Lab 2 - pthreads

Threading model in Ubuntu

There are several threading models that have been found out.
To learn more about the different threading models you can refer to this

Ubuntu uses the one-to-one threading model.

Threading in C-Language

What is pthread.h Library

The POSIX thread (pthread) libraries are a standards-based thread API for C/C++. It allows one to spawn a new concurrent process flow. It is most effective on multi-processor or multi-core systems where the process flow can be scheduled to run on another processor thus gaining speed through parallel or distributed processing. Threads require less overhead than "forking" or spawning a new process because the system does not initialize a new system virtual memory space and environment for the process. While most effective on a multiprocessor system, gains are also found on uni-processor systems which exploit latency in I/O and other system functions which may halt process execution.

How to compile C-Language code that uses pthread.h library

For compiling a c-code that uses pthread.h library, we do the following:

Assume the code is written in a file called **main.c**, then to compile it we run:

```
gcc main.c -lpthread
```

and to run the compiled code we run:

```
./a.out
```

Both can be combined into one single command:

```
gcc main.c -lpthread -o a && ./a
```

Resource for learning more about pthread.h library:

Link

Why did we use the -lpthread flag?

Add support for multithreading using the POSIX threads library. This option sets flags for both the preprocessor and the linker.

Multi-threading and performance

Does making the program multi-threaded always improve the performance?

Short answer: **No**.

Multi threading improves performance by allowing multiple CPUs to work on a problem at the same time; but it only helps if two things are true: as long as the CPU speed is the limiting factor (as opposed to memory, disk, or network bandwidth) AND so long as multithreading doesn't introduce so much additional work (aka "overhead") that the benefit is negated.

Too much multithreading is just as much a problem as too little. Consider the task of assembling a house: it's a big job, and having more than one worker is helpful. But there are limits. Ten workers is great! A hundred is challenging. Ten thousand probably means the house will get built more slowly than with just twenty (do you have enough tools? Enough bathrooms? Is there even enough work to do?). And there are also jobs that cannot be split; such as the old joke of hiring 9 women in order to gestate a baby in only one month instead of nine.

When to use/not use multi-threading

When to use Multi-threading:

- Improved application responsiveness - any program in which many activities are not dependent upon each other can be redesigned so that each activity is defined as a thread.
- Improved program structure - many programs are more efficiently structured as multiple independent or semi-independent units of execution instead of as a single, monolithic thread. Multithreaded programs can be more adaptive to variations in demands than single threaded programs.
- Rendering of animation
- GUI programming often helps to have at least two threads when doing something slow, e.g. processing large number of files - this allows the interface to remain responsive whilst the worker does the hard work

Other than those reasons (and similar ones) its not a good idea to use multi-threading.

💻 Assignment 2 - pthreads