Universidad de Antioquia
Facultad de Ingeniería
Laboratorio de Sistemas Operativos

Hilos

Objetivos

- Explorar el uso de hilos como herramienta básica para aprovechar la concurrencia en los procesos.
- Conocer y usar las principales herramientas para la gestión de hilos en la librería **pthread**.

Contenido

1 Hilos

Los hilos son el medio por el cual un proceso puede implementar concurrencia. Éstos permiten que un proceso posea varios caminos de ejecución y a su vez compartan cierta región de memoria de manera directa. Por otro lado, en un sistema con KLT (Kernel Level Thread) y con multiprocesamiento o multithreading (por ejemplo, el hyperthreading de Intel), los hilos aprovecharían al máximo los recursos de hardware disponibles y aumentarían considerablemente el desempeño de una aplicación.

Generalmente, cada sistema operativo tiene una forma propia de enfrentar el desafío de la implementación de los hilos. A continuación se presenta un resumen de dos lecturas tomadas del libro de Silberschatz [1] relacionadas con la implementación de los hilos en Windows XP y en Linux.

1.1 Hilos en Linux

Linux proporciona en el llamado al sistema fork() con la funcionalidad tradicional de duplicar un proceso. Linux también proporciona la funcionalidad de crear hilos usando el llamado al sistema clone(). Sin embargo, Linux no distingue entre procesos e hilos. De hecho, Linux generalmente usa el término de tarea más que proceso o hilo cuando se refiere a un flujo de control dentro de un programa. Cuando se invoca el llamado al sistema clone(), éste es pasado con un conjunto de banderas u opciones, las cuales determinan la cantidad de intercambio que se llevará a cabo entre la tarea padre y su nuevo hijo. Algunas de estas banderas se muestran a continuación:

flag	meaning
CLONE_FS	File-system information is shared.
CLONE_VM	The same memory space is shared.
CLONE_SIGHAND	Signal handlers are shared.
CLONE_FILES	The set of open files is shared.

Figura 1. Hilos en Linux.

Por ejemplo, si clone es llamado con las banderas CLONE_FS, CLONE_VM, CLONE_SIGHAND, y CLONE_FILES, las tareas padre e hija, compartirán la misma información del sistema de archivos (como el actual directorio de trabajo), el mismo espacio de memoria, los mismos manejadores de señales y el mismo conjunto de archivos abiertos. Usar el llamado al sistema clone() de este modo es equivalente a crear un hilo como se ha descrito de manera teórica, ya que la tarea padre comparte la mayoría de sus recursos con su tarea hija. Sin embargo, si ninguna de esas banderas es fijada cuando se hace el llamado a clone(), ningún intercambio es realizado y la funcionalidad será similar a la proporcionada por la llamada al sistema fork().

El nivel de variación de intercambio es posible debido a la manera como una tarea es representada en el kernel de Linux. Solo existe una estructura de datos en el kernel para cada tarea en el sistema (específicamente, struct task_struct). Esta estructura de datos en lugar de almacenar datos para la tarea, contiene apuntadores a otras estructuras de datos donde la información está almacenada, por ejemplo estructuras de datos que representan la lista de archivos abiertos, información del manejo de señales y la memoria virtual. Cuando fork() es invocado, una nueva tarea es creada y junto con ella una copia de todas las estructuras de datos del proceso padre. Una nueva tarea se crea también con el llamada al sistema clone(), la diferencia radica es que en lugar de una copia de todas las estructuras de datos, la nueva tarea apunta a las estructuras de datos de la tarea padre, dependiendo del conjunto de opciones pasadas en el llamado a clone().

1.2 Librería *pthread* para la implementación de Hilos en C bajo Linux

Tras la revisión anterior es claro verificar que cada sistema operativo tiene, generalmente, una forma diferente de implementar los hilos a bajo nivel, sin embargo cuando estamos interactuando con un Lenguaje de Alto Nivel (LAN), éstos poseen o usan librerías que permiten abstraer aún más el problema. De acuerdo a nuestro programa basado en herramientas libres y el sistema Linux, nosotros en la presente guía de laboratorio trabajaremos la implementación de hilos a través del lenguaje de programación C, la librería pthread y bajo el sistema operativo Linux. Recomendamos que el estudiante explore otros lenguajes, librerías y SO y se sugiere el siguiente material ([2]).

1.3 Ejercicios

Compile los siguientes ejercicios, analice el código y la salida. Para la compilar el código con la librería pthread es necesario utilizar la opción **–lpthread**, como se muestra este ejemplo:

gcc codigo.c –o ejecutable -lpthread

1.3.1 Creación de un hilo

```c
#include <pthread.h>
#include <stdio.h>

/* Imprime 'x' en la salida de error stderr.
El parámetro no es usuado. No tiene retorno*/
void* imprime_x (void* unused)
{
    while (1) fputc ('x', stderr);
    return NULL;
}
int main (){
    pthread_t id_hilo;
    /* Crea un nuevo hilo sobre la función imprime_x */
    pthread_create (&id_hilo, NULL, &imprime_x, NULL);
    /* Imprime 'o' en la salida de error stderr*/
    while (1) fputc ('o', stderr);
    return 0;
}

```

Nota: Puede finalizar la ejecución con <Ctrl + C>.

1. ¿Cuáles son los argumentos de la función pthread_create? ¿Para qué sirven?
2. ¿Cómo es la salida en pantalla? ¿Cuál es la razón para este tipo de salida?

1.3.2 Varios Hilos y modo de ejecución

```c
#include <pthread.h>
#include <stdio.h>

/* Esta estructura sirve para pasar parámetros a el hilo */
struct parametros_hilo {
    /* Parametro 1: caracter a imprimir */
    char caracter;
    /* Parametro 2: número de veces que se desea imprimir */
    int contador;
};

/* Esta función imprime un numero de caracteres a la
salida de error, tal y como
lo indica el parámetro de la función.*/
void* imprimir_caracter (void* parametros) {

/* Se hace un cast a tipo de dato correcto */
    struct parametros_hilo* p = (struct parametros_hilo*) parametros;
    int i;
    for (i = 0; i < p->contador; ++i)
        fputc (p->caracter, stderr);
    return NULL;
}

```

```
24
25  /* Programa principal*/
26  int main () {
27      pthread_t id_hilo_1;
28      pthread_t id_hilo_2;
29      struct parametros_hilo hilo1_param;
30      struct parametros_hilo hilo2_param;
31
32  /* Crea un hilo para imprimir 30,000 x */
33      hilo1_param.caracter = 'x';
34      hilo1_param.contador = 30000;
35      pthread_create (&id_hilo_1, NULL, &imprimir_caracter, &hilo1_param);
36
37  /* Crea un hilo para imprimir 20,000 o */
38      hilo2_param.caracter = 'O';
39      hilo2_param.contador = 20000;
40      pthread_create (&id_hilo_2, NULL, &imprimir_caracter, &hilo2_param);
41
42  /*-------INSERTAR AQUÍ-------*/
43      return 0;
44  }
45
```

3. ¿Cuál es el resultado de la ejecución? ¿Usted esperaba este resultado? ¿Por qué?
4. ¿Para qué se usa un apuntador a un tipo de dato void? ¿Qué sentido tiene hacer esto?

1.3.3 Conectando hilos (Join)

Modifique el código anterior justo después de la marca que dice `/*-------INSERTAR AQUÍ-------*/` inserte las siguientes líneas:

```
pthread_join (id_hilo_1, NULL);
pthread_join (id_hilo_2, NULL);
```

5. ¿Qué sucede ahora con la ejecución de este código? ¿Por qué?
6. ¿Cuál es la funcionalidad de hacer un join en los hilos?
7. ¿Cuáles son los parámetros de la función pthread_join? ¿Cuál es su uso?

1.3.4 La misma operación con el mismo dato

```c
#include<stdio.h>
#include<pthread.h>
#include<stdlib.h>
#include <sys/types.h>
#include <unistd.h>

/* Variable Global x*/
int x = 0;

void ft()
{
  int i;
  printf("Identificador de hilo: %d. \nx tiene el valor de  %d \n
  antes de ser incrementado 1000 veces por este hilo \n", (int) getpid (),x);
  for( i=1;i<=1000;i++) x++;
}

int main(void)
{
  pthread_t hilos_ids[4];
  int i;
  for (i = 0; i < 4; ++i) pthread_create (&hilos_ids[i],NULL,(void *) ft ,NULL);
  for (i = 0; i < 4; ++i) pthread_join (hilos_ids [i], NULL);
  printf("Hilo principal: x= %d \n",x);
  return 0;
}
```

8. Analice el código del presente ejemplo. ¿Qué hace?
9. Ejecute el código muchas veces. ¿Los hilos se ejecutan en el mismo orden siempre? ¿esto es lo que usted esperaba?
10. ¿Qué opina al respecto de los identificadores para los hilos? ¿Cuál es la razón para tener este valor?

1.3.5 Retorna un valor desde el hilo

```c
#include <pthread.h>
#include <stdio.h>

void* calcular_primo (void* arg);

int main () {
   pthread_t hilo_id;
   int cual_primo = 5000;
   int primo;

   /* Inicia el hilo, se requiere el 5000-ésimo número primo. */
   pthread_create (&hilo_id, NULL, &calcular_primo, &cual_primo);
   /* Puedo hacer algo mientras... si quiero */
   /* Espero que el número sea calculado y me sea entregado. */
   pthread_join (hilo_id, (void*) &primo);
   /* Imprimo el número entregado. */
   printf("El número primo es %d.\n", primo);
   return 0;
}
```

```
20
21 /* Calcula los números primos sucesivamente
22    Retorna el N-esimo numero primo
23    donde N es el valor apuntado por *ARG. */
24 void* calcular_primo (void* arg) {
25    int candidato = 2;
26    int n = *((int*) arg);
27    while (1) {
28       int factor;
29       int es_primo = 1;
30             for (factor = 2; factor < candidato; ++factor)
31          if (candidato % factor == 0) {
32             es_primo = 0;
33          break;
34          }
35          if (es_primo) {
36             if (--n == 0)
37                return (void*) candidato;
38          }
39          ++candidato;
40    }
41    return NULL;
42 }
```

11. ¿Cómo es el funcionamiento del código presentado?
12. Según lo visto en este ejercicio ¿Cuál es la forma de retornar valores desde un hilo? Explique de manera clara y en sus propias palabras.
13. Modifique el programa e) del punto anterior, de manera que pueda obtener el tiempo que demora la ejecución del hilo (Investigue sobre la función **gettimeofday**).

1.3.6 Ejercicios propuestos

14. Se requiere un programa que reciba un vector de números a través de un archivo de texto. La idea es que el programa sume todos los números del vector. Implemente el programa de dos maneras, la primera de una forma estrictamente secuencial. La segunda forma es creando dos hilos, de manera que cada uno de ellos realice la sumatoria de la mitad de los componentes del vector. El hilo 1 sumará los primeros datos del vector y el hilo 2 los últimos. Luego cuando los dos hilos finalicen muestre en pantalla el resultado.

    a. Realice el programa de manera genérica, de tal forma que sea posible ingresar vectores de cualquier tamaño.
    b. Mida el tiempo de ejecución de ambas implementaciones para varios tamaños del vector.
    c. ¿El resultado obtenido es acorde a lo que usted esperaba?
    d. Describa la técnica que usó para realizar la medición del tiempo. ¿Cuáles son las debilidades de esta técnica? ¿Existe otra forma de medir el tiempo de ejecución de un programa?

15. ***Medida de Dispersión:*** el profesor de un curso desea un programa en lenguaje C que calcule la desviación estándar (símbolo **σ** o **s**) de las notas obtenidas por sus estudiantes en el curso.

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \qquad \overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

Requisitos:

a. El número de notas es variable (se debe usar memoria dinámica).
b. El programa debe crear tantos hilos como se especifique en el parámetro de entrada **cantidad_hilos**, se debe ejecutar así:

   $ ./nombre_ejecutable fichero_notas.csv cantidad_hilos

c. Se debe calcular la desviación estándar, implemente la función:

   calculate_standard_deviation().

Referencias

[1] Silberschatz, A., Cagne, G., Galvin, P. Operating system concepts. Wiley, 2005.
[2] Java2 API – Thread. Available online: http://download.oracle.com/javase/1.3/docs/api/java/lang/Thread.html. Last visited: 22/09/11