CSE 333 – SECTION 8

Threads

Threads

- A (single) thread is a sequential execution of a program.
- Contained within a process.
- Multiple threads can exist within the same process.
  - Every process starts with one thread, can spawn more.
- Threads in a single process share one address space
  - Instructions (code)
  - Static (global) data
  - Dynamic (heap) data
  - Environment variables, open files, sockets, etc.
- Each thread has it’s own stack.

POSIX threads (Pthreads)

- The POSIX standard provides APIs for creating and manipulating threads.
- Part of the standard C/C++ libraries, declared in pthread.h.
- Use -pthread option on gcc/g++ to compile/load.

Core pthread functions

- pthread_create(thread, attr, start_routine, arg)
- pthread_exit(status)
- pthread_join(thread, status)
- pthread_cancel (thread)

pthread_create

```
#include <pthread.h>
int pthread_create( pthread_t *thread,
                    const pthread_attr_t *attr,
                    void *(*start_routine) (void *),
                    void *arg );
```

- Create a new thread and run start_routine with arg as its parameter.
- Arguments:
  - **thread:** A unique identifier for the new thread.
  - **attr:** An object that may be used to set thread attributes. Use NULL for defaults.
  - **start_routine:** The C routine the thread will execute once it is created.
  - **arg**: A single argument that is passed to *start_routine*. Can be anything, but must cast to void* in the call. Use NULL if no appropriate argument.

Terminating Threads

- There are several ways in which a thread may be terminated:
  - Thread starting routine does a normal return.
  - The thread calls `pthread_exit` to terminate the thread.
  - The thread is canceled by another thread using `pthread_cancel`.
  - The entire process is terminated by a call to `exec(), exit()`or by a return from `main()`.

pthread_exit

```
void pthread_exit(void *retval);
```

- Terminate the current thread; `retval` can be retrieved by another thread after a successful join (use NULL if no useful information).
- Often not needed if the initial function in the thread returns normally.
- `main()`can call `pthread_exit()` to finish and leave other threads running; all other threads terminate when `main()` returns or exits by calling `exit()`.

pthread_join

```
int pthread_join(pthread_t thread, void **retval);
```

Synchronization between threads.

- `pthread_join` blocks the calling thread until the specified thread terminates and then the calling thread continues (i.e., “joining” the terminated thread).
- Only threads that are created as joinable can be joined; a thread created as detached can never be joined. (See `pthread_create`)
- The target thread's termination return status can be obtained if it was specified in the target thread's call to `pthread_exit()`.

Demo: *pthread_demo.c*

Section exercise (not to be turned in)

- Create a program that spawns two or three different threads, each of which prints a numeric sequence. Examples:
  - First n odd numbers
  - First n factorials
  - First n primes
- Use pthread_demo.c for ideas, but the structure might not be the same.
- Can you do something in the threads (maybe sleep()) so that different runs of the program don’t always produce the same output?

Exercise 11

- Implement a chat program in C++.
- Create two threads – Server and the Client.
- The Client thread reads from stdin, and writes anything the user types to the network.
- The Server thread reads from the network, and writes anything that it receives to stdout.
- Feel free to use any sample code from lectures or other exercises to implement the above functions.

Questions?