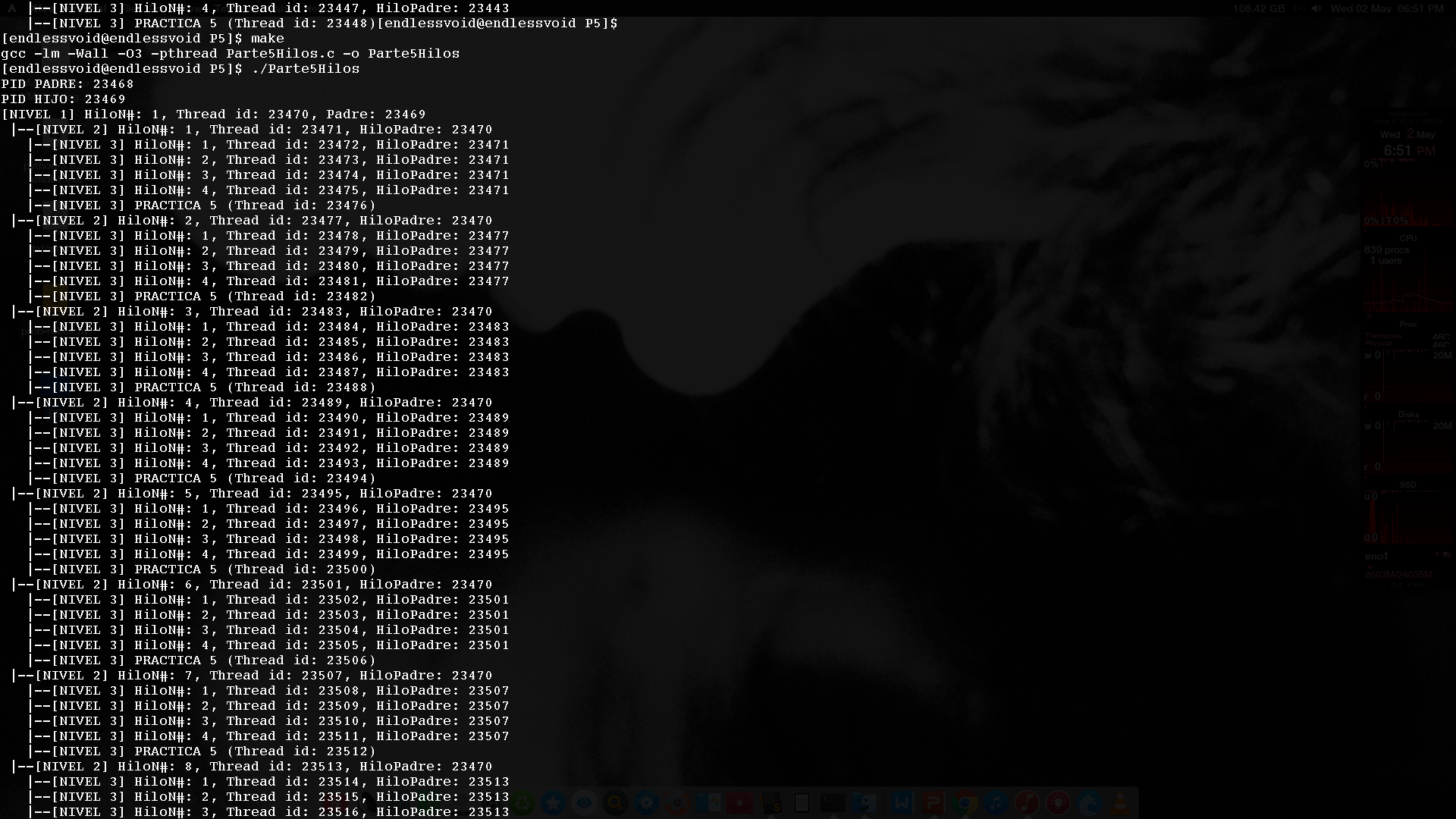
**int ObtenerIDHilo(void ){**

**pid\_t threadid = syscall(SYS\_gettid);**

**return threadid; //Sirve para obtener el identificador del hilo**

**}**

**Capturas de pantalla:**

****

**Salida semicompleta a continuación.**

PID PADRE: 23468

PID HIJO: 23469

[NIVEL 1] HiloN#: 1, Thread id: 23470, Padre: 23469

|--[NIVEL 2] HiloN#: 1, Thread id: 23471, HiloPadre: 23470

|--[NIVEL 3] HiloN#: 1, Thread id: 23472, HiloPadre: 23471

|--[NIVEL 3] HiloN#: 2, Thread id: 23473, HiloPadre: 23471

|--[NIVEL 3] HiloN#: 3, Thread id: 23474, HiloPadre: 23471

|--[NIVEL 3] HiloN#: 4, Thread id: 23475, HiloPadre: 23471

|--[NIVEL 3] PRACTICA 5 (Thread id: 23476)

|--[NIVEL 2] HiloN#: 2, Thread id: 23477, HiloPadre: 23470

|--[NIVEL 3] HiloN#: 1, Thread id: 23478, HiloPadre: 23477

|--[NIVEL 3] HiloN#: 2, Thread id: 23479, HiloPadre: 23477

|--[NIVEL 3] HiloN#: 3, Thread id: 23480, HiloPadre: 23477

|--[NIVEL 3] HiloN#: 4, Thread id: 23481, HiloPadre: 23477

|--[NIVEL 3] PRACTICA 5 (Thread id: 23482)

|--[NIVEL 2] HiloN#: 3, Thread id: 23483, HiloPadre: 23470

|--[NIVEL 3] HiloN#: 1, Thread id: 23484, HiloPadre: 23483

|--[NIVEL 3] HiloN#: 2, Thread id: 23485, HiloPadre: 23483

|--[NIVEL 3] HiloN#: 3, Thread id: 23486, HiloPadre: 23483

|--[NIVEL 3] HiloN#: 4, Thread id: 23487, HiloPadre: 23483

|--[NIVEL 3] PRACTICA 5 (Thread id: 23488)

|--[NIVEL 2] HiloN#: 4, Thread id: 23489, HiloPadre: 23470

|--[NIVEL 3] HiloN#: 1, Thread id: 23490, HiloPadre: 23489

|--[NIVEL 3] HiloN#: 2, Thread id: 23491, HiloPadre: 23489

|--[NIVEL 3] HiloN#: 3, Thread id: 23492, HiloPadre: 23489

|--[NIVEL 3] HiloN#: 4, Thread id: 23493, HiloPadre: 23489

|--[NIVEL 3] PRACTICA 5 (Thread id: 23494)

|--[NIVEL 2] HiloN#: 5, Thread id: 23495, HiloPadre: 23470

|--[NIVEL 3] HiloN#: 1, Thread id: 23496, HiloPadre: 23495

|--[NIVEL 3] HiloN#: 2, Thread id: 23497, HiloPadre: 23495

|--[NIVEL 3] HiloN#: 3, Thread id: 23498, HiloPadre: 23495

|--[NIVEL 3] HiloN#: 4, Thread id: 23499, HiloPadre: 23495

|--[NIVEL 3] PRACTICA 5 (Thread id: 23500)

|--[NIVEL 2] HiloN#: 6, Thread id: 23501, HiloPadre: 23470

|--[NIVEL 3] HiloN#: 1, Thread id: 23502, HiloPadre: 23501

|--[NIVEL 3] HiloN#: 2, Thread id: 23503, HiloPadre: 23501

|--[NIVEL 3] HiloN#: 3, Thread id: 23504, HiloPadre: 23501

|--[NIVEL 3] HiloN#: 4, Thread id: 23505, HiloPadre: 23501

|--[NIVEL 3] PRACTICA 5 (Thread id: 23506)

|--[NIVEL 2] HiloN#: 7, Thread id: 23507, HiloPadre: 23470

|--[NIVEL 3] HiloN#: 1, Thread id: 23508, HiloPadre: 23507

|--[NIVEL 3] HiloN#: 2, Thread id: 23509, HiloPadre: 23507

|--[NIVEL 3] HiloN#: 3, Thread id: 23510, HiloPadre: 23507

|--[NIVEL 3] HiloN#: 4, Thread id: 23511, HiloPadre: 23507

|--[NIVEL 3] PRACTICA 5 (Thread id: 23512)

|--[NIVEL 2] HiloN#: 8, Thread id: 23513, HiloPadre: 23470

|--[NIVEL 3] HiloN#: 1, Thread id: 23514, HiloPadre: 23513

|--[NIVEL 3] HiloN#: 2, Thread id: 23515, HiloPadre: 23513

|--[NIVEL 3] HiloN#: 3, Thread id: 23516, HiloPadre: 23513

|--[NIVEL 3] HiloN#: 4, Thread id: 23517, HiloPadre: 23513

|--[NIVEL 3] PRACTICA 5 (Thread id: 23518)

|--[NIVEL 2] HiloN#: 9, Thread id: 23519, HiloPadre: 23470

|--[NIVEL 3] HiloN#: 1, Thread id: 23520, HiloPadre: 23519

|--[NIVEL 3] HiloN#: 2, Thread id: 23521, HiloPadre: 23519

|--[NIVEL 3] HiloN#: 3, Thread id: 23522, HiloPadre: 23519

|--[NIVEL 3] HiloN#: 4, Thread id: 23523, HiloPadre: 23519

|--[NIVEL 3] PRACTICA 5 (Thread id: 23524)

|--[NIVEL 2] HiloN#: 10, Thread id: 23525, HiloPadre: 23470

|--[NIVEL 3] HiloN#: 1, Thread id: 23526, HiloPadre: 23525

|--[NIVEL 3] HiloN#: 2, Thread id: 23527, HiloPadre: 23525

|--[NIVEL 3] HiloN#: 3, Thread id: 23528, HiloPadre: 23525

|--[NIVEL 3] HiloN#: 4, Thread id: 23529, HiloPadre: 23525

|--[NIVEL 3] PRACTICA 5 (Thread id: 23530)

[NIVEL 1] HiloN#: 2, Thread id: 23531, Padre: 23469

|--[NIVEL 2] HiloN#: 1, Thread id: 23532, HiloPadre: 23531

|--[NIVEL 3] HiloN#: 1, Thread id: 23533, HiloPadre: 23532

|--[NIVEL 3] HiloN#: 2, Thread id: 23534, HiloPadre: 23532

|--[NIVEL 3] HiloN#: 3, Thread id: 23535, HiloPadre: 23532

///////////////////[SALIDA TRUNCADA]///////////////////

[NIVEL 1] HiloN#: 14, Thread id: 24263, Padre: 23469

|--[NIVEL 2] HiloN#: 1, Thread id: 24264, HiloPadre: 24263

|--[NIVEL 3] HiloN#: 1, Thread id: 24265, HiloPadre: 24264

|--[NIVEL 3] HiloN#: 2, Thread id: 24266, HiloPadre: 24264

|--[NIVEL 3] HiloN#: 3, Thread id: 24267, HiloPadre: 24264

|--[NIVEL 3] HiloN#: 4, Thread id: 24268, HiloPadre: 24264

|--[NIVEL 3] PRACTICA 5 (Thread id: 24269)

|--[NIVEL 2] HiloN#: 2, Thread id: 24270, HiloPadre: 24263

|--[NIVEL 3] HiloN#: 1, Thread id: 24271, HiloPadre: 24270

|--[NIVEL 3] HiloN#: 2, Thread id: 24272, HiloPadre: 24270

|--[NIVEL 3] HiloN#: 3, Thread id: 24273, HiloPadre: 24270

|--[NIVEL 3] HiloN#: 4, Thread id: 24274, HiloPadre: 24270

|--[NIVEL 3] PRACTICA 5 (Thread id: 24275)

|--[NIVEL 2] HiloN#: 3, Thread id: 24276, HiloPadre: 24263

|--[NIVEL 3] HiloN#: 1, Thread id: 24277, HiloPadre: 24276

|--[NIVEL 3] HiloN#: 2, Thread id: 24278, HiloPadre: 24276

|--[NIVEL 3] HiloN#: 3, Thread id: 24279, HiloPadre: 24276

|--[NIVEL 3] HiloN#: 4, Thread id: 24280, HiloPadre: 24276

|--[NIVEL 3] PRACTICA 5 (Thread id: 24281)

|--[NIVEL 2] HiloN#: 4, Thread id: 24282, HiloPadre: 24263

|--[NIVEL 3] HiloN#: 1, Thread id: 24283, HiloPadre: 24282

|--[NIVEL 3] HiloN#: 2, Thread id: 24284, HiloPadre: 24282

|--[NIVEL 3] HiloN#: 3, Thread id: 24285, HiloPadre: 24282

|--[NIVEL 3] HiloN#: 4, Thread id: 24286, HiloPadre: 24282

|--[NIVEL 3] PRACTICA 5 (Thread id: 24287)

|--[NIVEL 2] HiloN#: 5, Thread id: 24288, HiloPadre: 24263

|--[NIVEL 3] HiloN#: 1, Thread id: 24289, HiloPadre: 24288

|--[NIVEL 3] HiloN#: 2, Thread id: 24290, HiloPadre: 24288

|--[NIVEL 3] HiloN#: 3, Thread id: 24291, HiloPadre: 24288

|--[NIVEL 3] HiloN#: 4, Thread id: 24292, HiloPadre: 24288

|--[NIVEL 3] PRACTICA 5 (Thread id: 24293)

|--[NIVEL 2] HiloN#: 6, Thread id: 24294, HiloPadre: 24263

|--[NIVEL 3] HiloN#: 1, Thread id: 24295, HiloPadre: 24294

|--[NIVEL 3] HiloN#: 2, Thread id: 24296, HiloPadre: 24294

|--[NIVEL 3] HiloN#: 3, Thread id: 24297, HiloPadre: 24294

|--[NIVEL 3] HiloN#: 4, Thread id: 24298, HiloPadre: 24294

|--[NIVEL 3] PRACTICA 5 (Thread id: 24299)

|--[NIVEL 2] HiloN#: 7, Thread id: 24300, HiloPadre: 24263

|--[NIVEL 3] HiloN#: 1, Thread id: 24301, HiloPadre: 24300

|--[NIVEL 3] HiloN#: 2, Thread id: 24302, HiloPadre: 24300

|--[NIVEL 3] HiloN#: 3, Thread id: 24303, HiloPadre: 24300

|--[NIVEL 3] HiloN#: 4, Thread id: 24304, HiloPadre: 24300

|--[NIVEL 3] PRACTICA 5 (Thread id: 24305)

|--[NIVEL 2] HiloN#: 8, Thread id: 24306, HiloPadre: 24263

|--[NIVEL 3] HiloN#: 1, Thread id: 24307, HiloPadre: 24306

|--[NIVEL 3] HiloN#: 2, Thread id: 24308, HiloPadre: 24306

|--[NIVEL 3] HiloN#: 3, Thread id: 24309, HiloPadre: 24306

|--[NIVEL 3] HiloN#: 4, Thread id: 24310, HiloPadre: 24306

|--[NIVEL 3] PRACTICA 5 (Thread id: 24311)

|--[NIVEL 2] HiloN#: 9, Thread id: 24312, HiloPadre: 24263

|--[NIVEL 3] HiloN#: 1, Thread id: 24313, HiloPadre: 24312

|--[NIVEL 3] HiloN#: 2, Thread id: 24314, HiloPadre: 24312

|--[NIVEL 3] HiloN#: 3, Thread id: 24315, HiloPadre: 24312

|--[NIVEL 3] HiloN#: 4, Thread id: 24316, HiloPadre: 24312

|--[NIVEL 3] PRACTICA 5 (Thread id: 24317)

|--[NIVEL 2] HiloN#: 10, Thread id: 24318, HiloPadre: 24263

|--[NIVEL 3] HiloN#: 1, Thread id: 24319, HiloPadre: 24318

|--[NIVEL 3] HiloN#: 2, Thread id: 24320, HiloPadre: 24318

|--[NIVEL 3] HiloN#: 3, Thread id: 24321, HiloPadre: 24318

|--[NIVEL 3] HiloN#: 4, Thread id: 24322, HiloPadre: 24318

|--[NIVEL 3] PRACTICA 5 (Thread id: 24323)

[NIVEL 1] HiloN#: 15, Thread id: 24324, Padre: 23469

|--[NIVEL 2] HiloN#: 1, Thread id: 24325, HiloPadre: 24324

|--[NIVEL 3] HiloN#: 1, Thread id: 24326, HiloPadre: 24325

|--[NIVEL 3] HiloN#: 2, Thread id: 24327, HiloPadre: 24325

|--[NIVEL 3] HiloN#: 3, Thread id: 24328, HiloPadre: 24325

|--[NIVEL 3] HiloN#: 4, Thread id: 24329, HiloPadre: 24325

|--[NIVEL 3] PRACTICA 5 (Thread id: 24330)

|--[NIVEL 2] HiloN#: 2, Thread id: 24331, HiloPadre: 24324

|--[NIVEL 3] HiloN#: 1, Thread id: 24332, HiloPadre: 24331

|--[NIVEL 3] HiloN#: 2, Thread id: 24333, HiloPadre: 24331

|--[NIVEL 3] HiloN#: 3, Thread id: 24334, HiloPadre: 24331

|--[NIVEL 3] HiloN#: 4, Thread id: 24335, HiloPadre: 24331

|--[NIVEL 3] PRACTICA 5 (Thread id: 24336)

|--[NIVEL 2] HiloN#: 3, Thread id: 24337, HiloPadre: 24324

|--[NIVEL 3] HiloN#: 1, Thread id: 24338, HiloPadre: 24337

|--[NIVEL 3] HiloN#: 2, Thread id: 24339, HiloPadre: 24337

|--[NIVEL 3] HiloN#: 3, Thread id: 24340, HiloPadre: 24337

|--[NIVEL 3] HiloN#: 4, Thread id: 24341, HiloPadre: 24337

|--[NIVEL 3] PRACTICA 5 (Thread id: 24342)

|--[NIVEL 2] HiloN#: 4, Thread id: 24343, HiloPadre: 24324

|--[NIVEL 3] HiloN#: 1, Thread id: 24344, HiloPadre: 24343

|--[NIVEL 3] HiloN#: 2, Thread id: 24345, HiloPadre: 24343

|--[NIVEL 3] HiloN#: 3, Thread id: 24346, HiloPadre: 24343

|--[NIVEL 3] HiloN#: 4, Thread id: 24347, HiloPadre: 24343

|--[NIVEL 3] PRACTICA 5 (Thread id: 24348)

|--[NIVEL 2] HiloN#: 5, Thread id: 24349, HiloPadre: 24324

|--[NIVEL 3] HiloN#: 1, Thread id: 24350, HiloPadre: 24349

|--[NIVEL 3] HiloN#: 2, Thread id: 24351, HiloPadre: 24349

|--[NIVEL 3] HiloN#: 3, Thread id: 24352, HiloPadre: 24349

|--[NIVEL 3] HiloN#: 4, Thread id: 24353, HiloPadre: 24349

|--[NIVEL 3] PRACTICA 5 (Thread id: 24354)

|--[NIVEL 2] HiloN#: 6, Thread id: 24355, HiloPadre: 24324

|--[NIVEL 3] HiloN#: 1, Thread id: 24356, HiloPadre: 24355

|--[NIVEL 3] HiloN#: 2, Thread id: 24357, HiloPadre: 24355

|--[NIVEL 3] HiloN#: 3, Thread id: 24358, HiloPadre: 24355

|--[NIVEL 3] HiloN#: 4, Thread id: 24359, HiloPadre: 24355

|--[NIVEL 3] PRACTICA 5 (Thread id: 24360)

|--[NIVEL 2] HiloN#: 7, Thread id: 24361, HiloPadre: 24324

|--[NIVEL 3] HiloN#: 1, Thread id: 24362, HiloPadre: 24361

|--[NIVEL 3] HiloN#: 2, Thread id: 24363, HiloPadre: 24361

|--[NIVEL 3] HiloN#: 3, Thread id: 24364, HiloPadre: 24361

|--[NIVEL 3] HiloN#: 4, Thread id: 24365, HiloPadre: 24361

|--[NIVEL 3] PRACTICA 5 (Thread id: 24366)

|--[NIVEL 2] HiloN#: 8, Thread id: 24367, HiloPadre: 24324

|--[NIVEL 3] HiloN#: 1, Thread id: 24368, HiloPadre: 24367

|--[NIVEL 3] HiloN#: 2, Thread id: 24369, HiloPadre: 24367

|--[NIVEL 3] HiloN#: 3, Thread id: 24370, HiloPadre: 24367

|--[NIVEL 3] HiloN#: 4, Thread id: 24371, HiloPadre: 24367

|--[NIVEL 3] PRACTICA 5 (Thread id: 24372)

|--[NIVEL 2] HiloN#: 9, Thread id: 24373, HiloPadre: 24324

|--[NIVEL 3] HiloN#: 1, Thread id: 24374, HiloPadre: 24373

|--[NIVEL 3] HiloN#: 2, Thread id: 24375, HiloPadre: 24373

|--[NIVEL 3] HiloN#: 3, Thread id: 24376, HiloPadre: 24373

|--[NIVEL 3] HiloN#: 4, Thread id: 24377, HiloPadre: 24373

|--[NIVEL 3] PRACTICA 5 (Thread id: 24378)

|--[NIVEL 2] HiloN#: 10, Thread id: 24379, HiloPadre: 24324

|--[NIVEL 3] HiloN#: 1, Thread id: 24380, HiloPadre: 24379

|--[NIVEL 3] HiloN#: 2, Thread id: 24381, HiloPadre: 24379

|--[NIVEL 3] HiloN#: 3, Thread id: 24382, HiloPadre: 24379

|--[NIVEL 3] HiloN#: 4, Thread id: 24383, HiloPadre: 24379

|--[NIVEL 3] PRACTICA 5 (Thread id: 24384)

6. Programe la misma aplicación del punto 5 de la práctica 4 pero utilizando hilos. Así mismo, dé sus observaciones tanto de funcionamiento como de los tiempos de ejecución resultantes.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/syscall.h>

#include <sys/wait.h>

#include <sys/types.h>

#include "operadores.h"

#define SIZ\_MAX 3

pthread\_mutex\_t mutex=PTHREAD\_MUTEX\_INITIALIZER;

void \*hilosuma(void \*data);

void \*hiloresta(void \*data);

void \*hilomult(void \*data);

void \*hilotransp(void \*data);

void \*hiloinv(void \*data);

void \*hiloLector(void \*data);

struct datos {

int iii; //Estructura para pasar datos entre hilos.

}; //Estructuras para pasar datos entre proceso e hilos

int main() //void ImpMat(float matrix[SIZ\_MAX][SIZ\_MAX], int row, int column, char\* nombre, int nodecim, int nofile, int nomostrar);

{ //void CargarEnArreglo(char \*NombreArchivo, float matrix[SIZ\_MAX][SIZ\_MAX],int tam, int skiprows);

struct datos \*datos = malloc(sizeof(struct datos));

printf("PID PADRE: %d\n",getpid());

float a[SIZ\_MAX][SIZ\_MAX],b[SIZ\_MAX][SIZ\_MAX]; int contc=1;

for(int i=0;i<SIZ\_MAX;i++) //debido a que es dificil crear de manera

for(int j=0;j<SIZ\_MAX;j++) //automatizada una matriz 10x10

{ //donde todas las entradas sean

a[i][j]=contc; //Lineal.Independientes (para que exista una inversa)

b[i][j]=contc; //DEMO

contc++; //DEMO

} //DEMO

a[2][1]=b[2][1]=9; //<= se hacen cambios en ciertos indices

a[2][2]=b[2][2]=8; //<= para tener una matriz L.I

/\*printf("Inserte Matriz A: \n"); //IngresaMatrizA

InsEnMat(a,SIZ\_MAX,SIZ\_MAX); //Se ordena guardar matriz en un archivo, sin imprimirla a pantalla

ImpMat(a,SIZ\_MAX,SIZ\_MAX,"OriginalA",1,0,1);

printf("Inserte Matriz B: \n");

InsEnMat(b,SIZ\_MAX,SIZ\_MAX);

ImpMat(b,SIZ\_MAX,SIZ\_MAX,"OriginalB",1,0,1);\*/

pthread\_t thread[5]; //Handlers de los primeros 5 hilos

int hilonum[5];

hilonum[0]=pthread\_create(&(thread[0]),NULL, hilosuma, datos);

pthread\_join(thread[0], NULL); //Espera a que todos los hilos acaben

hilonum[1]=pthread\_create(&(thread[1]),NULL, hiloresta, datos);

pthread\_join(thread[1], NULL); //Espera a que todos los hilos acaben

hilonum[2]=pthread\_create(&(thread[2]),NULL, hilomult, datos);

pthread\_join(thread[2], NULL); //Espera a que todos los hilos acaben

hilonum[3]=pthread\_create(&(thread[3]),NULL, hilotransp, datos);

pthread\_join(thread[3], NULL); //Espera a que todos los hilos acaben

hilonum[4]=pthread\_create(&(thread[4]),NULL, hiloinv, datos);

pthread\_join(thread[4], NULL); //Espera a que todos los hilos acaben

hilonum[5]=pthread\_create(&(thread[5]),NULL, hiloLector, datos);

pthread\_join(thread[5], NULL); //Espera a que todos los hilos acaben

for(int i=0;i<6;i++)

if (hilonum[i])

printf("No fue posible crear un hilo.\n");

exit(0);

}

void \*hilosuma(void \*data)

{

struct datos \*datos = data;

printf("\n===SUMA=== [HILO | ThreadID: %d PID: %d]\n",ObtenerIDHilo(),getpid());

float a[SIZ\_MAX][SIZ\_MAX],b[SIZ\_MAX][SIZ\_MAX],c[SIZ\_MAX][SIZ\_MAX];

CargarEnArreglo("OriginalA",a,SIZ\_MAX,1);

CargarEnArreglo("OriginalB",b,SIZ\_MAX,1);

SumaOResta(a,b,c,SIZ\_MAX,SIZ\_MAX,1); //solo si es 1 las va sumar

ImpMat(c,SIZ\_MAX,SIZ\_MAX,"suma",1,0,0);

pthread\_exit(NULL); //Sale del hilo

}

void \*hiloresta(void \*data)

{

struct datos \*datos = data;

printf("\n===RESTA=== [HILO | ThreadID: %d PID: %d]\n",ObtenerIDHilo(),getpid());

float a[SIZ\_MAX][SIZ\_MAX],b[SIZ\_MAX][SIZ\_MAX],c[SIZ\_MAX][SIZ\_MAX];

CargarEnArreglo("OriginalA",a,SIZ\_MAX,1);

CargarEnArreglo("OriginalB",b,SIZ\_MAX,1);

SumaOResta(a,b,c,SIZ\_MAX,SIZ\_MAX,0); //solo si es 1 las va sumar

ImpMat(c,SIZ\_MAX,SIZ\_MAX,"resta",1,0,0);

printf("\n");

pthread\_exit(NULL); //Sale del hilo

}

void \*hilomult(void \*data)

{

struct datos \*datos = data;

printf("\n===MULTIPLICACION=== [HILO | ThreadID: %d PID: %d]\n",ObtenerIDHilo(),getpid());

float a[SIZ\_MAX][SIZ\_MAX],b[SIZ\_MAX][SIZ\_MAX],c[SIZ\_MAX][SIZ\_MAX];

CargarEnArreglo("OriginalA",a,SIZ\_MAX,1);

CargarEnArreglo("OriginalB",b,SIZ\_MAX,1);

Multiplicar(a,b,c);

ImpMat(c,SIZ\_MAX,SIZ\_MAX,"multiplicacion",1,0,0);

printf("\n");

pthread\_exit(NULL); //Sale del hilo

}

void \*hilotransp(void \*data)

{

struct datos \*datos = data;

printf("\n===TRANSPUESTAS=== [HILO | ThreadID: %d PID: %d]\n",ObtenerIDHilo(),getpid());

float a[SIZ\_MAX][SIZ\_MAX],b[SIZ\_MAX][SIZ\_MAX],c[SIZ\_MAX][SIZ\_MAX];

CargarEnArreglo("OriginalA",a,SIZ\_MAX,1);

CargarEnArreglo("OriginalB",b,SIZ\_MAX,1);

Transpuesta(a,c,SIZ\_MAX,SIZ\_MAX);

printf("Transpuesta de A:\n");

ImpMat(c,SIZ\_MAX,SIZ\_MAX,"TranspuestaA",1,0,0);

Transpuesta(b,c,SIZ\_MAX,SIZ\_MAX);

printf("Transpuesta de B:\n");

ImpMat(c,SIZ\_MAX,SIZ\_MAX,"TranspuestaB",1,0,0);

printf("\n");

pthread\_exit(NULL); //Sale del hilo

}

void \*hiloinv(void \*data)

{

struct datos \*datos = data;

printf("\n===INVERSAS=== [HILO | ThreadID: %d PID: %d]\n",ObtenerIDHilo(),getpid());

float a[SIZ\_MAX][SIZ\_MAX],b[SIZ\_MAX][SIZ\_MAX],c[SIZ\_MAX][SIZ\_MAX];

CargarEnArreglo("OriginalA",a,SIZ\_MAX,1);

CargarEnArreglo("OriginalB",b,SIZ\_MAX,1);

if(determ(a,SIZ\_MAX)!=0 && determ(b,SIZ\_MAX)!=0){

TreInversaCaller(a,c,SIZ\_MAX);

printf("\nInversa Matriz A:\n");

ImpMat(c,SIZ\_MAX,SIZ\_MAX,"InversaA",0,0,0);

TreInversaCaller(b,c,SIZ\_MAX);

printf("\nInversa Matriz B:\n");

ImpMat(c,SIZ\_MAX,SIZ\_MAX,"InversaB",0,0,0);

}

printf("\n");

pthread\_exit(NULL); //Sale del hilo

}

void \*hiloLector(void \*data)

{

struct datos \*datos = data;

printf("\n===LECTURA ARCHIVOS CREADOS=== [HILO | ThreadID: %d PID: %d]\n",ObtenerIDHilo(),getpid());

LeerArchivos("OriginalA");

LeerArchivos("OriginalB");

LeerArchivos("suma");

LeerArchivos("resta");

LeerArchivos("multiplicacion");

LeerArchivos("TranspuestaA");

LeerArchivos("TranspuestaB");

LeerArchivos("InversaA");

LeerArchivos("InversaB");

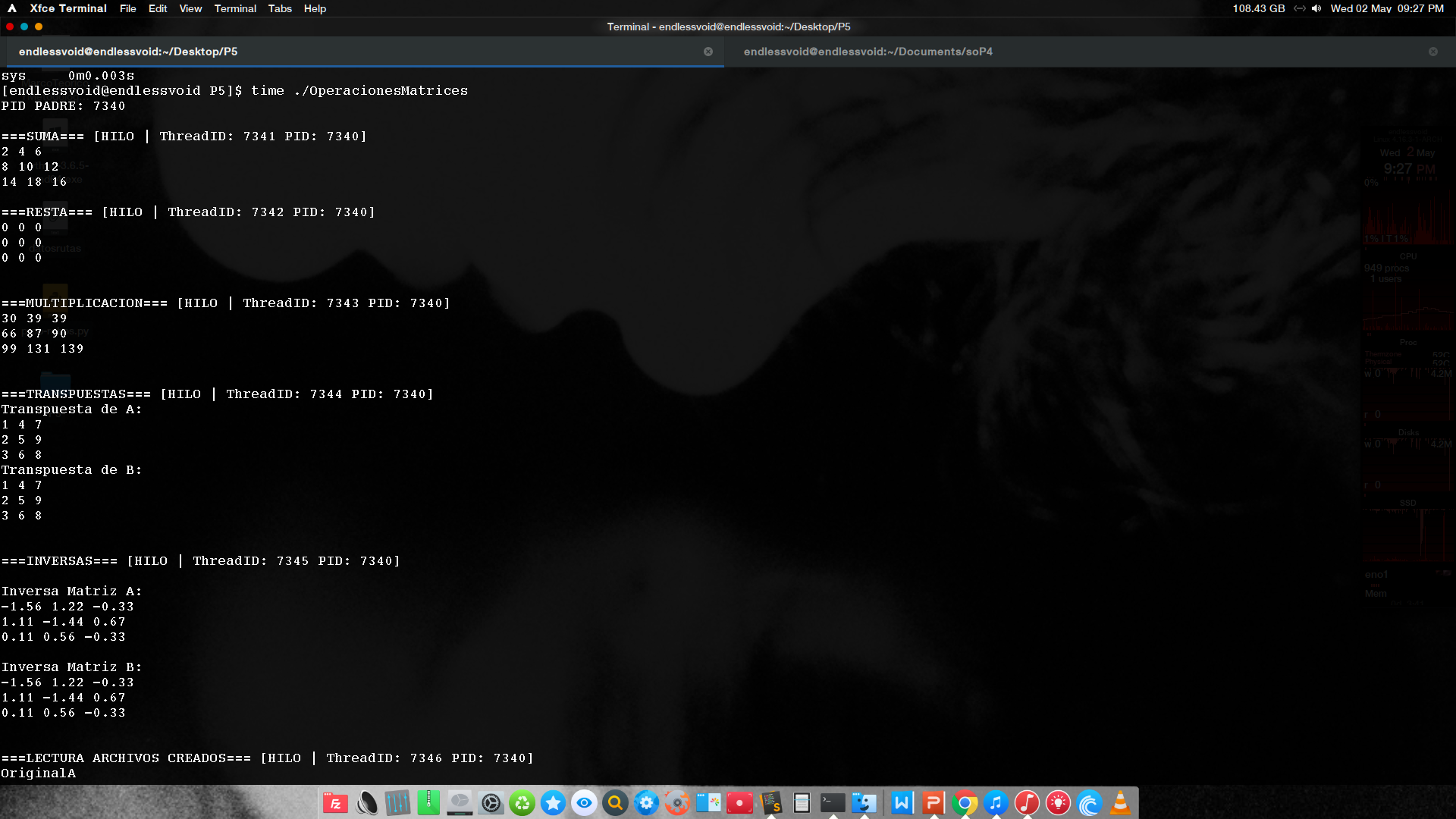
pthread\_exit(NULL); //Sale del hilo

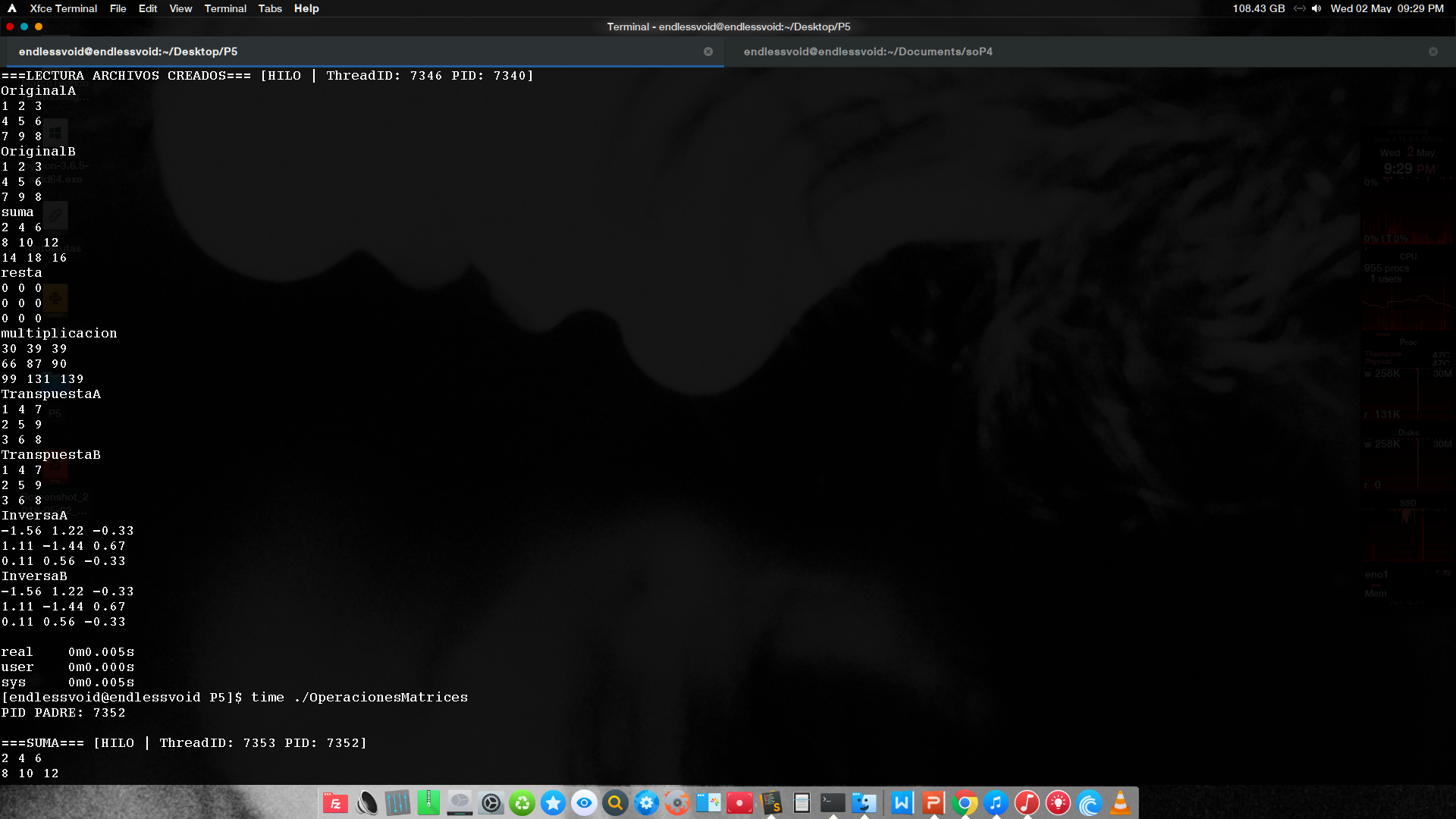
}

**Archivo de biblioteca donde se encuentran las funciones utilizadas en el programa:**

#include<stdio.h>  
#include<math.h>  
#include<string.h>  
#define SIZ\_MAX 3  
float determ(float a[SIZ\_MAX][SIZ\_MAX], float tam);  
// int < determ(matrizA, tamMatriz)  
void TreInversaCaller(float [SIZ\_MAX][SIZ\_MAX], float dest[SIZ\_MAX][SIZ\_MAX], float);  
// void < TreInversaCaller(matrizOrigen, matrizDestino, tamMatriz)  
void treinversa(float [][SIZ\_MAX], float [][SIZ\_MAX], float dest[][SIZ\_MAX], float); //llamada indirecta  
void ImpMat(float matrix[SIZ\_MAX][SIZ\_MAX], int row, int column, char\* nombre, int nodecim, int nofile, int nomostrar);  
// void < ImpMatriz (matrizAImprimir, TamFila, TamColumna, EtiquetaParaArchivo, NoMostrarDecimales(1), NoImprimirA-Archivo(1), NoMostrar(1))  
void InsEnMat(float matrix[SIZ\_MAX][SIZ\_MAX], int row, int column);  
// void < InsertarEnMatriz (matrizAInsertar, TamFila, TamColumna)  
void SumaOResta (float matrix[SIZ\_MAX][SIZ\_MAX], float matrix2[SIZ\_MAX][SIZ\_MAX], float matrixDest[SIZ\_MAX][SIZ\_MAX], int row, int column, int pain);  
// void < SumaOResta (matrizA, matrizB, matrizDestino, TamFila, TamColumna, FlagSuma(1))  
void Transpuesta(float matrix[SIZ\_MAX][SIZ\_MAX], float matrixDest[SIZ\_MAX][SIZ\_MAX], int row, int column);  
// void < transpuesta(matrizOrigen, matrizDestino, TamFila, TamColumna)  
void Multiplicar(float matrix[SIZ\_MAX][SIZ\_MAX], float matrix2[SIZ\_MAX][SIZ\_MAX], float matrixDest[SIZ\_MAX][SIZ\_MAX]);  
// void < Multiplicar(matrizA, matrizB, matrizDestino)  
void CargarEnArreglo(char \*NombreArchivo, float matrix[SIZ\_MAX][SIZ\_MAX],int tam, int skiprows);  
// void < CargarEnArreglo("nombreArchivo",matrizDestino, TamFila, TamColumna, SaltarNLineas)  
int ObtenerIDHilo(void );  
float determ(float a[SIZ\_MAX][SIZ\_MAX], float tam)  
{  
 float s = 1, det = 0, b[SIZ\_MAX][SIZ\_MAX];  
 int i, j, m, n, c;  
 if (tam == 1)  
 {  
 return (a[0][0]);  
 }  
 else  
 {  
 det = 0;  
 for (c = 0; c < tam; c++)  
 {  
 m = 0;  
 n = 0;  
 for (i = 0;i < tam; i++)  
 {  
 for (j = 0 ;j < tam; j++)  
 {  
 b[i][j] = 0;  
 if (i != 0 && j != c)  
 {  
 b[m][n] = a[i][j];  
 if (n < (tam - 2))  
 n++;  
 else  
 {  
 n = 0;  
 m++;  
 }  
 }  
 }  
 }  
 det = det + s \* (a[0][c] \* determ(b, tam - 1));  
 s = -1 \* s;  
 }  
 }  
 return (det);  
}  
void TreInversaCaller(float num[SIZ\_MAX][SIZ\_MAX],float dest[SIZ\_MAX][SIZ\_MAX], float f)  
{  
 float b[SIZ\_MAX][SIZ\_MAX], fac[SIZ\_MAX][SIZ\_MAX];  
 int p, q, m, n, i, j;  
 for (q = 0;q < f; q++)  
 {  
 for (p = 0;p < f; p++)  
 {  
 m = 0;  
 n = 0;  
 for (i = 0;i < f; i++)  
 {  
 for (j = 0;j < f; j++)  
 {  
 if (i != q && j != p)  
 {  
 b[m][n] = num[i][j];  
 if (n < (f - 2))  
 n++;  
 else  
 {  
 n = 0;  
 m++;  
 }  
 }  
 }  
 }  
 fac[q][p] = pow(-1, q + p) \* determ(b, f - 1);  
 }  
 }  
 treinversa(num, fac, dest, f);  
}  
void treinversa(float num[SIZ\_MAX][SIZ\_MAX], float fac[SIZ\_MAX][SIZ\_MAX], float dest[SIZ\_MAX][SIZ\_MAX], float r)  
{  
 int i, j;  
 float b[SIZ\_MAX][SIZ\_MAX], d;  
 for (i = 0;i < r; i++)  
 for (j = 0;j < r; j++)  
 {  
 b[i][j] = fac[j][i];  
 }  
 d = determ(num, r);  
for (i = 0;i < r; i++)  
 {  
 for (j = 0;j < r; j++)  
 {  
 dest[i][j] = b[i][j] / d;  
 }  
 }  
 }  
 void InsEnMat (float matrix[SIZ\_MAX][SIZ\_MAX], int row, int column)  
{  
 int i, j;  
 for (i=0;i<row;i++)  
 {  
  
 for (j=0;j<column;j++)  
 {  
 printf ("POS [%d | %d] ", i+1, j+1);  
 scanf ("%f", &matrix[i][j]);  
 }  
 }  
}  
void ImpMat (float matrix[SIZ\_MAX][SIZ\_MAX],int row, int column, char\* operacion, int nodecim, int nofile, int nomostrar)  
{  
 int i, j;  
 FILE \*NuevoArchivo;  
 if(nofile!=1)  
 NuevoArchivo = fopen(operacion, "w");  
 if(NuevoArchivo!=NULL && nofile!=1)  
 fprintf(NuevoArchivo,"%s \n",operacion);  
 for(i=0;i<row;i++)  
 {  
 for(j=0;j<column;j++)  
 {  
 if(nodecim==1)  
 {  
 if(nomostrar!=1)  
 printf("%.0lf ", matrix[i][j]);  
 if(NuevoArchivo!=NULL && nofile!=1)  
 fprintf(NuevoArchivo,"%.0lf ",matrix[i][j]);  
 }  
 else  
 {  
 if(nomostrar!=1)  
 printf("%.2lf ", matrix[i][j]);  
 if(NuevoArchivo!=NULL && nofile!=1)  
 fprintf(NuevoArchivo,"%.2lf ",matrix[i][j]);  
 }  
 }  
 if(nomostrar!=1)  
 printf("\n");  
 if(NuevoArchivo!=NULL && nofile!=1)  
 fprintf(NuevoArchivo,"\n");  
 }  
 if(NuevoArchivo!=NULL && nofile!=1)  
 fclose(NuevoArchivo);  
 }  
void SumaOResta (float matrix[SIZ\_MAX][SIZ\_MAX], float matrix2[SIZ\_MAX][SIZ\_MAX], float matrixDest[SIZ\_MAX][SIZ\_MAX], int row, int column, int pain)  
{  
 int i;  
 int j;  
 for(i=0;i<row;i++)  
 for(j=0;j<column;j++)  
 {  
 if(pain==1)  
 matrixDest[i][j]=matrix[i][j]+matrix2[i][j];  
 else  
 matrixDest[i][j]=matrix[i][j]-matrix2[i][j];  
 }  
}  
void Transpuesta(float matrix[SIZ\_MAX][SIZ\_MAX], float matrixDest[SIZ\_MAX][SIZ\_MAX], int row, int column)  
{  
 int i;  
 int j;  
 for(i=0;i<row;i++)  
 for(j=0;j<column;j++)  
 {  
 matrixDest[i][j]=matrix[j][i];  
 }  
}  
void Multiplicar(float matrix[SIZ\_MAX][SIZ\_MAX], float matrix2[SIZ\_MAX][SIZ\_MAX], float matrixDest[SIZ\_MAX][SIZ\_MAX])  
{  
 int i, j, k;  
 for (i=0;i<SIZ\_MAX; i++)  
 {  
 for (j=0;j<SIZ\_MAX;j++)  
 {  
 matrixDest[i][j] = 0;  
 for (k = 0; k < SIZ\_MAX; k++)  
 matrixDest[i][j] += matrix[i][k]\*matrix2[k][j];  
 }  
 }   
}  
void LeerArchivos(char \*NombreArchivo)  
{  
 char c; FILE \*PTRNomArch;  
 PTRNomArch = fopen(NombreArchivo, "r");  
 if (PTRNomArch == NULL)  
 {  
 printf("No se pudo abrir el archivo %s\n",NombreArchivo);  
 exit(0);  
 }  
 c = fgetc(PTRNomArch);  
 while (c != EOF)  
 {  
 printf ("%c", c);  
 c = fgetc(PTRNomArch);  
 }  
 fclose(PTRNomArch);  
}  
 void CargarEnArreglo(char \*NombreArchivo, float matrix[SIZ\_MAX][SIZ\_MAX], int tam, int skiprows)  
{  
 int counter=0, numberRead=0, i, j,cont=0;  
 int array[100];  
 FILE \*PTRNomArch;  
 PTRNomArch = fopen(NombreArchivo, "r");  
 if (PTRNomArch == NULL)  
 {  
 printf("No se pudo abrir el archivo %s\n",NombreArchivo);  
 exit(0);  
 }  
 char buffer[100]; //la primera linea se asume que es de 100   
 if (skiprows>0)  
 for(int rowskipped=0;rowskipped<skiprows;rowskipped++)  
 fgets(buffer, 100, PTRNomArch);  
 char arr[666];  
 char\* ptr;  
 fread(arr , 1,sizeof arr , PTRNomArch);  
 ptr = strtok(arr , " ");  
 while(ptr)  
 {  
 array[counter++] = strtol(ptr , NULL , 10);  
 ++numberRead;  
 ptr = strtok(NULL , " ");  
 }  
 for(i=0;i<tam;i++)  
 for(j=0;j<tam;j++)  
 {  
 matrix[i][j]=(double)array[cont];  
 cont++;  
 }   
 fclose(PTRNomArch);  
}   
  
int ObtenerIDHilo(void ){  
 pid\_t threadid = syscall(SYS\_gettid);   
 return threadid; //Sirve para obtener el identificador del hilo  
}

Capturas de pantalla:

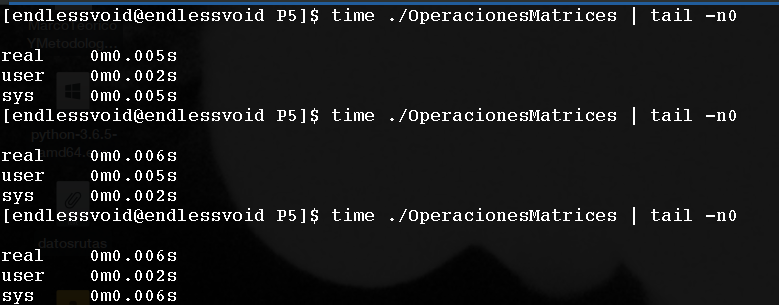




**NOTA: Similar al programa en la práctica 4, se utilizó una matriz de 3x3 *hardcodeada* que es linealmente independiente para poder probar las funciones de matriz inversa. El código soporta matrices de hasta 10x10 pero esto haría más tediosas las pruebas de tiempo de ejecución, además de que es poco práctico *hardcodear* una matriz de tales magnitudes para probar los tiempos de ejecución.**

Observaciones de la práctica (utilizando hilos e hijos):

**Tiempo de ejecución utilizando hilos:**



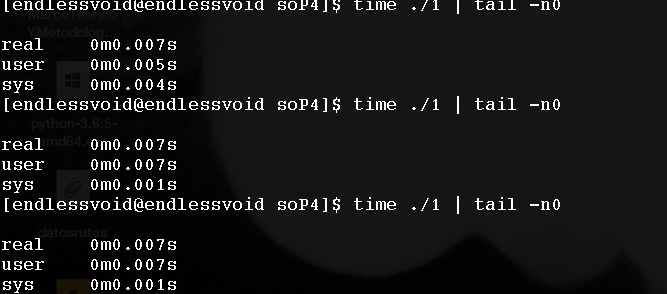
Primera ejecución: .002s + .005s = .007s

Segunda ejecución: .005s + .002s = .007s

Tercera ejecución: .006s + .002s = .008s

Promedio: 0.0073s (7.3ms)

**Tiempo de ejecución utilizando hijos (programa 5 de la práctica 4):**



Primera ejecución: .005s + .004s = .009s

Segunda ejecución: .007s + .001s = .008s

Tercera ejecución: .007s + .001s = .008s

Promedio: .0083s (8.3ms)

Se pudo observar que con los hijos el procesamiento tardó ligeramente más, a pesar de que, aún leyendo datos desde disco utilizando hilos estos tardaron menos. Podemos concluir que hay menos *overhead* al utilizar hilos y que el procesador se utiliza de manera más eficiente.

7.- Programe una aplicación (tanto en Linux como en Windows) que copie los archivos y directorios contenidos dentro de una ruta específica. Por cada directorio que se encuentre al momento de copiar, se deberá crear un hilo que se encargará de copiar los archivos existentes en ese directorio. Nuevamente, si se encuentra otro directorio se creará otro hilo, así sucesivamente. Todos los hilos deberán correr concurrentemente. Las rutas origen y destino de copia se aceptará

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <string.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <dirent.h>

#include <fcntl.h>

#define BUFFER 1024

struct copia

{

const char \*ruta\_copia;

const char \*ruta\_original;

};

typedef struct copia hilo\_args;

struct stat st = {0};

int copiar(const char \*original, const char \*copia );

void directorio( const char \*origen, const char \*copia);

void \*hilo( void \*rutas );

int main( int argc, char \*argv[] ) {

if( argc == 3 ) {

const char \* nombre\_dir = argv[1];

const char \* nombre\_copia = argv[2];

directorio(nombre\_dir, nombre\_copia);

}

else

printf("Primer argumento: Directorio a copiar\n Segundo argumento: Directorio destino (puede no existir).\n");

return -1;

}

void directorio( const char \*origen, const char \*copia ){

DIR \* dirp;

struct dirent \* dir;

if(!(dirp=opendir(origen)))

printf( "%s\n", "No se puede acceder" );

else{ //En caso de que no exista el directorio destino

if (stat(copia, &st)==-1)

{

mkdir(copia, 0700);

printf("[OK/Directorio] Directorio '%s' creado\n",copia);

}

else

printf("[INFO/Directorio] '%s' ya existe. \n",copia);

hilo\_args args;

args.ruta\_original = origen;

args.ruta\_copia = copia;

int devolucion\_hilo = 0;

pthread\_t id\_hilo;

pthread\_create( &id\_hilo, NULL, hilo, ( void \* )&args );

pthread\_join( id\_hilo, (void\*)&devolucion\_hilo );

}

while((dir=readdir(dirp))!= NULL )

if(dir->d\_type == DT\_DIR &&

strcmp(dir->d\_name, "." ) != 0 && //omitir el directorio padre

strcmp(dir->d\_name, ".." ) != 0 ) // y las carpetas u archivos ocultos

{

int longitud = strlen(origen)+strlen(dir->d\_name)+2;

int longitud\_copia=strlen(copia)+strlen(dir->d\_name)+2;

char ruta[longitud];

char ruta\_copia[longitud\_copia];

strcpy(ruta, origen ); //Se concatena la ruta relativa del archivo a copiar

strcat(ruta, "/" );

strcat(ruta, dir -> d\_name);

strcat(ruta\_copia, copia); //Se concatena la ruta relativa del archivo destino

strcat(ruta\_copia, "/");

strcat(ruta\_copia, dir -> d\_name);

directorio(ruta, ruta\_copia);

}

closedir( dirp );

}

void \* hilo(void \* rutas ) {

hilo\_args \* args = rutas;

DIR \*dirp;

struct dirent \* dir;

//Si existe la ruta o 'path'

if( !( dirp = opendir( args -> ruta\_original ) ) ){

printf("[ERROR] No se puede acceder a %s",args->ruta\_original);

exit(-1);

pthread\_exit(NULL); //Sale del hilo

}

while( ( dir = readdir( dirp ) ) != NULL )

if( dir -> d\_type == DT\_REG ){

int longitud\_original = strlen( args -> ruta\_original ) + strlen( dir -> d\_name ) + 2;

int longitud\_copia = strlen( args -> ruta\_copia ) + strlen( dir -> d\_name ) + 2;

char nueva\_ruta\_original[ longitud\_original ];

char nueva\_ruta\_copia[ longitud\_copia ];

strcpy(nueva\_ruta\_original, args -> ruta\_original );

strcat(nueva\_ruta\_original, "/" );

strcat(nueva\_ruta\_original, dir->d\_name );

strcpy(nueva\_ruta\_copia, args -> ruta\_copia );

strcat(nueva\_ruta\_copia, "/" );

strcat(nueva\_ruta\_copia, dir->d\_name );

if( copiar(nueva\_ruta\_original,nueva\_ruta\_copia ) )

printf("[OK/Archivo] '%s' copiado correctamente\n", nueva\_ruta\_original);

else

printf("[ERROR] No se ha podido copiar %s",nueva\_ruta\_original);

}

closedir( dirp );

pthread\_exit(NULL);

}

int copiar( const char \* original, const char \* copia ){

int retorno = 1;

int archivo\_1 = open( original, O\_RDONLY );

if( archivo\_1 != -1 ) {

int longitud;

char buffer[ BUFFER ];

int archivo\_2 = open( copia, O\_RDWR|O\_CREAT, 0666 );

while( ( longitud = read( archivo\_1, buffer, BUFFER ) ) )

if( archivo\_2 != -1 )

if( write( archivo\_2, buffer, longitud ) != longitud )

retorno = 0;

close( archivo\_1 );

close( archivo\_2 );

}else{

printf( "No se pudo abrir el archivo: %s\n", original );

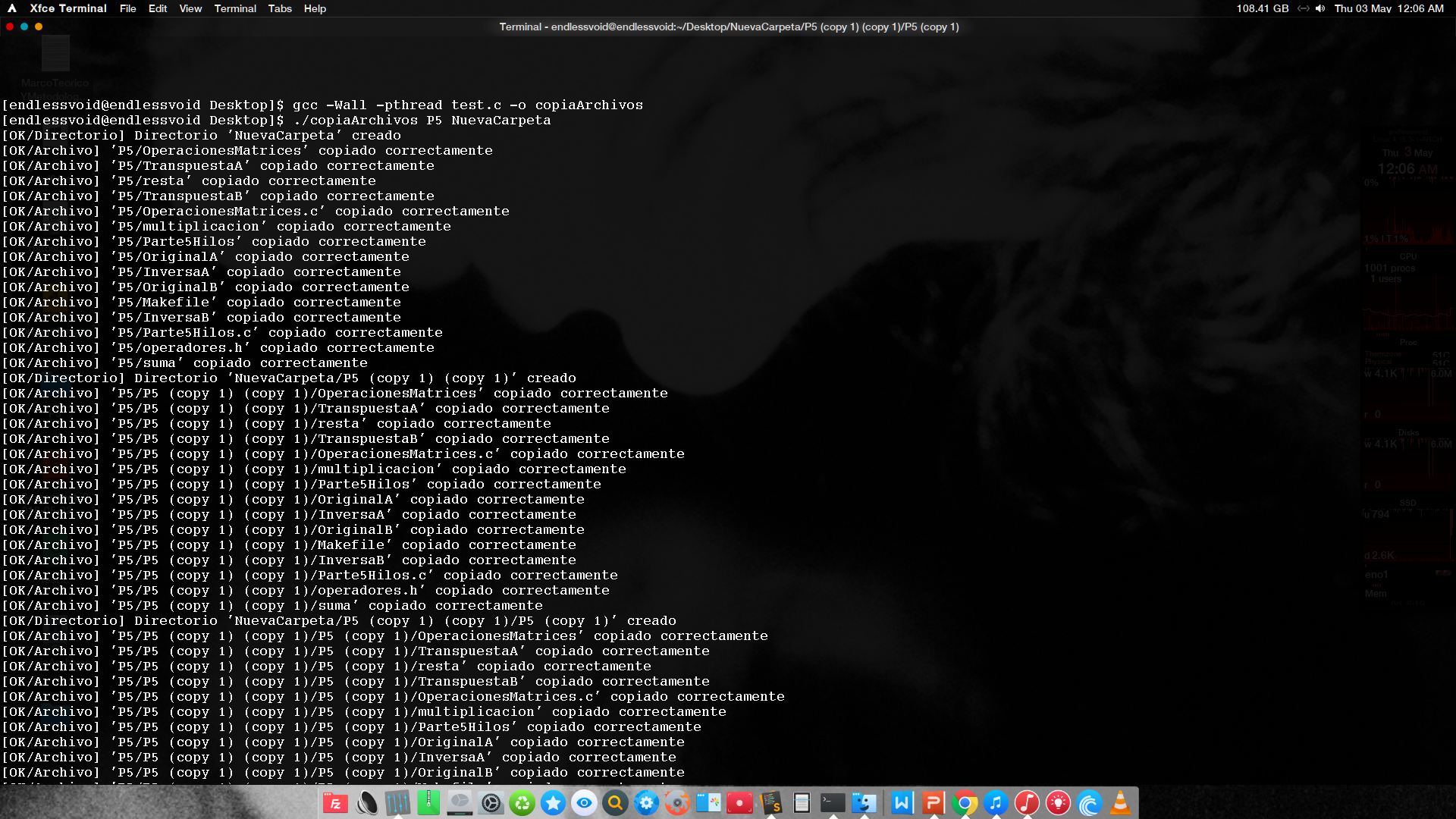
return 0;

}

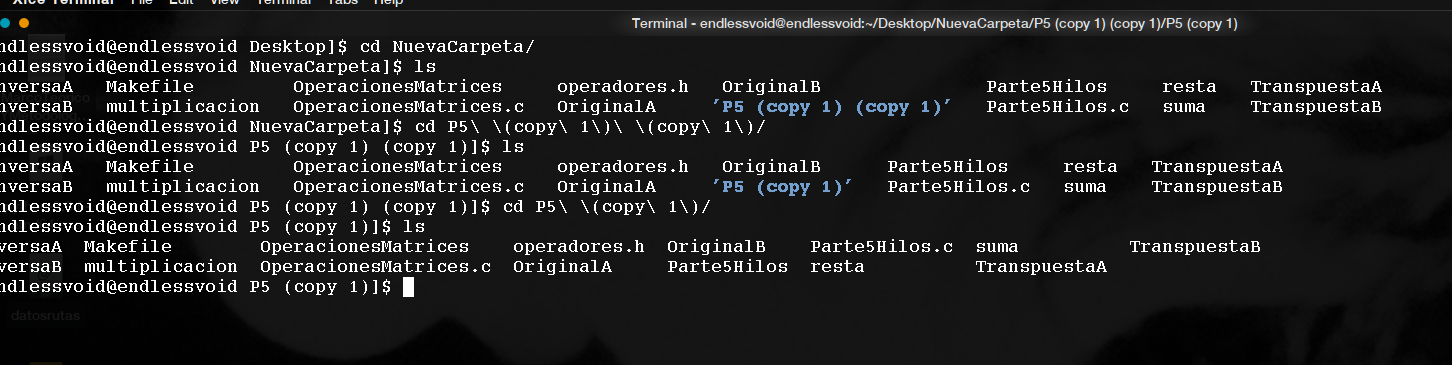
return retorno;

}

**Captura de pantalla:**

****

Copia de archivos en progreso.

****

Verificación de que en efecto, los nuevos archivos estuvieran ahí.

****

**Ejemplo:**

**#include <windows.h>**

**#include <stdio.h>**

**DWORD WINAPI funcionHilo(LPVOID IpParam);**

**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;**

**//Creación del hilo**

**manHilo = CreateThread(NULL, 0, funcionHilo, &argumentos, 0, &idHilo);**

**//Espera la finalzación del hilo**

**WaitForSingleObject(manHilo, INFINITE);**

**printf("Valores al salir del hilo: %i %i\n", argumentos.val\_1, argumentos.val\_2);**

**//Cierre del manejador de hilo creado**

**CloseHandle(manHilo);**

**return 0;**

**}**

**DWORD WINAPI funcionHilo(LPVOID IpParam)**

**{**

**info \*datos=(info \*)IpParam;**

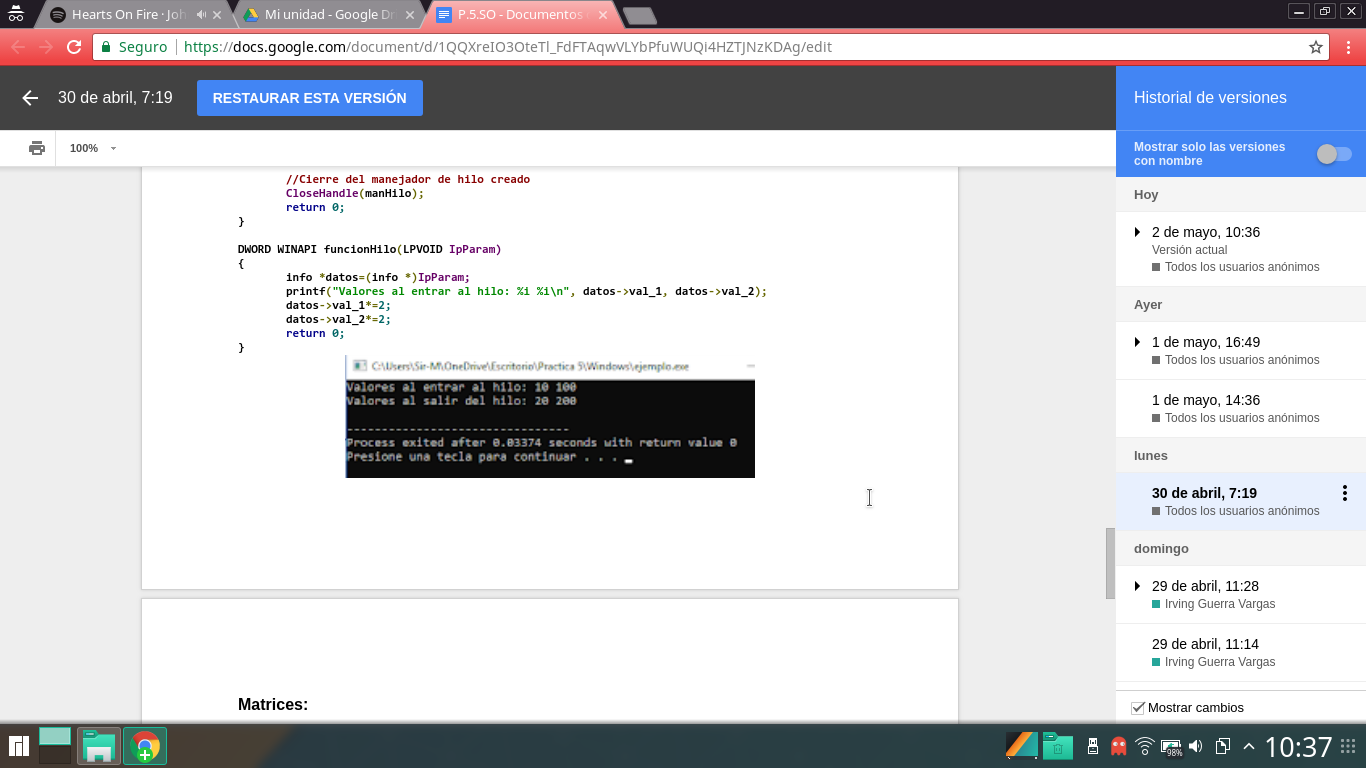
**printf("Valores al entrar al hilo: %i %i\n", datos->val\_1, datos->val\_2);**

**datos->val\_1\*=2;**

**datos->val\_2\*=2;**

**return 0;**

**}**

****

**Matrices:**

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

#include <windows.h>

#include <math.h>

typedef struct informacion info;

float determinant(float a[10][10], float k);

void cofactor(float num[10][10], float f,int a);

void transpose(float num[10][10], float fac[10][10], float r,int a);

struct informacion

{

DWORD idHilo;

};

float determinant(float a[10][10], float k)

{

float s = 1, det = 0, b[10][10];

int i, j, m, n, c;

if (k == 1)

{

return (a[0][0]);

}

else

{

det = 0;

for (c = 0; c < k; c++)

{

m = 0;

n = 0;

for (i = 0;i < k; i++)

{

for (j = 0 ;j < k; j++)

{

b[i][j] = 0;

if (i != 0 && j != c)

{

b[m][n] = a[i][j];

if (n < (k - 2))

n++;

else

{

n = 0;

m++;

}

}

}

}

det = det + s \* (a[0][c] \* determinant(b, k - 1));

s = -1 \* s;

}

}

return (det);

}

void cofactor(float num[10][10], float f,int a)

{

float b[10][10], fac[10][10];

int p, q, m, n, i, j;

for (q = 0;q < f; q++)

{

for (p = 0;p < f; p++)

{

m = 0;

n = 0;

for (i = 0;i < f; i++)

{

for (j = 0;j < f; j++)

{

if (i != q && j != p)

{

b[m][n] = num[i][j];

if (n < (f - 2))

n++;

else

{

n = 0;

m++;

}

}

}

}

fac[q][p] = pow(-1, q + p) \* determinant(b, f - 1);

}

}

transpose(num, fac, f,a);

}

void transpose(float num[10][10], float fac[10][10], float r,int a)

{

FILE \*ptr\_file;

int i, j;

float b[10][10], inverse[10][10], d;

for (i = 0;i < r; i++)

{

for (j = 0;j < r; j++)

{

b[i][j] = fac[j][i];

}

}

d = determinant(num, r);

for (i = 0;i < r; i++)

{

for (j = 0;j < r; j++)

{

inverse[i][j] = b[i][j] / d;

}

}

if(a==1){

ptr\_file =fopen("inversa1.txt", "w");

for (i = 0;i < r; i++){

for (j = 0;j < r; j++){

fprintf(ptr\_file,"%.3f\t" ,inverse[i][j]);

}

fprintf(ptr\_file, "\n");

}

fclose(ptr\_file);

printf("Archivo matriz inversa 1 guardado\n");

}else{

ptr\_file =fopen("inversa2.txt", "w");

for (i = 0;i < r; i++){

for (j = 0;j < r; j++){

fprintf(ptr\_file,"%.3f\t" ,inverse[i][j]);

}

fprintf(ptr\_file, "\n");

}

fclose(ptr\_file);

printf("Archivo matriz inversa 2 guardado\n");

}

}

DWORD WINAPI Suma(LPVOID lpParam){

int matrixa[10][10],matrixb[10][10],k,j;

FILE \*ptr\_file;

char buff;

ptr\_file =fopen("matriz1.txt", "r");

k=j=0;

while((buff = getc(ptr\_file)) != EOF){

if(buff=='0'|buff=='1'|buff=='2'|buff=='3'|buff=='4'|buff=='5'|buff=='6'|buff=='7'|buff=='8'|buff=='9'){

matrixa[k][j]=buff-'0';

j++;

if(j==10){

j=0;

k++;

}

}

}

fclose(ptr\_file);

ptr\_file =fopen("matriz2.txt", "r");

j=0;k=0;

while((buff = getc(ptr\_file)) != EOF){

if(buff=='0'|buff=='1'|buff=='2'|buff=='3'|buff=='4'|buff=='5'|buff=='6'|buff=='7'|buff=='8'|buff=='9'){

matrixb[k][j]=buff-'0';

j++;

if(j==10){

j=0;

k++;

}

}

}

fclose(ptr\_file);

ptr\_file =fopen("suma.txt", "w");

for(k=0;k<10;k++){

for(j=0;j<10;j++){

fprintf(ptr\_file,"%d\t" ,matrixa[k][j]+matrixb[k][j]);

}

fprintf(ptr\_file, "\n");

printf("\n");

}

fclose(ptr\_file);

printf("Archivo suma guardado\n");

}

DWORD WINAPI Resta(LPVOID lpParam){

int matrixa[10][10],matrixb[10][10],k,j;

FILE \*ptr\_file;

char buff;

ptr\_file =fopen("matriz1.txt", "r");

k=j=0;

while((buff = getc(ptr\_file)) != EOF){

if(buff=='0'|buff=='1'|buff=='2'|buff=='3'|buff=='4'|buff=='5'|buff=='6'|buff=='7'|buff=='8'|buff=='9'){

matrixa[k][j]=buff-'0';

j++;

if(j==10){

j=0;

k++;

}

}

}

fclose(ptr\_file);

ptr\_file =fopen("matriz2.txt", "r");

j=0;k=0;

while((buff = getc(ptr\_file)) != EOF){

if(buff=='0'|buff=='1'|buff=='2'|buff=='3'|buff=='4'|buff=='5'|buff=='6'|buff=='7'|buff=='8'|buff=='9'){

matrixb[k][j]=buff-'0';

j++;

if(j==10){

j=0;

k++;

}

}

}

fclose(ptr\_file);;

ptr\_file =fopen("resta.txt", "w");

for(k=0;k<10;k++){

for(j=0;j<10;j++){

fprintf(ptr\_file,"%d\t" ,matrixa[k][j]-matrixb[k][j]);

}

fprintf(ptr\_file, "\n");

}

fclose(ptr\_file);

printf("Archivo resta guardado\n");

}

DWORD WINAPI Producto(LPVOID lpParam){

int matrixa[10][10],matrixb[10][10],k,j;

FILE \*ptr\_file;

char buff;

ptr\_file =fopen("matriz1.txt", "r");

k=j=0;

while((buff = getc(ptr\_file)) != EOF){

if(buff=='0'|buff=='1'|buff=='2'|buff=='3'|buff=='4'|buff=='5'|buff=='6'|buff=='7'|buff=='8'|buff=='9'){

matrixa[k][j]=buff-'0';

j++;

if(j==10){

j=0;

k++;

}

}

}

fclose(ptr\_file);

ptr\_file =fopen("matriz2.txt", "r");

j=0;k=0;

while((buff = getc(ptr\_file)) != EOF){

if(buff=='0'|buff=='1'|buff=='2'|buff=='3'|buff=='4'|buff=='5'|buff=='6'|buff=='7'|buff=='8'|buff=='9'){

matrixb[k][j]=buff-'0';

j++;

if(j==10){

j=0;

k++;

}

}

}

fclose(ptr\_file);

ptr\_file =fopen("prod.txt", "w");

for(k=0;k<10;k++){

for(j=0;j<10;j++){

fprintf(ptr\_file,"%d\t" ,matrixa[k][j]\*matrixb[j][k]);

}

fprintf(ptr\_file, "\n");

}

fclose(ptr\_file);

printf("Archivo producto guardado\n");

}

DWORD WINAPI Transpuesta(LPVOID lpParam){

int matrixa[10][10],matrixb[10][10],k,j;

FILE \*ptr\_file;

char buff;

ptr\_file =fopen("matriz1.txt", "r");

k=j=0;

while((buff = getc(ptr\_file)) != EOF){

if(buff=='0'|buff=='1'|buff=='2'|buff=='3'|buff=='4'|buff=='5'|buff=='6'|buff=='7'|buff=='8'|buff=='9'){

matrixa[k][j]=buff-'0';

j++;

if(j==10){

j=0;

k++;

}

}

}

fclose(ptr\_file);

ptr\_file =fopen("matriz2.txt", "r");

j=0;k=0;

while((buff = getc(ptr\_file)) != EOF){

if(buff=='0'|buff=='1'|buff=='2'|buff=='3'|buff=='4'|buff=='5'|buff=='6'|buff=='7'|buff=='8'|buff=='9'){

matrixb[k][j]=buff-'0';

j++;

if(j==10){

j=0;

k++;

}

}

}

fclose(ptr\_file);

ptr\_file =fopen("trans1.txt", "w");

for(k=0;k<10;k++){

for(j=0;j<10;j++){

fprintf(ptr\_file,"%d\t" ,matrixa[j][k]);

}

fprintf(ptr\_file, "\n");

}

fclose(ptr\_file);

printf("Archivo matriz transpuesta 1 guardado\n");

ptr\_file =fopen("trans2.txt", "w");

for(k=0;k<10;k++){

for(j=0;j<10;j++){

fprintf(ptr\_file,"%d\t" ,matrixb[j][k]);

}

fprintf(ptr\_file, "\n");

}

fclose(ptr\_file);

printf("Archivo matriz transpuesta 2 guardado\n");

}

DWORD WINAPI Inversa(LPVOID lpParam){

float matrixa[10][10],matrixb[10][10];

float d;

int k,j;

int m = 10;

FILE \*ptr\_file;

char buff;

ptr\_file =fopen("matriz1.txt", "r");

k=j=0;

while((buff = getc(ptr\_file)) != EOF){

if(buff=='0'|buff=='1'|buff=='2'|buff=='3'|buff=='4'|buff=='5'|buff=='6'|buff=='7'|buff=='8'|buff=='9'){

matrixa[k][j]=(float)buff-'0';

j++;

if(j==10){

j=0;

k++;

}

}

}

fclose(ptr\_file);

ptr\_file =fopen("matriz2.txt", "r");

j=0;k=0;

while((buff = getc(ptr\_file)) != EOF){

if(buff=='0'|buff=='1'|buff=='2'|buff=='3'|buff=='4'|buff=='5'|buff=='6'|buff=='7'|buff=='8'|buff=='9'){

matrixb[k][j]=(float)buff-'0';

j++;

if(j==10){

j=0;

k++;

}

}

}

fclose(ptr\_file);

d = determinant(matrixa, 10);

if (d == 0)

printf("\nLa inversa de A no existe\n");

else

cofactor(matrixa, 10,1);

d = determinant(matrixb,10);

if (d == 0)

printf("\nLa inversa de B no existe\n");

else

cofactor(matrixb, 10,2);

}

DWORD WINAPI Imprime(LPVOID lpParam){

FILE \*ptr\_file;

char buff;

ptr\_file =fopen("suma.txt", "r");

printf("Suma de matrices\n");

while((buff = getc(ptr\_file)) != EOF){

printf("%c",buff);

}

fclose(ptr\_file);

printf("\n");

ptr\_file =fopen("resta.txt", "r");

printf("resta de matrices\n");

while((buff = getc(ptr\_file)) != EOF){

printf("%c",buff);

}

fclose(ptr\_file);

printf("\n");

ptr\_file =fopen("prod.txt", "r");

printf("producto de matrices\n");

while((buff = getc(ptr\_file)) != EOF){

printf("%c",buff);

}

fclose(ptr\_file);

printf("\n");

ptr\_file =fopen("trans1.txt", "r");

printf("matriz transpuesta 1\n");

while((buff = getc(ptr\_file)) != EOF){

printf("%c",buff);

}

fclose(ptr\_file);

printf("\n");

ptr\_file =fopen("trans2.txt", "r");

printf("matriz transpuesta 2\n");

while((buff = getc(ptr\_file)) != EOF){

printf("%c",buff);

}

fclose(ptr\_file);

printf("\n");

ptr\_file =fopen("inversa1.txt", "r");

printf("matriz inversa 1\n");

while((buff = getc(ptr\_file)) != EOF){

printf("%c",buff);

}

fclose(ptr\_file);

printf("\n");

ptr\_file =fopen("inversa2.txt", "r");

printf("matriz inversa 2\n");

while((buff = getc(ptr\_file)) != EOF){

printf("%c",buff);

}

fclose(ptr\_file);

printf("\n");

}

int main(int argc, char \*argv[])

{

FILE \*ptr\_file;

char matrixa[10][10],matrixb[10][10],k=0,j=0;

srand(time(NULL));

//Creación de matrices de manera aleatoria

printf("Matriz 1\n");

ptr\_file =fopen("matriz1.txt", "w");

for(k=0;k<10;k++){

for(j=0;j<10;j++){

matrixa[k][j]=rand()%9;

printf("%d\t",matrixa[k][j]);

fprintf(ptr\_file, "%d\t" ,matrixa[k][j]);

}

fprintf(ptr\_file,"\n");

printf("\n");

}

fclose(ptr\_file);

printf("Matriz 2\n");

ptr\_file =fopen("matriz2.txt", "w");

for(k=0;k<10;k++){

for(j=0;j<10;j++){

matrixb[k][j]=rand()%9;

printf("%d\t",matrixb[k][j]);

fprintf(ptr\_file, "%d\t" ,matrixb[k][j]);

}

fprintf(ptr\_file,"\n");

printf("\n");

}

fclose(ptr\_file);

HANDLE manHilo,manHilo2,manHilo3,manHilo4,manHilo5,manHilo6; /\*Manejador de hilo\*/

info argumentos,argumentos2,argumentos3,argumentos4,argumentos5,argumentos6;

//Creación de hilo 1

manHilo = CreateThread(NULL, 0, Suma, &argumentos,0,&(argumentos.idHilo));

WaitForSingleObject(manHilo, INFINITE);

//Terminar Hilo 1

CloseHandle(manHilo);

//Creación de hilo 2

manHilo2 = CreateThread(NULL, 0, Resta, &argumentos2,0,&(argumentos2.idHilo));

WaitForSingleObject(manHilo2, INFINITE);

//Terminar hilo 2

CloseHandle(manHilo2);

// Creación de hilo 3

manHilo3 = CreateThread(NULL, 0, Producto, &argumentos3,0,&(argumentos3.idHilo));

WaitForSingleObject(manHilo3, INFINITE);

//Terminar Hilo 3

CloseHandle(manHilo3);

//Creación de hilo 4

manHilo4 = CreateThread(NULL, 0, Transpuesta, &argumentos4,0,&(argumentos4.idHilo));

WaitForSingleObject(manHilo4, INFINITE);

//Terminar Hilo 4

CloseHandle(manHilo4);

// Creación de hilo 5

manHilo5 = CreateThread(NULL, 0, Inversa, &argumentos5,0,&(argumentos5.idHilo));

WaitForSingleObject(manHilo5, INFINITE);

//Terminar Hilo 5

CloseHandle(manHilo5);

//Creación de hilo 6

manHilo6 = CreateThread(NULL, 0, Imprime, &argumentos6,0,&(argumentos6.idHilo));

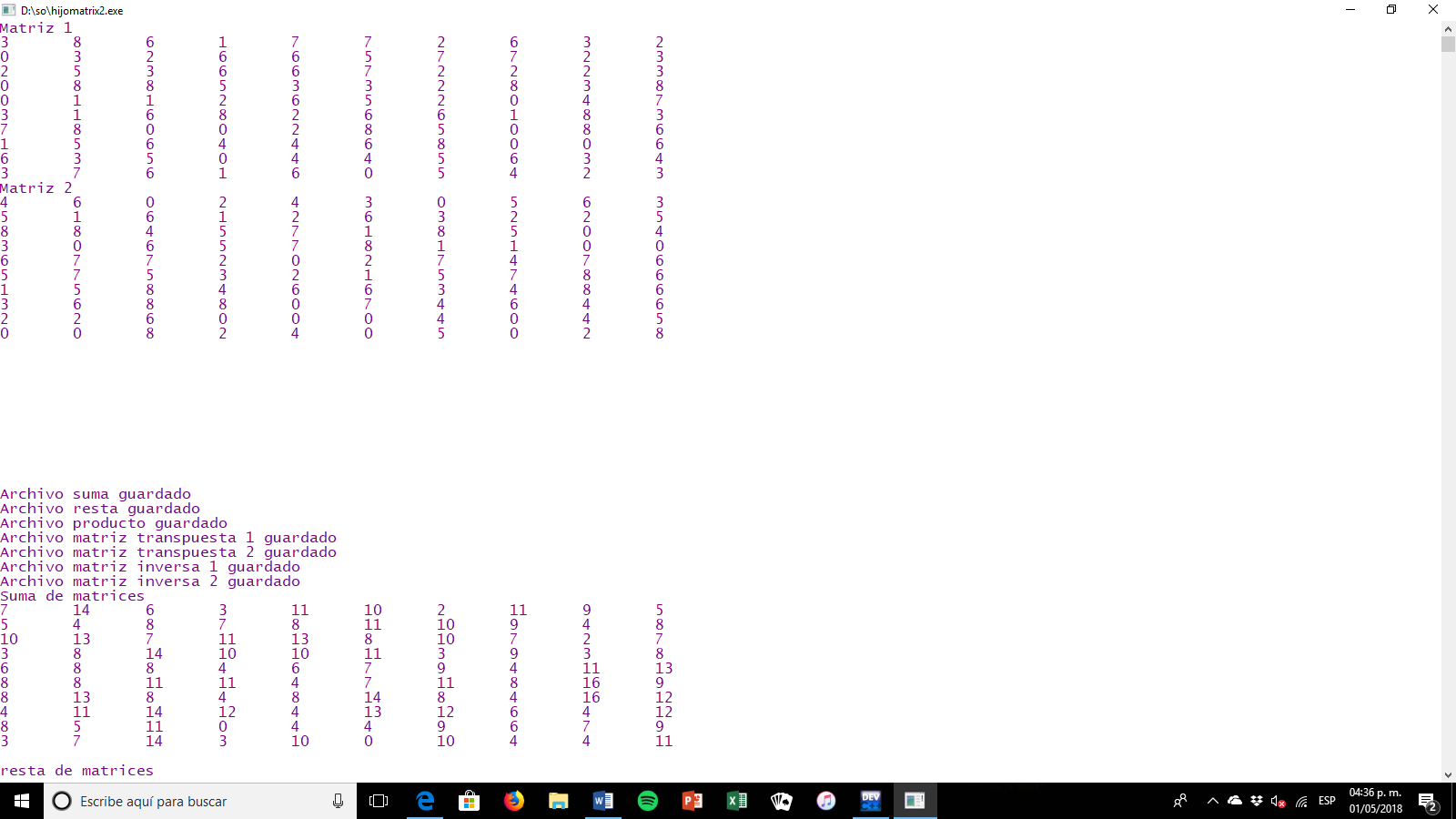
WaitForSingleObject(manHilo6, INFINITE);

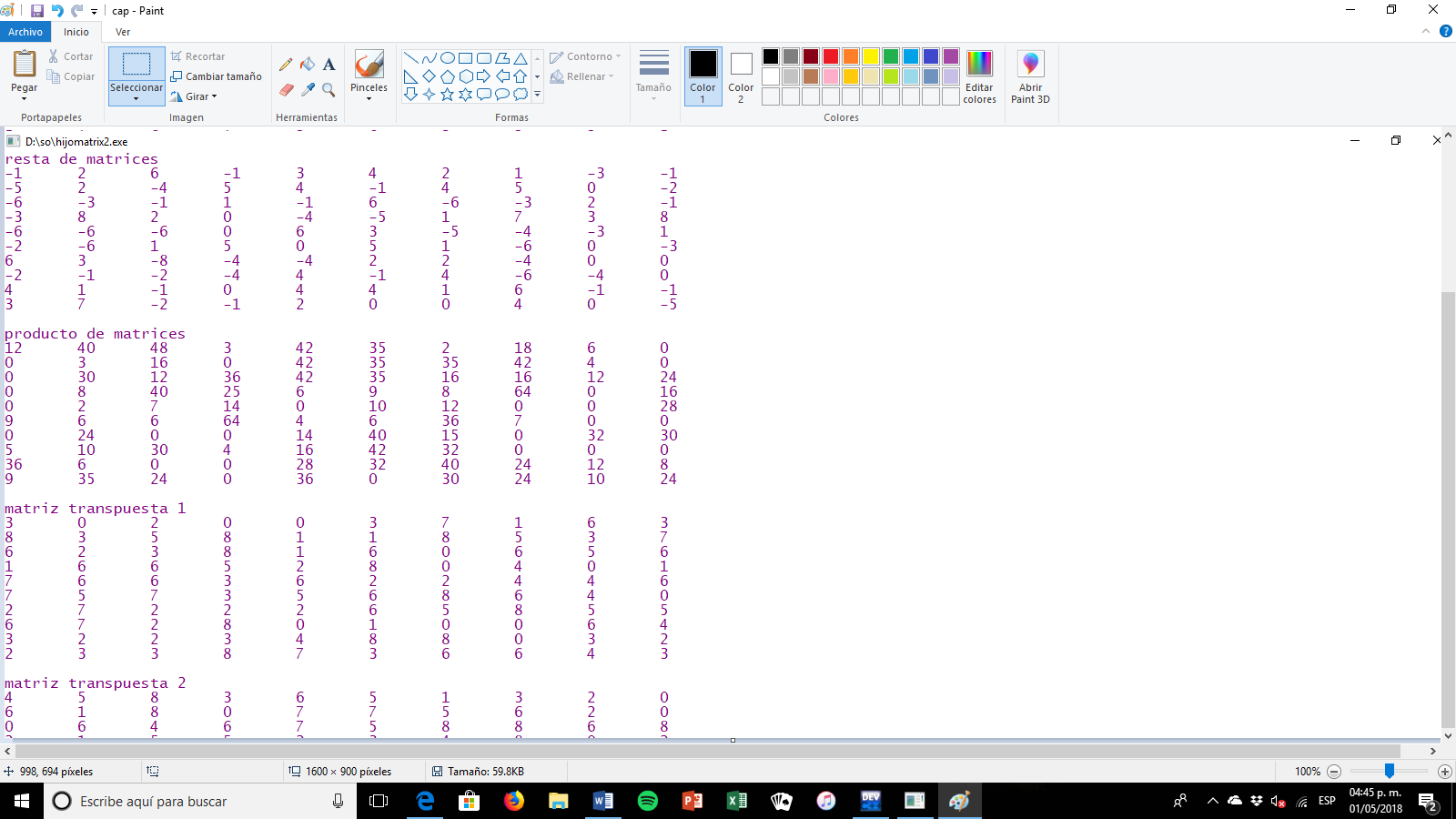
//Terminar Hilo 6

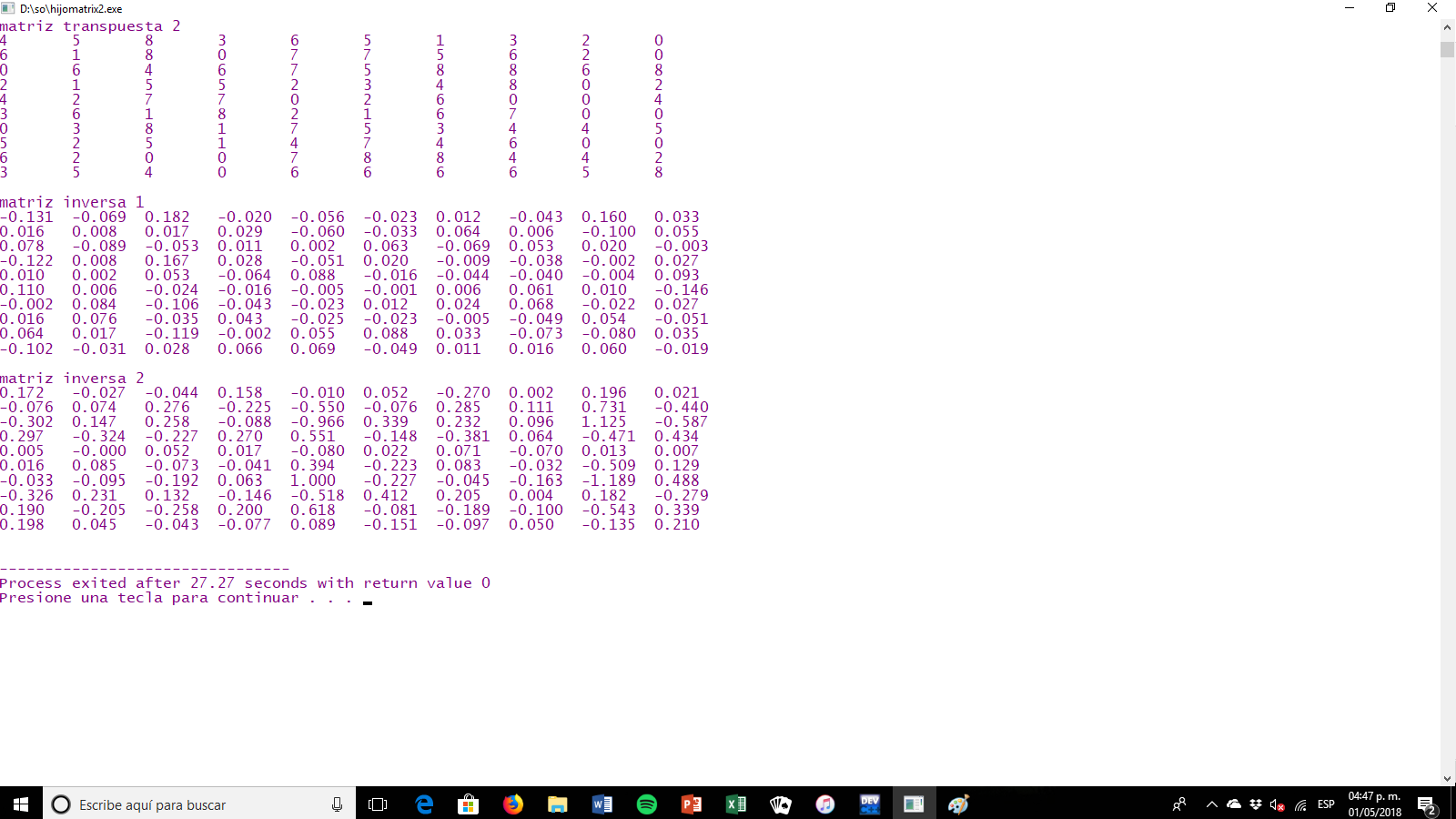
CloseHandle(manHilo6);

return 0;

}







**Creación de hilos a través de procesos:**

Archivo Proceso Padre

#include <stdlib.h>

#include <stdio.h>

#include <windows.h>

int main(int argc, char \*argv[])

{

//Pedimos el id del proceso padre, el cual se encargará de crear un proceso hijo

printf("PID PADRE: %d\n", GetCurrentProcessId());

STARTUPINFO si;

PROCESS\_INFORMATION pi;

ZeroMemory(&si, sizeof(si));

si.cb = sizeof(si);

ZeroMemory(&pi, sizeof(pi));

//Creación del proceso hijo

if(!CreateProcess(NULL, "hijohilo.exe", NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))

{

printf("Ha ocurrido un error:(%d)\n", GetLastError());

return;

}

WaitForSingleObject(pi.hProcess, INFINITE);

CloseHandle(pi.hProcess);

CloseHandle(pi.hThread);

Ñ

return 0;

}

Archivo Proceso HIjo

#include <windows.h>

#include <stdio.h>

DWORD WINAPI hilonoterminal(LPVOID lpParam);

typedef struct informacion info;

struct informacion

{

DWORD idHilo;

};

int main(void)

{

int i=0;

//Creamos los 15 hilos

printf("PID HIJO: %d\n", GetCurrentProcessId());

for (i = 0; i < 15; ++i)

{

DWORD idHilo; /\*Identificador de hilo\*/

HANDLE manHilo; /\*Manejador de hilo\*/

info argumentos;

//Creación de hilo

manHilo = CreateThread(NULL, 0, hilonoterminal, &argumentos,0,&(argumentos.idHilo));

//Esperar al hilo

WaitForSingleObject(manHilo, INFINITE);

//Destruir manejador

CloseHandle(manHilo);

}

return 0;

}

DWORD WINAPI imprimir(LPVOID lpParam)

{

printf("\t\t\t\tPractica 5\n");

return 0;

}

DWORD WINAPI hiloterminal(LPVOID lpParam)

{

int i;

info \*datos = (info\*)lpParam;

//Imprimimos los id's de los hilos terminales

printf("\t\tID Hilo TERMINAL: %i\n", datos->idHilo);

for (i = 0; i < 10; ++i)

{

DWORD idHilo; /\*Identificador de hilo\*/

HANDLE manHilo; /\*Manejador de hilo\*/

info argumentos;

//Creación de hilo, pasaando como argumento el id del hilo, el cuál llamará a la función de imprimir, ya que el hilo es terminal

manHilo = CreateThread(NULL, 0, imprimir, &argumentos,0,&(argumentos.idHilo));

//Esperar al hilo

WaitForSingleObject(manHilo, INFINITE);

//Destruir manejador

CloseHandle(manHilo);

}

return 0;

}

DWORD WINAPI hilonoterminal(LPVOID lpParam)

{

int i;

info \*datos = (info\*)lpParam;

//Imprimimos los id's de los hilos no terminales

printf("\tID Hilo NO TERMINAL: %i\n", datos->idHilo);

for (i = 0; i < 15; ++i)

{

DWORD idHilo; /\*Identificador de hilo\*/

HANDLE manHilo; /\*Manejador de hilo\*/

info argumentos;

//Creación de hilo, pasando como argumento el id del hilo, el cuál llamará al hilo terminal

manHilo = CreateThread(NULL, 0, hiloterminal, &argumentos,0,&(argumentos.idHilo));

//Esperar al hilo

WaitForSingleObject(manHilo, INFINITE);

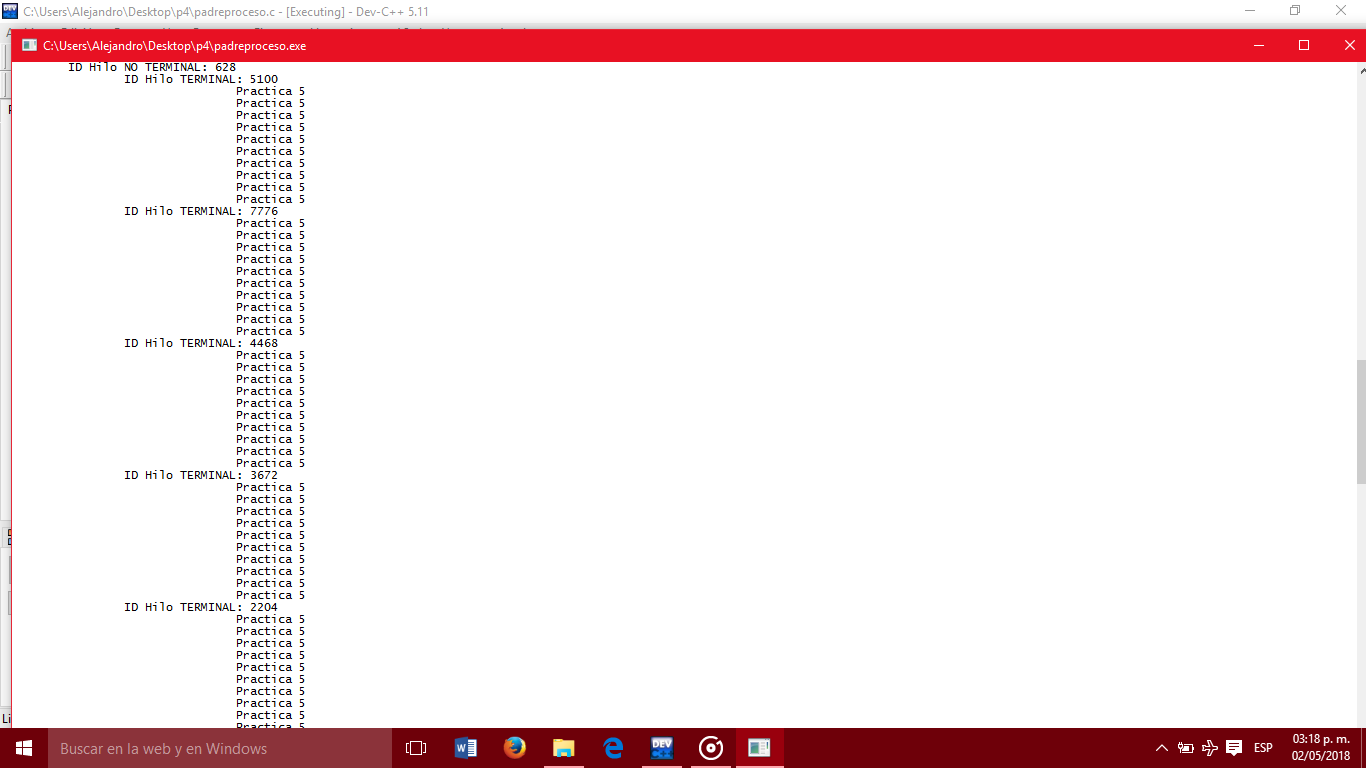
//Destruir manejador

CloseHandle(manHilo);

}

return 0;

}



7.-

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <dirent.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <string.h>

#include <errno.h>

#define LONGI 1024

DWORD WINAPI funcionHilo( LPVOID lpParam );

typedef struct Informacion info;

struct Informacion {

char \* original;

char \* copia;

};

void directorio( char \* ruta, char \* copia );

int copiar( char \* ruta, char \* copia );

int main( int argc, char \* argv[] ) {

if( argc == 3 )

directorio( argv[ 1 ], argv[ 2 ] );

else

printf("Ingrese los tres argumentos");

return 1;

}

void directorio( char \* ruta, char \* copia ){

DIR \* dir;

struct dirent \* file;

info argumentos;

struct \_stat buf;

int existe;

existe = \_stat( copia, &buf );

if( existe != 0 )

switch (errno){

case ENOENT:

if( !CreateDirectory( copia, NULL ) ){

printf( "%s: %s\n", "No se creo el directorio", copia );

exit( 1 );

}

else

printf( "%s %s\n", "Se copio el directorio:", copia );

break;

}

if( ( dir = opendir( ruta ) ) == NULL ) {

printf( "No se pudo acceder: %s", ruta );

exit( 1 );

}else {

DWORD idHilo;

HANDLE manHilo;

argumentos.original = ruta;

argumentos.copia = copia;

manHilo = CreateThread( NULL, 0, funcionHilo, &argumentos, 0, &idHilo );

WaitForSingleObject( manHilo, INFINITE );

CloseHandle( manHilo );

}

while( ( file = readdir( dir ) ) != NULL )

if( file -> d\_type == 16

&& strcmp( file -> d\_name, "." ) != 0

&& strcmp( file -> d\_name, ".." ) != 0 )

{

int longitud = strlen( ruta ) + strlen( file -> d\_name ) + 2;

int longitud\_copia = strlen( copia ) + strlen( file -> d\_name ) + 2;

char folder[ longitud ];

char folder\_copia[ longitud\_copia ];

for( int i = 0; i < longitud\_copia; i++ ){

folder\_copia[ i ] = '\0';

folder[ i ] = '\0';

}

strcpy( folder, ruta );

strcat( folder, "/" );

strcat( folder, file -> d\_name );

strcat( folder\_copia, copia );

strcat( folder\_copia, "/" );

strcat( folder\_copia, file -> d\_name );

directorio( folder, folder\_copia );

}

closedir( dir );

}

DWORD WINAPI funcionHilo( LPVOID lpParam ){

info \* args = ( info \* ) lpParam;

DIR \*dirp;

struct dirent \* dir;

if( ( dirp = opendir( args -> original ) ) == NULL ) {

printf( "%s\n", "No se pudo acceder" );

exit( 1 );

}

int resultado\_hilo = 0;

while( ( dir = readdir( dirp ) ) != NULL )

if( dir -> d\_type == DT\_REG ){

printf( "Archivo %s\n", dir -> d\_name );

int longitud\_original = strlen( args -> original ) + strlen( dir -> d\_name ) + 2;

int longitud\_copia = strlen( args -> copia ) + strlen( dir -> d\_name ) + 2;

char nueva\_original[ longitud\_original ];

char nueva\_copia[ longitud\_copia ];

strcpy( nueva\_original, args -> original );

strcat( nueva\_original, "/" );

strcat( nueva\_original, dir -> d\_name );

strcpy( nueva\_copia, args -> copia );

strcat( nueva\_copia, "/" );

strcat( nueva\_copia, dir -> d\_name );

if( copiar( nueva\_original, nueva\_copia ) )

printf( "%s\n", "Archivo copiado" );

else

printf( "%s\n", "No se pudo copiar" );

}

closedir( dirp );

return 0;

}

int copiar( char \* original, char \* copia )

{

int retorno = 0;

HANDLE archivo\_original, archivo\_copia;

DWORD numero\_bytes\_l = 0, numero\_bytes\_e = 0;

OFSTRUCT buff;

BYTE buffer[ LONGI ] = { 0 };

archivo\_original = CreateFile( original,

GENERIC\_READ,

FILE\_SHARE\_READ,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

if( archivo\_original == NULL ){

printf( "%s %s\n", "No se pudo abrir: ", original );

return 0;

}

archivo\_copia = CreateFile( copia,

GENERIC\_WRITE,

FILE\_SHARE\_READ,

NULL,

CREATE\_ALWAYS,

FILE\_ATTRIBUTE\_NORMAL,

NULL );

BOOL success;

if( archivo\_copia != NULL )

do{

if( ReadFile( archivo\_original, buffer, LONGI, &numero\_bytes\_l, NULL ) ){

success = WriteFile( archivo\_copia, buffer, numero\_bytes\_l, &numero\_bytes\_e, NULL );

if( !success )

return 0;

else

retorno = 1;

}

else

return 0;

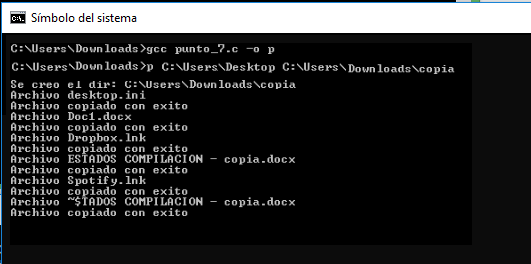
}while( numero\_bytes\_l == LONGI );

else

printf( "%s\n", "Archivo no copiado" );

return retorno;

}



# 

**Análisis Crítico**

El desarrollo de la práctica no fue complicado, ya que la metodología fue similar a la aplicada en la práctica de procesos, donde segmentamos las tareas que realizarán los programas, al principio existieron problemas de compilación, debido a que necesitábamos comprender en su totalidad los hilos, y que al invocarlos necesitamos hacerlo con argumentos dentro de un struct, el cual variaba con base a lo que necesitábamos.

Dichos pasos nos presentaron complicaciones, que afortunadamente se pudieron sobrellevar.

**Observaciones:** Comparando los hilos con los procesos podemos darnos cuenta que comparten muchas similitudes, practicamente se podria decir que hacen lo mismo. Sin embargo en esta práctica nos damos cuenta que los hilos son menos seguros en cuestión de la información, y que por esa razón se usan para hacer tareas mucho más pequeñas que las que un proceso haría.

**Conclusiones**

**Guerra Vargas:** Gracias a esta práctica se entendió mejor el funcionamiento de cómo funcionan los hilos en los sistemas operativos de Windows y de Linux, realmente es muy parecido a los proceso, incluso se podría decir que implementarlos es mucho más fácil que un proceso.

Es ideal para cumplir pequeñas tareas o pequeños programas que no necesitan grandes recursos y se harán en un tiempo corto, ya que la seguridad en hilos no es nada comparada con la de los procesos.

**Dávila Méndez:** A diferencia de la práctica anterior, en esta se pudieron observar las diferencias que hay entre utilizar hijos así como hilos. Una de las principales problemáticas que se presentó al realizar la misma fueron las *race conditions* entre hilos para acceder a un mismo recurso compartido, mismas que se solucionan ya sea poniendo en orden la ejecución de los hilos o utilizando funciones como *pthread\_mutex\_lock*, para administrar bloqueos. En general, uno puede darse cuenta que los hilos aceleran el procesamiento de cierta tarea utilizando más recursos (como se vio comparándolo vs. hijos), por lo que el utilizar hilos puede resultar útil para realizar tareas que requieran de procesamiento paralelo.

# **Reyes Valenzuela:** Con esta práctica pudimos observar funcionamiento de los hilos en los distintos sistemas operativos además de crear las mismas aplicaciones de la práctica pasada a través de la de la nueva aplicación de hilos, la cual tiene aspectos parecidos a los procesos, pero son más eficientes para la realización de tareas pequeñas