CS221
C and Systems Programming

Threads

A computer thread is a single stream of instructions

```
double score = 0.0;
while (!isAwinner()) {
   score += thisGame();
}
printFinal(score);
```


CPU starts running instructions "at the top", continues step-by-step

Parallel processing

Every computer CPU has multiple "cores" i.e. processors
- Can run multiple programs at once

How many processors?
- **lscpu**

Talk about Threads per Core (hyperthreading) and Cores per socket and sockets per server

Multitasking

So how can I run >12 programs at the same time?

- **Task Switching**

Task Switching gives faster responsiveness i.e. lower latency to a program

Can we get more computing power?

An Operating System feature, not a CPU feature.
Switch between ALL active tasks ~1000x/second.
Show all Windows tasks

Parallel Processing

Yeah, if we can allocate more than one thread to a program

…and if all of those threads get a CPU core

pthread library

```
pthread_t thr1;
pthread_create(&thr1, NULL, methodToRun, (void*) &inputVar);
```

`methodToRun` returns a `void*` and only input arg is a `void*`

`NULL` is a set of thread attributes, which we won't discuss

There are several multithreading libraries in C – pthread seems most widely available (on most OS's)
Cast a struct address to void* for input and output, if need >1 arg

pthread library

```
pthread_t thr1;
pthread_create(&thr1, NULL, methodToRun, (void*) &inputVar);
```

`methodToRun` returns a `void*` and only input arg is a `void*`

`NULL` is a set of thread attributes, which we won't discuss

`pthread_create` creates and starts a new thread that runs in parallel with "main thread"

pthread library

```
long RetVal;
pthread_join(thr1, (void**) &RetVal);
```

`pthread_join()` is called in main thread. Waits until `thr1` exits.

Also can modify values in input struct!

Let's do a CODE-ALONG, to print a bunch of numbers out. (demo.c)
Use usleep() method so it doesn't complete in a microsecond
After first try, add third thread.

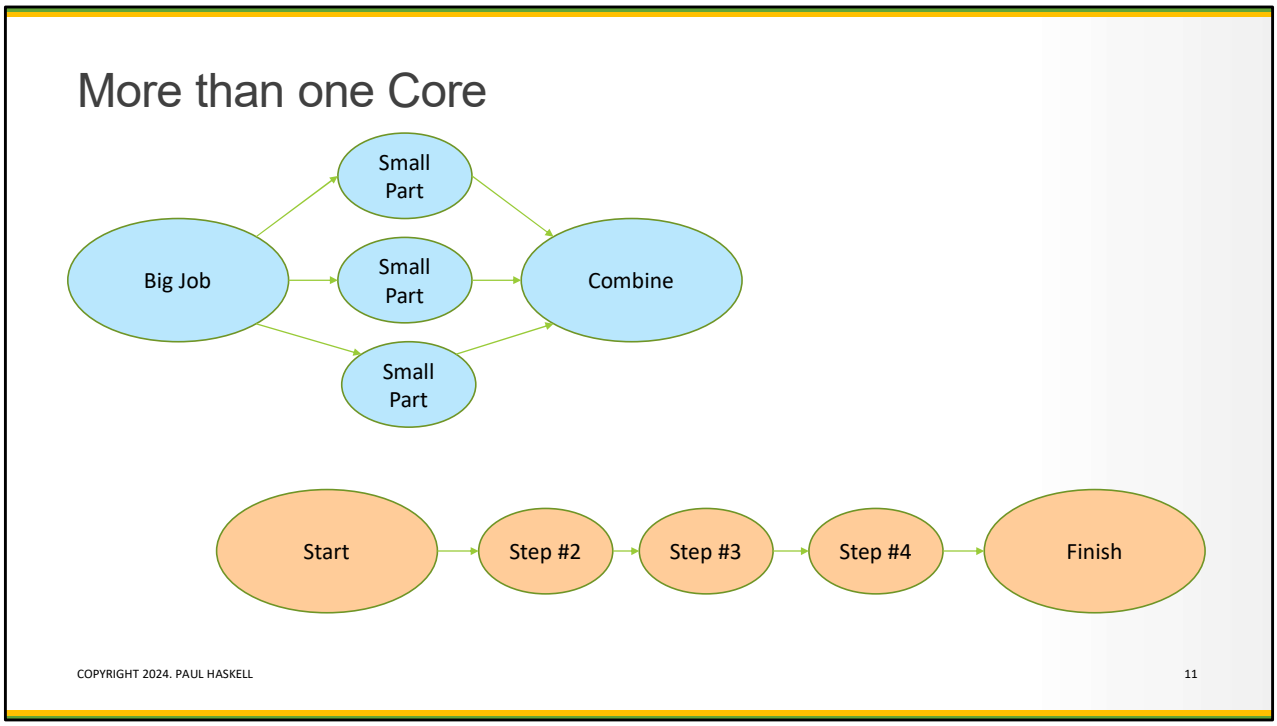
Multithread models

Why use multiple threads?

Want to use more than one CPU core at once, to complete big jobs faster

Want part of the program to be more responsive i.e. have lower latency

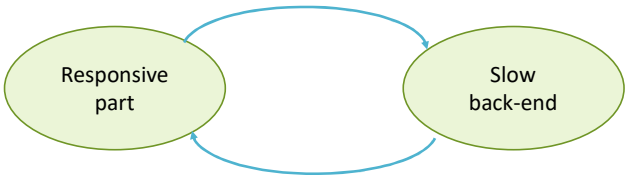
Example: audio mixing program: GUI gives instant responsiveness. Audio processing backend is busy and slow.



Parallelization and Pipelining.
Choose a model (or both) based on delay variability of the steps and requirements of the application.
All of these can synchronize via join():

- parallel: combine() waits for each SmallPart to finish. Done SmallParts can go onto the next job.
- Pipeline: item(N), item(N+1), item(N+2), …

Maximize Responsiveness



Often: GUI or TextUI
Each part feeds data/requests to other. Need some way to synchronize.
Topic of next lecture!

Threads vs Processes

Threads vs Processes

You might have heard me sneak the word "process" into our discussion.

An Operating System runs multiple **processes**

Each process can contain one or more **threads**

A computer **program** can consist of one or more processes

What's the difference?

Threads vs Processes

Each process contains unique:

- memory address space
- user id and permissions
- open file descriptors
- executable code
- environment variables
- resource limits ('ulimit')

It is much slower to create and join processes than threads.

Unless you need some of the above resources to be unique, use **threads**