CS 4410
Operating Systems

Processes and Threads

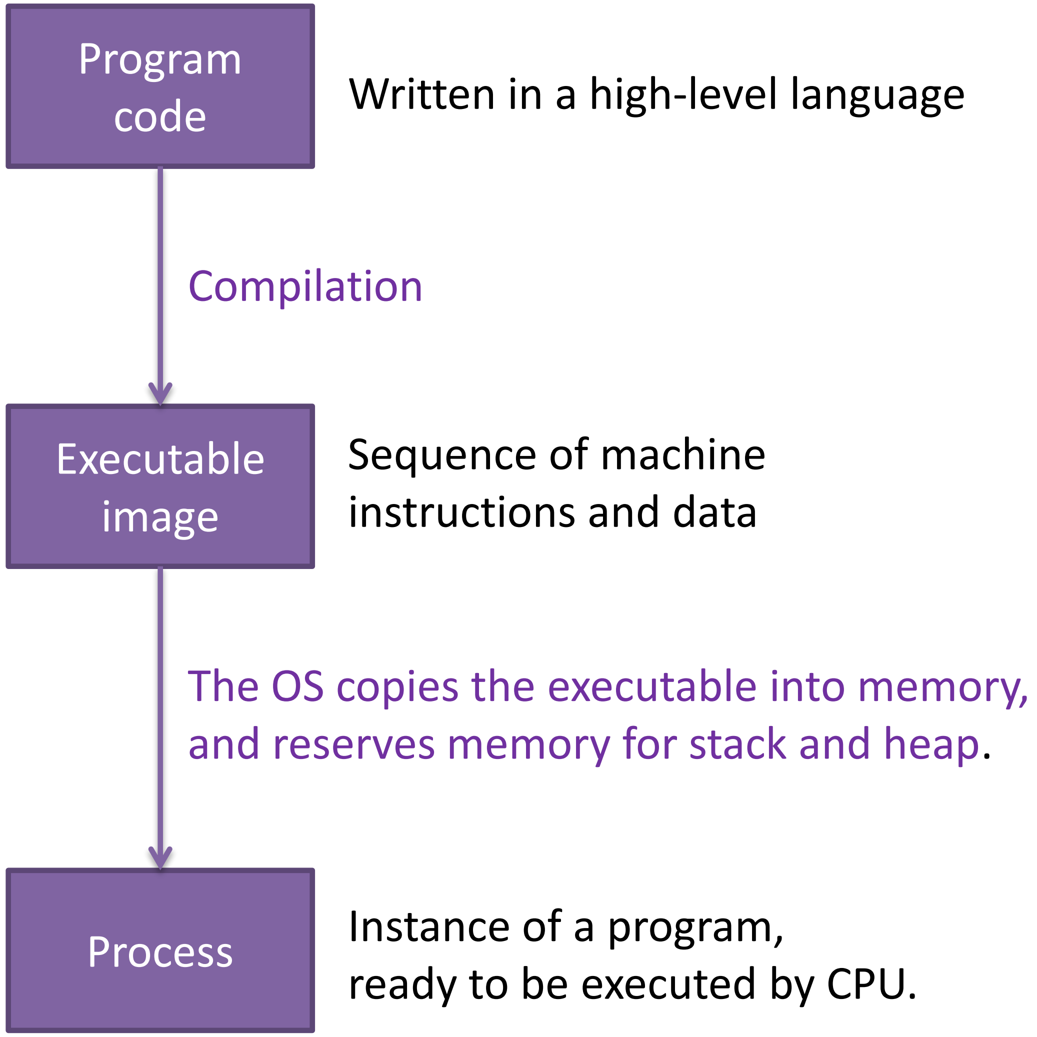
Summer 2016

Cornell University

Today

- Process
- Thread

From program code to a process

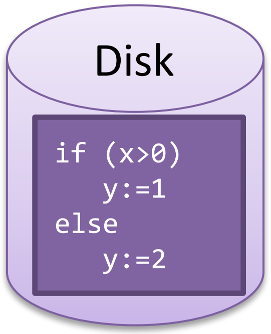


Program code
Compilation
Executable image
The OS copies the executable into memory, and reserves memory for stack and heap.
Process

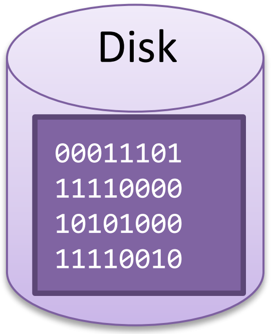
Program code: Written in a high-level language

Executable image: Sequence of machine instructions and data

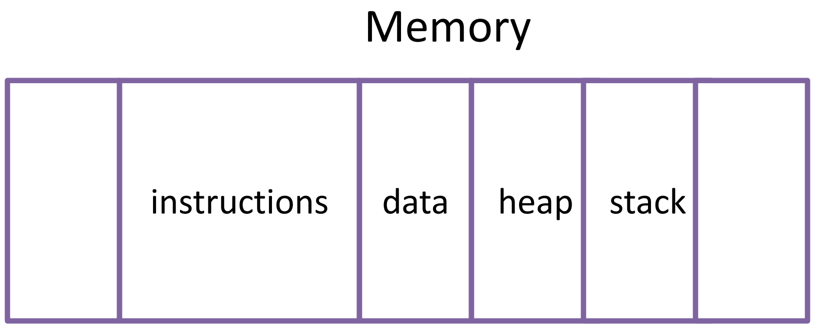
Process: Instance of a program, ready to be executed by CPU.



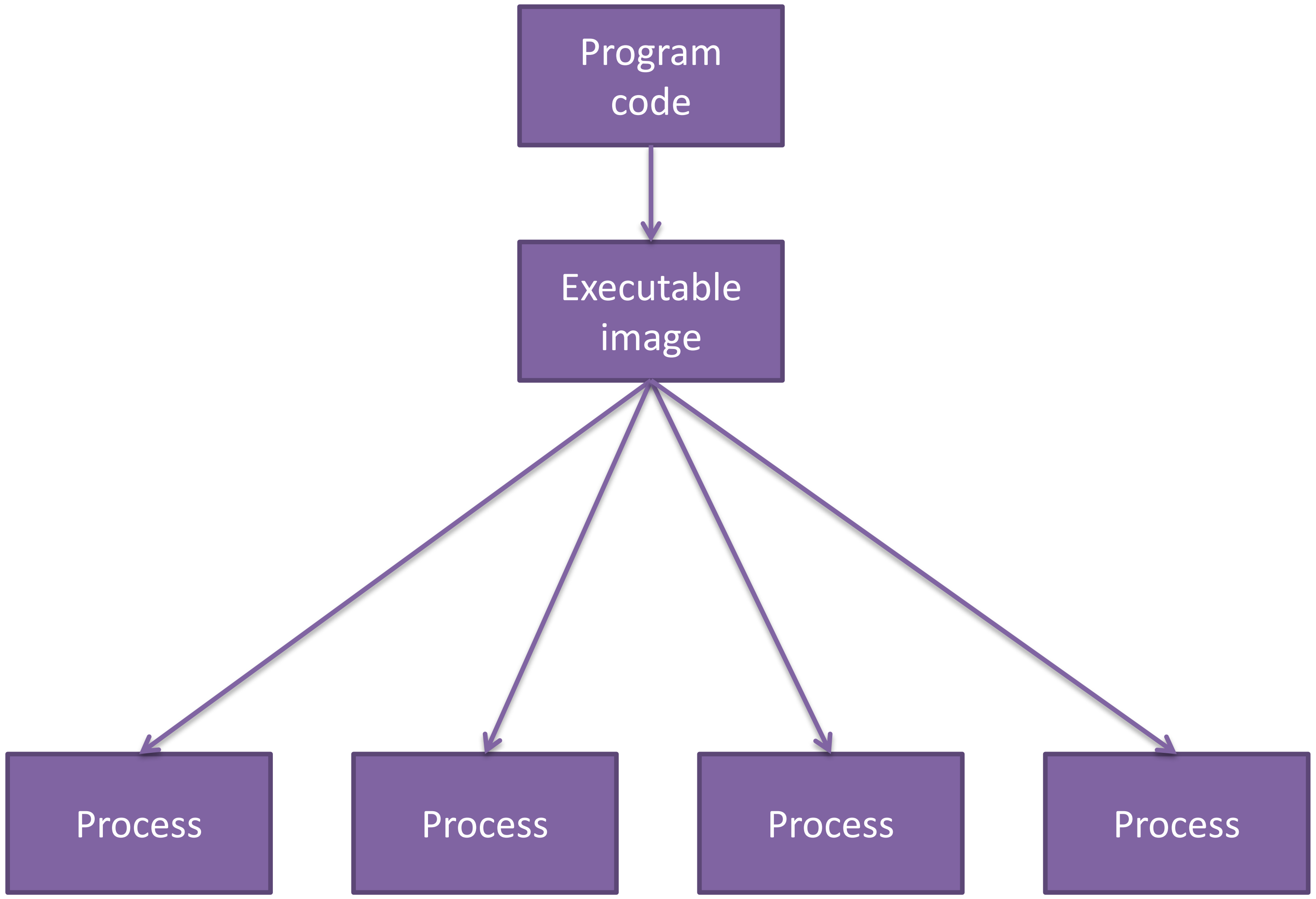
```
if (x>0)
   y:=1
else
   y:=2
```



```
00011101
11110000
10101000
11110010
```



From program code to processes



The process abstraction

- One of the fundamental contributions of operating systems!
- Processes can receive inputs or generate outputs at any time—contrary to the Turing view that processes get all their input at the start and produce all their output at the end.
  - Interactive computations can implement functions that non-interactive computations cannot. [1]

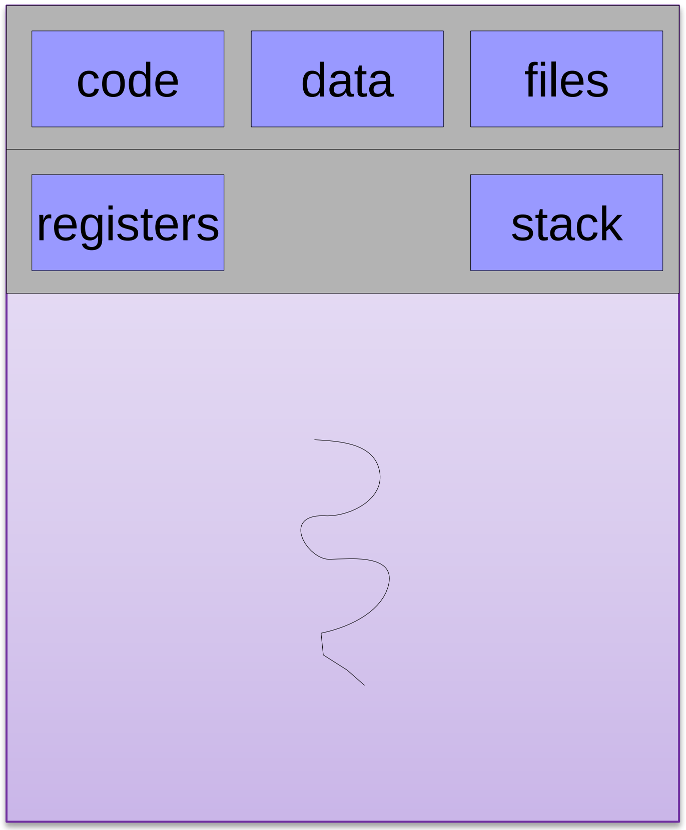
From a process to threads

- Traditional programming model: one logical sequence of steps.
- Traditional model is not appropriate for applications:
  - that serve multiple requests,
  - consisting of logically concurrent tasks (e.g. user interfaces + background computation, OS),
  - that exploit multiple processors.

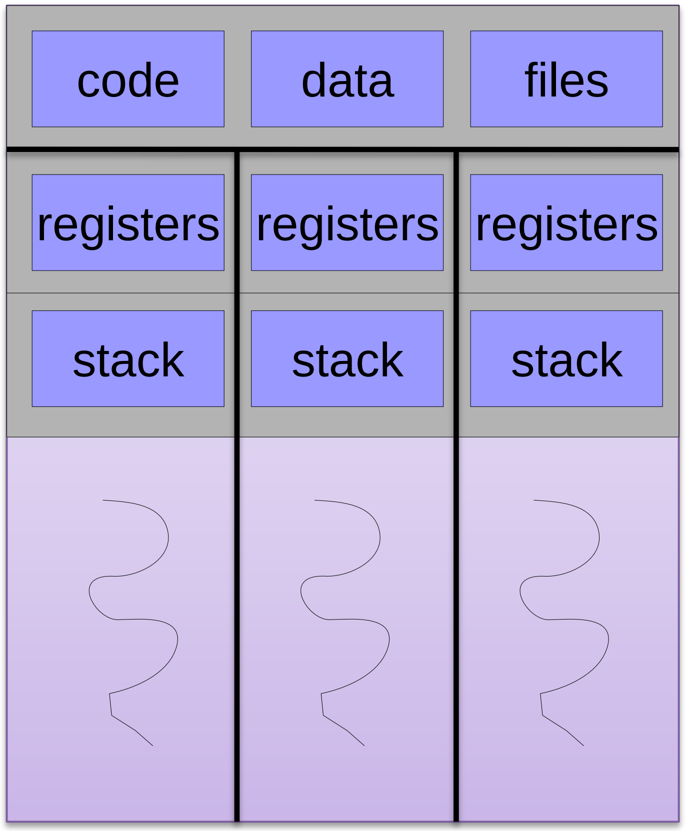
From a process to threads

- Multi-threaded programming model: a set of logical sequences of steps.
  - These sequences can be executed concurrently.
- *Thread*: single execution sequence.
- Traditional program → single-threaded process
- Multi-threaded program → multi-threaded process

From a process to threads



single-threaded process



multi-threaded process

Processes VS Threads

- Many processes of a program:
  - Isolation between processes.
  - Slow communication between processes (through OS).
  - Useful when many machines execute the same program.
- Many threads of a program:
  - Fast communication between threads (through shared memory).
  - Economy of resources (e.g. instructions, open files).
  - Useful when one machine executes the program.

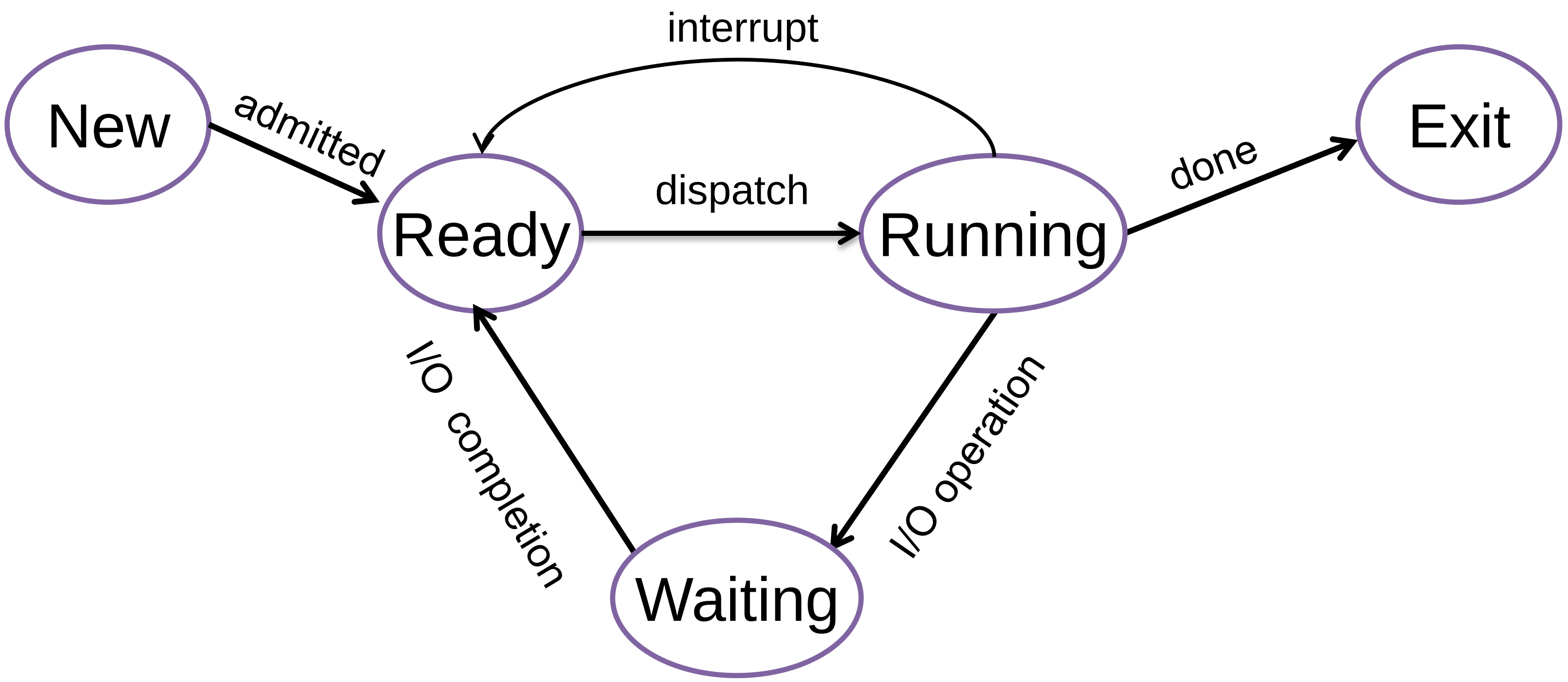
Managing threads

- A process may consist of a large number of threads.
- A machine has a small number of CPUs.
- The OS should create the illusion that all threads produce work “at the same time”.
- The OS does that by scheduling the execution of threads on CPUs.
  - For example, the OS gives some CPU cycles to each thread iteratively, until all threads complete their job.
- Thread: separately schedulable task.
  - Its execution may be initiated, stopped and resumed by the OS.

Thread State

- The execution of a thread on the CPU may be disrupted by synchronous and asynchronous events.
- So, the state of a thread may change.
- For example, a thread may be
    - waiting to be assigned to the CPU, or
    - executing instructions on the CPU, or
    - waiting for an event, e.g., I/O completion.
- The OS needs to keep track of threads’ state.

Thread State Transitions



Thread Control Block (TCB)

The TCB stores all the information the OS needs about a particular thread, such as:

- thread state,
- where it is stored in memory,
- what privileges the thread has,
- the program counter, indicating the next instruction,
- a set of general-purpose registers with current values,
- a set of operating system resources (open files, connections to other programs, etc.).

The TCB represents the computation state and metadata of a thread.

State Queues

- The OS maintains a collection of queues that represent the state of all thread in the system
- There is typically one queue for each state, e.g., ready, waiting for I/O, etc.
- Each TCB is queued onto a state queue according to its current state.
- As a thread changes state, its TCB is unlinked from one queue and linked onto another.

Context Switch

- Whenever a thread leaves the “Running” state, it leaves CPU.
- CPU becomes available to execute another job.
- The OS is responsible to switch the context in which CPU is working.
- The OS should save the execution state of the current thread and restore the execution state of a different one.
- Time consuming

Context Switch: an example

- Suppose thread T is at the “Running” state and executes a system call.
- The OS saves the state of thread T to the TCB of T.
- The OS handles the system call.
- The OS restore the state of thread T from the TCB of T.

Today

- Process
- Thread
- Context switch

[1] Peter J. Denning, The Profession of IT: Fifty Years of Operating Systems, Communications of the ACM, March 2016, Vol. 59, No. 3

Coming up…

- Next lecture: CPU scheduling
- HW1: due today at 10pm

Discussion

- Consider a web-browser.
  - How would you implement the creation of a new tab: as a new process or as a new thread?
  - How would you implement the creation of a new private window: as a new process or as a new thread?
  - You are checking out your favorite news portal and you decide to log-in to your bank account. Would you open a new tab, or a new private window?
- Consider Google search. Which steps of processing a search query may involve multithreaded programming?