COS40003 Concurrent Programming

Lecture 4a: Thread

Outline

- Why we need threads?
- What is a thread?
- How threads are working?
- Thread in Java

SWINBURNE UNIVERSITY OF TECHNOLOGY

Why we need threads?

Recall - why people came up with processes?

- Period 3: - Process is invented!
  - A program is capsuled in a process.
  - Every process is allocated a chunk of memory, and can only use its own memory. The state of the process and the resources the process is using can be saved.
  - When a process is switched back, easily restore the previous state and continue
  - Different processes will not affect each other.

  → ☺

Recall - why people came up with processes?

- Period 3: - Process is invented!
  - It seems operation system is multi-tasking. although, only one program is using CPU at one time.
  - It is “process” that made it!
  - People never get completely satisfied!

Why people came up with Threads?

- Scenario, a monitoring system (One Program!)
  - function 1: get data from recording end;
  - function 2: display data on screen;
  - function 3: interact with operator now and then;

  Suppose: while we are in function 1 (function 1 takes 5s), operator presses a button. Operator may need to wait for 5s to get response.

  - Not acceptable. People want more real-time! → ☹

Why people came up with Threads?

- Period 3: - Process is invented!
  - Questions: Can we separate the functions into subtasks?
  - Eg. while we are getting data from recording end, operator presses a button;
    - Pause the video transmission
    - Respond to the operator first (this usually takes very little time)
    - Get back to go on video transmission

Why people came up with Threads?

- Period 4: - Thread is invented!
  - Each thread runs a subtask.
  - A process runs multiple threads. Each thread runs an independent subtask.
  - when operator presses a button:
    - Pause video-transmission thread;
    - Run UI thread;
    - Resuming video-transmission thread.
  - Note that, the threads are working in the same working space. ☺

Why people came up with Threads?

- Why not processes in the previous example?

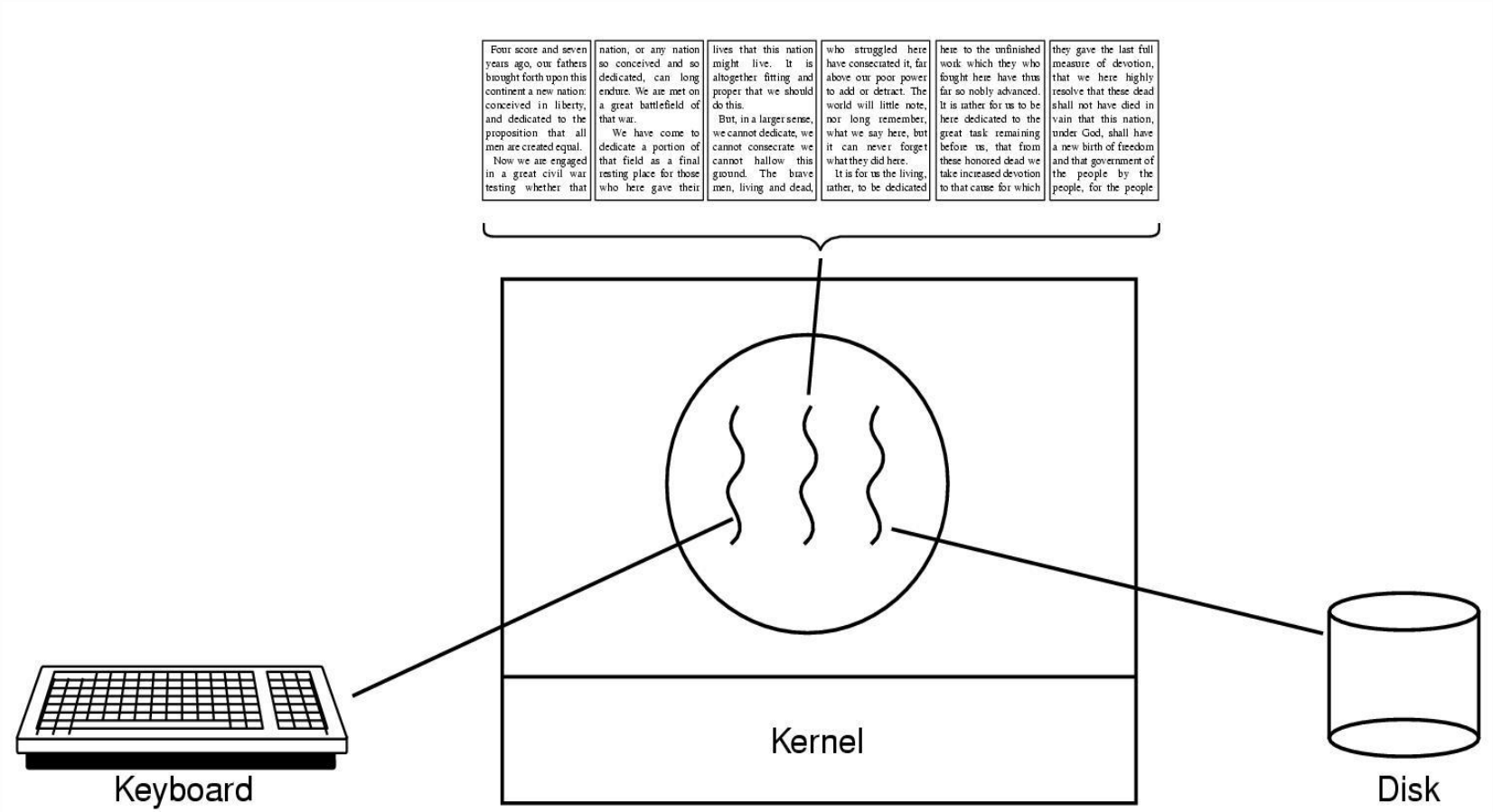
- Processes have different address spaces
  - Data cost
    - Data redundancy (multiple duplicate data copies)
    - Data consistency
  - Communication cost
    - Higher than threads

SWINBURNE UNIVERSITY OF TECHNOLOGY

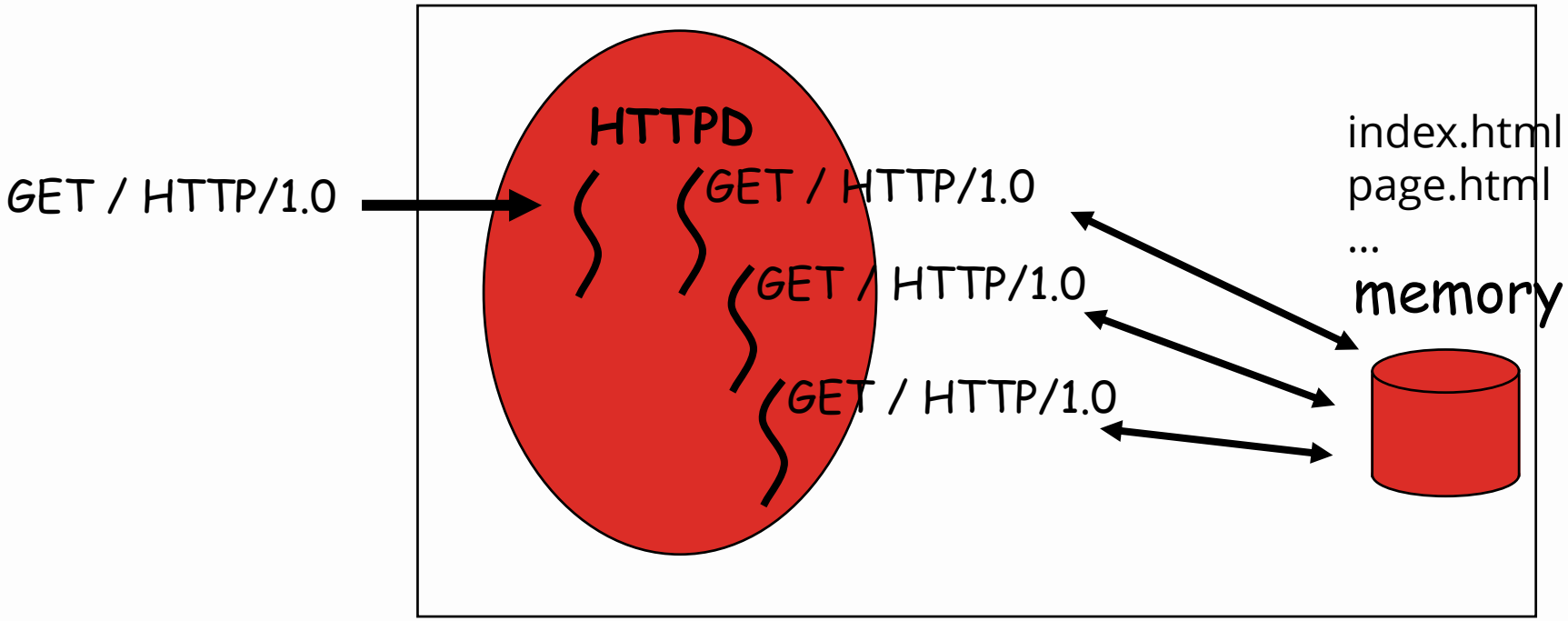
More thread application examples

Word Processor Program

A word processor with three threads



Simple web server



Australian Census 2016

Aussies face more frustration as Census servers continue to fail despite repeated attempts

FRUSTRATED Australians are still trying but are failing to fill in the Census online, just days after the ABS claimed to have fixed it.

Rod Chester

News Corp Australia Network AUGUST 16, 2016 2:41PM

Australian Statistician David Kalisch says the census server is up but many Australians would disagree. Picture: Kym Smith *Source:News Corp Australia*

FRUSTRATED Australians are still trying, and failing, to use the Census online days

Attacked by millions of Australian People??

http://www.news.com.au/technology/aussies-face-more-frustration-as-census-servers-continue-to-fail-despite-repeated-attempts/news-story/4e43f5e92dda5685778601b9b66fddf9

Outline

- Why we need threads?
- What is a thread?
- How threads are working?
- Thread in Java

SWINBURNE UNIVERSITY OF TECHNOLOGY

What is a thread?

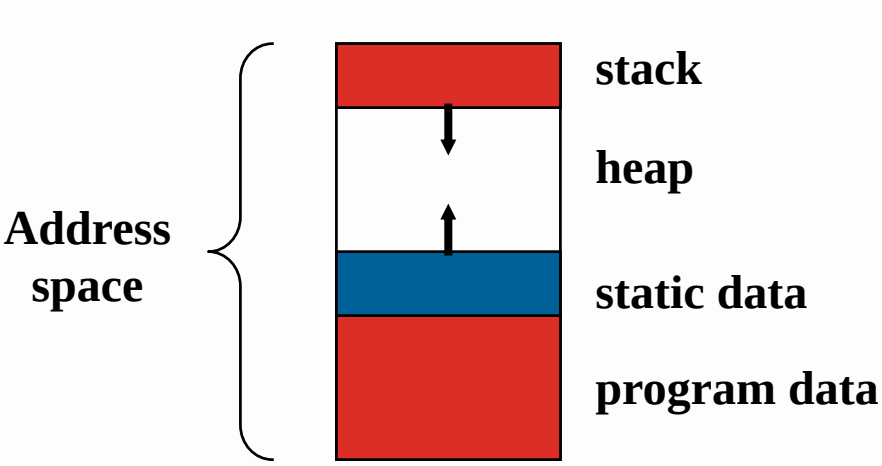
What is a thread?

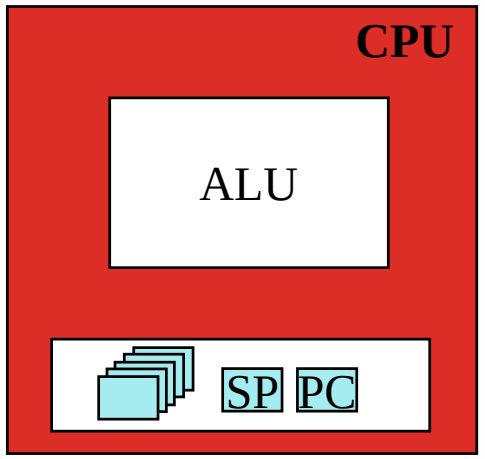
Simple but not very precise

- Process
  - A running program/program in action
- Thread
  - A running subtask of a program

Recall: processes

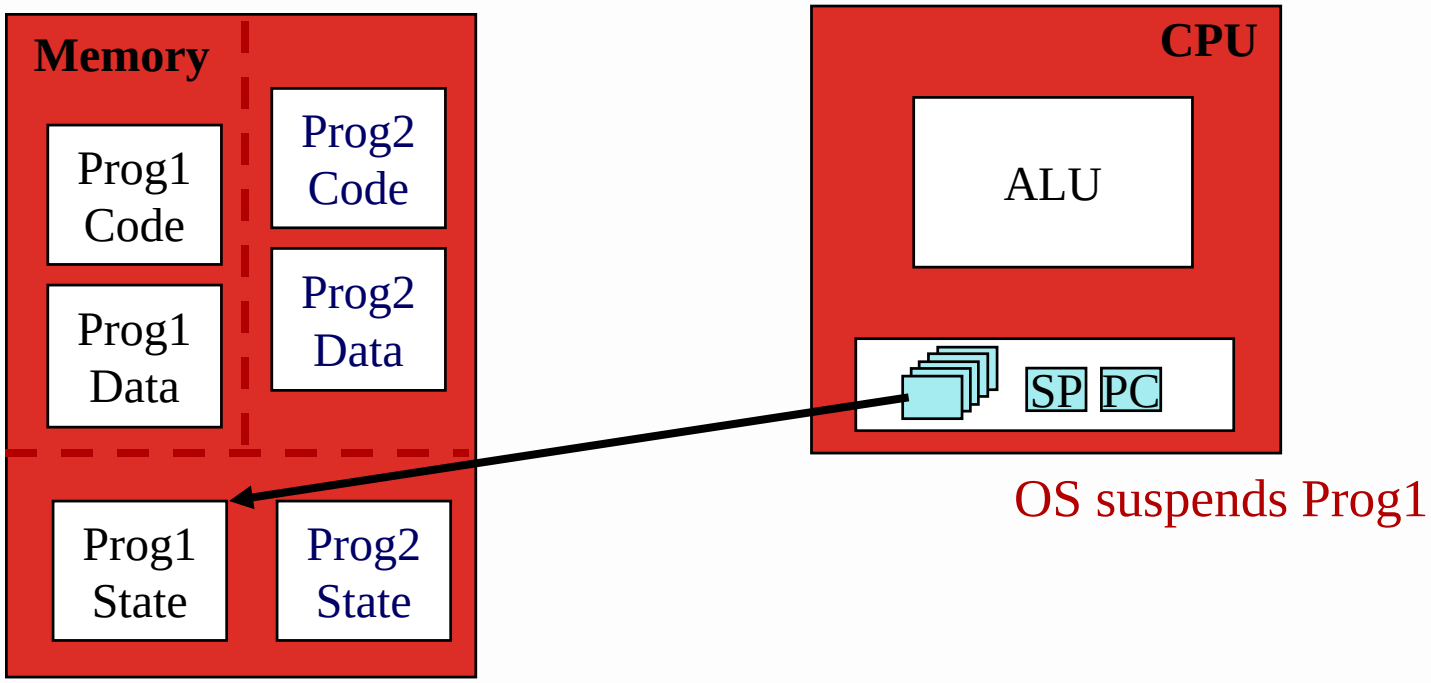
- Processes have the following components:
  - an address space
  - a collection of operating system states
  - a CPU context





Recall: process context switch

- Save registers, program counter, stack pointer of Prog1



Threads

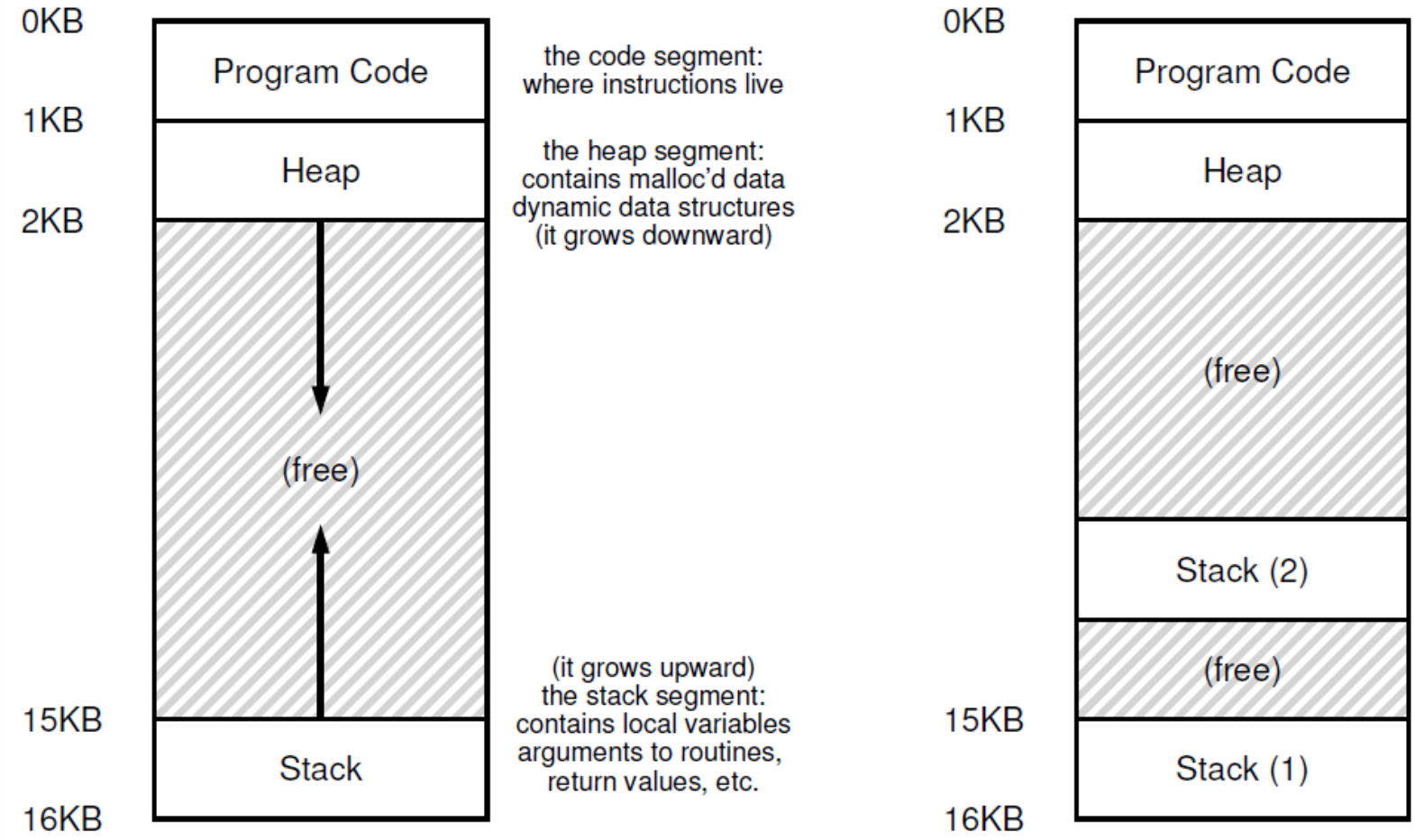
- In order to run multiple subtasks in a process, multiple CPU contexts need to be recorded

- Threads have their own CPU contexts
  - Program counter, Stack Pointer, Register states (CPU)
  - Stack (memory)

- Threads share one process address space with zero or more other threads

- A traditional process could be viewed as a memory address space with a single thread

Single-Threaded And Multi-Threaded Address Spaces



What Is a Thread?

- A thread executes a stream of instructions
  - It is an abstraction for control-flow

- Practically, it is a processor context and stack
  - Allocated to CPU by a scheduler
  - Executes in the memory address space of the holding process

Private Per-Thread State

- Things that define the state of a particular flow of control in an executing program
  - –Program counter
  - –Stack pointer
  - –Registers
  - –Stack (local variables)
  - –Scheduling properties (i.e., priority)

Shared State Among Threads

- Things that relate to an instance of an executing program
  - –User ID, group ID, process ID
  - –Address space: Program instructions, Static data (eg., global variables), Heap memory (dynamic data), stack memory
  - –Opened files, sockets, lock

Why use threads? (summary)

- Overlap computation and blocking on a single CPU
  - Blocking due to I/O
  - Computation and communication
- Utilize multiple CPU's concurrently
- Low cost communication via shared memory

Processes and Threads

- Some comparisons:
  - Process makes operating system multitasking
  - Thread makes process multitasking

  - Processes do not share resources in an operating system;
  - Threads of a process share resources and memory addresses;

  - Processes are the basic units of resource allocation of operation system;
  - Threads are the basic units of CPU scheduling of operation system;

Outline

- Why we need threads?
- What is a thread?
- How threads are working?
- Thread in Java

SWINBURNE UNIVERSITY OF TECHNOLOGY

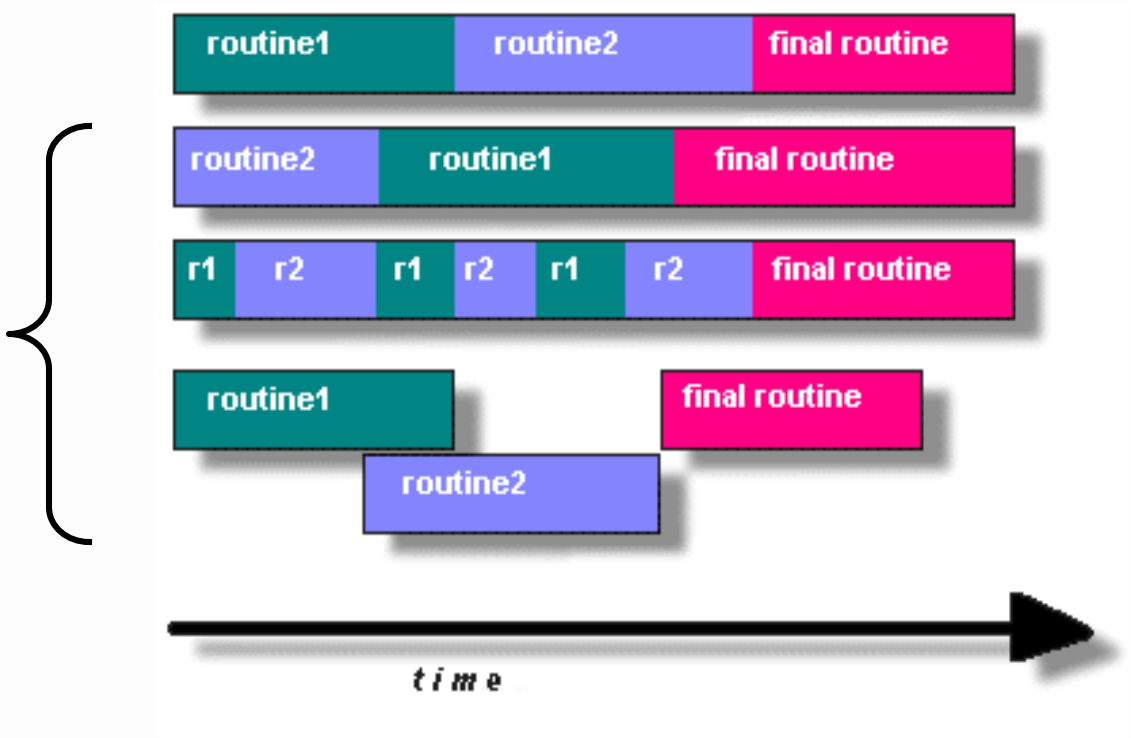
How threads are working?

Programming With Threads

Split program into routines to execute in parallel

- True or pseudo (interleaved) parallelism

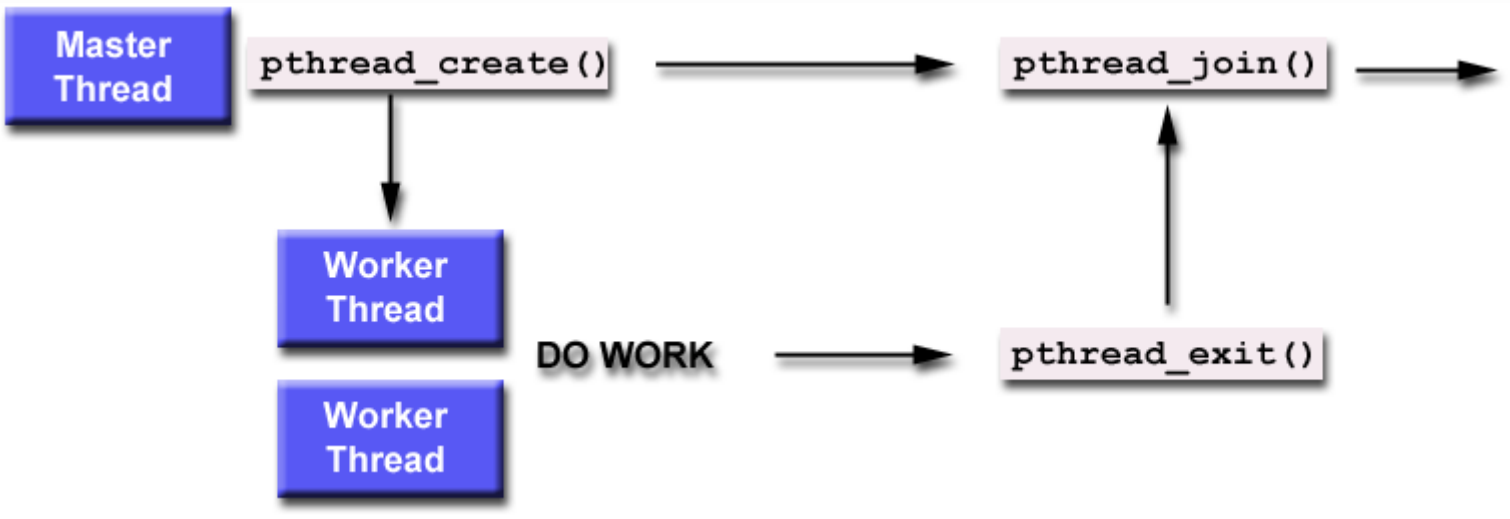
Alternative strategies for executing multiple rountines



Common Thread Strategies

- Manager/worker
  - Manager thread handles I/O
  - Manager assigns work to worker threads
  - Worker threads created dynamically
  - … or allocated from a *thread-pool*
- Pipeline
  - Each thread handles a different stage of an assembly line
  - Threads hand work off to each other in a *producer-consumer* relationship

Using Create, Join and Exit



Pthreads: A Typical Thread API

- Pthreads: POSIX standard threads
- First thread exists in main(), creates the others

- pthread_create (thread,attr,start_routine,arg)
  - Returns new thread ID in “thread”
  - Executes routine specified by “start_routine” with argument specified by “arg”
  - Exits on return from routine or when told explicitly

Pthreads (continued)

- pthread_exit (status)
  - Terminates the thread and returns “status” to any joining thread
- pthread_join (threadid,status)
  - Blocks the calling thread until thread specified by “threadid” terminates
  - Return status from pthread_exit is passed in “status”
  - One way of synchronizing between threads
- pthread_yield ()
  - Thread gives up the CPU and enters the run queue

Pthread Online Tutorial (C)

- https://hpc-tutorials.llnl.gov/posix/

- Full introduction
- Code available

Pros & Cons of Threads

- Pros:
  - Overlap I/O with computation!
  - Cheaper context switches
  - Better mapping to multiprocessors
- Cons:
  - Potential thread interactions
  - Complexity of debugging
  - Complexity of multi-threaded programming
  - Backwards compatibility with existing code

Acknowledgement

- Chapter 26
  - Operating Systems: Three Easy Pieces

- 3.ppt
  - Intro to Operating System at Portland State University
  - by Jonathan Walpole

SWINBURNE UNIVERSITY OF TECHNOLOGY

Questions?