

Axel Rodríguez 1229715

Lester García 1003115

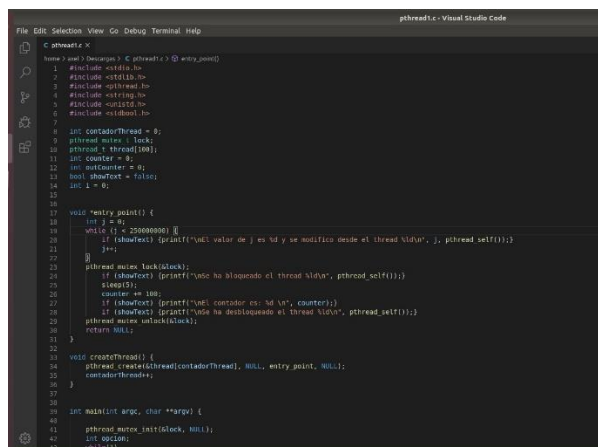
Edwin Hilario 1298816

Manuel Catalán 1038416

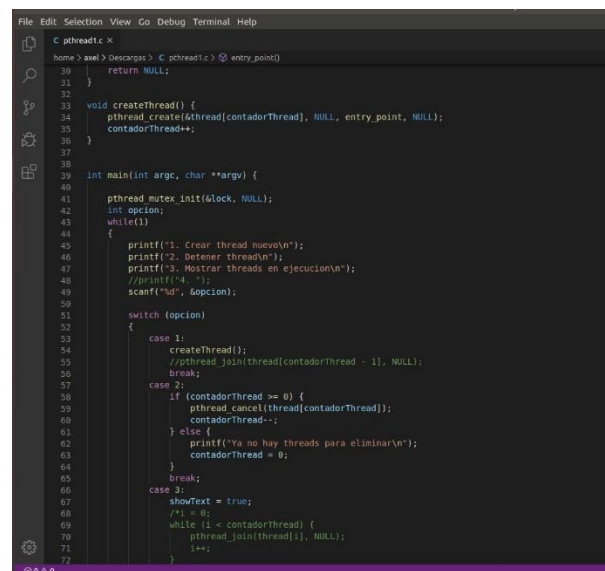
Steven Villatoro 1129215

Práctica #3 (Screenshots parte práctica)

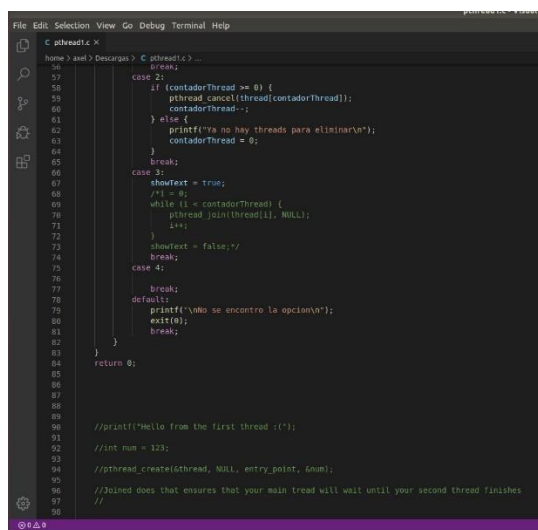
1) Código de creación y eliminación de threads



```
File Edit Selection View Go Debug Terminal Help
C pthread.c x
home > asel > Descargas > C pthread.c > @ entry_point
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <string.h>
5 #include <unistd.h>
6 #include <time.h>
7
8 int contadorThread = 0;
9 pthread_mutex_t lock;
10 pthread_t thread[100];
11 int counter = 0;
12 int mutexCounter = 0;
13 bool showText = false;
14 int i = 0;
15
16 void *entry_point() {
17     int j = 0;
18     while (j < 25000000) {
19         if (showText) printf("del valor de j es %d y se modifica desde el thread %d\n", j, pthread_self());
20         j++;
21     }
22     pthread_mutex_lock(&lock);
23     if (showText) printf("ya se bloquea el thread %d\n", pthread_self());
24     sleep(5);
25     counter += 100;
26     if (showText) printf("@@ contador es: %d\n", counter);
27     if (showText) printf("ya se desbloquea el thread %d\n", pthread_self());
28     pthread_mutex_unlock(&lock);
29     return NULL;
30 }
31
32 void createThread() {
33     pthread_create(&thread[contadorThread], NULL, entry_point, NULL);
34     contadorThread++;
35 }
36
37 int main(int argc, char **argv) {
38     pthread_mutex_init(&lock, NULL);
39     int opcion;
40     while(1) {
41         printf("1. Crear thread nuevo\n");
42         printf("2. Detener thread\n");
43         printf("3. Mostrar threads en ejecucion\n");
44         //printf("4. ");
45         scanf("%d", &opcion);
46
47         switch (opcion) {
48             case 1:
49                 createThread();
50                 //pthread_join(thread[contadorThread - 1], NULL);
51                 break;
52             case 2:
53                 if (contadorThread >= 0) {
54                     pthread_cancel(thread[contadorThread]);
55                     contadorThread--;
56                 } else {
57                     printf("Ya no hay threads para eliminar\n");
58                     contadorThread = 0;
59                 }
60                 break;
61             case 3:
62                 showText = true;
63                 //i = 0;
64                 while (i < contadorThread) {
65                     pthread_join(thread[i], NULL);
66                     i++;
67                 }
68                 showText = false;
69                 break;
70             case 4:
71                 break;
72             default:
73                 printf("\nNo se encontro la opcion\n");
74                 exit(0);
75                 break;
76         }
77     }
78     return 0;
79 }
```



```
File Edit Selection View Go Debug Terminal Help
C pthread.c x
home > asel > Descargas > C pthread.c > @ entry_point
39 }
40 return NULL;
41 }
42
43 void createThread() {
44     pthread_create(&thread[contadorThread], NULL, entry_point, NULL);
45     contadorThread++;
46 }
47
48 int main(int argc, char **argv) {
49     pthread_mutex_init(&lock, NULL);
50     int opcion;
51     while(1) {
52         printf("1. Crear thread nuevo\n");
53         printf("2. Detener thread\n");
54         printf("3. Mostrar threads en ejecucion\n");
55         //printf("4. ");
56         scanf("%d", &opcion);
57
58         switch (opcion) {
59             case 1:
60                 createThread();
61                 //pthread_join(thread[contadorThread - 1], NULL);
62                 break;
63             case 2:
64                 if (contadorThread >= 0) {
65                     pthread_cancel(thread[contadorThread]);
66                     contadorThread--;
67                 } else {
68                     printf("Ya no hay threads para eliminar\n");
69                     contadorThread = 0;
70                 }
71                 break;
72             case 3:
73                 showText = true;
74                 //i = 0;
75                 while (i < contadorThread) {
76                     pthread_join(thread[i], NULL);
77                     i++;
78                 }
79                 showText = false;
80                 break;
81             case 4:
82                 break;
83             default:
84                 printf("\nNo se encontro la opcion\n");
85                 exit(0);
86                 break;
87         }
88     }
89     return 0;
90 }
```



```
File Edit Selection View Go Debug Terminal Help
C pthread.c x
home > asel > Descargas > C pthread.c > ...
39 }
40 return NULL;
41 }
42
43 void createThread() {
44     pthread_create(&thread[contadorThread], NULL, entry_point, NULL);
45     contadorThread++;
46 }
47
48 int main(int argc, char **argv) {
49     pthread_mutex_init(&lock, NULL);
50     int opcion;
51     while(1) {
52         printf("1. Crear thread nuevo\n");
53         printf("2. Detener thread\n");
54         printf("3. Mostrar threads en ejecucion\n");
55         //printf("4. ");
56         scanf("%d", &opcion);
57
58         switch (opcion) {
59             case 1:
60                 createThread();
61                 //pthread_join(thread[contadorThread - 1], NULL);
62                 break;
63             case 2:
64                 if (contadorThread >= 0) {
65                     pthread_cancel(thread[contadorThread]);
66                     contadorThread--;
67                 } else {
68                     printf("Ya no hay threads para eliminar\n");
69                     contadorThread = 0;
70                 }
71                 break;
72             case 3:
73                 showText = true;
74                 //i = 0;
75                 while (i < contadorThread) {
76                     pthread_join(thread[i], NULL);
77                     i++;
78                 }
79                 showText = false;
80                 break;
81             case 4:
82                 break;
83             default:
84                 printf("\nNo se encontro la opcion\n");
85                 exit(0);
86                 break;
87         }
88     }
89     return 0;
90 }
```

Axel Rodríguez 1229715

Lester García 1003115

Edwin Hilario 1298816

Manuel Catalán 1038416

Steven Villatoro 1129215

2) Menú de opciones para crear, detener y mostrar threads activos

```
axel@axel-VirtualBox:~/Descargas$ ./pthread1
1. Crear thread nuevo
2. Detener thread
3. Mostrar threads en ejecucion
```

3) Consola de procesos activos con el consumo de CPU de cada thread creado

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
4332	axel	20	0	72256	928	848	S	0.0	0.0	0:00.00	./pthread1
4336	axel	20	0	72256	928	848	S	0.0	0.0	0:00.67	./pthread1
4338	axel	20	0	72256	928	848	S	0.0	0.0	0:00.66	./pthread1
4339	axel	20	0	72256	928	848	S	0.0	0.0	0:00.68	./pthread1
4341	axel	20	0	72256	928	848	S	0.0	0.0	0:00.71	./pthread1
4342	axel	20	0	72256	928	848	S	0.0	0.0	0:00.71	./pthread1

4) Thread corriendo

```
El valor de j es 179063494 y se modifco desde el thread 139999881815808
El valor de j es 179063495 y se modifco desde el thread 139999881815808
El valor de j es 179063496 y se modifco desde el thread 139999881815808
El valor de j es 179063497 y se modifco desde el thread 139999881815808
El valor de j es 179063498 y se modifco desde el thread 139999881815808
El valor de j es 179063499 y se modifco desde el thread 139999881815808
El valor de j es 179063500 y se modifco desde el thread 139999881815808
El valor de j es 179063501 y se modifco desde el thread 139999881815808
El valor de j es 179063502 y se modifco desde el thread 139999881815808
El valor de j es 179063503 y se modifco desde el thread 139999881815808
El valor de j es 179063504 y se modifco desde el thread 139999881815808
El valor de j es 179063505 y se modifco desde el thread 139999881815808
El valor de j es 179063506 y se modifco desde el thread 139999881815808
El valor de j es 179063507 y se modifco desde el thread 139999881815808
El valor de j es 65916570 y se modifco desde el thread 139999865030400
El valor de j es 65916571 y se modifco desde el thread 139999865030400
El valor de j es 65916572 y se modifco desde el thread 139999865030400
El valor de j es 65916573 y se modifco desde el thread 139999865030400
El valor de j es 65916574 y se modifco desde el thread 139999865030400
El valor de j es 65916575 y se modifco desde el thread 139999865030400
El valor de j es 65916576 y se modifco desde el thread 139999865030400
El valor de j es 65916577 y se modifco desde el thread 139999865030400
El valor de j es 65916578 y se modifco desde el thread 139999865030400
El valor de j es 65916579 y se modifco desde el thread 139999865030400
El valor de j es 65916580 y se modifco desde el thread 139999865030400
```

Axel Rodríguez 1229715

Lester García 1003115

Edwin Hilario 1298816

Manuel Catalán 1038416

Steven Villatoro 1129215

Práctica #3 (Preguntas teóricas)

1) ¿Cuál es la principal diferencia entre codificar un proceso y codificar threads?

Un proceso se puede crear una a la vez mientras que un thread se pueden crear varios hijos los cuales serán parte de un proceso principal general. En otras palabras, un proceso es independiente, y un thread son varios procesos bajo un proceso padre.

2) Describa la lógica (con código) para Crear y detener un thread.

```
pthread_create(&thread[contadorThread], NULL, entry_point, NULL);  
    contadorThread++;
```

Para crear el Thread el primer parámetro recibido es el puntero de thread, el segundo no se utiliza y se pone NULL, el tercer es la función que se va a ejecutar y lo demás son los parámetros restantes, pero como no se utilizan serán NULL.

```
pthread_cancel(thread[contadorThread]);  
    contadorThread--;
```

Para detener un thread se manda directamente el Thread para que se detenga.

Axel Rodríguez 1229715

Lester García 1003115

Edwin Hilario 1298816

Manuel Catalán 1038416

Steven Villatoro 1129215

- 3) Describa el ciclo de vida de un thread al momento de codificarlo, como se crea, como se maneja y si tiene algún mecanismo a tomar en cuenta en particular.

Al momento de ser creado automáticamente inicia su ejecución, se maneja adentro del parámetro del método, se crea la codificación que se desea y empieza el thread a correr en segundo plano. Se tomó en consideración utilizar el Mutex para poder tener un control de apagado y encendido de las variables compartidos.

- 4) ¿Es necesario hacer un "join" al finalizar el proceso con un thread? ¿Por qué si o por qué no?

No, ya que el Join lo que hace es esperar a que el thread termine para continuar la ejecución del siguiente thread o del resto del código.

COMANDOS PARA VER PROCESOS Y SUS THREADS

```
top -H -c -p $(pgrep -d',' -f pthread1)
```