Operating Systems!

Processes to Threads (part 1)

Prof. Travis Peters

Montana State University

CS 460 - Operating Systems

Fall 2020

https://www.cs.montana.edu/cs460

Some diagrams and notes used in this slide deck have been adapted from Sean Smith's OS courses @ Dartmouth. Thanks, Sean!

What's a thread?

Julia Evans
@b0rk

drawings.jvns.ca

a process can have lots of threads

process id	thread id
1888	1888
1888	1892
1888	1893
1888	2007

threads share memory

thread 1: I'm going to write "3" to address 2977886

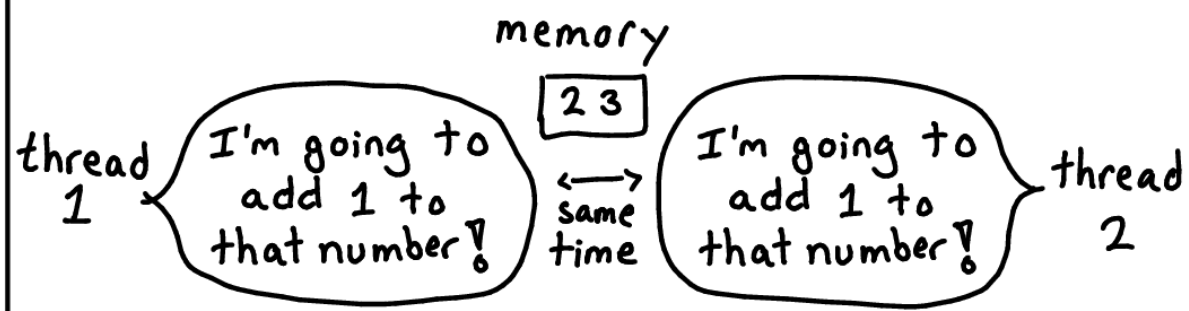
I can overwrite that if I want! — thread 2

but they can run different code

thread 1: I'm doing a calculation!

I'm waiting for a network request to finish! — thread 2

Sharing memory can cause problems
"race conditions"



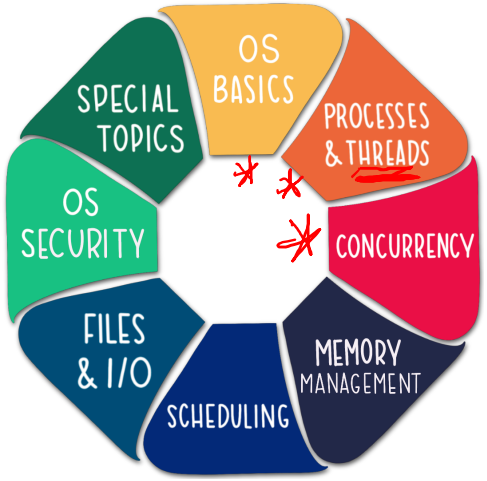
RESULT: 24 ← WRONG (should be 25)

if you have 8 CPU cores, you can run code for 8 threads at the SAME TIME

1 5
2 6
3 7
4 8
CPU cores

SO BUSY

Today



- Announcements
  - *PA1 Done! Nice Work!*
  - *Heads up…. Exam 1:* ***Friday [10/02/2020] – due @ 11:59 PM (MST)***
  - *Heads up…. PA2 due* ***Sunday [10/04/2020] @ 11:59 PM (MST)***

- Learning Objectives
  - Understand threads
    - *more theory (create, terminate, states, etc.)*
    - *more reality (POSIX Threads – "PThreads")*
    - *common issues when working with threads*
      - *race conditions*
      - *synchronization strategies*

Processes *(Recap)*

What is a process?

- An **address space** and an **execution context**

Why all this noise about processes?

- A useful abstraction – it is convenient to package "one task" as "one process"
- Efficiency of concurrency *(multiprogramming / multitasking / potential parallelism)*

1 Process (w/ 1 Thread)

Concurrency (Above the Surface)

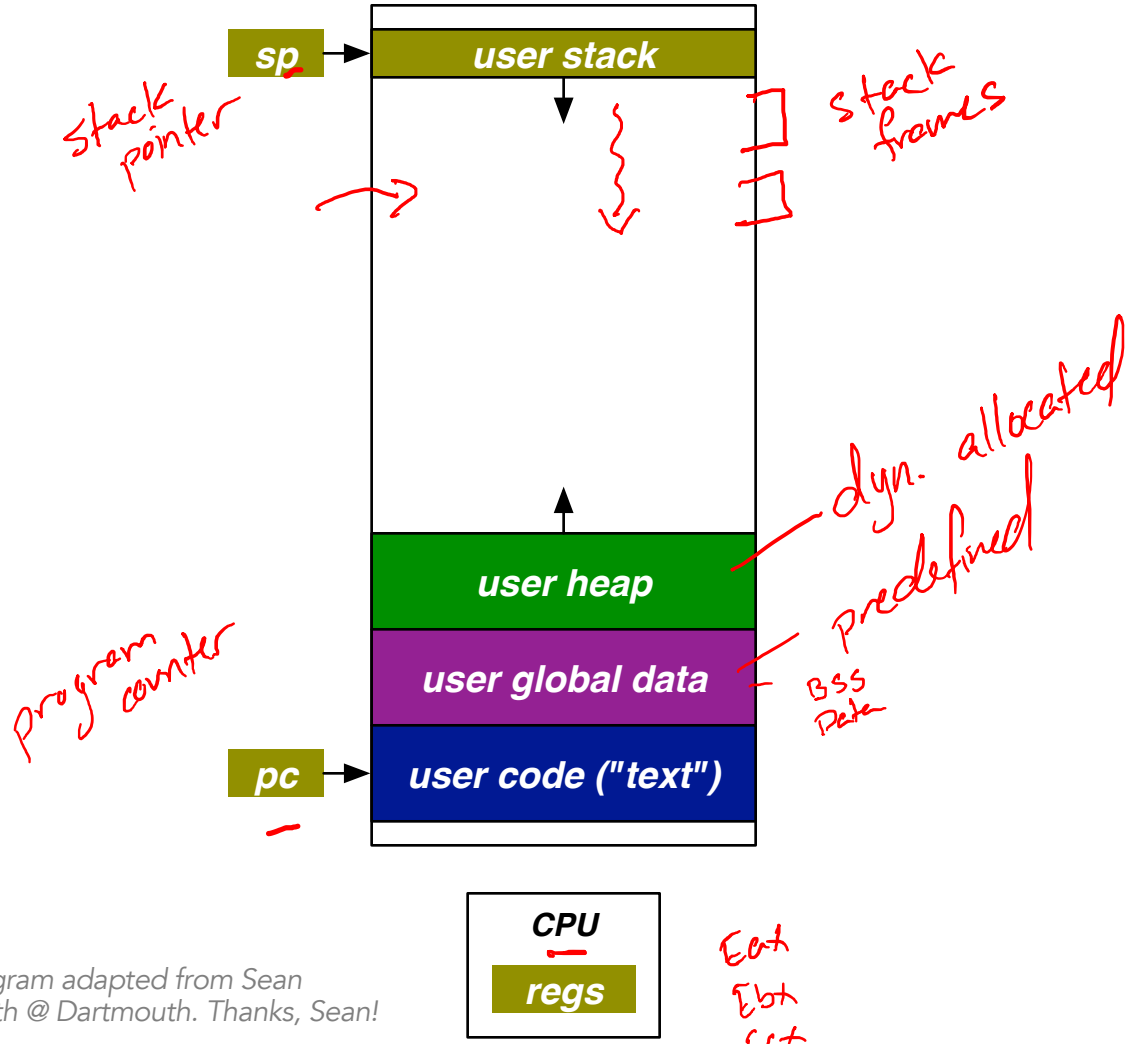


Diagram adapted from Sean Smith @ Dartmouth. Thanks, Sean!

see demo:

- ***sf.c***

Multiple, Concurrent Processes (Each w/ 1 Thread)

Concurrency (Above the Surface)

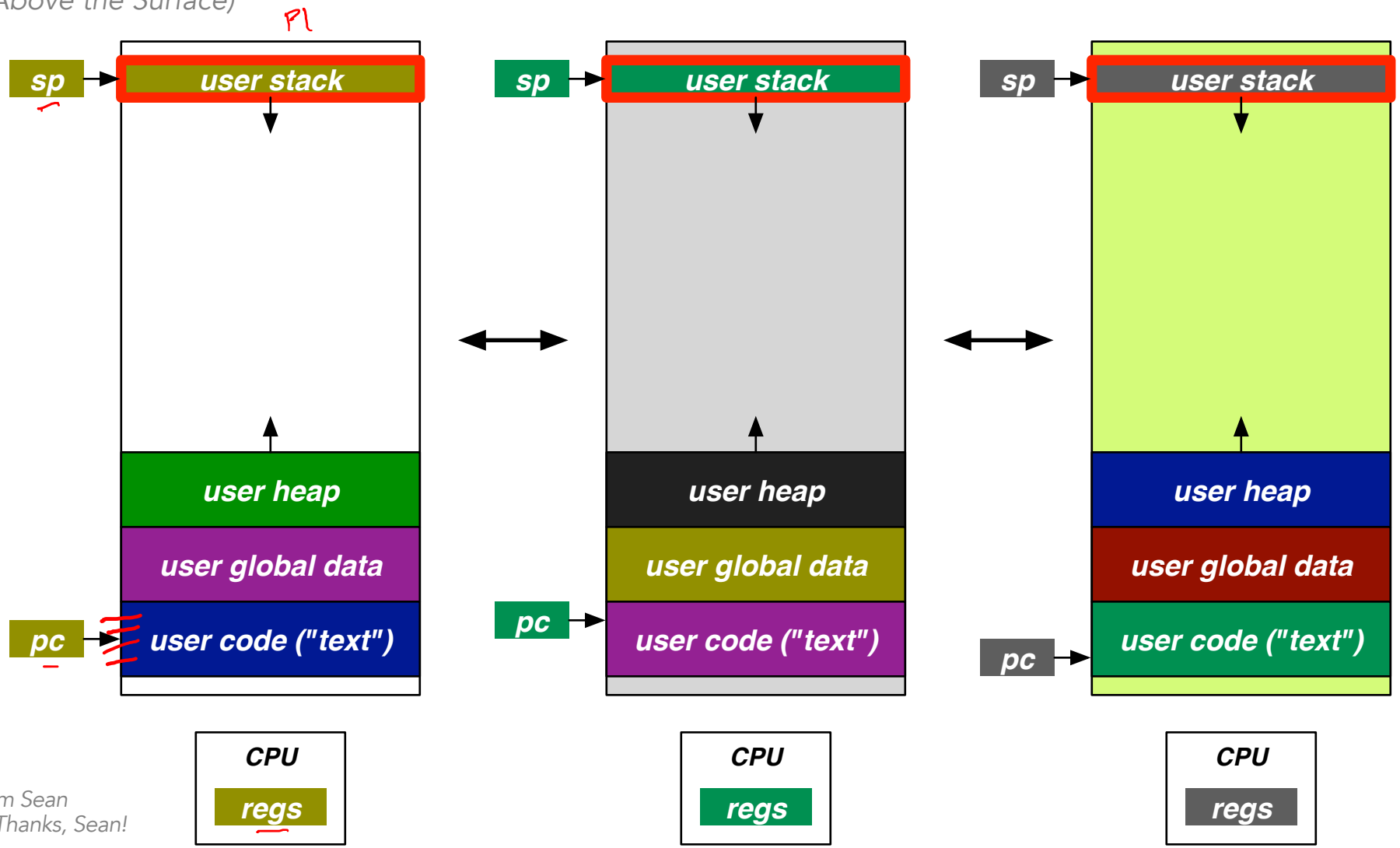


Diagram adapted from Sean Smith @ Dartmouth. Thanks, Sean!

see demo:

- ***sf.c***

"Humans are actually quite good at doing two or three things at a time, and seem to get offended if their computer cannot do as much." (Andrew Birrell)

Processes vs. Threads

- What is the difference between a **process** and a **thread**?

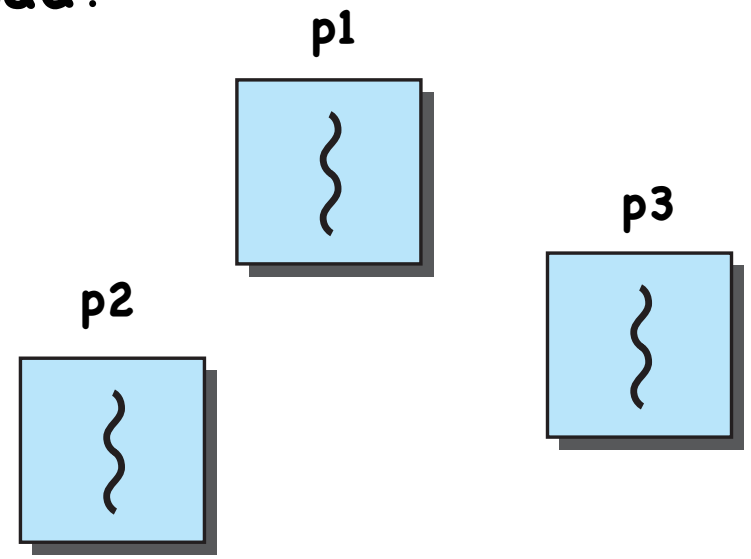
resources

runs on a separate processor

↳ mult. threads PER process

Processes vs. Threads

- What is the difference between a **process** and a **thread**?



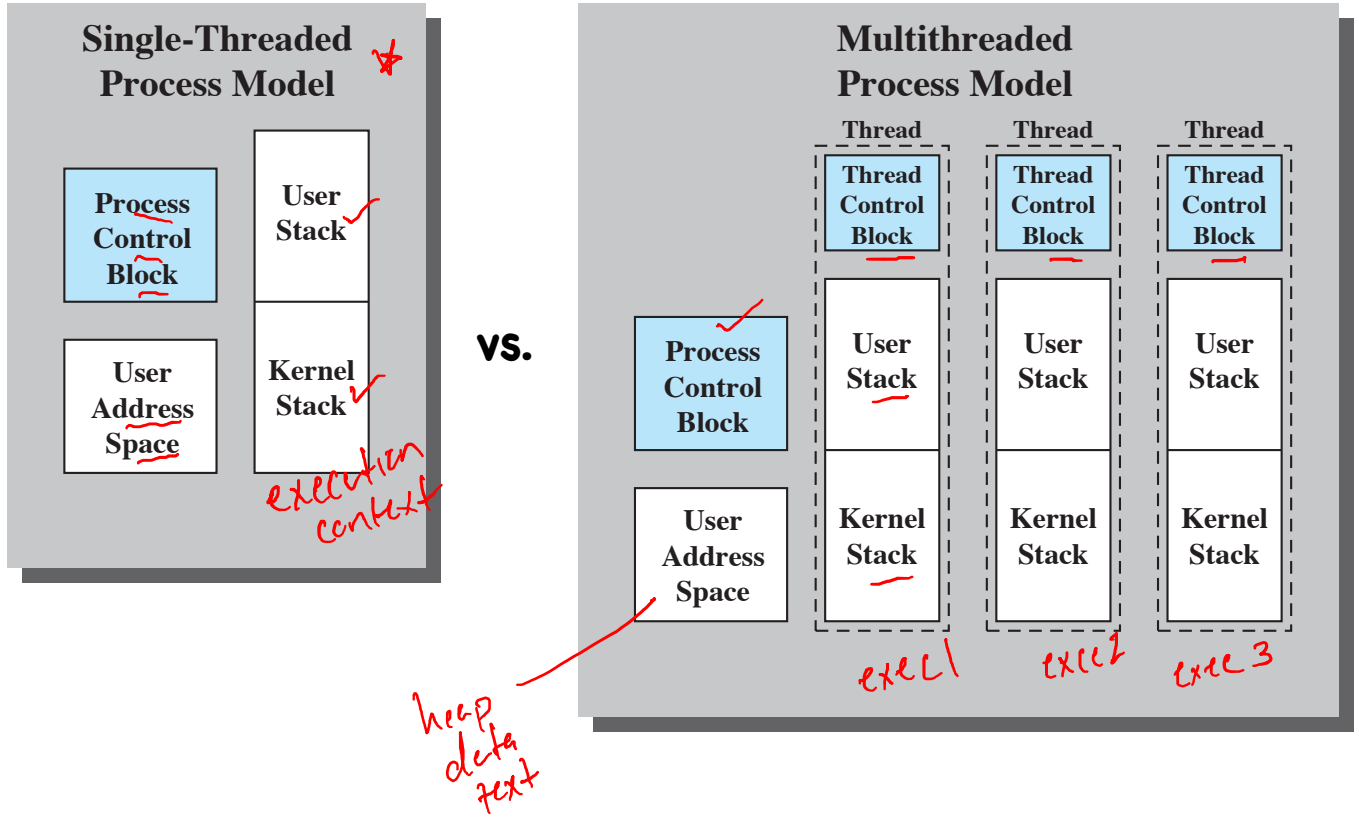
Processes vs. Threads

- What is the difference between a **process** and a **thread**?

- Divide process responsibilities:

  1) **resource ownership**
     (a *"process"* or *"task"*)

  2) **scheduling/execution**
     (a *"thread"* or *"lightweight process"*)



Processes vs. Threads

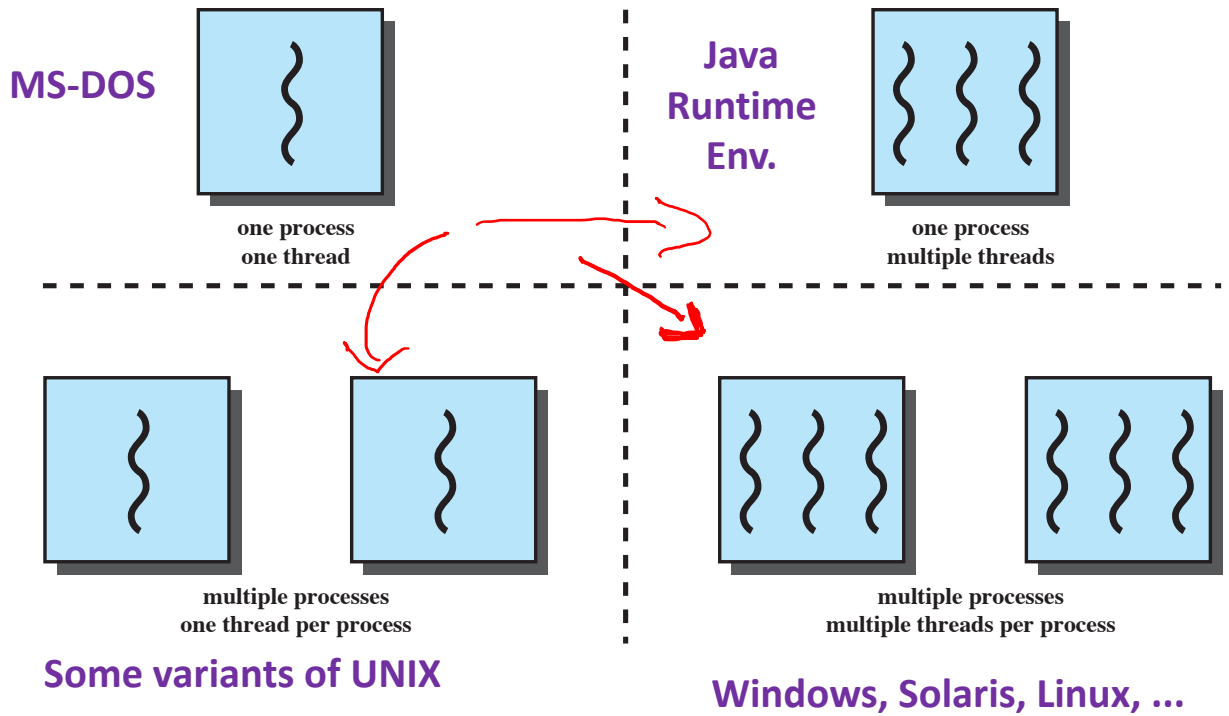
- What is the difference between a **process** and a **thread**?

- Divide process responsibilities:

  1) **resource ownership**
     (a *"process"* or *"task"*)

  2) **scheduling/execution**
     (a *"thread"* or *"lightweight process"*)

- **Multithreading** =
  multiple, concurrent paths of execution within a single process

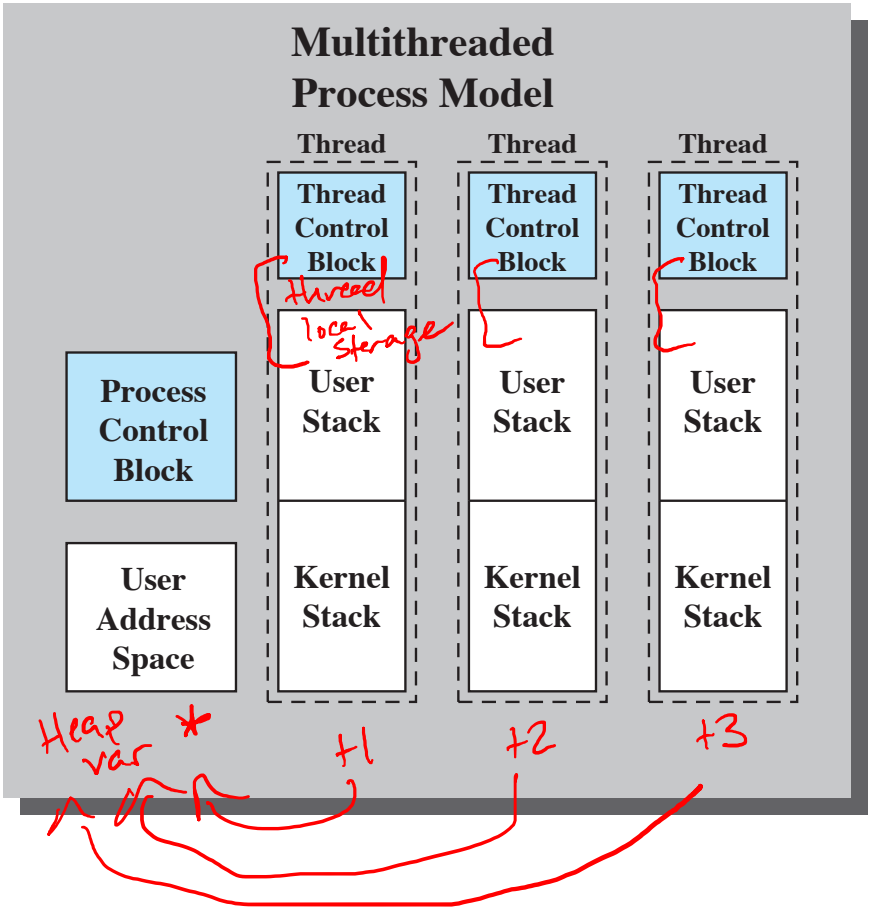


Processes vs. Threads ***(Going Deeper!)***

- Both provide independent execution sequences, but...

- Each **process** has their own...
  - **private memory space**
  - **resources** *(e.g., protected access to processors, other processes (IPC), files, and I/O devices and channels)*

- **Threads**...
  - run within a **shared memory space** *(i.e., within the process)*
  - have their own...
    - **execution state** *(Running, Ready, Blocked, etc.),*
    - **execution context** *(PC, general purpose registers, etc.),*
    - **execution stack** *(think "function calls" / "call stack"), and*
    - **thread local storage** *(think "global variables"... but only for that specific thread)*



1 Process (w/ 1 Thread)

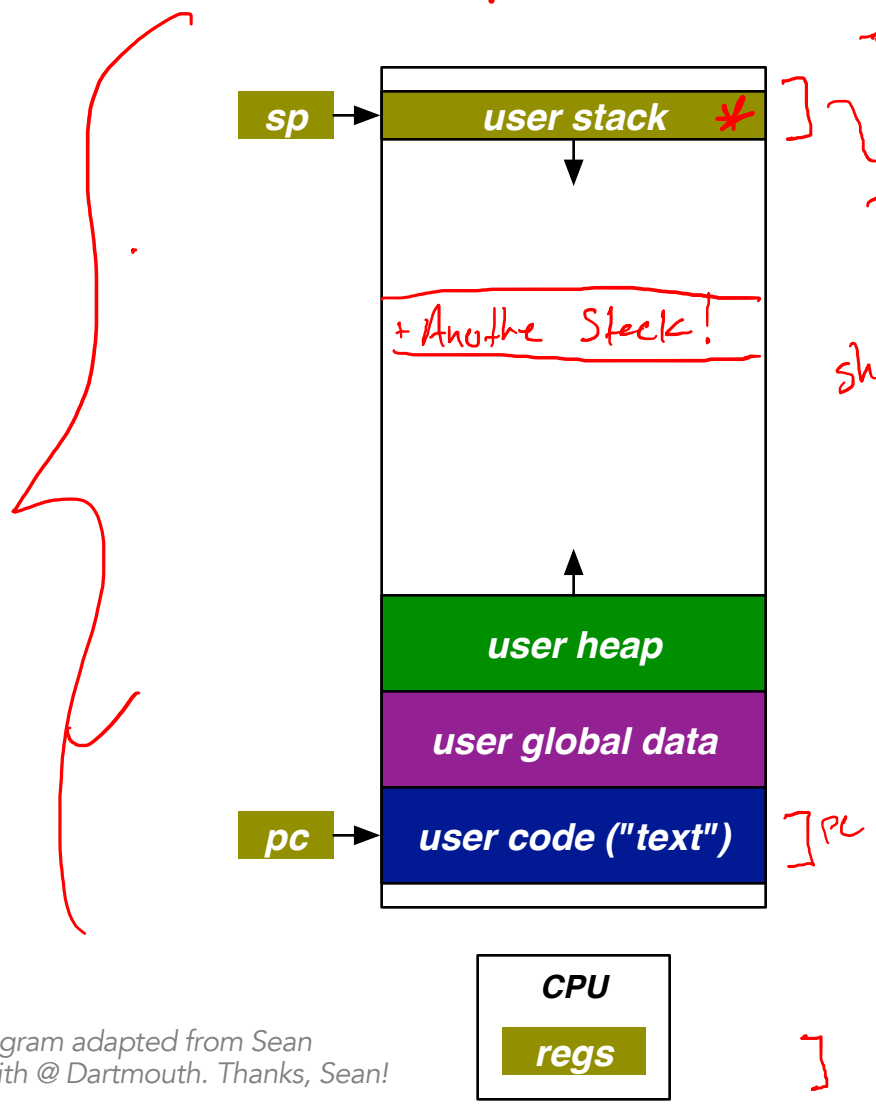
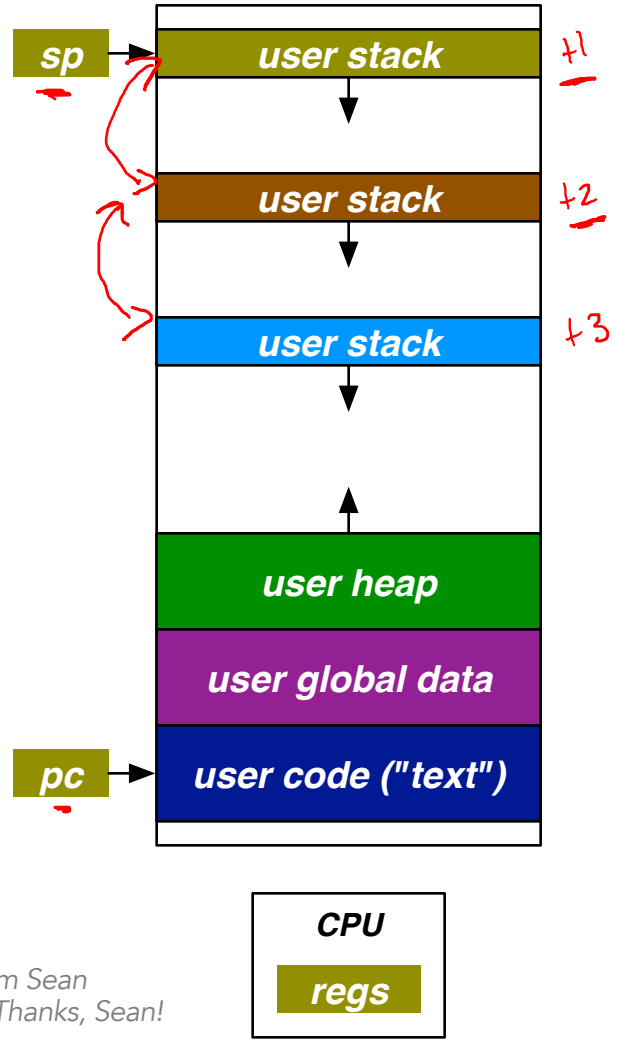


Diagram adapted from Sean Smith @ Dartmouth. Thanks, Sean!

What do you think we need to do here to realize new threads?

1 Process (w/ Many Threads!)



What do you think we need to do here to realize new threads?

A separate stack for each new thread!

+ a few other things, like IDs and registers

Diagram adapted from Sean Smith @ Dartmouth. Thanks, Sean!

Many Processes (w/ Many Thread!)

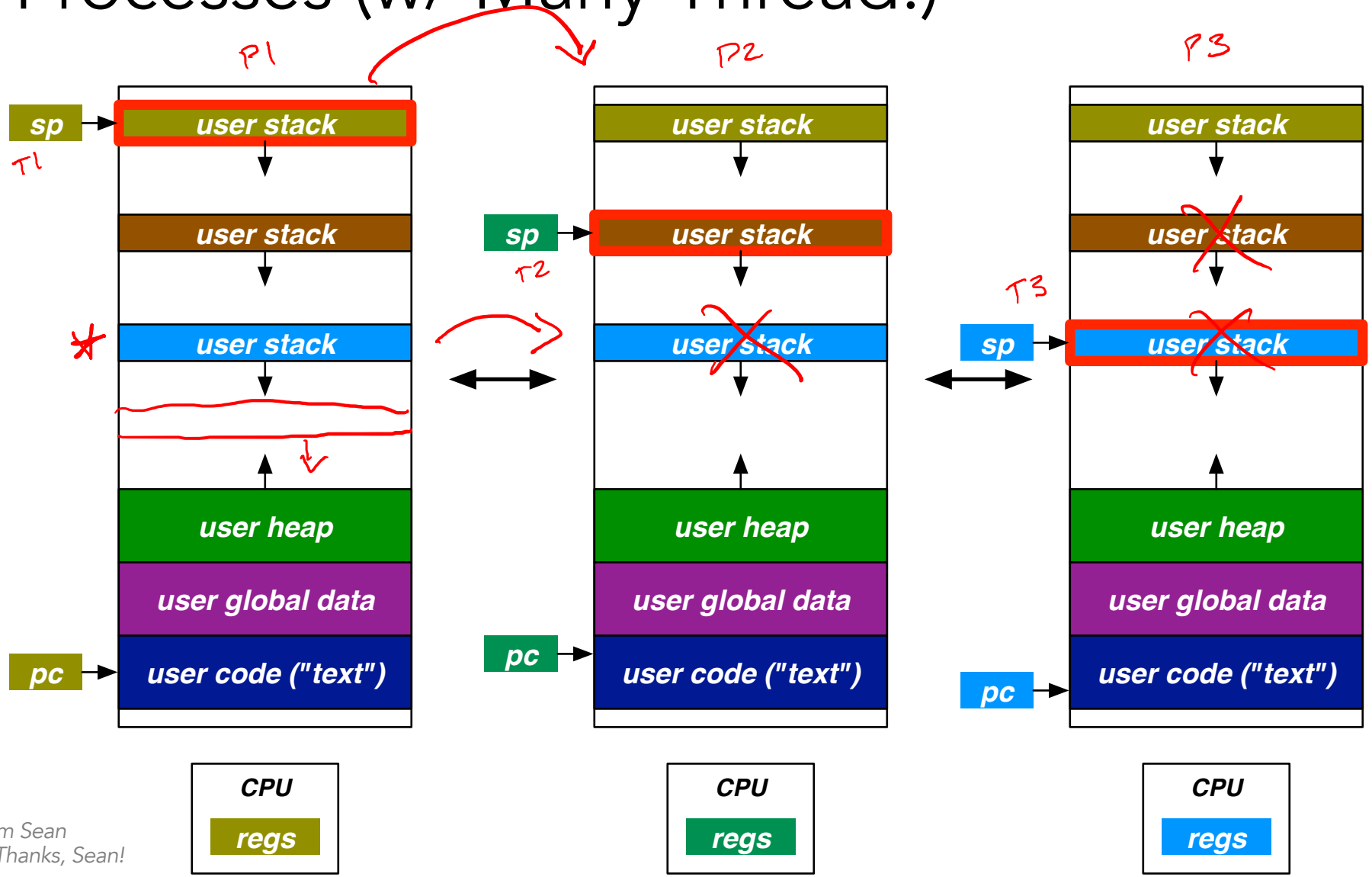


Diagram adapted from Sean Smith @ Dartmouth. Thanks, Sean!

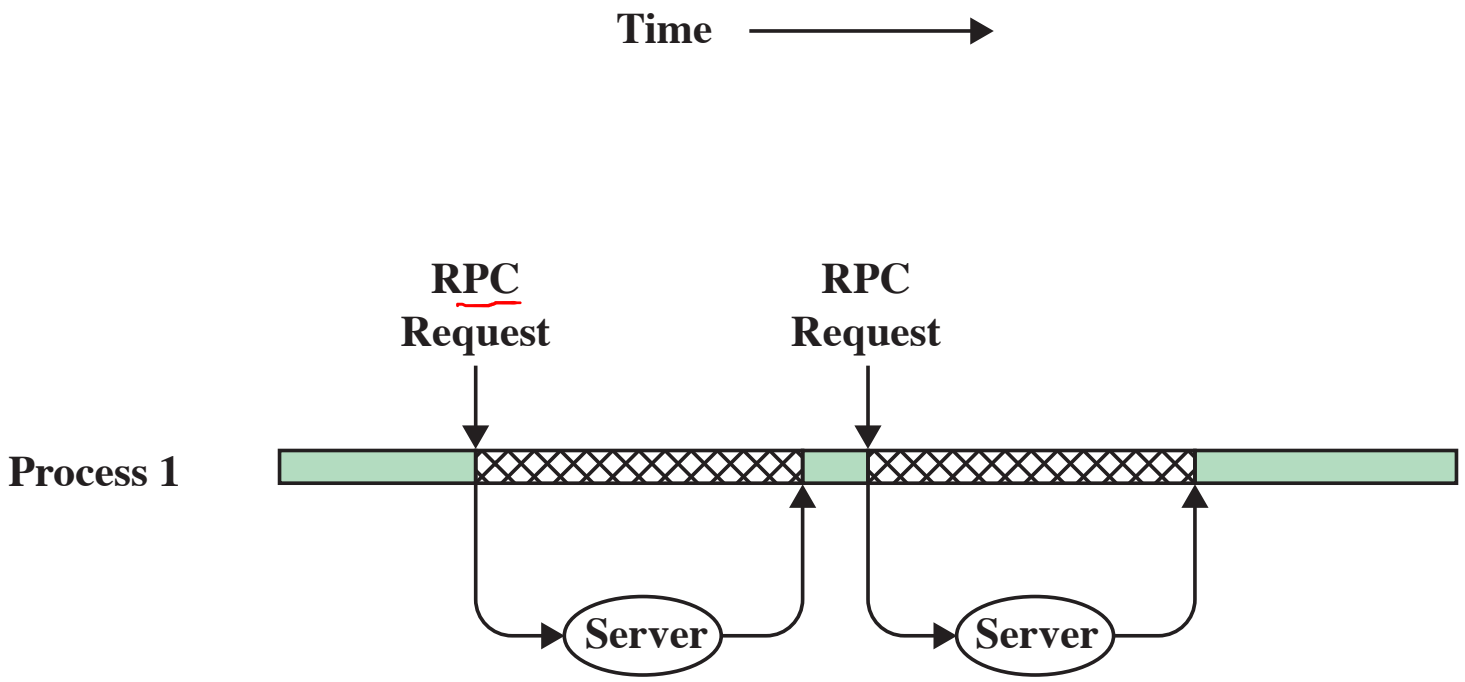
Why Use Threads?

- Much faster to **create/terminate threads**
  How much faster? Great question! Be patient....
- Much faster to **switch between threads** (within the same process)
  How much faster? Great question! Be patient....
- Must faster to **communicate between threads** (as compared to processes)
  How much faster? Great question! Be patient....
- They can prevent a process from **blocking** entirely!
  - IPC requires intervention from the kernel.
  - Using shared memory and files can be done without help from the kernel.*

**Requires more work on behalf of the process / software developer to do this correctly though!*

Why Use Threads?

Example: RPC using a (single-threaded) process vs. multi-threaded process



Why Use Threads?

Example: RPC using a (single-threaded) process vs. multi-threaded process