RUTGERS

Principles of Programming Languages
CS 314

Recitation 11

Topics Today

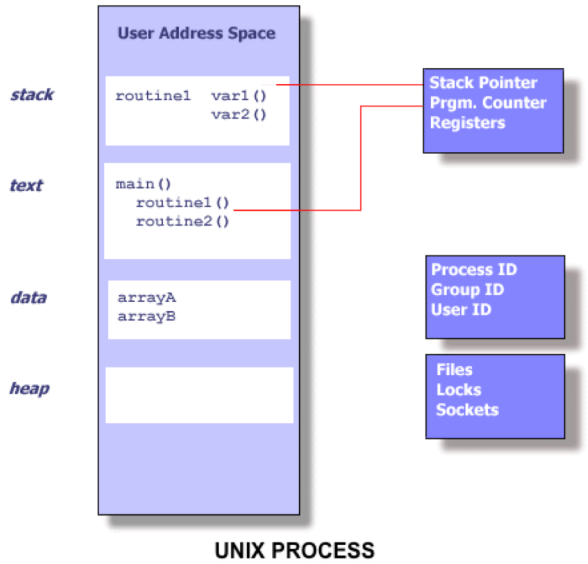
- Thread & Process
- Pthreads overview
- Pthreads API
- Project 3

What is a thread?

- A thread is defined as an independent stream of instructions that can be scheduled to run by the operating system.
- In most case, a thread is a component of a process.
- Multiple threads can exist within one process, executing concurrently and sharing resources
- So, what is a (UNIX) process?

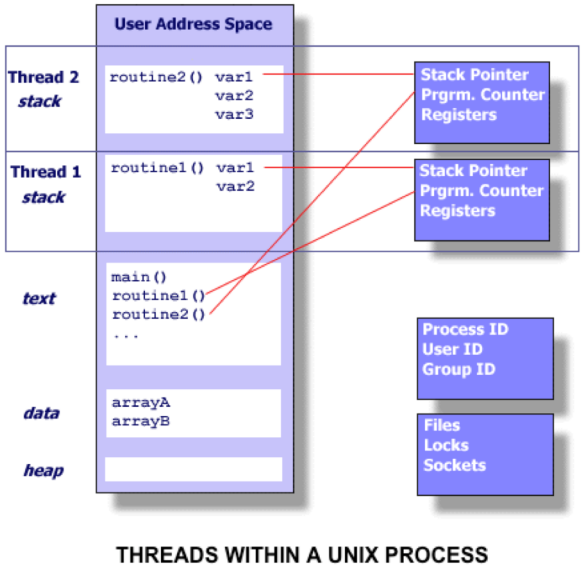
UNIX Process

- A process is an instance of a program in execution
- Each process is associated with a User Address Space (UAS)
  - Program code (Text)
  - Program data (Data)
  - Stack
  - Heap
  - …



Threads within a UNIX Process

- Threads exist within a process and share the process resource
- A thread has its own
  - Thread ID
  - Register set
  - Stack
  - …
- Threads within the same process share
  - Program code (Text)
  - Program data (Data)
  - Heap
  - …



THREADS WITHIN A UNIX PROCESS

Pthreads overview

- Pthreads defines a set of C programming language types, functions and constants.
- It is implemented with a pthread.h header and a thread library.
- Pthreads is …
  - light weight
  - efficient communication/data exchange
  - …

Pthreads Hello World

- Here is the hello world program of pthreads (hello.c)
- Compile it with:

  gcc hello.c –o hello **–pthread**

- All pthreads examples in this recitation will be uploaded to Sakai later

Pthreads API

- pthread_create()
- pthread_exit()
- pthread_join()
- pthread_barrier_*()
- Passing arguments to threads
- …

pthread_create()

- pthread_create creates a new thread and makes it executable.
- This routine can be called any number of times from anywhere within in your code

- pthread_create(thread, attr, start_routine, arg)
  - **thread**: An opaque, unique thread identifier.
  - **attr**: Thread attributes or NULL for default values
  - **start_routine**: the C routine that the thread will execute once it is created.
  - **arg**: A single argument that may be passed to *start_routine* or NULL if no argument is to be passed

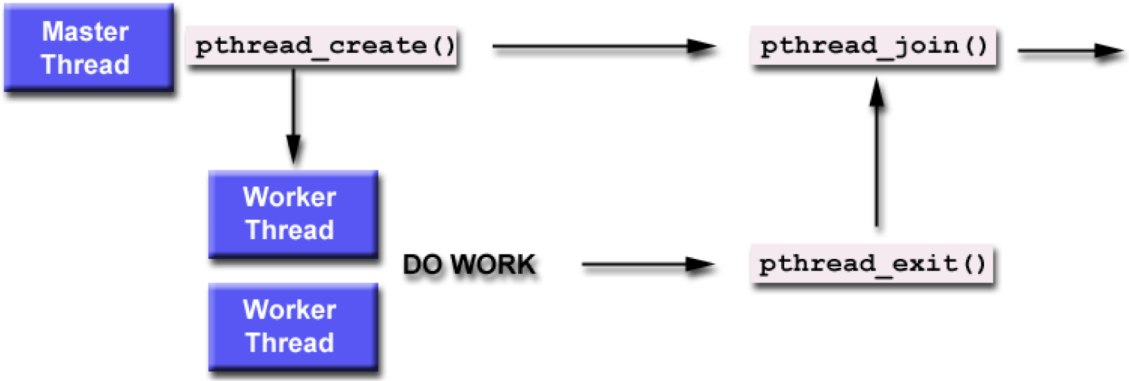
pthread_exit()

- Threads terminate by
  - explicitly calling pthread_exit()
  - letting the function return
  - a call to the function exit() which will terminate the process including any threads.

- void pthread_exit(void *retval);
  - **retval**: return value of pthread_exit()

pthread_join()

- "Joining" can accomplish synchronization between threads.



- pthread_join() blocks the calling thread until the certain thread terminates
- int pthread_join(pthread_t *thread*, void ***retval*)

pthread_join()

- Let's see an example (join.c)

```
gcc join.c –o join -pthread -lm
```

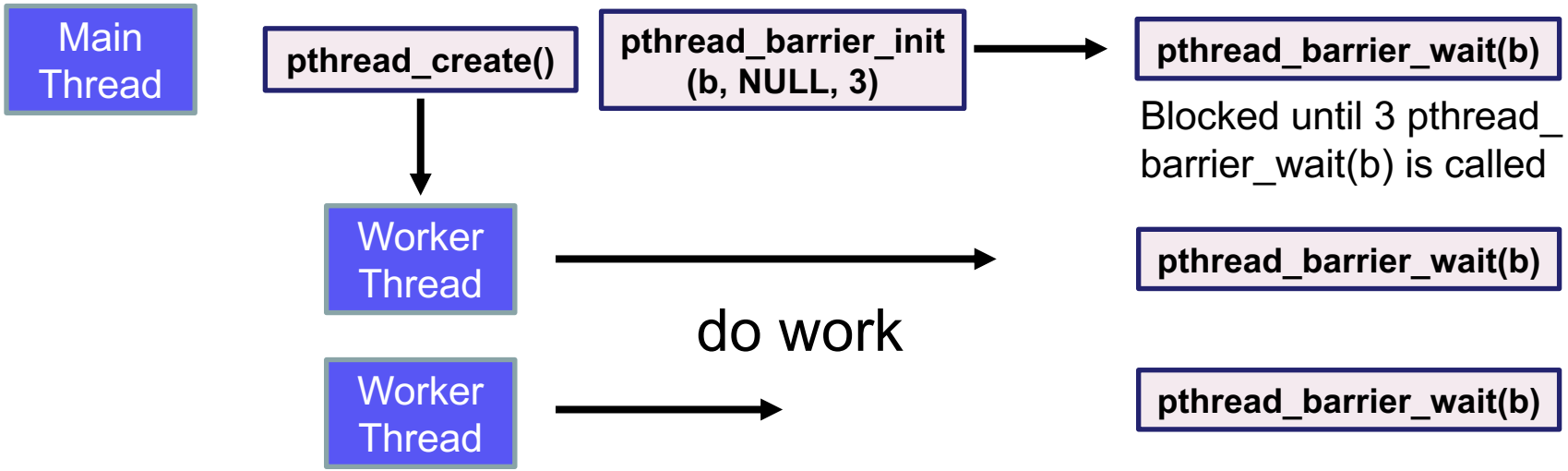
pthread_barrier_*()

- pthread_barrier_t barrier
- pthread_barrier_init(barrier, attr, count)
- pthread_barrier_wait()
- pthread_barrier_destroy(barrier)
- pthread_barrier_* is used for synchronization at the barrier

pthread_barrier_*()

- Let's see an example (barrier.c)

  gcc barrier.c –o barrier -pthread



Passing Arguments to Threads

- The pthread_create() routine permits the programmer to pass one argument to the thread start routine.
- If multiple arguments must be passed, this limitation is easily overcome by creating a structure which contains all of the arguments, and then passing a pointer to that structure in the pthread_create() routine.
- We pass the thread id in hello.c
- Let's see an example for multiple arguments (arg.c)

```
gcc arg.c –o arg -pthread
```

Project 3

- It will be released this week
- Start early!
- Read the project documentation
- Understand the algorithms and the pseudo codes before you implement them
- Understand what is pthreads and how to use pthreads API
- Come to office hour if you need help