

CSC 1600: Operating Systems

Basics of Threads

POSIX Threads

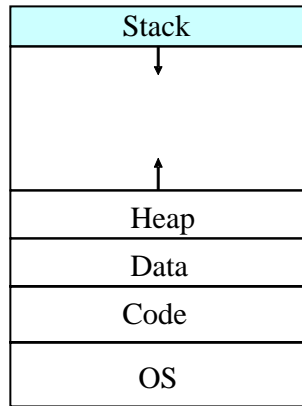
What are Threads?

- Thread
 - Independent stream of instructions
 - Basic unit of CPU utilization

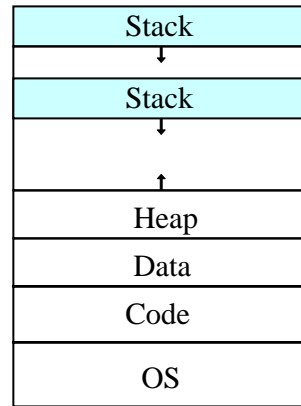
- A thread contains
 - A thread ID
 - A register set (including the Program Counter PC)
 - An execution stack

- A thread shares with its sibling threads
 - The code, data and heap section
 - Other OS resources, such as open files and signals

Process Address Space Revisited

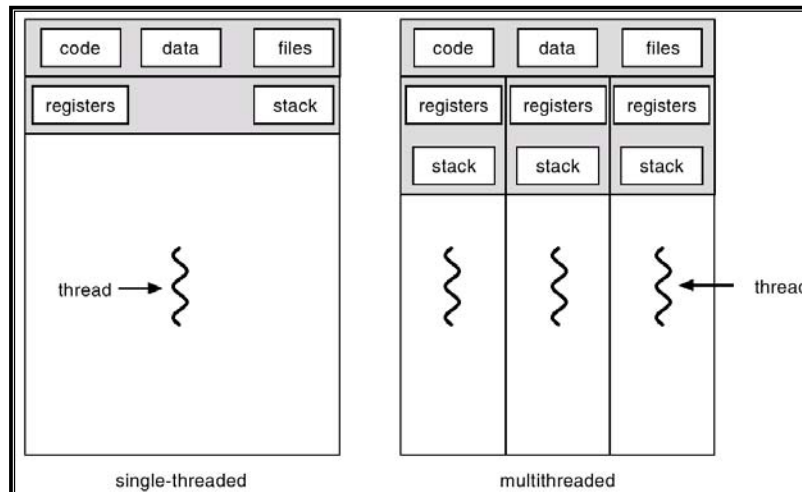


(a) Process with Single Thread



(b) Process with Two Threads

Single and Multi-Threaded Processes



Multi-Threaded Processes

- ❑ Each thread has a private stack
- ❑ But threads share the process address space!
- ❑ **There's no memory protection!**
- ❑ Threads could potentially write into each other's stack

Why use Threads?

- ❑ A specific example, a Web server:

```
do {  get web page request from client
      check if page exists and client has permissions
      transmit web page back to client
} while(1);
```
- ❑ If transmission takes very long time, server is unable to answer other client's requests. Solution:

```
do {  get web page request from client
      check if page exists and client has permissions
      create a thread to transmit web page back to client
} while(1);
```

Benefits of Threads

- ❑ Responsiveness
 - ❑ Program continues even one part is blocked
- ❑ Resource Sharing
 - ❑ Point to the same process: memory and resources are shared!
- ❑ Economy
 - ❑ Memory allocation for process is costly
 - ❑ Context switching for process is costly
- ❑ Utilization of Multiprocessor Architectures
 - ❑ Each thread is assigned one unique processor

Threading Issues

- ❑ A badly-behaved thread can damage other threads
 - ❑ Threads share the data of the master process
 - ❑ Threads have read/write access to other threads' memory

POSIX Threads (1)

Thread call	Description
Pthread_create	Create a new thread
Pthread_exit	Terminate the calling thread
Pthread_join	Wait for a specific thread to exit

Figure 2-14. Some of the Pthreads function calls.

POSIX Threads (2)

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

#define NUMBER_OF_THREADS 10

void *print_hello_world(void *tid)
{
    /* This function prints the thread's identifier and then exits. */
    printf("Hello World. Greetings from thread %d0, tid);
    pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
    /* The main program creates 10 threads and then exits. */
    pthread_t threads[NUMBER_OF_THREADS];
    int status, i;

    for(i=0; i < NUMBER_OF_THREADS; i++) {
        printf("Main here. Creating thread %d0, i);
        status = pthread_create(&threads[i], NULL, print_hello_world, (void *)i);

        if (status != 0) {
            printf("Oops. pthread_create returned error code %d0, status);
            exit(-1);
        }
    }
    exit(NULL);
}
```

Figure 2-15. An example program using threads.