REPORTE DE LA PRÁCTICA 9

SISTEMAS OPERATIVOS

ALUMNO:
HERNÁNDEZ TAPIA LUIS ENRIQUE

PROFESOR:
HERMES FRANCISCO MONTES CASIANO

Escuela Superior de Cómputo, IPN

25 de mayo de 2018

Índice general

CAPÍTULO 1

Introducción

1.1. Hilos (Thread)

En la computación, si queremos que los programas se ejecuten varias cosas a la vez, es decir, al mismo tiempo, tenemos dos opciones. Por una parte podemos crear un nuevo proceso y por otro, podemos crear un nuevo hilo de ejecución (thread), entonces el sistema operativo irá ejecutando segmentos de programas por turnos de forma rápida, simulando la simultaneidad, a menos que poseas varias CPUs.

Un hilo pertenece a un proceso, es decir, es un subconjunto del susodicho, entonces un proceso puede tener varios hilos sin la necesidad de programación adicional, pero un hilo no puede tener varios procesos. Una de las ventajas que destaca sobre los hilos, es que nos son copias como los procesos por lo que consumen menos recursos del sistema, la aplicación de los multihilos es favorable y funciona de manera eficiente en multiprocesadores.

Como los paquetes de transporte trabajan mediante UDP (User Datagram Protocol) que está basado en el intercambio de diagramas, funciona de manera asíncrona y es la manera en que un hilo recibe los mensajes, es necesario que los hilos estén comunicados en todo momento. En esta práctica se trabajará mediante el sistema POSIX (Portable Operating System Interface y X de UNIX)término sugerido por Richard Stallman en la década de 1980, en respuesta a la demanda de la IEEE, para un nombre fácil de recordar.

CAPÍTULO 2

Desarrollo

2.1. Programa 12-1

A continuación se nos pide crear el siguiente programa, con la finalidad de iniciar un hilo principal que crea dos hilos hijo dependientes.

El hilo principal espera a que sus hijos terminen para poder finalizar.

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

void funcion(void)
{
printf("Hilo %lu \n", pthread_self());
sleep(2);
}

int main(){
pthread_t th1, th2;
/*Se crean dos hilos con atributos predeterminados */
pthread_create(&th1, NULL, (void *)funcion, NULL);
pthread_create(&th2, NULL, (void *)funcion, NULL);
printf("El hilo principal espera a sus hijos\n");

/*Se espera su terminacion*/
pthread_join(th1, NULL);
pthread_join(th2, NULL);
printf("El hilo principal termina\n");

exit(0);
}
```

Para compilar, se debe hacer uso del parametro *-pthread* como a continuación se describe:

gcc programa12-1.c -pthread -o programa12-1
La salida es la siguiente:

```
El hilo principal espera a sus hijos
Hilo 139940447405824
Hilo 139940455798528
El hilo principal termina
```

2.2. Pregunta 12-1

Respecto al programa 12-1 se nos pide responder las siguientes preguntas.

1. ¿Qué sucede si el proceso principal termina un instante después de haber creado los hilos?
R: Como los hilos dependen del proceso principal, termina inmediatamente.

2. ¿Puede un hilo hijo crear otro hilo (nieto) y esperar a que este nieto termine?
Sí, la programación adecuada sería la que se presente a continuación:

```
1  #include <stdio.h>
2  #include <pthread.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5
6  void Nieto(void * data){
7          char *texto =(char*)data;
8          printf("%s\n", texto);
9          sleep(3);
10 }
11
12 void Hijo(void * data){
13         char *texto =(char*)data;
14         printf("%s\n", texto);
15         pthread_t proceso2;
16         pthread_create(&proceso2, NULL, (void*)Nieto, "Hola soy un hilo nieto");
17         pthread_join(proceso2, NULL);
18         printf("Termino el nieto\n");
19 }
20
21 int main(void){
22         pthread_t proceso1;
23         /*Se crean dos hilos con atributos predeterminados */
24         pthread_create(&proceso1, NULL, (void*)Hijo, "Hola soy un hilo hijo");
25         /*Se espera su terminacion*/
26         pthread_join(proceso1, NULL);
27         printf("Termino el hijo\n");
28         return 0;
29
30 }
```

Con salida en la terminal:

```
Hola soy un hilo hijo
Hola soy un hilo nieto
Termino el nieto
Termino el hijo
```

2.3. Pregunta 12-2

Se pide crear una variable entera global, con un valor inicial de cero. Un hijo debe incrementar la variable en *a* unidades y el otro hijo la debe decrementar en *b* unidades.

Responder las siguientes preguntas.

1. ¿Los hilos comparten la variable?
R: Los hilos comparten de forma concurrente la variable global.

2. ¿Y qué sucede si la variable se declara dentro de la función main()?
R: La variable trabaja de forma independiente, es decir, por un lado va disminuyendo y por otro aumentando a la par, por trabajar con hilos (paralelismo).

Se presenta el código con el que se comprobó los resultados:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <pthread.h>
4  #include <unistd.h>
5
6  int i = 0; // Variable compartida
7
8  void Hilo1(void){
9          while(1){
10                 printf("Soy el hilo 1 (%lu) y el valor de i = %d\n", pthread_self
                       (), i++);
11                 sleep(1);
12         }
13 }
14 void Hilo2(void){
15         while(1){
16                 printf("Soy el hilo 2 (%lu) y el valor de i = %d\n", pthread_self
                       (), i--);
17                 sleep(1);
18         }
19 }
20
21 int main(void)
22 {
23         pthread_t proceso1, proceso2;
24
25         //Crear hilos
26         pthread_create(&proceso1, NULL, (void*)Hilo1, NULL);
27         pthread_create(&proceso2, NULL, (void*)Hilo2, NULL);
```

```
28
29          //Espera a que los hilos terminen
30          pthread_join(proceso1, NULL);
31          pthread_join(proceso2, NULL);
32
33          exit(0);
34  }
```

Con salida en la terminal para la pregunta **1.**:

```
Soy el hilo 2 (140657187337984) y el valor de i = 0
Soy el hilo 1 (140657195730688) y el valor de i = -1
Soy el hilo 2 (140657187337984) y el valor de i = 0
Soy el hilo 1 (140657195730688) y el valor de i = -1
Soy el hilo 2 (140657187337984) y el valor de i = 0
Soy el hilo 1 (140657195730688) y el valor de i = -1
Soy el hilo 2 (140657187337984) y el valor de i = 0
Soy el hilo 1 (140657195730688) y el valor de i = -1
```

Con salida en la terminal para la pregunta **2.**:

```
Soy el hilo 2 (140693554886400) y el valor de i = 0
Soy el hilo 1 (140693563279104) y el valor de i = 0
Soy el hilo 2 (140693554886400) y el valor de i = -1
Soy el hilo 1 (140693563279104) y el valor de i = 1
Soy el hilo 2 (140693554886400) y el valor de i = -2
Soy el hilo 1 (140693563279104) y el valor de i = 2
Soy el hilo 2 (140693554886400) y el valor de i = -3
```

2.4. Programa 12-2

El siguiente ejemplo sirve para crear 10 hilos de forma independientes:

```
1   #include <pthread.h>
2   #include <stdio.h>
3   #include <unistd.h>
4   #include <stdlib.h>
5   #define MAX_HILOS 10
6
7   void funcion(){
8           printf("Hilo %lu\n", pthread_self());
9           sleep(3);
10          pthread_exit(NULL);
11  }
12
13  void main(){
14          int i;
15          pthread_attr_t atributos;
16          pthread_t thid[MAX_HILOS];
17
18          /*Se inicializan los atributos como independientes*/
19          pthread_attr_init(&atributos);
20          pthread_attr_setdetachstate(&atributos, PTHREAD_CREATE_DETACHED);
21
```

```
22          for (i = 0; i<MAX_HILOS; i++){
23                  pthread_create(&thid[i], &atributos, (void*)funcion, NULL);
24                  /*El hilo principal se supende 6 segundos, para esperar a que los
                        hilos hijos terminen.
25                  De lo contrario al terminar el proceso principal, tambien todos su
                         hijos terminarian*/
26                  sleep(6);
27          }
28  }
```

Para compilar, se debe hacer uso del parametro *-pthread* como a continuación se describe:

gcc programa12-2.c -pthread -o programa12-2
La salida es la siguiente:

```
Hilo 140235436726016
Hilo 140235436726016
Hilo 140235436726016
Hilo 140235436726016
Hilo 140235436726016
Hilo 140235436726016
Hilo 140235436726016
Hilo 140235436726016
Hilo 140235436726016
Hilo 140235436726016
```

2.5. Programa 12-3

El ejemplo a continuación sincroniza dos hilos mediante el uso de semáforos para hilos.

```
1   /*Este programa muestra la sincronizacion en la impresion de dos hilos*/
2   #include <stdio.h>
3   #include <stdlib.h>
4   #include <pthread.h>
5   #include <semaphore.h>
6   #include <pthread.h>
7   int i = 10; // Variable compartida
8   sem_t mutex1;
9   sem_t mutex2;
10
11  void Hilo1(void){
12          while(i>0){
13                  sem_wait(&mutex1);
14                  printf("Soy el hilo 1 (%lu) y esta es la impresion %d\n",
                        pthread_self(), i--);
15                  sem_post(&mutex2);
16          }
17                  pthread_exit(0);
18  }
19  void Hilo2(void){
20          while(i>0){
21                  sem_wait(&mutex2);
22                  printf("Soy el hilo 2 (%lu) y esta es la impresion %d\n",
                        pthread_self(), i--);
23                  sem_post(&mutex1);
```

```
24          }
25                  pthread_exit(0);
26  }
27
28  int main(void)
29  {
30          pthread_t th1, th2;
31
32          //Inicializa los semaforos
33          sem_init(&mutex1, 0 ,1);
34          sem_init(&mutex2, 0 ,0);
35
36          //Crear hilos
37          pthread_create(&th1, NULL, (void*)Hilo1, NULL);
38          pthread_create(&th2, NULL, (void*)Hilo2, NULL);
39
40          //Espera a que los hilos terminen
41          pthread_join(th1, NULL);
42          pthread_join(th2, NULL);
43
44          exit(0);
45  }
```

Para compilar, se debe hacer uso del parametro *-pthread* como a continuación se describe:

gcc programa12-3.c -pthread -o programa12-3
La salida es la siguiente:

```
Soy el hilo 1 (140011130406656) y esta es la impresion 10
Soy el hilo 2 (140011122013952) y esta es la impresion 9
Soy el hilo 1 (140011130406656) y esta es la impresion 8
Soy el hilo 2 (140011122013952) y esta es la impresion 7
Soy el hilo 1 (140011130406656) y esta es la impresion 6
Soy el hilo 2 (140011122013952) y esta es la impresion 5
Soy el hilo 1 (140011130406656) y esta es la impresion 4
Soy el hilo 2 (140011122013952) y esta es la impresion 3
Soy el hilo 1 (140011130406656) y esta es la impresion 2
Soy el hilo 2 (140011122013952) y esta es la impresion 1
Soy el hilo 1 (140011130406656) y esta es la impresion 0
```

CAPÍTULO 3

Pruebas

3.1. Ejercicio 12-1

Para el ejercicio siguiente, se pide elaborar un programa donde se comparta una variable con un valor inicial de cero. Posteriormente se debe crear dos hilos independientes, uno de ellos incrementa permanentemente la variable en uno y el otro la decrementa en uno. Después de "n"segundos el proceso debe imprimir el valor final de la variable y terminar. El número de segundos que el proceso padre espera se debe pasar en la línea de comandos.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

int i = 0, seconds; // Variable compartida

void Hilo1(void){
        for (int j = 0; j < seconds; j++){
                i++;
                //printf("%d\n",i);
                sleep(1);
        }
}
void Hilo2(void){
        for (int j = 0; j < seconds; j++){
                i--;
                //printf("%d\n",i);
                sleep(1);
        }
}

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

```
{
        if (argc == 1)
        {
                printf("Debes ingresar el numero de segundos...\n");
                exit(0);
        }

        //Crear hilos
        pthread_t proceso1, proceso2;
        seconds = (int)*argv[1]-'0';//Recibimos parametros de la linea de comandos
        pthread_create(&proceso1, NULL, (void*)Hilo1, NULL);
        pthread_create(&proceso2, NULL, (void*)Hilo2, NULL);

        //Espera a que los hilos terminen
        pthread_join(proceso1, NULL);
        pthread_join(proceso2, NULL);

        printf("El valor final de la variable i es : %d\n", i);
        exit(0);
}
```

Para compilar, se debe hacer uso del parametro *-pthread* como a continuación se describe:

gcc Ejercicio12-1.c -pthread -o Ejercicio12-1
Para ejecutar, pasando los segundos por la línea de comandos:
./Ejercicio12-1 10
La salida es la siguiente:

```
El valor final de la variable i es : 0
```

3.2. Ejercicio 12-2

Se nos pide realizar un programa donde el hilo principal, al crear un hilo nuevo, le pase como parámetro la dirección de una estructura con los miembros que se definan.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

struct fraccion
{
        float num;
        float dem;
}f1;

float res(struct fraccion *f){
        return (f->num)/(f->dem);
}

```

```
16 void Hilo1(struct fraccion *f){
17         printf("La fracion en decimal de %d/%d es %f\n", (int)f->num, (int)f->dem,
               res(f));
18 }
19
20 int main(int argc, char *argv[])
21 {
22         //Crear hilos
23         pthread_t proceso1;
24         struct fraccion *f;
25         f->num = 1;//Inicializamos valores a la estructura
26         f->dem = 8;
27         pthread_create(&proceso1, NULL, (void*)Hilo1, f);
28
29         //Espera a que los hilos terminen
30         pthread_join(proceso1, NULL);
31
32         exit(0);
33 }
```

Para compilar, se debe hacer uso del parametro *-pthread* como a continuación se describe:

gcc Ejercicio12-2.c -pthread -o Ejercicio12-2
La salida es la siguiente:

```
La fracion en decimal de 1/8 es 0.125000
```

3.3. Ejercicio 12-3

Además se pide modificar el programa 12-3 para que se sincronice las impresiones consecutivas de tres hilos en vez de que sean dos hilos.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <pthread.h>
4  #include <semaphore.h>
5  int i = 10; // Variable compartida
6  sem_t mutex1;
7  sem_t mutex2;
8  sem_t mutex3;
9
10 void Hilo1(void){
11         while(i>0){
12                 sem_wait(&mutex1);
13                 printf("Soy el hilo 1 (%lu) y esta es la impresion %d\n",
                       pthread_self(), i--);
14                 sem_post(&mutex2);
15         }
16                 pthread_exit(0);
17 }
18 void Hilo2(void){
19         while(i>0){
20                 sem_wait(&mutex2);
21                 printf("Soy el hilo 2 (%lu) y esta es la impresion %d\n",
```

```
                    pthread_self(), i--);
22                  sem_post(&mutex3);
23          }
24                  pthread_exit(0);
25  }
26  void Hilo3(void){
27          while(i>0){
28                  sem_wait(&mutex3);
29                  printf("Soy el hilo 3 (%lu) y esta es la impresion %d\n",
                        pthread_self(), i--);
30                  sem_post(&mutex1);
31          }
32                  pthread_exit(0);
33  }
34  int main(void)
35  {
36          pthread_t proceso1, proceso2, proceso3;
37
38          //Inicializa los semaforos
39          sem_init(&mutex1, 0 ,1);
40          sem_init(&mutex2, 0 ,0);
41          sem_init(&mutex3, 0 ,0);
42
43
44          //Crear hilos
45          pthread_create(&proceso1, NULL, (void*)Hilo1, NULL);
46          pthread_create(&proceso2, NULL, (void*)Hilo2, NULL);
47          pthread_create(&proceso3, NULL, (void*)Hilo3, NULL);
48
49
50          //Espera a que los hilos terminen
51          pthread_join(proceso1, NULL);
52          pthread_join(proceso2, NULL);
53          pthread_join(proceso3, NULL);
54
55
56          exit(0);
57  }
```

Para compilar, se debe hacer uso del parametro *-pthread* como a continuación se describe:

gcc Ejercicio12-3.c -pthread -o Ejercicio12-3
La salida es la siguiente:

```
Soy el hilo 1 (140378478241536) y esta es la impresion 10
Soy el hilo 2 (140378469848832) y esta es la impresion 9
Soy el hilo 3 (140378461456128) y esta es la impresion 8
Soy el hilo 1 (140378478241536) y esta es la impresion 7
Soy el hilo 2 (140378469848832) y esta es la impresion 6
Soy el hilo 3 (140378461456128) y esta es la impresion 5
Soy el hilo 1 (140378478241536) y esta es la impresion 4
Soy el hilo 2 (140378469848832) y esta es la impresion 3
Soy el hilo 3 (140378461456128) y esta es la impresion 2
```

CAPÍTULO 4

Conclusión

4.1. Deducción

Nos encontramos ante la definición de lo que es la concurrencia, es decir, la propiedad en los sistemas en la cual los procesos de cómputo se hacen simultáneamente y, pueden interactuar entre ellos.

Así que podemos hacer uso de los hilos para que sean empleados bajo situaciones donde se necesitan largos procesamientos en segundo plano o tareas de Entrada/Salida.

Para concluir, destacamos que sobrecargar los recursos en algunas situaciones por no mencionar todas, sería algo inaceptable. Sin embargo, hay grandes beneficios como; simular a un multiprocesador y compartir el tiempo de eficiencia. En definitiva, el uso de hilos convierte a las aplicaciones en una respuesta eficiente y veloz a varios problemas en las ciencias de la computación.