Programación Avanzada(TC2025)

Tema 5. Programación concurrente

Instituto Tecnológico y de Estudios Superiores de Monterrey. Campus Santa Fe
Departamento de Tecnologías de Información y Electrónica
Dr. Vicente Cubells (vcubells@itesm.mx)

Temario

- Los hilos de POSIX
  - Variables privadas a los hilos
  - Conociendo el ID de un hilo
  - Cediendo el procesador
  - Trabajando con prioridades
  - Señales entre hilos
  - Atributos de los hilos
  - Políticas de planificación
  - El tamaño de la pila

Creando variables privadas

<pthread.h>

- Los procesos mono-hilos trabajan con dos tipos de variables: globales y locales
- Los procesos multi-hilos incorporan un nuevo tipo: variables privadas al hilo (TSD)
- Es la única forma de referir datos privados a un hilo
- Todos los TSD son asociados con una llave global a todos los hilos del proceso
- Al crearse una llave, esta tiene valor NULL en todos los hilos
- Una vez que la llave es creada, cada hilo puede asignarle su propio valor

Creando variables privadas

<pthread.h>

```
int pthread_key_create(pthread_key_t *key,
void (*destructor) (void *));
```

- Un ejemplo:

```
#include <pthread.h>

pthread_key_t key;
int ret;

/* se crea una llave sin destructor */
ret = pthread_key_create(&key, NULL);

/* se crea una llave con destructor */
ret = pthread_key_create(&key, destructor);
```

Estableciendo el valor de variables privadas <pthread.h>

```
int pthread_setspecific(pthread_key_t key, const void *value);
```

- Un ejemplo:

```
#include <pthread.h>

pthread_key_t key;
void *value;
int ret;

/* llave ya creada, se le establece un valor */
ret = pthread_setspecific(key, value);
```

Esta función no libera memoria, si se vuelve a invocar sin liberar la memoria, esta se queda ocupada

Recuperando el valor de variables privadas <pthread.h>

```
void * pthread_getspecific(pthread_key_t key);
```

- Un ejemplo:

```
#include <pthread.h>

pthread_key_t key;
void *value;

/* llave ya creada, se recupera su valor */
value = pthread_getspecific(key);
```

Ver ejemplo: t5c5e1

Obteniendo y comparando IDs

<pthread.h>

pthread_t
pthread_self(**void**);

```
#include <pthread.h>

pthread_t tid;
tid = pthread_self();
```

int pthread_equal(pthread_t tid1, pthread_t tid2);

```
#include <pthread.h>

pthread_t tid1, tid2;
int ret;

ret =
pthread_equal(tid1,
tid2);
```

Ver ejemplo: t5c5e2

Cediendo el procesador

<sched.h>

```
int sched_yield(void);
```

- Un ejemplo:

```
#include <sched.h>

int ret;
ret = sched_yield();
```

Estableciendo la prioridad

<pthread.h>

```
int pthread_setschedparam(pthread_t tid, int policy,
const struct sched_param *param);
```

- Un ejemplo:

```
#include <pthread.h>
pthread_t tid;
int ret;
struct sched_param param;
int priority;
int policy;

/* sched_priority será la prioridad del hilo */
param.__sched_priority = priority;

policy = SCHED_OTHER;

/* parámetros de calendarización del hilo */
ret = pthread_setschedparam(tid, policy, &param);
```

Obteniendo la prioridad

<pthread.h>

```
int pthread_getschedparam(pthread_t tid, int policy,
struct schedparam *param)
```

- Un ejemplo:

```
#include <pthread.h>
pthread_t tid;
int ret;
struct sched_param param;
int priority;
int policy;

/* parámetros de calendarización del hilo */
ret = pthread_getschedparam(tid, policy, &param);

/* sched_priority contiene la prioridad del hilo */
priority = param.__sched_priority;
```

Ver ejemplo: t5c5e3

Enviando señales a un hilo

<pthread.h>

- Funciona similar a los procesos
- Solo se pueden mandar señales entre hilos del mismo proceso

- Un ejemplo:

```
#include <pthread.h>
#include <signal.h>

int sig;
pthread_t tid;
int ret;

ret = pthread_kill(tid, sig);
```

Examinando la máscara de señales <pthread.h>

```
int pthread_sigmask(int how, const sigset_t *new,
sigset_t *old);
```

- Un ejemplo:

```
#include <pthread.h>
#include <signal.h>

int sig;
pthread_t tid;
int ret;
sigset_t old, new;

ret = pthread_sigmask(SIG_SETMASK, &new, &old); /* set new mask */
ret = pthread_sigmask(SIG_BLOCK, &new, &old);   /* blocking mask */
ret = pthread_sigmask(SIG_UNBLOCK, &new, &old); /* unblocking */
```

Ver ejemplo: t5c5e4

Atributos de los hilos

<pthread.h>

- Los atributos especifican un comportamiento
- Solo pueden especificarse durante la creación, luego no pueden modificarse
- Tres funciones se invocan consecutivamente
  - Inicialización -- pthread_attr_init()
  - Cambio de valores: -- una variedad de pthread_attr_*()
    - Una por cada atributo a cambiar
  - Creación-- pthread_create()

Atributos de los hilos

<pthread.h>

- Un ejemplo:

```
#include <pthread.h>

pthread_attr_t tattr;
pthread_t tid;
void *start_routine;
void arg
int ret;

/* Inicialización */
ret = pthread_attr_init(&tattr);

/* Cambiar un valor predeterminado */
ret = pthread_attr_*(&tattr,SOME_ATRIBUTE_VALUE_PARAMETER);

/* Creación */
ret = pthread_create(&tid, &tattr, start_routine, arg);
```

Atributos de los hilos

<pthread.h>

- Atributos predeterminados

Atributo	Valor	Descripción
scope	PTHREAD_SCOPE_PROCESS	Se puede desenlazar del proceso
detachstate	PTHREAD_CREATE_JOINABLE	Estado de salida y el hilo se preservan después de terminar
stackaddr	NULL	Pila definida por el sistema
stacksize	1 megabyte	Tamaño de la pila
inheritsched	PTHREAD_INHERIT_SCHED	Hereda la prioridad del padre

Atributos de los hilos

<pthread.h>

- Algunas funciones para establecer atributos

```
int pthread_attr_setdetachstate(pthread_attr_t *tattr,int detachstate);
          PTHREAD_CREATE_DETACHED, PTHREAD_CREATE_JOINABLE
int pthread_attr_setscope(pthread_attr_t *tattr,int scope);
          PTHREAD_SCOPE_SYSTEM, PTHREAD_SCOPE_PROCESS
int pthread_attr_setschedpolicy(pthread_attr_t *tattr, int policy);
          SCHED_FIFO, SCHED_RR, SCHED_OTHER
int pthread_attr_setinheritsched(pthread_attr_t *tattr, int inherit);
          PTHREAD_INHERIT_SCHED, PTHREAD_EXPLICIT_SCHED
```

Controlando el tamaño de la pila

<pthread.h>

- Tamaño mínimo de la pila definido por PTHREAD_STACK_MIN

- Función que establece el tamaño de la pila en bytes:

```
int
pthread_attr_setstacksize(pthread_attr_t
*tattr, int stacksize);
```

Resumiendo…

- Los hilos pueden tener variables privadas mediante llaves
- Se puede conocer el ID de un hilo al igual que el de un proceso
- Se puede conocer o establecer la prioridad de un hilo pero no influye en el algoritmo de calendarización de procesos

Resumiendo

- Los hilos pueden enviarse señales como los procesos
- Se pueden controlar los atributos conque se crean los hilos a través de funciones set
- El tamaño de la pila de cada hilo puede administrarse pero hay que tener en cuenta que el tamaño mínimo de la pila tiene que ser el correspondiente a una función vacía