CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Processes

Imagine a one-program computer

Running program could access all memory (though it wouldn't want to overwrite OS) and would have CPU all the time

OS would be relegated to a library that the program could call

For the purposes of our discussion, let's assume we are working with a single single-core processor. The ideas generalize fairly easily to more cores, though the OS implementation gets messier.

On our actual computers, we are constantly switching back and forth between different applications

Sometimes this is done for efficiency, such as on a batch system

Other times, it is done to give the feeling of running more than one application at once

What is the right word for the "things" that we keep switching between?

Program? Not really. A program is a piece of code, something that can be written on paper. A program is passive and does not have state.

"Process" is the word we will use for this

(Others may use "task" or "job" to mean the same thing)

Process is "a program in execution"

Process is a unit of work -- what we can schedule on the CPU

Process is a unit of resource ownership -- processes are assigned memory segments and open files

More than one *process* can run the same *program*

What is everything that makes up a particular process?

Or, the same question another way:

What do we need to keep track of to allow a process to resume executing after it has stopped as though nothing happened?

- Memory regions
- Program counter
- Registers
- Open files and I/O devices

All of these will be enough to uniquely identify what a process was doing when it stopped running.

Recall that the program itself is stored in the process's memory

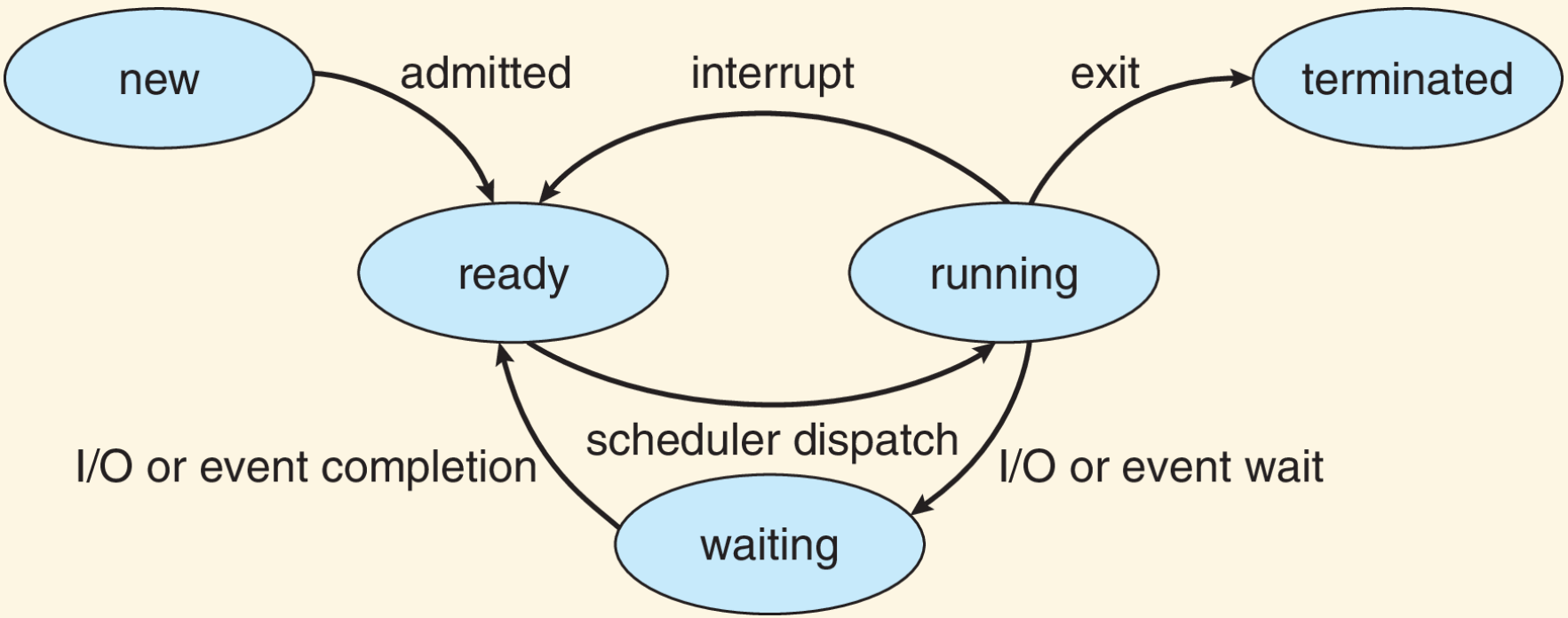
Another way to think about it is that each process is a different single-program computer being simulated by our actual computer

When a different process is run, it just means our computer has gone on to simulating something else

(Just ignore this part if you don't find it helpful)

One other thing we would like to know about a process is what it is doing right now.

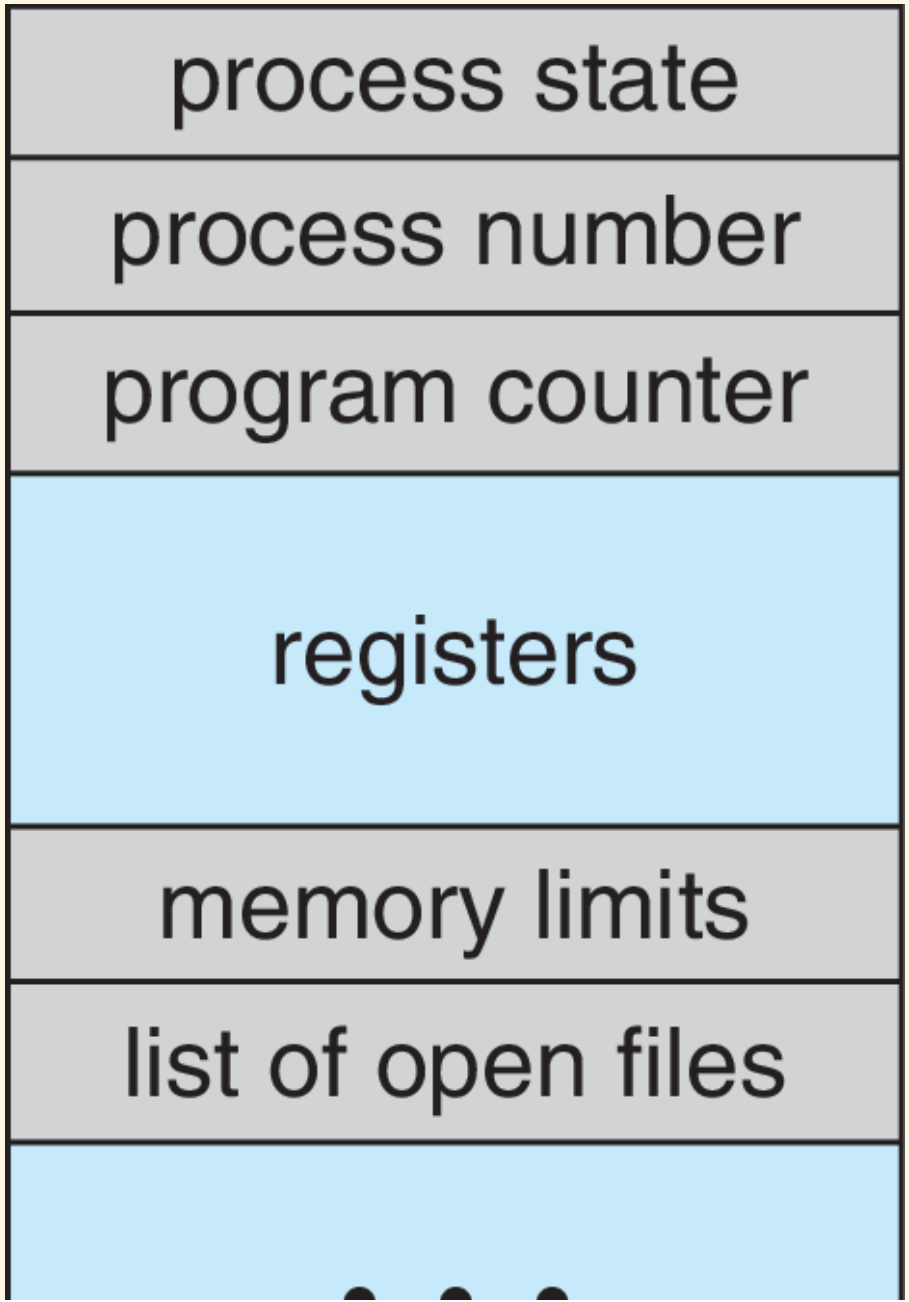
Is it on the CPU, waiting for I/O, ...?


new
admitted
interrupt
exit
terminated
ready
running
scheduler dispatch
I/O or event completion
I/O or event wait
waiting

Process is an abstract concept, but the OS needs to represent processes to manage them

All the things we've talked about so far must be included in the representation, along with bookkeeping data the OS keeps about processes

Representation is the Process Control Block (PCB)


process state
process number
program counter
registers
memory limits
list of open files

In the PCB

Context

- state
- PC
- registers
- memory
- I/O
- files

Bookkeeping

- scheduling info
- accounting

In Linux, data type for storing PCB is `task_struct` in `sched.h`

Linux "tasks" are essentially what we call processes