

Introduction to C

Threads

Instructor: Yin Lou

02/16/2011

Processes vs. Threads

Processes

- ▶ Multiple simultaneous programs
- ▶ Independent memory space
- ▶ Independent open file-descriptors

Threads

- ▶ Multiple simultaneous functions
- ▶ Shared memory space
- ▶ Shared open file-descriptors

Threads Examples

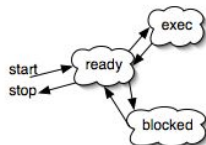
- ▶ Graphical User Interfaces (GUIs)
 - ▶ The GUI is usually put on a separate thread from the “app engine”
 - ▶ GUI remains responsive even if app blocks for processing
- ▶ Web Browser Tabs
 - ▶ Each tab is managed by a separate thread for rendering
 - ▶ Web pages render “simultaneously”
 - ▶ Note: Google Chrome actually uses a separate process per tab

Threads

- ▶ One copy of the heap
- ▶ One copy of the code
- ▶ **Multiple** stacks

Threads

- ▶ One copy of the heap
- ▶ One copy of the code
- ▶ **Multiple** stacks
- ▶ Life Cycle



pthread

- ▶ `#include <pthread.h>`

pthread

- ▶ `#include <pthread.h>`
- ▶ Define a worker function

pthread

- ▶ `#include <pthread.h>`
- ▶ Define a worker function
 - ▶ `void *foo(void *args) { }`

pthread

- ▶ `#include <pthread.h>`
- ▶ Define a worker function
 - ▶ `void *foo(void *args) { }`
- ▶ Initialize `pthread_attr_t`

pthread

- ▶ `#include <pthread.h>`
- ▶ Define a worker function
 - ▶ `void *foo(void *args) { }`
- ▶ Initialize `pthread_attr_t`
 - ▶ `pthread_attr_t attr;`
 - ▶ `pthread_attr_init(&attr);`

pthread

- ▶ `#include <pthread.h>`
- ▶ Define a worker function
 - ▶ `void *foo(void *args) { }`
- ▶ Initialize `pthread_attr_t`
 - ▶ `pthread_attr_t attr;`
 - ▶ `pthread_attr_init(&attr);`
- ▶ Create a thread

pthread

- ▶ `#include <pthread.h>`
- ▶ Define a worker function
 - ▶ `void *foo(void *args) { }`
- ▶ Initialize `pthread_attr_t`
 - ▶ `pthread_attr_t attr;`
 - ▶ `pthread_attr_init(&attr);`
- ▶ Create a thread
 - ▶ `pthread_t thread;`
 - ▶ `pthread_create(&thread, &attr, worker_function, arg);`

pthread

- ▶ `#include <pthread.h>`
- ▶ Define a worker function
 - ▶ `void *foo(void *args) { }`
- ▶ Initialize `pthread_attr_t`
 - ▶ `pthread_attr_t attr;`
 - ▶ `pthread_attr_init(&attr);`
- ▶ Create a thread
 - ▶ `pthread_t thread;`
 - ▶ `pthread_create(&thread, &attr, worker_function, arg);`
- ▶ Exit current thread

- ▶ `#include <pthread.h>`
- ▶ Define a worker function
 - ▶ `void *foo(void *args) { }`
- ▶ Initialize `pthread_attr_t`
 - ▶ `pthread_attr_t attr;`
 - ▶ `pthread_attr_init(&attr);`
- ▶ Create a thread
 - ▶ `pthread_t thread;`
 - ▶ `pthread_create(&thread, &attr, worker_function, arg);`
- ▶ Exit current thread
 - ▶ `pthread_exit(status)`

Example

```
#include <stdio.h>
#include <pthread.h>
#define NUM_THREADS    5

void *print_hello(void *threadid)
{
    long tid;
    tid = (long) threadid;
    printf("Hello World! It's me, thread #%ld!\n", tid);
    pthread_exit(NULL);
}

int main (int argc, char *argv[])
{
    pthread_t threads[NUM_THREADS];
    int rc;
    long t;
    for (t = 0; t < NUM_THREADS; t++)
    {
        printf("In main: creating thread %ld\n", t);
        rc = pthread_create(threads + t, NULL, print_hello, (void *) t);
        if (rc)
        {
            printf("ERROR; return code from pthread_create() is %d\n", rc);
            exit(-1);
        }
    }
    pthread_exit(NULL);
}
```

Thread Management

- ▶ `pthread_join(threadid, status)`
- ▶ `pthread_detach(threadid)`

Example

```
#include <stdio.h>
#include <pthread.h>
#define NUM_THREADS    5

void *print_hello(void *threadid) {
    long tid;
    tid = (long) threadid;
    printf("Hello World! It's me, thread %ld!\n", tid);
    pthread_exit(NULL);
}

int main (int argc, char *argv[]) {
    pthread_t threads[NUM_THREADS];
    int rc;
    long t;
    for (t = 0; t < NUM_THREADS; t++) {
        printf("In main: creating thread %ld\n", t);
        rc = pthread_create(threads + t, NULL, print_hello, (void *) t);
        if (rc) {
            printf("ERROR; return code from pthread_create() is %d\n", rc);
            exit(-1);
        }
    }
    /* wait for all threads to complete */
    for (t = 0; t < NUM_THREADS; t++) {
        pthread_join(threads[t], NULL);
    }
    pthread_exit(NULL);
}
```

Compiling

- ▶ Use “-pthread”

Compiling

- ▶ Use “-pthread”

Makefile

```
CC:=gcc
OPTIONS:=-O2
LIB_PATH:=-pthread

SRC_DIR:=src
DST_DIR:=bin

default:
    $(CC) $(OPTIONS) $(LIB_PATH) \
    $(SRC_DIR)/*.c -o $(DST_DIR)/test
clean:
    cd $(DST_DIR); rm test
```