High

Performance

Distributed

System

KUAS – High Performance Distributed System

Linux Programming - Pthread

Reporter: Po-Sen Wang

Pthread Function(1/3)

- #include <pthread.h>
- int pthread_create(pthread_t **thread*, pthread_attr_t **attr*, void *(*start_routine)(void *), void **arg*);
  //**create a thread**
- void pthread_exit(void **retval*);
  //**terminate a thread**
- int pthread_join(pthread_t *th*, void ***thread_return*);
  //**wait for thread termination**

Pthread Function(2/3)

- int pthread_create(pthread_t **thread*, pthread_attr_t **attr*, void *(*start_routine)(void *), void **arg*);
  - pthread_t **thread*: thread 的識別字
  - pthread_attr_t **attr*: thread 的屬性。設定為 NULL 表示使用預設
  - void *(*start_routine)(void*): thread 要執行的 function
  - void **arg*: 傳遞給 thread 的參數

Pthread Function(3/3)

- void pthread_exit(void **retval*);
  - void **retval*: thread 結束時回傳的變數
- int pthread_join(pthread_t *th*, void ***thread_return*);
  - pthread_t *th*: thread 識別字
  - void ***thread_return*: 接收 pthread_exit 傳回的變數

HPDS

Example 1 (1/3)

- gcc thread1.c -o thread1 -L/usr/lib/nptl –lpthread
- thread1.c

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

void *thread_function(void *arg);

char message[] = "Hello World";

int main() {
    int res;
    pthread_t a_thread;
    void *thread_result;
    res = pthread_create(&a_thread, NULL, thread_function, (void *)message);
    if (res != 0) {
        perror("Thread creation failed");
        exit(EXIT_FAILURE);
    }
```

Example 1 (2/3)

- thread1.c

```
    printf("Waiting for thread to finish...\n");
    res = pthread_join(a_thread, &thread_result);
    if (res != 0) {
        perror("Thread join failed");
        exit(EXIT_FAILURE);
    }
    printf("Thread joined, it returned %s\n", (char *)thread_result);
    printf("Message is now %s\n", message);
    exit(EXIT_SUCCESS);
}

void *thread_function(void *arg) {
    printf("thread_function is running. Argument was %s\n", (char *)arg);
    sleep(3);
    strcpy(message, "Bye!");
    pthread_exit("Thank you for the CPU time");
}
```

Example 1 (3/3)

- thread1.c

kevin@localhost:~/linux_program/ch11 [80x24]

連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

```
[kevin@localhost ch11]$ ./thread2
thread_function is running. Argument was Hello World
Waiting for thread to finish...
Thread joined, it returned Thank you for the CPU time
Message is now Bye!
[kevin@localhost ch11]$
```

Synchronization – Using Semaphore (1/3)

- #include <semaphore.h>
- int sem_init(sem_t **sem*, int *pshared*, unsigned int *value*);
  //create a semaphore
- int sem_wait(sem_t **sem*);
  //lock a semaphore
- int sem_post(sem_t **sem*);
  //unlock a semaphore
- int sem_destroy(sem_t **sem*);
  //delete a semaphore

Synchronization – Using Semaphore (2/3)

- int sem_init(sem_t **sem*, int *pshared*, unsigned int *value*);
  - sem_t **sem*: semaphore 識別字
  - int *pshared*: 設定為 0 表示僅供目前的 process 及其 thread 使用。非零值表示此 semaphore 與其他 process 共用
  - unsigned int *value*: semaphore 的初始值

Synchronization – Using Semaphore (3/3)

- int sem_wait(sem_t *sem);
  - 若 semaphore 為非零值，則 semaphore 值減 1；若 semaphore 為 0，則呼叫此 function 的 thread 會被 block，直到 semaphore 值不為零。
- int sem_post(sem_t *sem);
  - 對 semaphore 值加 1。

Example 2 – Using Semaphore (1/4)

- thread3.c

kevin@localhost:~/linux_program/ch11 [80x24]

連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

```
[kevin@localhost ch11]$ ./thread4
Input some text. Enter 'end' to finish
hello
You input 5 characters
hihi
You input 4 characters
end

Waiting for thread to finish...
Thread joined
[kevin@localhost ch11]$
```

Example 2 – Using Semaphore (2/4)

- thread3.c

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

void *thread_function(void *arg);
sem_t bin_sem;

#define WORK_SIZE 1024
char work_area[WORK_SIZE];

int main() {
    int res;
    pthread_t a_thread;
    void *thread_result;

    res = sem_init(&bin_sem, 0, 0);
    res = pthread_create(&a_thread, NULL, thread_function, NULL);
```

Example 2 – Using Semaphore (3/4)

- thread3.c

```
    printf("Input some text. Enter 'end' to finish\n");
    while(strncmp("end", work_area, 3) != 0) {
        fgets(work_area, WORK_SIZE, stdin);
        sem_post(&bin_sem);
    }
    printf("\nWaiting for thread to finish...\n");
    res = pthread_join(a_thread, &thread_result);
    if (res != 0) {
        perror("Thread join failed");
        exit(EXIT_FAILURE);
    }
    printf("Thread joined\n");
    sem_destroy(&bin_sem);
    exit(EXIT_SUCCESS);
}
```

Example 2 – Using Semaphore (4/4)

- thread3.c

```
void *thread_function(void *arg) {
    sem_wait(&bin_sem);
    while(strncmp("end", work_area, 3) != 0) {
        printf("You input %d characters\n", strlen(work_area) -1);
        sem_wait(&bin_sem);
    }
    pthread_exit(NULL);
}
```

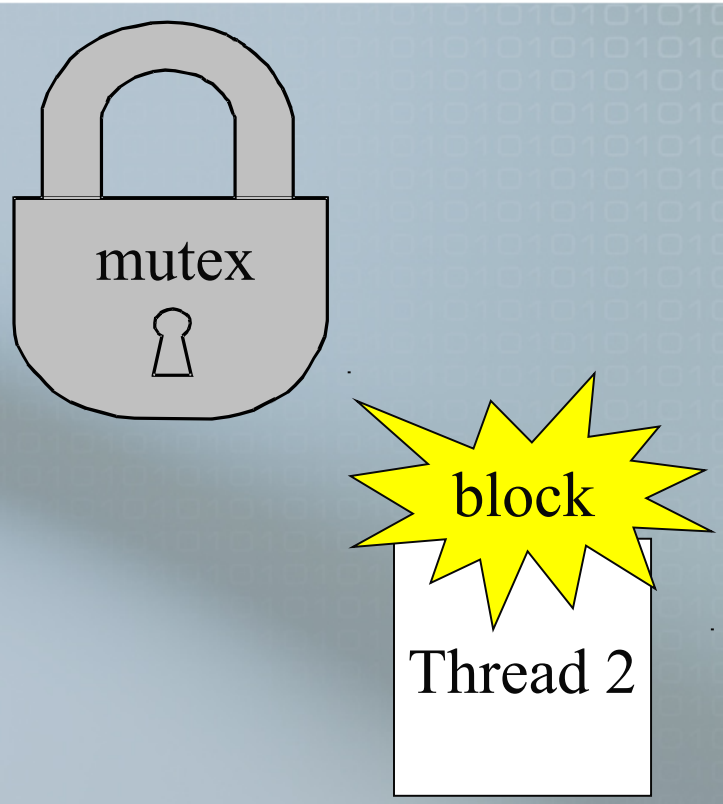
Synchronization – Using Mutex (1/3)

- #include <pthread.h>
- int pthread_mutex_init(pthread_mutex_t **mutex*, const pthread_mutexattr_t **mutexattr*);
  //create a mutex
- int pthread_mutex_lock(pthread_mutex_t **mutex*);
  //lock a mutex
- int pthread_mutex_unlock(pthread_mutex_t **mutex*);
  //unlock a mutex
- int pthread_mutex_destroy(pthread_mutex_t **mutex*);
  //delete a mutex

Synchronization – Using Mutex (2/3)

- int pthread_mutex_init(pthread_mutex_t **mutex*, const pthread_mutexattr_t **mutexattr*);
  - pthread_mutex_t **mutex*: mutex 識別字
  - const pthread_mutexattr_t **mutexattr*: mutex 的屬性。設定為 NULL 表示使用預設。

Synchronization – Using Mutex (2/3)



pthread_mutex_lock

pthread_mutex_unlock

pthread_mutex_lock

Example 3 – Using Mutex(1/4)

- thread4.c

kevin@localhost:~/linux_program/ch11 [80x24]

連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

```
[kevin@localhost ch11]$ ./thread5
Input some text. Enter 'end' to finish
test
You input 4 characters
hpds lab
You input 8 characters
end

Waiting for thread to finish...
Thread joined
[kevin@localhost ch11]$
```

Example 3 – Using Mutex(2/4)

- thread4.c

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

void *thread_function(void *arg);
pthread_mutex_t work_mutex;

#define WORK_SIZE 1024
char work_area[WORK_SIZE];
int time_to_exit = 0;

int main() {
    int res;
    pthread_t a_thread;
    void *thread_result;
    res = pthread_mutex_init(&work_mutex, NULL);
    res = pthread_create(&a_thread, NULL, thread_function, NULL);
```

Example 3 – Using Mutex(3/4)

- thread4.c

```
    pthread_mutex_lock(&work_mutex);
    printf("Input some text. Enter 'end' to finish\n");
    while(!time_to_exit) {
        fgets(work_area, WORK_SIZE, stdin);
        pthread_mutex_unlock(&work_mutex);
        while(1) {
            pthread_mutex_lock(&work_mutex);
            if (work_area[0] != '\0') {
                pthread_mutex_unlock(&work_mutex);
                sleep(1);
            }
            else {
                break;
            }
        }
    }
    pthread_mutex_unlock(&work_mutex);
    printf("\nWaiting for thread to finish...\n");
    res = pthread_join(a_thread, &thread_result);
    printf("Thread joined\n");
    pthread_mutex_destroy(&work_mutex);
    exit(EXIT_SUCCESS);
}
```

Example 3 – Using Mutex(4/4)

- thread4.c

```
void *thread_function(void *arg) {
    sleep(1);
    pthread_mutex_lock(&work_mutex);
    while(strncmp("end", work_area, 3) != 0) {
        printf("You input %d characters\n", strlen(work_area) -1);
        work_area[0] = '\0';
        pthread_mutex_unlock(&work_mutex);
        sleep(1);
        pthread_mutex_lock(&work_mutex);
        while (work_area[0] == '\0' ) {
            pthread_mutex_unlock(&work_mutex);
            sleep(1);
            pthread_mutex_lock(&work_mutex);
        }
    }
    time_to_exit = 1;
    work_area[0] = '\0';
    pthread_mutex_unlock(&work_mutex);
    pthread_exit(0);
}
```

Cancellation(1/2)

- #include <pthread.h>
- int pthread_cancel(pthread_t *thread*);
  //cancel a thread
- int pthread_setcancelstate(int *state*, int **oldstate*);
  //set cancellation state
- int pthread_setcanceltype(int *type*, int **oldtype*);
  //set cancellation type

Cancellation(2/2)

- int pthread_setcancelstate(int *state*, int **oldstate*);

  int *state*: 設定為 PTHREAD_CANCEL_ENABLE 即表示允許取消 thread 的請求；
  設定為 PTHREAD_CANCEL_DISABLE 即表示忽略取消的請求。

  int **oldstate*: 此指標指向前一個狀態

- int pthread_setcanceltype(int *type*, int **oldtype*);

  int type: 設定為 PTHREAD_CANCEL_ASYNCHRONOUS 則立即取消 thread；
  設定為 PTHREAD_CANCEL_DEFERRED 則會遇到取消點才會取消 thread。取消點即是下列函數：
  pthread_join、pthread_cond_wait、pthread_testcancel 等

  int **oldtype*: 此指標指向前一個型態

Example 4 (1/3)

- thread7.c

kevin@localhost:~/linux_program/ch11 [80x24]

連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

```
[kevin@localhost ch11]$ ./thread8
thread_function is running
Thread is still running (0)...
Thread is still running (1)...
Cancelling thread...
Waiting for thread to finish...
[kevin@localhost ch11]$
```

Example 4 (2/3)

- thread7.c

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

void *thread_function(void *arg);

int main() {
    int res;
    pthread_t a_thread;
    void *thread_result;

    res = pthread_create(&a_thread, NULL, thread_function, NULL);
    sleep(2);
    printf("Cancelling thread...\n");
    res = pthread_cancel(a_thread);
    printf("Waiting for thread to finish...\n");
    res = pthread_join(a_thread, &thread_result);
    exit(EXIT_SUCCESS);
}
```

Example 4 (3/3)

- thread7.c

```
void *thread_function(void *arg) {
    int i, res, j;
    res = pthread_setcancelstate(PTHREAD_CANCEL_ENABLE, NULL);
    res = pthread_setcanceltype(PTHREAD_CANCEL_DEFERRED, NULL);
    printf("thread_function is running\n");
    for(i = 0; i < 10; i++) {
        printf("Thread is still running (%d)...\n", i);
        sleep(1);
    }
    pthread_exit(0);
}
```

Example – Multi-Thread (1/4)

- thread8a.c

kevin@localhost:~/linux_program/ch11 [80x24]

連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

```
[kevin@localhost ch11]$ ./thread9a
thread_function is running. Argument was 0
thread_function is running. Argument was 1
thread_function is running. Argument was 2
thread_function is running. Argument was 3
thread_function is running. Argument was 4
thread_function is running. Argument was 5
Waiting for threads to finish...
Bye from 5
Picked up a thread
Bye from 1
Bye from 0
Bye from 2
Bye from 3
Bye from 4
Picked up a thread
Picked up a thread
Picked up a thread
Picked up a thread
Picked up a thread
All done
[kevin@localhost ch11]$
```

Example – Multi-Thread (2/4)

- thread8a.c

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

#include <pthread.h>

#define NUM_THREADS 6

void *thread_function(void *arg);

int main() {

  int res;
  pthread_t a_thread[NUM_THREADS];
  void *thread_result;
  int lots_of_threads;
```

Example – Multi-Thread (3/4)

- thread8a.c

```
  for(lots_of_threads = 0; lots_of_threads < NUM_THREADS; lots_of_threads++)
    res = pthread_create(&(a_thread[lots_of_threads]), NULL,
                         thread_function, (void *)lots_of_threads);
  printf("Waiting for threads to finish...\n");
  for(lots_of_threads = NUM_THREADS - 1; lots_of_threads >= 0; lots_of_threads--) {
    res = pthread_join(a_thread[lots_of_threads], &thread_result);
    if (res == 0) {
      printf("Picked up a thread\n");
    } else {
      perror("pthread_join failed");
    }
  }

  printf("All done\n");

  exit(EXIT_SUCCESS);
}
```

Example – Multi-Thread (4/4)

- thread8a.c

```
void *thread_function(void *arg) {
    int my_number = (int)arg;
    int rand_num;

    printf("thread_function is running. Argument was %d\n", my_number);
    rand_num=1+(int)(9.0*rand()/(RAND_MAX+1.0));
    sleep(rand_num);
    printf("Bye from %d\n", my_number);

    pthread_exit(NULL);
}
```

Condition Variables

- pthread_cond_init (condition,attr)
  pthread_cond_destroy (condition)
- pthread_condattr_init (attr)
- pthread_condattr_destroy (attr)

Condition Variables

- Example :condition_variable

Homework 1

- 撰寫一個程式， create 10 threads ，共同計算計算 1+2+3+4+…..+10000
- Thread 0, 計算 1~1000 總和，並將總和累計至共用變數 total
- Thread 1, 計算 1001~2000 總和，並將總和累計至共用變數 total
- 以此類推
- MAIN thread join 10 個 child thread 後，列印出總和 total

Homework 2

- 定義一結構 barrier 如下：
  - count_mutex
  - cond_var
  - count // the number of threads that has arrived at the barrier
  - limit // the number of threads that will arrive at the barrier
- void barrier_init(struct barrier *, int num);
- void barrier_arrive(struct barrier *);
- 寫一個程式 create 10 threads
- 每一個 thread 都執行下列 function

Homework 2

```
work(void* arg)
{  int myid= (int) arg;
   int i;
    for(i=0 ; i<3 ; i++){
        printf("Thread %d echo in %d  iteration\n", myid, i);
          barrier_arrive(&bar);
}
```