Process III, Concurrency

Instructor: Jocelyn Chen

Fork/Exec Separation Enables …

Backgrounding

```
$ myprog &
```

```
……
    else if (pid > 0) {              // parent process
         if (!background) {
             wait(0);                // wait only if it's a foreground process
         } else {
             printf("[1] %d\n", pid); // Print job info for background process
         }
     }
……
```

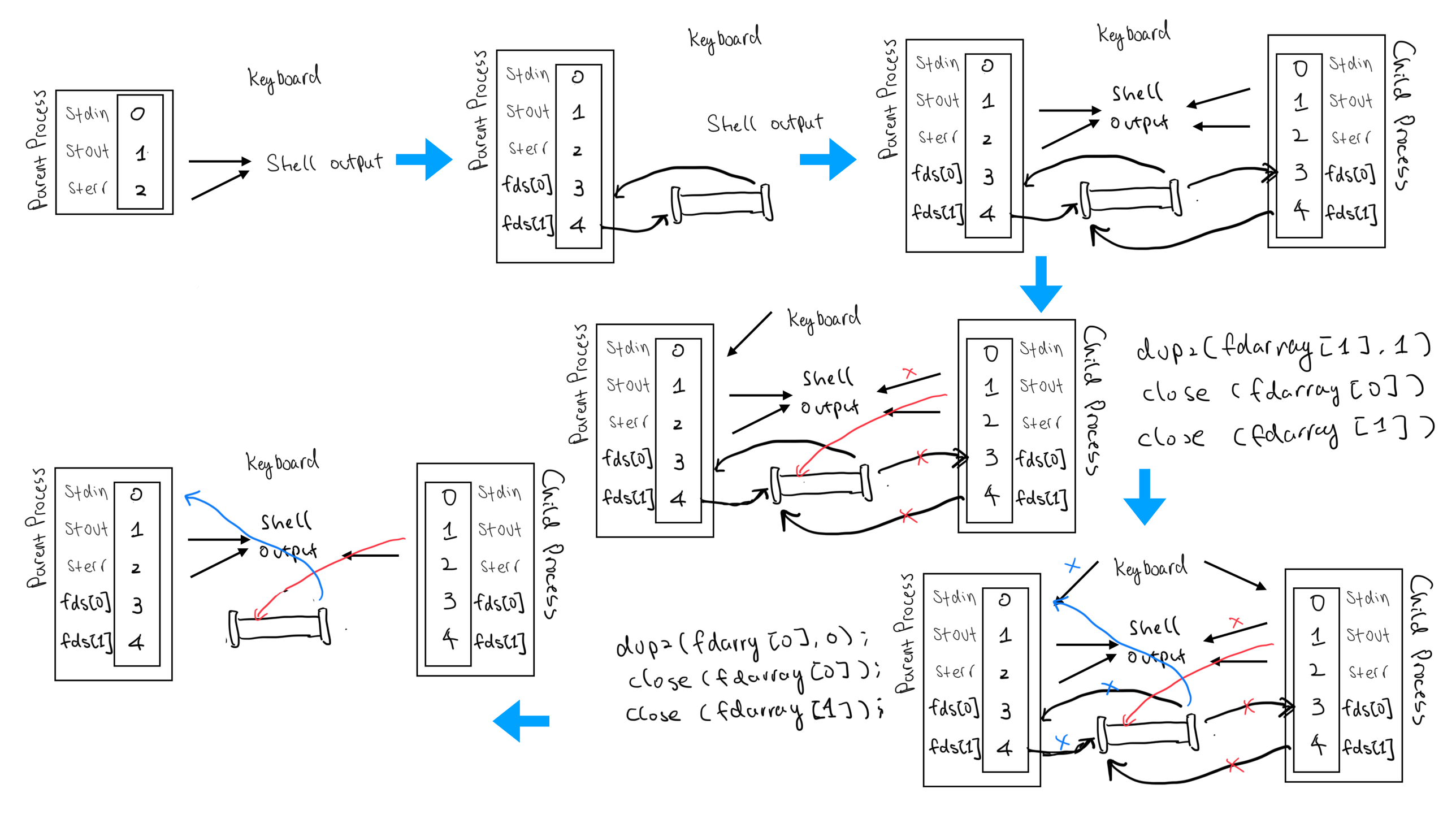
Fork/Exec Separation Enables …

Pipe

$ ps xc | grep …

```
void handle_pipeline(l_command, r_command) {
    int fdarray[2];
    pipe(fdarray);

    if ((pid = fork()) == 0) {  // child (left end of pipe)
        dup2(fdarray[1], 1); // make fd 1 the same as fdarray[1]
                             // which is the write end of the pipe
        close(fdarray[0]);
        close(fdarray[1]);
        parse(command1, args1, l_command);
        exec (command1, args1, 0);
    } else if (pid > 0) {   // parent (right end of pipe)
        dup2(fdarray[0], 0); // make fd 0 the same as fdarray[0]
                             // which is the read end of the pipe
        close(fdarray[0]);
        close(fdarray[1]);
        parse(command2, args2, r_command);
        exec(command2, args2, 0);
    }......
}
```

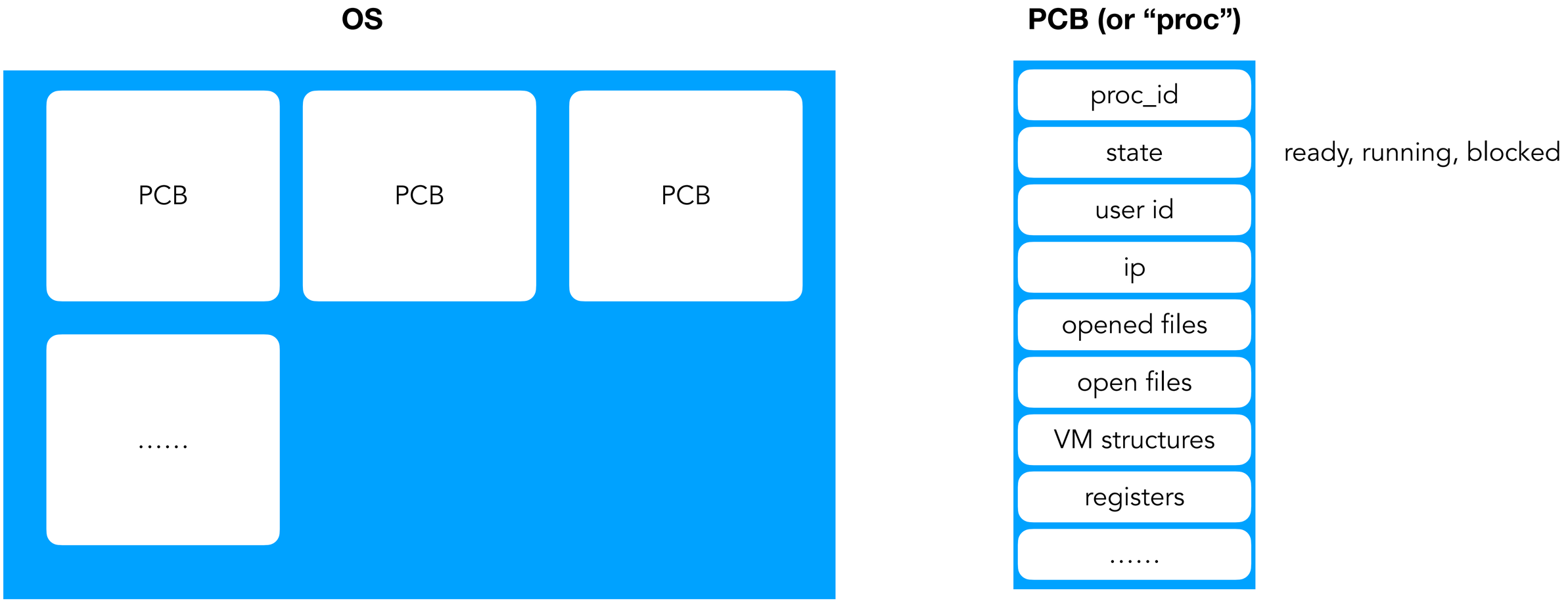

Parent Process
Stdin 0
Stout 1
Sterr 2
Keyboard
Shell output
fds[0] 3
fds[1] 4
Child Process
0 Stdin
1 Stout
2 Sterr
3 fds[0]
4 fds[1]
Shell output
dup2(fdarray[1], 1)
close(fdarray[0])
close(fdarray[1])
dup2(fdarry[0], 0);
close(fdarray[0]);
close(fdarray[1]);

To understand process…

How process see an abstract machine?

How OS implement the process abstraction?

What information of a process does OS keep track of?



Hmmm only single process so far?

What happen if i want to run two tasks at the same time in a program?

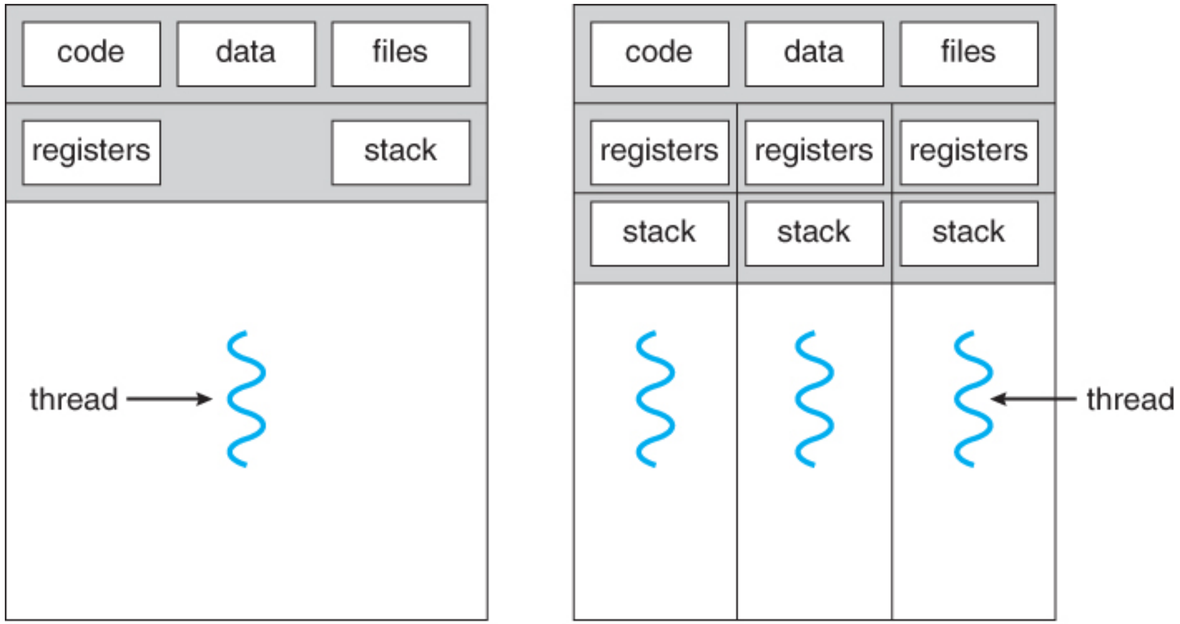
What happen if i want to speed up the computation using multiple processors?

How can we make use of CPU if it is waiting for some non-CPU instructions to finish?

Threading

Lightweight units of execution within a process

That means, you can do concurrent execution within a process using thread

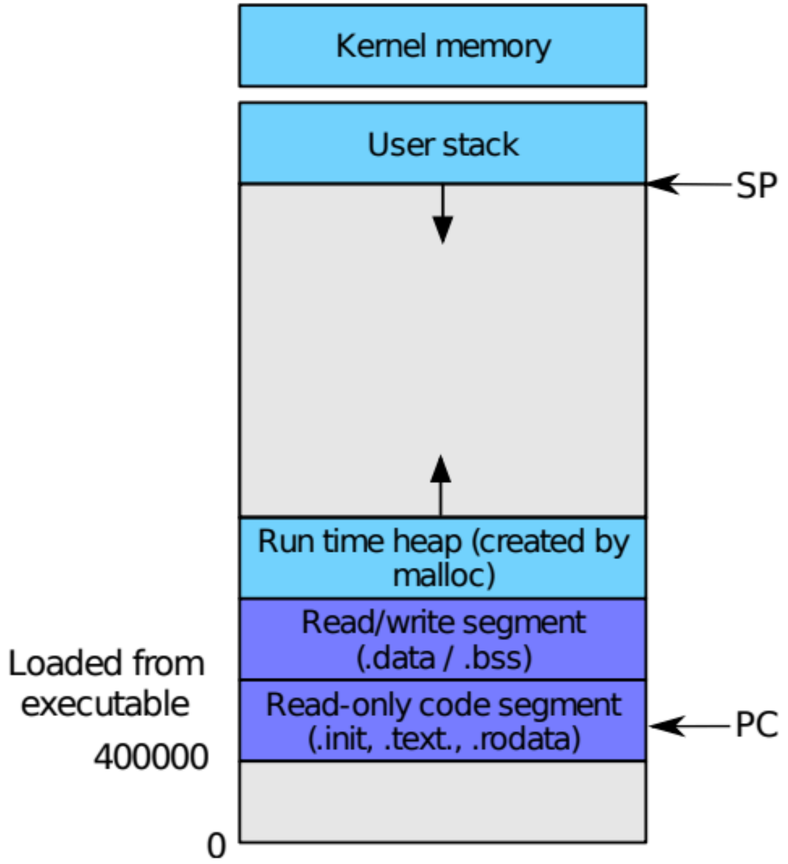


single-threaded process

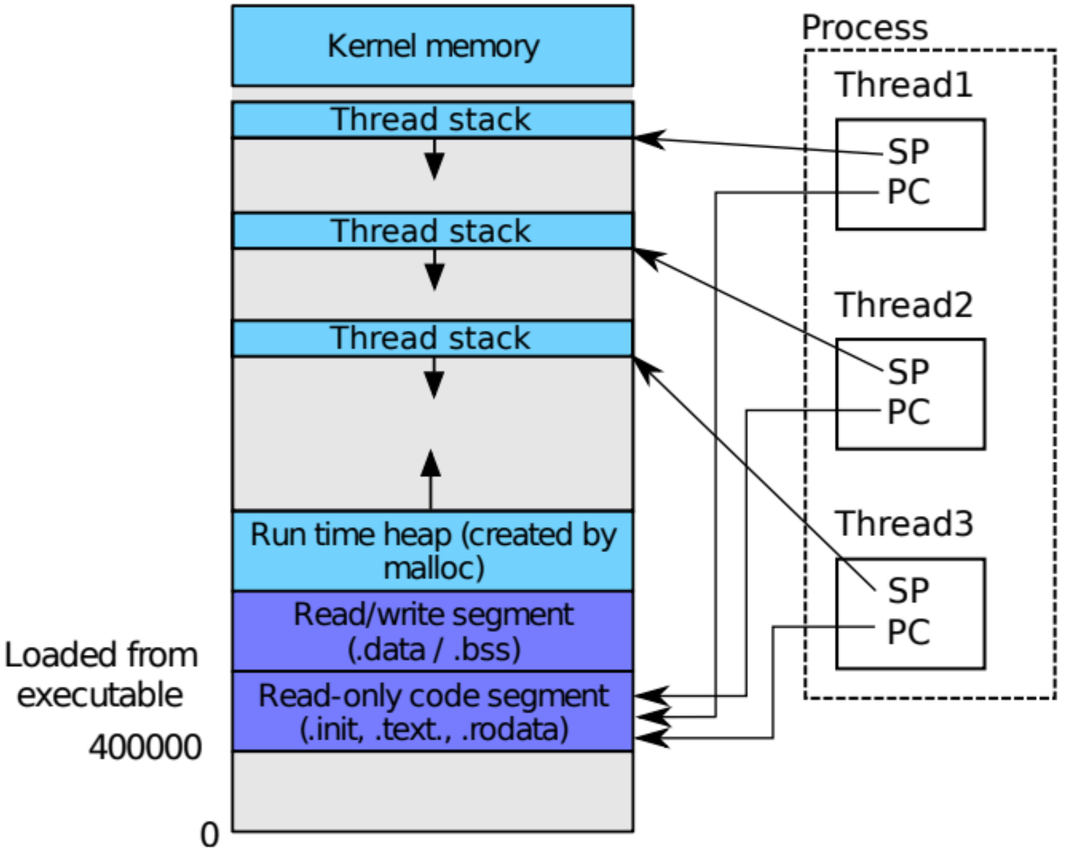
multithreaded process

https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/4_Threads.html

Process address space



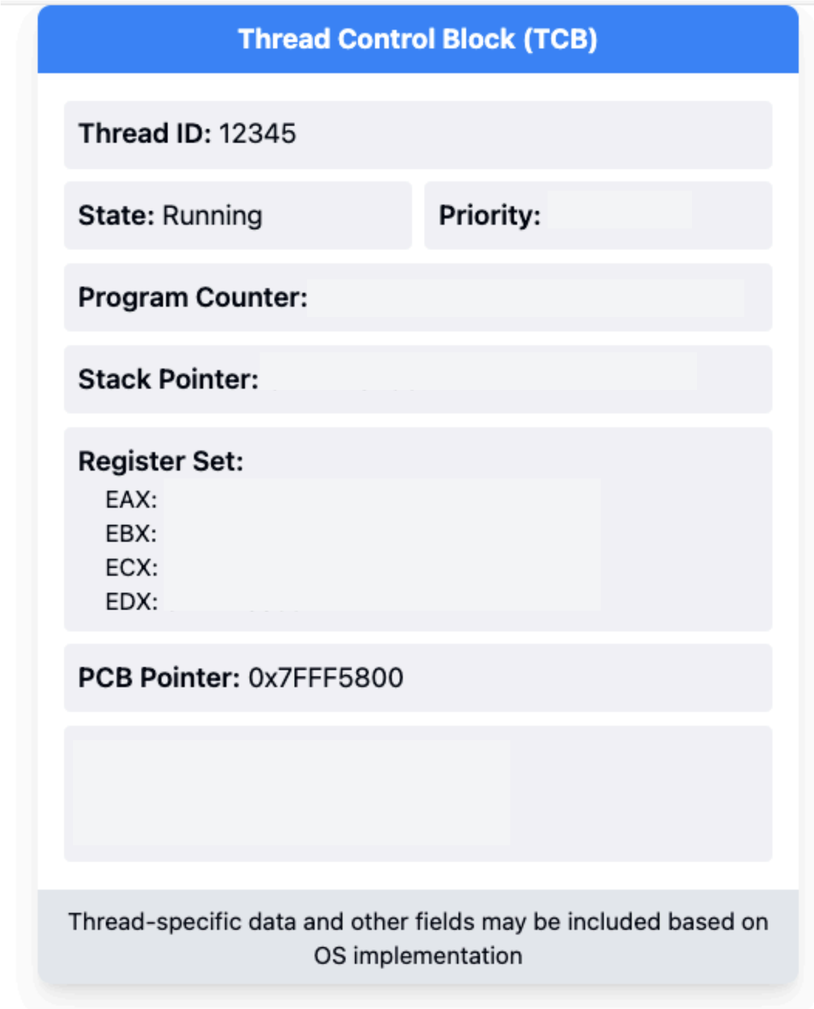
Process address space with threads



https://jhuopsys.github.io/spring2024/lectures/lecture04.pdf

TCB (thread control block)

a data structure **inside the kernel** that contains thread-specific information needed for managing the thread.



Interface to Threads

How do we create threads?

```
tid thread_create(void (*fn) (void *), void *arg);

void thread_exit();

void thread_join(tid thr);
```

And a lot more synchronization primitives

Threading is not the only way!

Concurrency

Simultaneous execution of multiple tasks

Broader concept than just threading

multiple CPUs and common memory

Multiple computers connected via a network

Multiple computers connected via a network

Allows CPU to work on other tasks while waiting for I/O to complete

the OS was the **first concurrent program**, and many techniques were created for use within the OS

Concurrency is **HARD**

Difficult to reason about all possible interleaving

Race conditions, deadlocks, lovelocks, starvations, …

Will talk more in the following lectures.

Handout 1(c)

```
  f()
1 movq 0x5000, %rbx     # load from address 0x5000 into register
2 addq $1, %rbx         # add 1 to the register's value
3 movq %rbx, 0x5000     # store back


  g()
4 movq 0x5000, %rbx     # load from address 0x5000 into register
5 addq $2, %rbx         # add 2 to the register's value
6 movq %rbx, 0x5000     # store back
```

Lab 2 is Released Today!