Unix intro / Processes / Bash / Java

THE UNIVERSITY
OF ARIZONA

Unix

Operating system

- Program that controls all other parts of a computer system, both hardware and software.
- It allocates computer resources and schedules tasks
- It allows us to make use of the facilities provided by the system
- Every computer requires an operating system

- Networking capabilities
- More secure than windows

Unix

Supports multiple users at once, running multiple tasks simultaneously

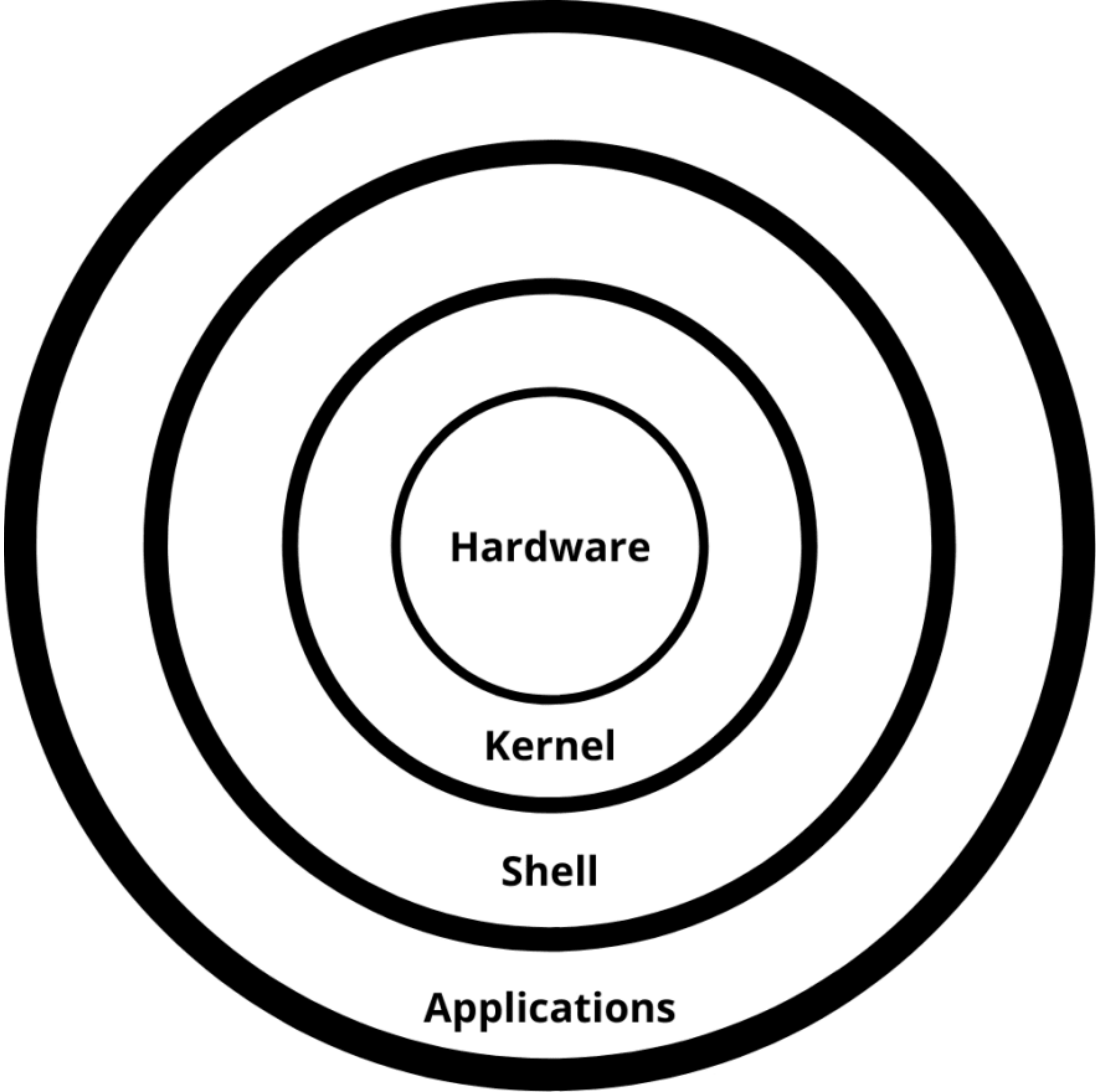
Unix is a machine-independent operating system

- Not specific to a single computer hardware
- Designed from the beginning to be independent of the computer hardware.

Unix is a software development environment.

- Was born in and designed to function within this type of environment

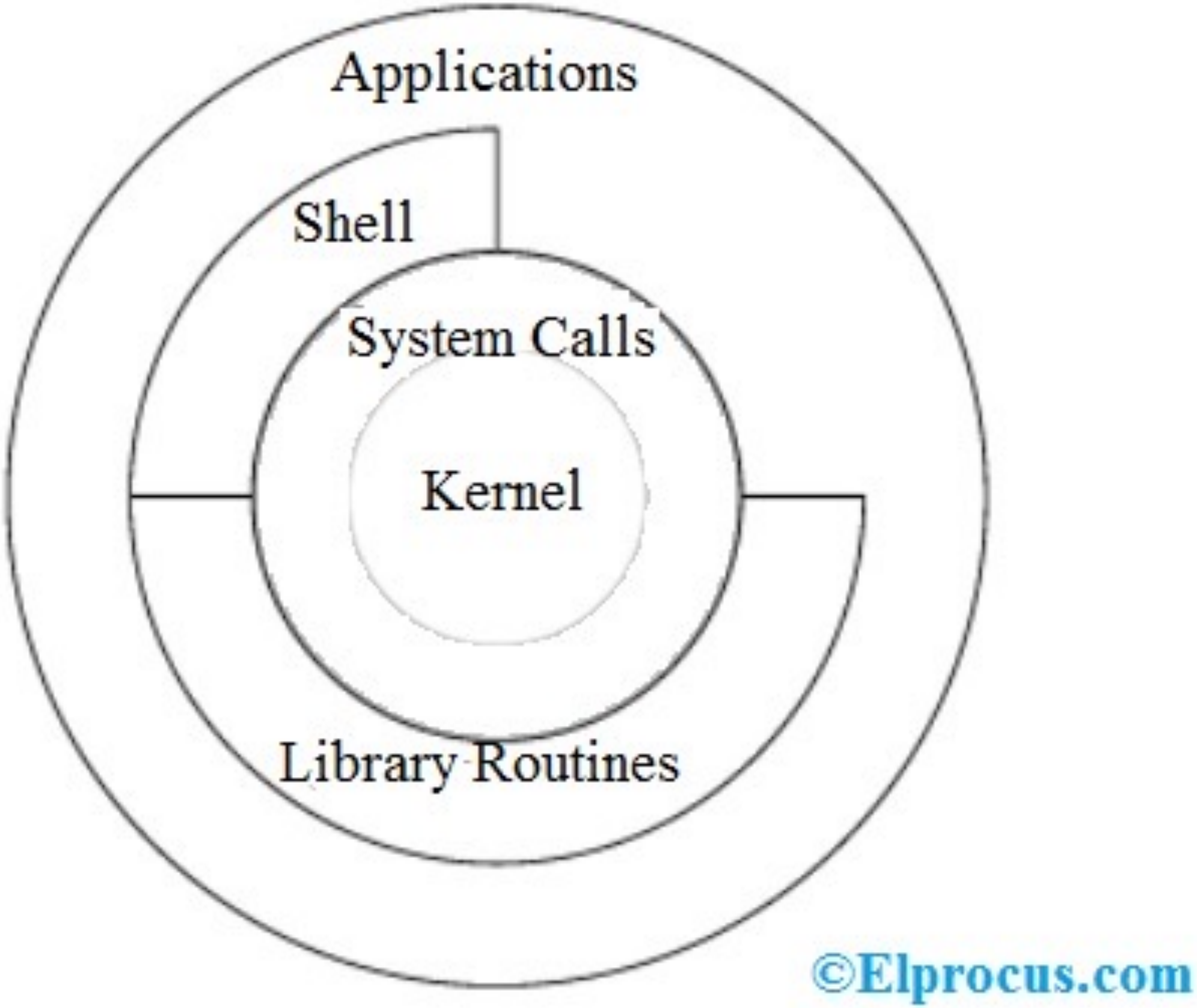
Unix

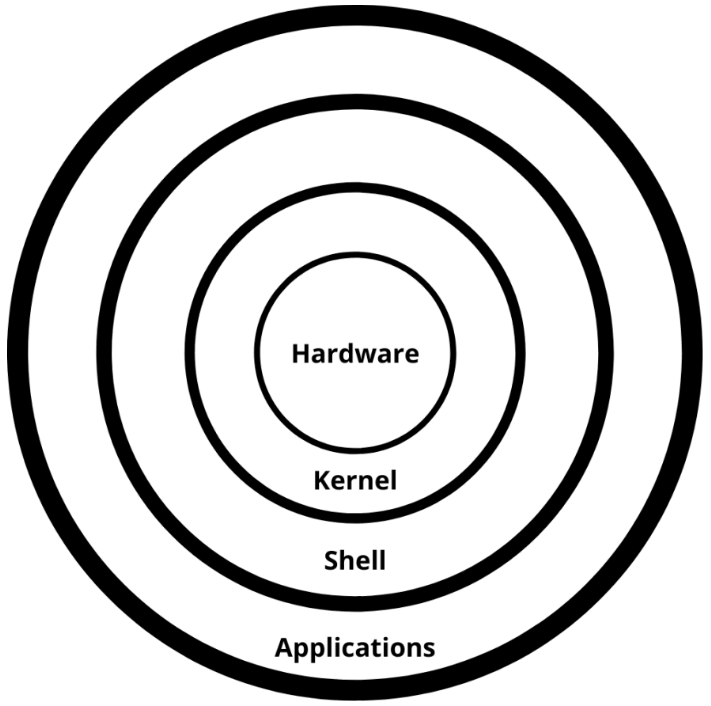


THE UNIVERSITY OF ARIZONA

https://dev.to/

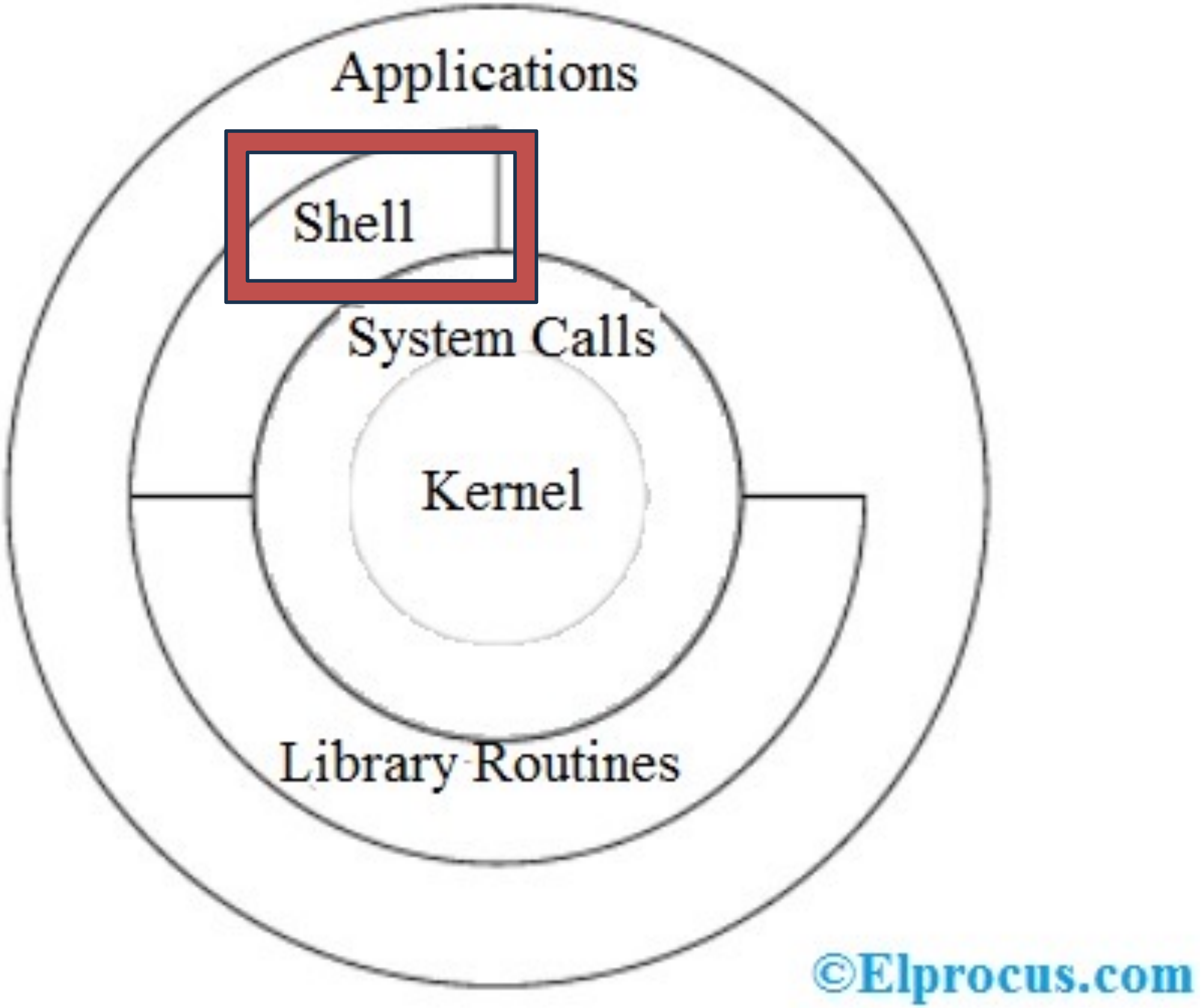
Unix





https://www.elprocus.com/

Unix



https://www.elprocus.com/

Kernel

Hub for the operating system

Allocates time and memory to programs and handles file storage and communication in response to calls.

Shell

THE UNIVERSITY OF ARIZONA

Shell

The shell is a command interpreter.

It is the layer between the operating system kernel and the user.

- Many command to manage operating system functions

Shell script

The first line must be “#!/bin/bash”.

–setup the shell path

chmod u+x scriptname (gives only the script owner execute permission)

./scripname.sh

Shell + Invoking the script

>: Redirect stdout to a file,

Creates the file if not present, otherwise overwrites it

< : Accept input from a file.

>>: Creates the file if not present, otherwise appends to it.

<<:

Forces the input to a command to be the shell’s input, which until there is a line that contains only *label*.

cat >> mshfile << .

|:pipe, similar to ">",

Shell + if

```
if [ condition ] then
   command1
elif   # Same as else if
   then
      command1
   else
   default-command
fi
```

Shell + loop

```
for [arg] in [list];
  do
  command
  done
while [condition];
do
   command...
done
```

Shell + Variables

`$: variable substitution`

If **variable1** is the name of a variable, then **$variable1** is a reference to its value.

Processes

Processes

A **program** is passive; a **process** is active

We can define a **program** as a group of instructions

We can define a **process** as executing a program

- Including current values of program counter, registers, and variables

Attributes held by a process include

- hardware state,
- memory,
- CPU,
- progress (executing)

Why do we have processes?

- Resource sharing (logical (files) and physical(hardware))

- Computation speedup
  - paralelism
  - taking advantage of multi-task programming
    - – i.e. example of a customer/server database system

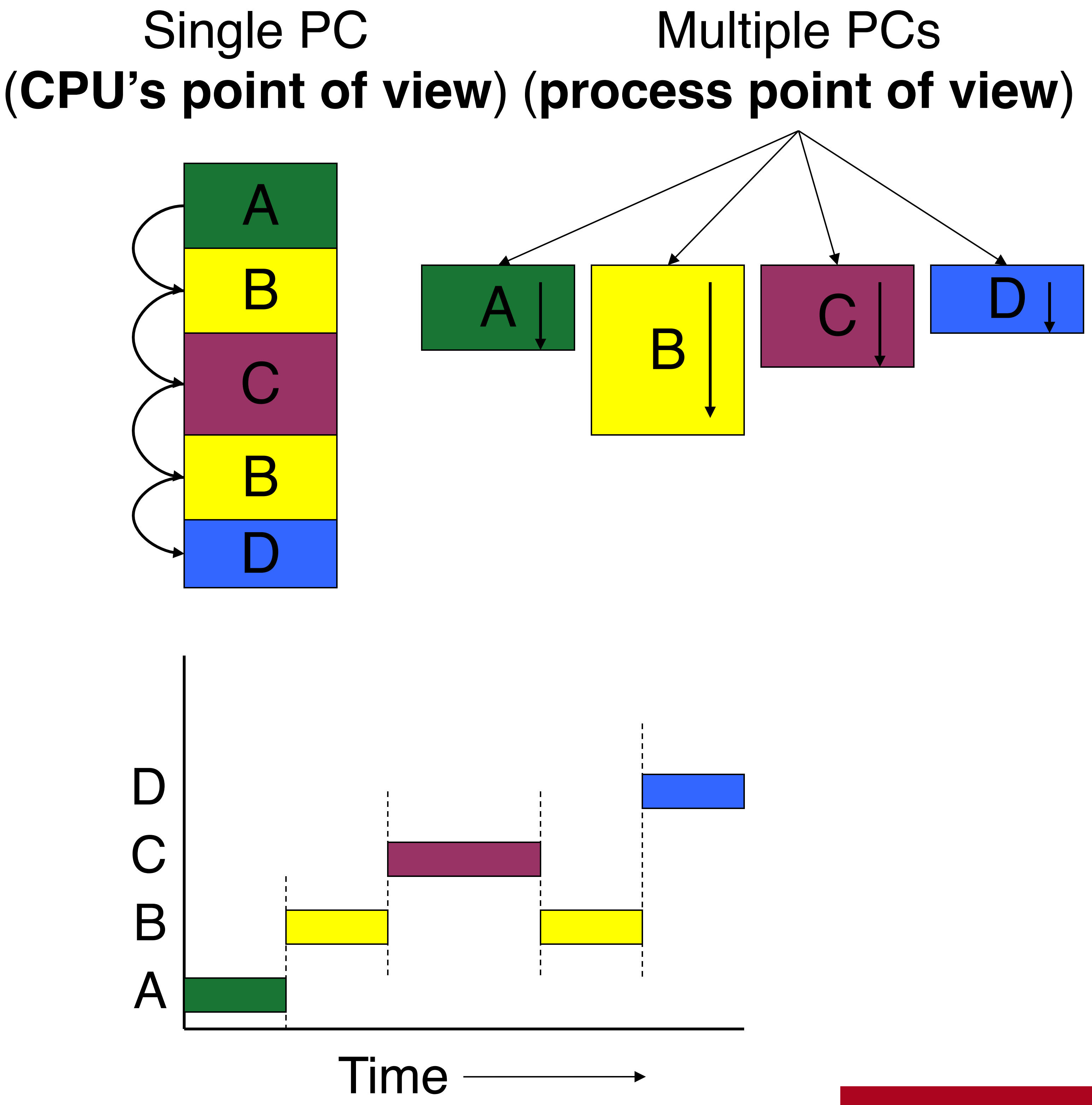
- Modularity for protection

An operating system executes a variety of programs

- Batch system - jobs
- Time-shared systems - user programs or tasks

*cs.pitt.edu

Process model



Multiprogramming of four programs

Conceptual model

- 4 independent processes
- Processes run sequentially

Only one program is active at any instant!

- That instant can be very short…

*cs.pitt.edu

When is a process created?

Processes can be created in two ways

- System initialization: one or more processes created when the OS starts up
- Execution of a process creation system call: something explicitly asks for a new process

System calls can come from

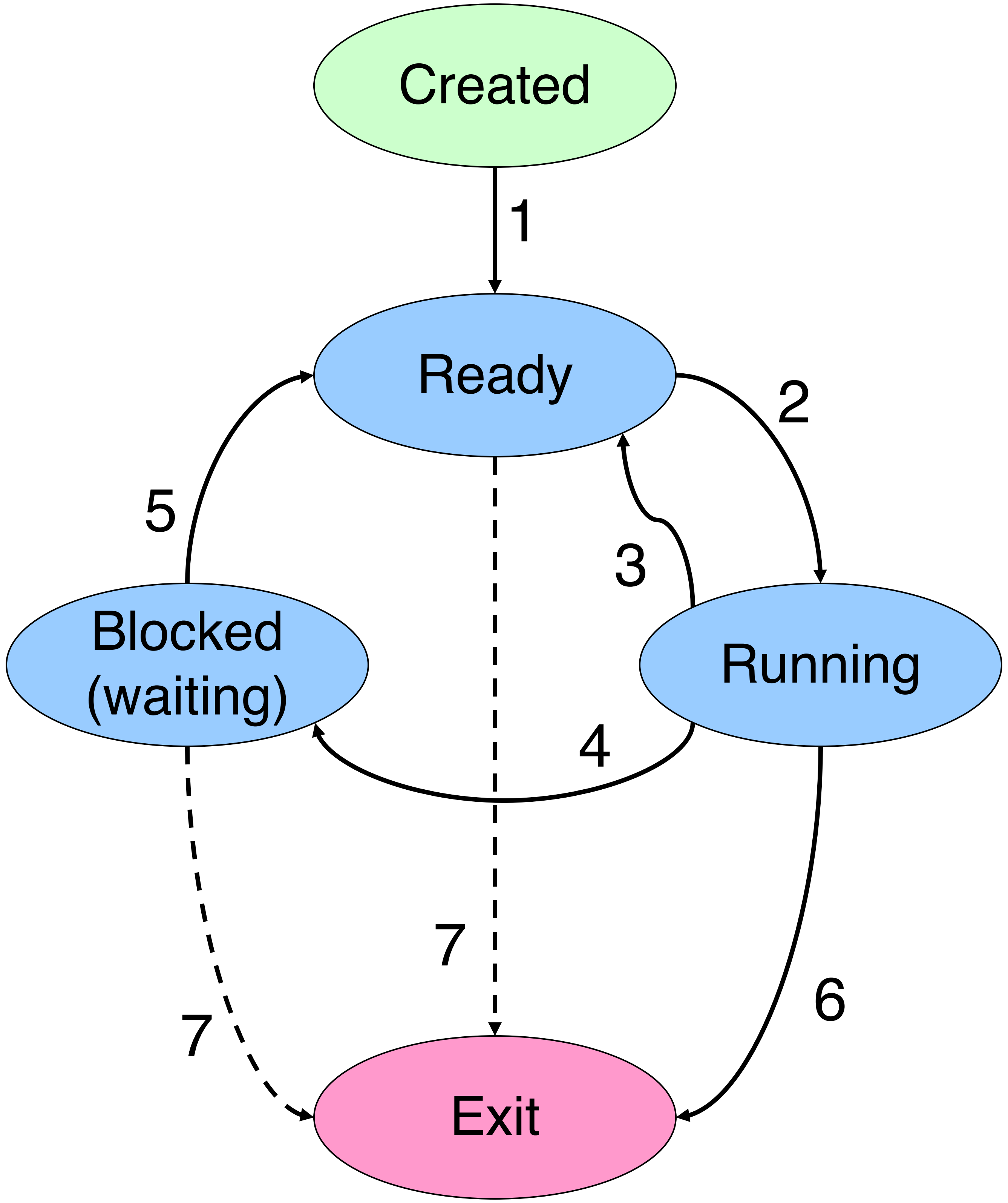
- User request to create a new process (system call executed from user shell)
- Already running processes
    - User programs
    - System daemons

Process Creation

Parent process creates children processes, which, in turn create other processes, forming a tree of processes

Generally, process identified and managed via a process identifier (**pid**)

Process States



Process in one of 5 states

- Created
- Ready
- Running
- Blocked
- Exit

Transitions between states

- 1 - Process enters ready queue
- 2 - Scheduler picks this process
- 3 - Scheduler picks a different process
- 4 - Process waits for event (such as I/O)
- 5 - Event occurs
- 6 - Process exits
- 7 - Process ended by another process

THE UNIVERSITY OF ARIZONA

*cs.pitt.edu

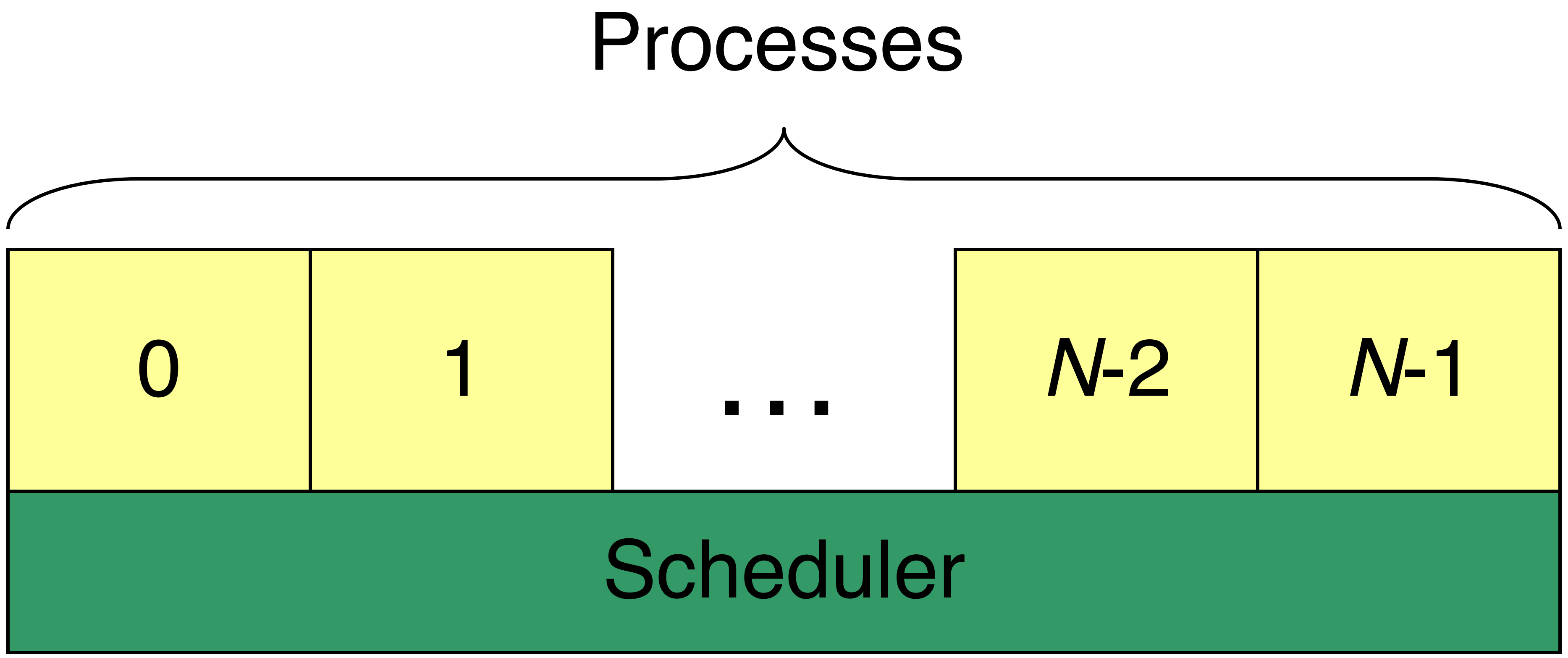
Processes in the OS

Two “layers” for processes

The lowest layer of process-structured OS handles interrupts, scheduling

Above that layer are sequential processes

- Processes tracked in the *process table*
- Each process has a *process table entry*



*cs.pitt.edu

What’s in a process table entry?

May be stored on stack

Process management

- Registers
- Program counter
- CPU status word
- Stack pointer
- Process state
- Priority / scheduling parameters
- Process ID
- Parent process ID
- Signals
- Process start time
- Total CPU usage

File management

- Root directory
- Working (current) directory
- File descriptors
- User ID
- Group ID

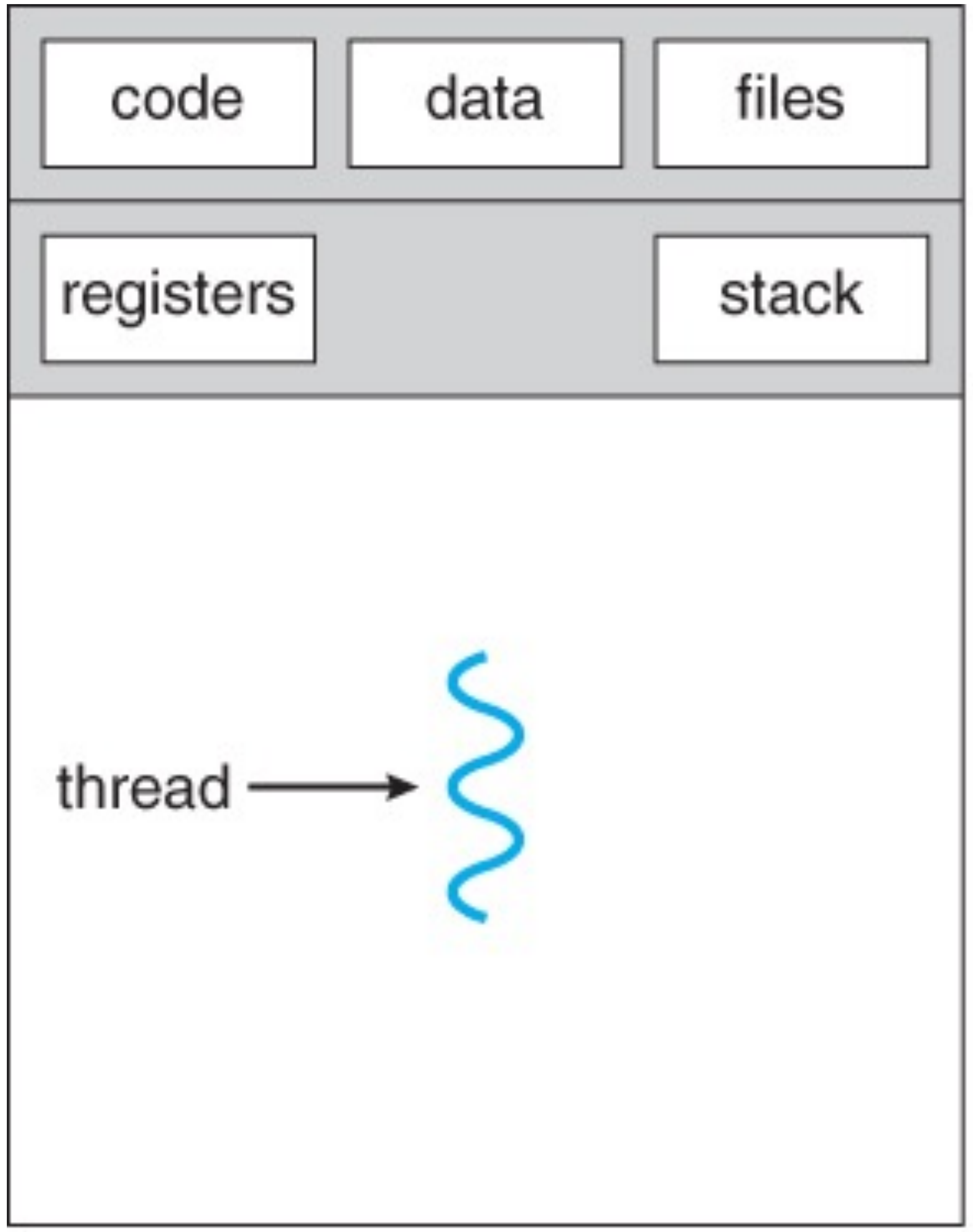
Memory management

Pointers to text, data, stack

or

Pointer to page table

A process has a thread



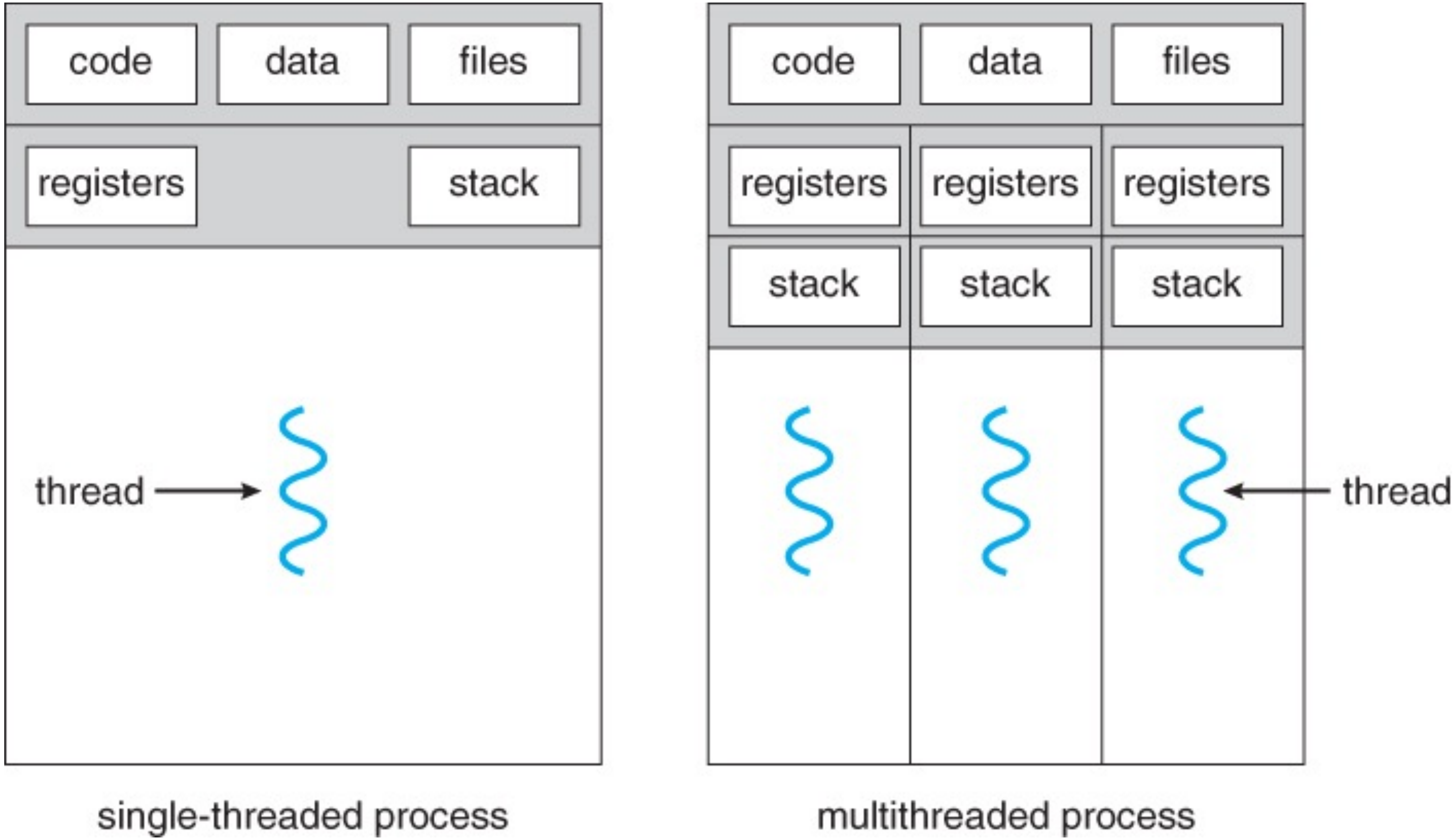
single-threaded process

*cs.uic.edu

Threads

THE UNIVERSITY OF ARIZONA

A process may have more threads



*cs.uic.edu

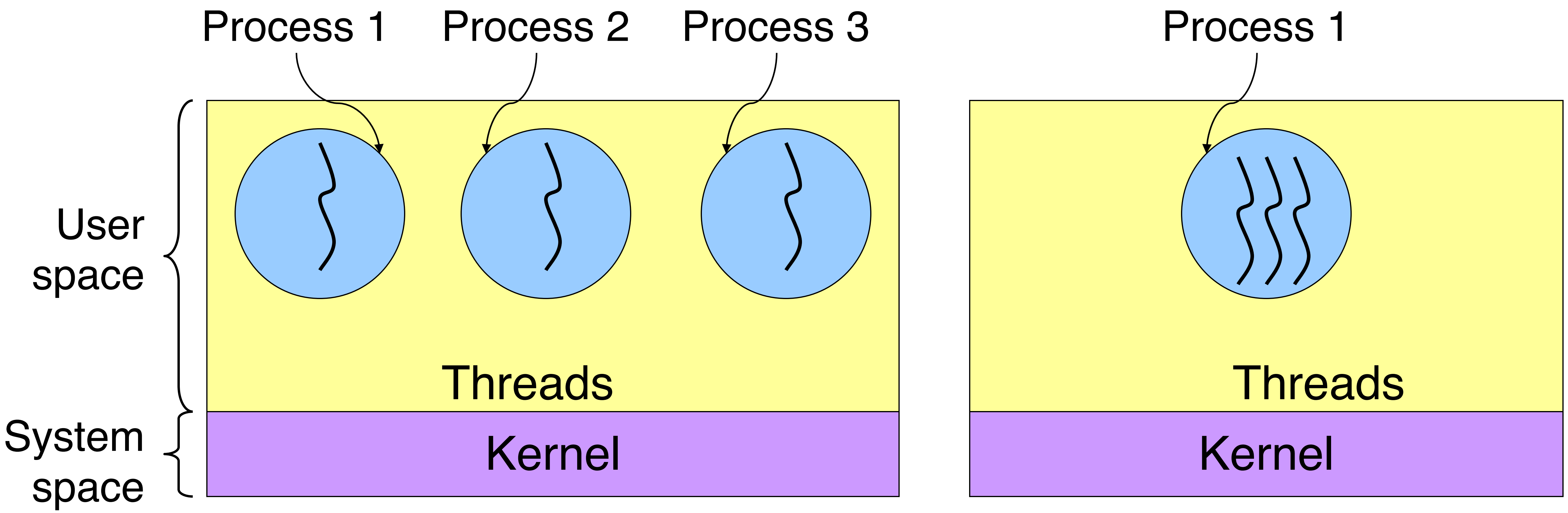
Threads: “processes” sharing memory

Process == address space

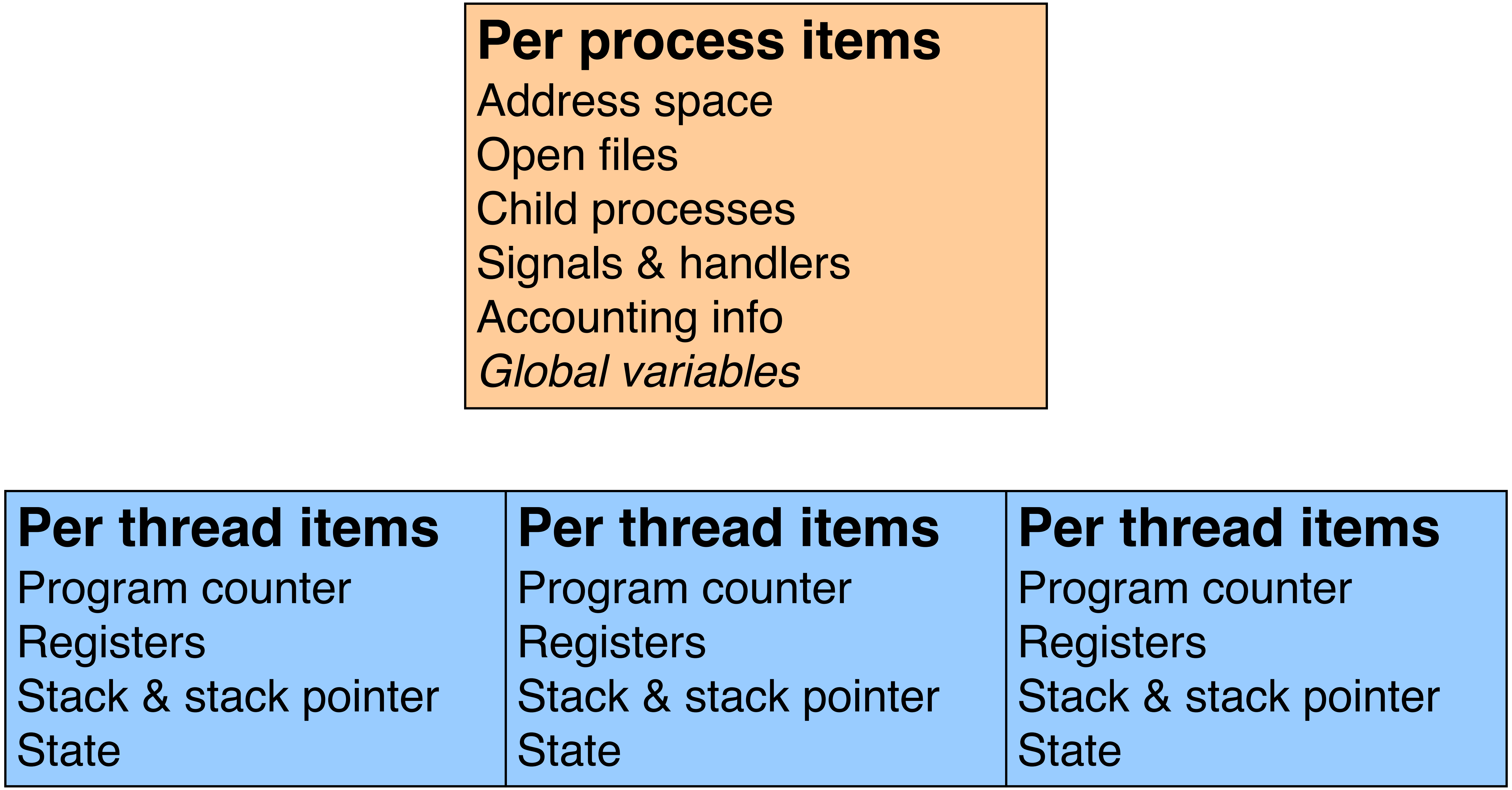
Thread == program counter / stream of instructions

Two examples

- Three processes, each with one thread
- One process with three threads



Process & thread information



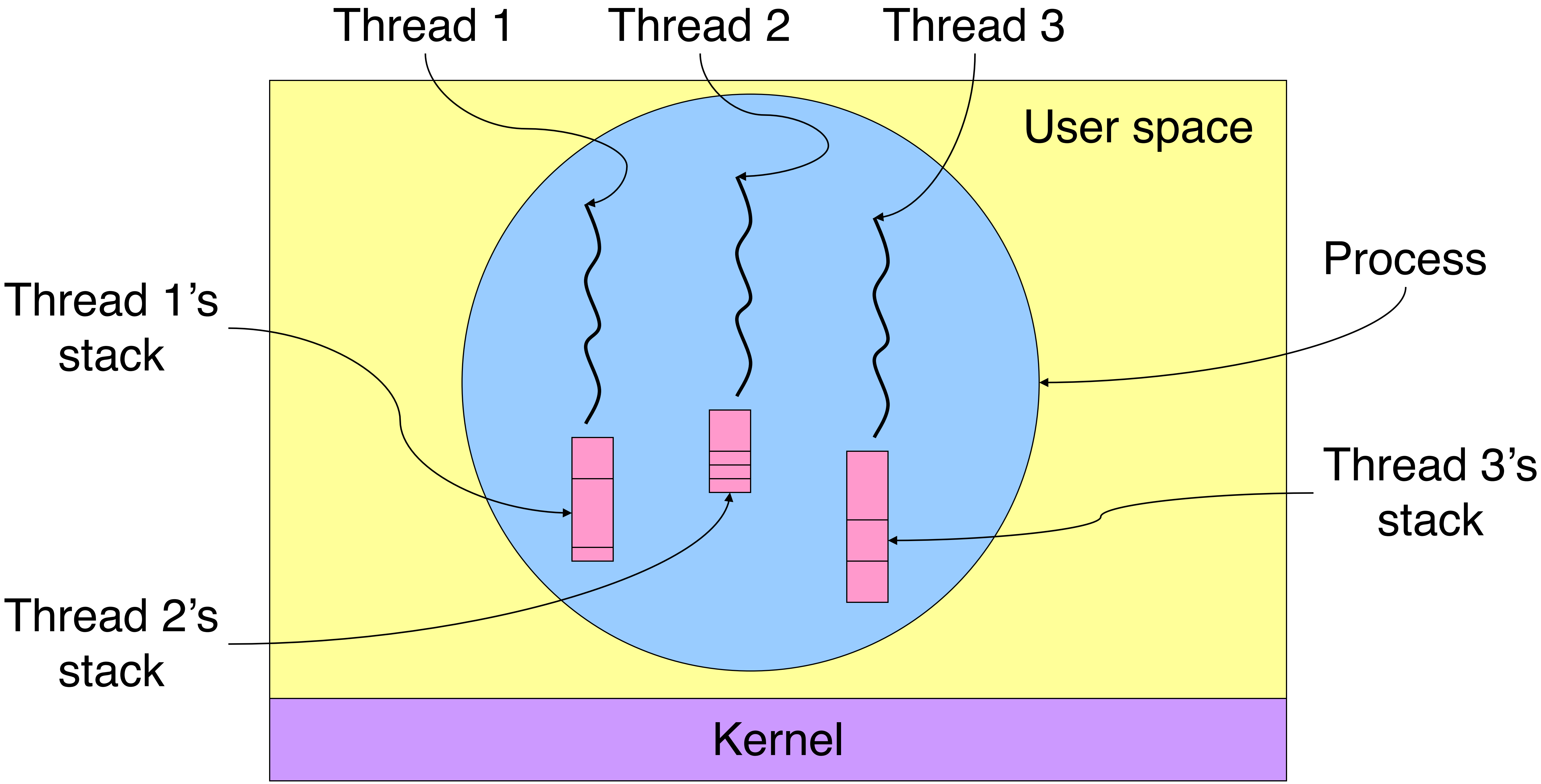
Per process items
Address space
Open files
Child processes
Signals & handlers
Accounting info
Global variables

Per thread items
Program counter
Registers
Stack & stack pointer
State

Per thread items
Program counter
Registers
Stack & stack pointer
State

Per thread items
Program counter
Registers
Stack & stack pointer
State

Threads & Stacks



=> Each thread has its own stack!

Why use threads?

Allow a single app to do multiple things at once

- Simpler programming model
- Less waiting

Threads are faster to create or destroy

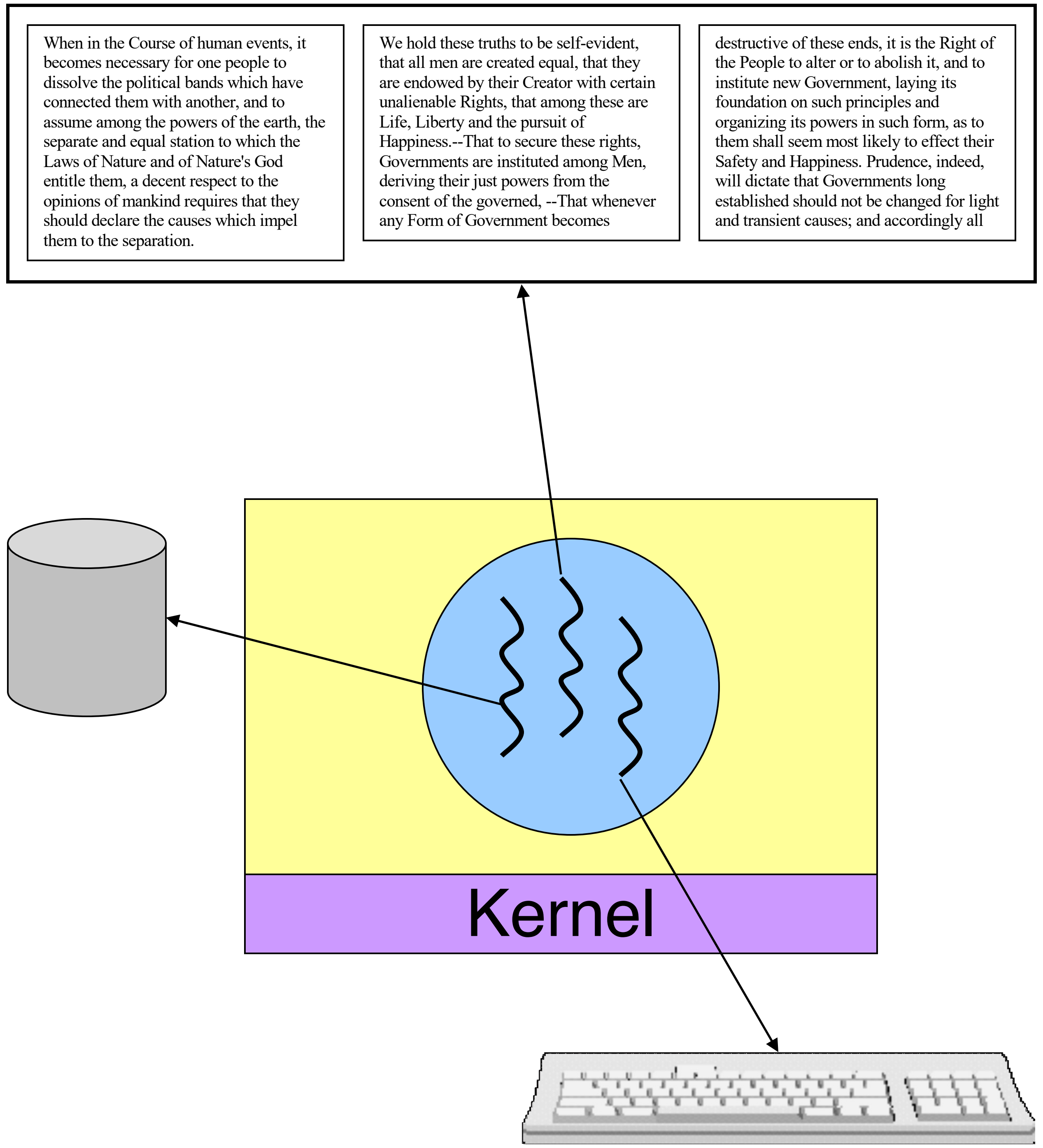
- No separate address space

Overlap computation and I/O

- It could be done without threads, but it's harder

Example: word processor

- Thread to read from the keyboard
- Thread to format document
- Thread to write to disk

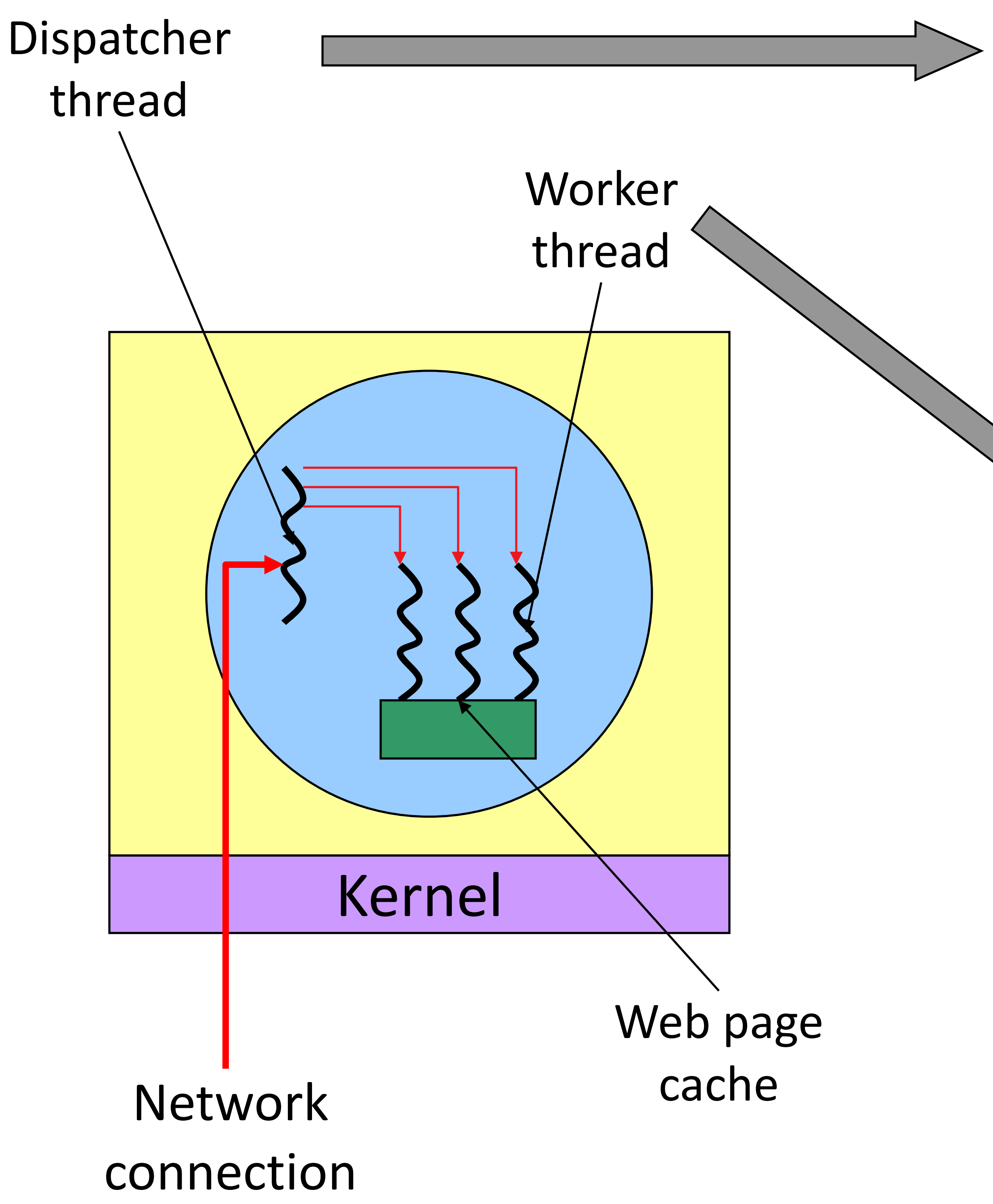


When in the Course of human events, it becomes necessary for one people to dissolve the political bands which have connected them with another, and to assume among the powers of the earth, the separate and equal station to which the Laws of Nature and of Nature's God entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the separation.

We hold these truths to be self-evident, that all men are created equal, that they are endowed by their Creator with certain unalienable Rights, that among these are Life, Liberty and the pursuit of Happiness.--That to secure these rights, Governments are instituted among Men, deriving their just powers from the consent of the governed, --That whenever any Form of Government becomes

destructive of these ends, it is the Right of the People to alter or to abolish it, and to institute new Government, laying its foundation on such principles and organizing its powers in such form, as to them shall seem most likely to effect their Safety and Happiness. Prudence, indeed, will dictate that Governments long established should not be changed for light and transient causes; and accordingly all

Multithreaded Web server



```
while(TRUE) {
  getNextRequest(&buf);
  handoffWork(&buf);
}
```

```
while(TRUE) {
  waitForWork(&buf);
  lookForPageInCache(&buf,&page);
  if(pageNotInCache(&page)) {
    readPageFromDisk(&buf,&page);
  }
  returnPage(&page);
}
```

Three ways to build a server

Thread model

- Parallelism
- Blocking system calls

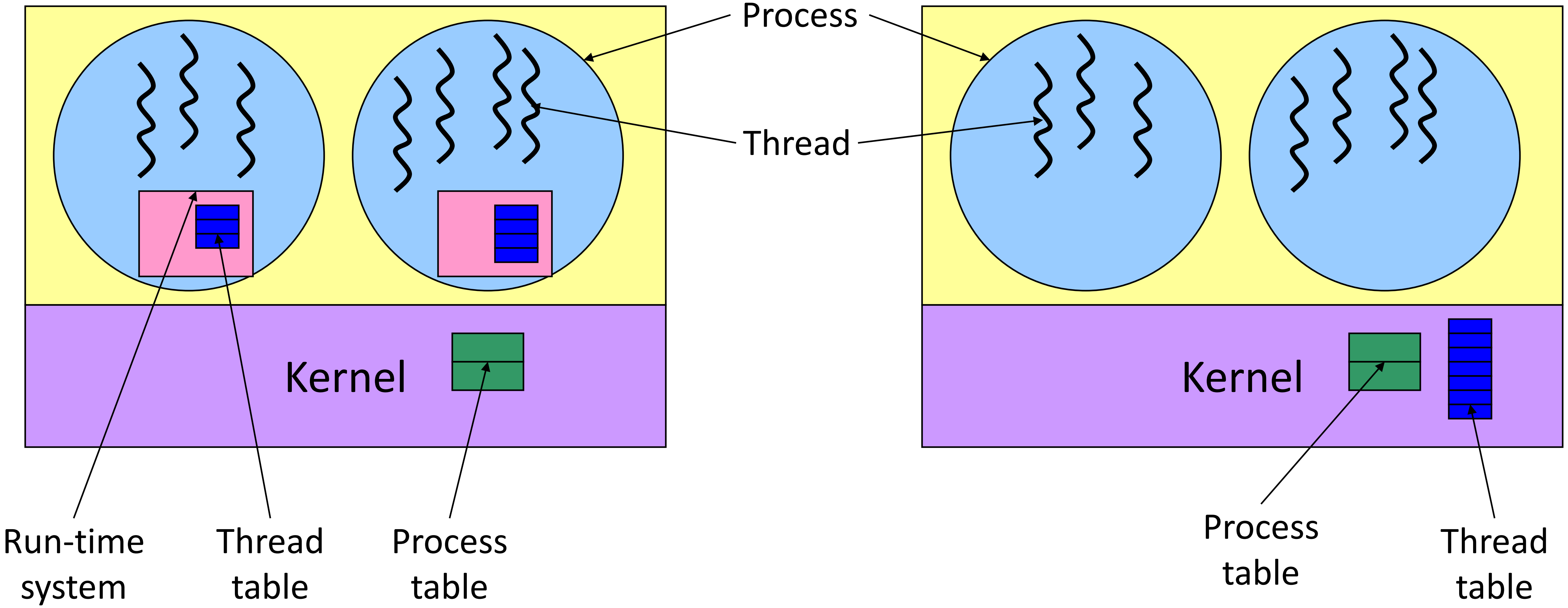
Single-threaded process: slow, but easier to do

- No parallelism
- Blocking system calls

Finite-state machine

- Each activity has its own state
- States change when system calls complete or interrupts occur
- Parallelism
- Nonblocking system calls
- Interrupts

Implementing threads



User-level threads
+ No need for kernel support
- May be slower than kernel threads
- Harder to do non-blocking I/O

Kernel-level threads
+ More flexible scheduling
+ Non-blocking I/O
- Not portable

Interprocess Communication

- Processes within a system may be **independent** or **cooperating**
- Cooperating process can affect or be affected by other processes, including sharing data

- Reasons for cooperating processes:
  - Information sharing
  - Computation speedup
  - Modularity
  - Convenience

- Cooperating processes need interprocess communication (IPC)
- Two models of IPC
  - Shared memory
  - Message passing

THE UNIVERSITY
OF ARIZONA