## THREADS

A thread is an execution of a portion of a program within a process. A thread calls a *certain procedure of a given program*. Since threads are similar to processes in some ways, it is also called *lightweight process*.

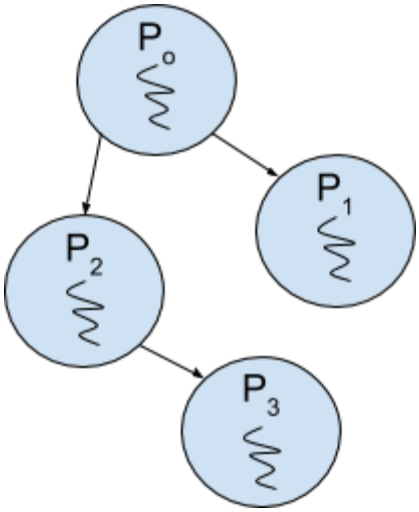


**Fig. 1.** A process having several children, each having its own kernel thread.

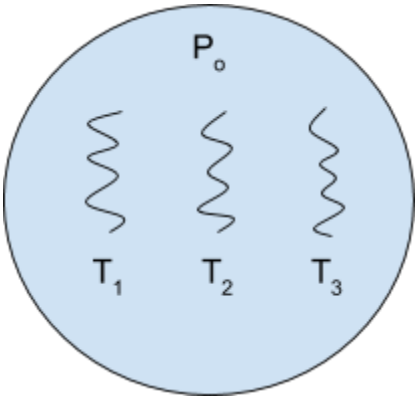


**Fig. 2.** A process having several threads within.

Like a process, a thread can either be a) *running*, b) *blocked*, c) *ready*, or d) *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.

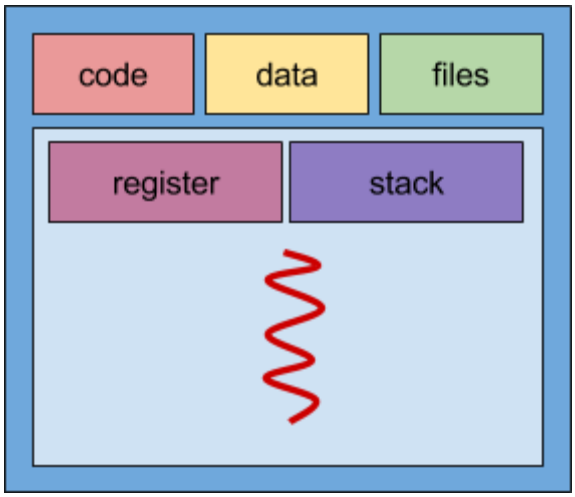


**Fig. 3.** A single-threaded process. The thread has its own register and stack.

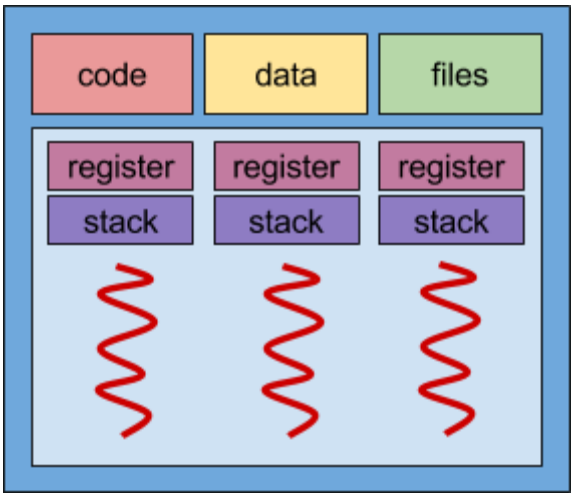


**Fig. 4.** A multithreaded process. Each thread has its own register and stack, however, they share the same code, data, and files.

## 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.
To use the default thread attributes, you can pass NULL to the second parameter.
The thread will terminate once `thread_function` terminates.
If your `thread_function` needs more than one parameter, you need to create a structure that holds the values you will pass.

### 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.

### REFERENCES:

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