PROGRAMMING WITH PTHREADS

Learning Outcomes

At the end of this session, the students should be able to:

1. Discuss how threads are created;
2. Understand how the Linux system handles threads; and
3. Create C programs which use threads.

Content

Parts of a Process

Recall that a process contains instructions (***code section*** of a program), variables (***data section*** of a program) and other information such as opened ***files***. The state of execution of a process in a computer system is stored in the ***registers***. A ***stack***, which is a designated memory area, is also needed for procedure and function calls as well as parameter passing. When the instructions of a process are executed by the CPU, it is called a ***thread of execution*** or simply a ***thread***. Normally, a process has only one thread but it is possible to have multiple threads. Unique to a thread is its own values of the *registers* and its *stack* to store its state of execution.

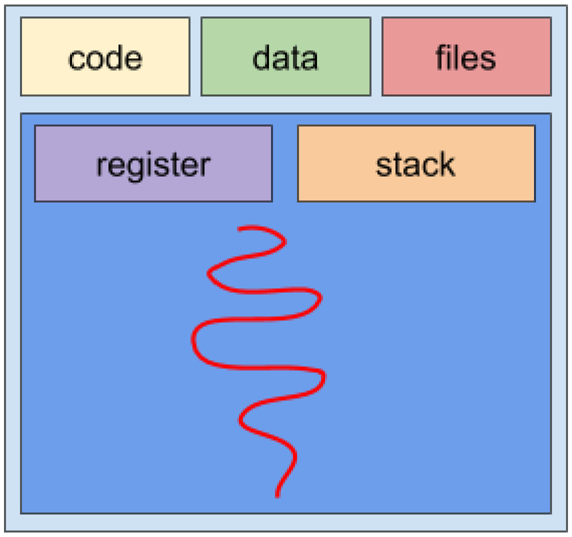

Fig. 1 *A single-threaded process which has its own reference to its register and stack.*

Threads

A ***thread*** is a basic unit of CPU utilization; it comprises a thread ID, a program counter (PC), a register set, and a stack. It shares with other threads belonging to the same process its code section, data section, and other operating-system resources, such as open files and signals.

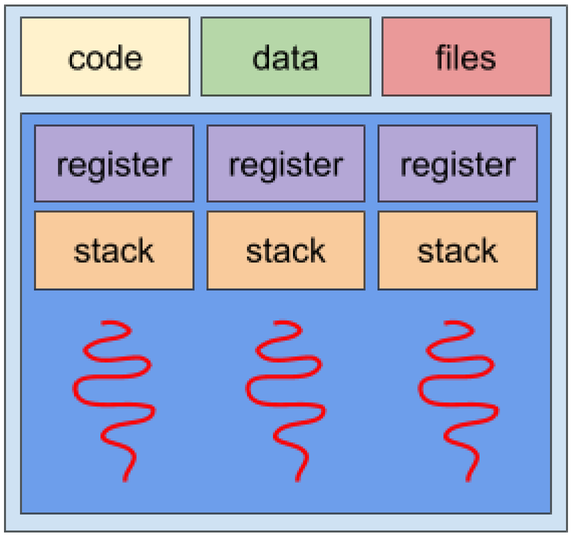

Fig. 2 *A multi-threaded process with each thread having its own register and stack but sharing the same code, data, and files of the process*

Also, a ***thread*** is an execution of a portion of a program within a process. A thread calls a certain procedure or function of a given program. Since threads are similar to processes in some ways, it is also called a *lightweight process*. *Fig. 3* shows a process tree with each of the processes having a single thread. *Fig. 4* shows a single process having multiple threads.

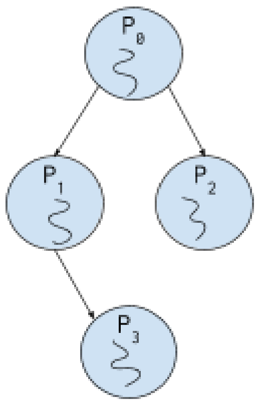

Fig. 3 *A process tree with four process each having a single thread.*

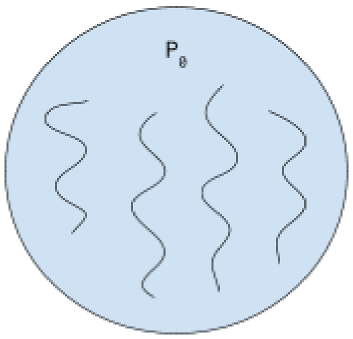

Fig. 4 *A single process having four threads.*

Like a process, a thread can either be: (1) running, (2) waiting, (3) ready or (4) terminated. Each thread in a process executes only a portion of the process. Processes have their own copy of the variables in a program while threads can share the variables of the process where they are created.

PThreads

Pthreads, or ***Portable Operating System Interface*** (POSIX) Threads, is the library of the C Programming Language for managing threads. These functions and types can be included in your C program using the **pthread.h** header file.

Thread Creation

```
int pthread_create(pthread_t *tid, const pthread_attr_t *attr,
      void* (*thread_function)(void *), void *arguments);
```

where `tid` is the address of the thread id,
`attr` is the attribute of the thread (values are defined in **pthread.h**),
`thread_function` is the pointer to the function to be executed, and
`arguments` are the arguments needed by thread_function.

- ❏ This function creates a new thread identified by `tid`.
- ❏ To use the default thread attributes, `attr`, you can pass **NULL** to the second parameter. See http://www.it.uom.gr/teaching/c_sys/node30.html to view the available thread attributes.
- ❏ The thread will terminate once `thread_function` terminates.
- ❏ If your `thread_function` needs more than one parameter, `arguments`, you need to create a **structure** that holds the values you will pass.
- ❏ Returns `0` if thread creation succeeds, else returns `error_number`.

Thread Waiting (*Joining*)

```
int pthread_join(pthread_t *tid, void **status_ptr);
```

where `tid` is the address of the thread id, and
`status_ptr` is the pointer to the exit status.

- ❏ The `status_ptr` pointer will point to the void pointer returned by the thread.

Thread Termination

```
int pthread_exit(void *status);
```

where `status` is the return value of the thread.

Learning Experiences

Students will be given sample codes for demonstration.

Assessment Tool

A programming exercise using threads.

References

[1] "The Pthreads Library." 22 Feb. 2016 <http://www.cs.nmsu.edu/~jcook/Tools/pthreads/library.html>