Construction of User Interfaces (SE/ComS 319)

Jinu Susan Kabala

Department of Computer Science

Iowa State University, Fall 2021

BASICS-OS

Outline

- Threads
  - Why threads are needed?
  - Threads vs Processes
  - Issues with threads and concurrency
    - How to handle it?

PROCESS AND THREADS

Questions (1)

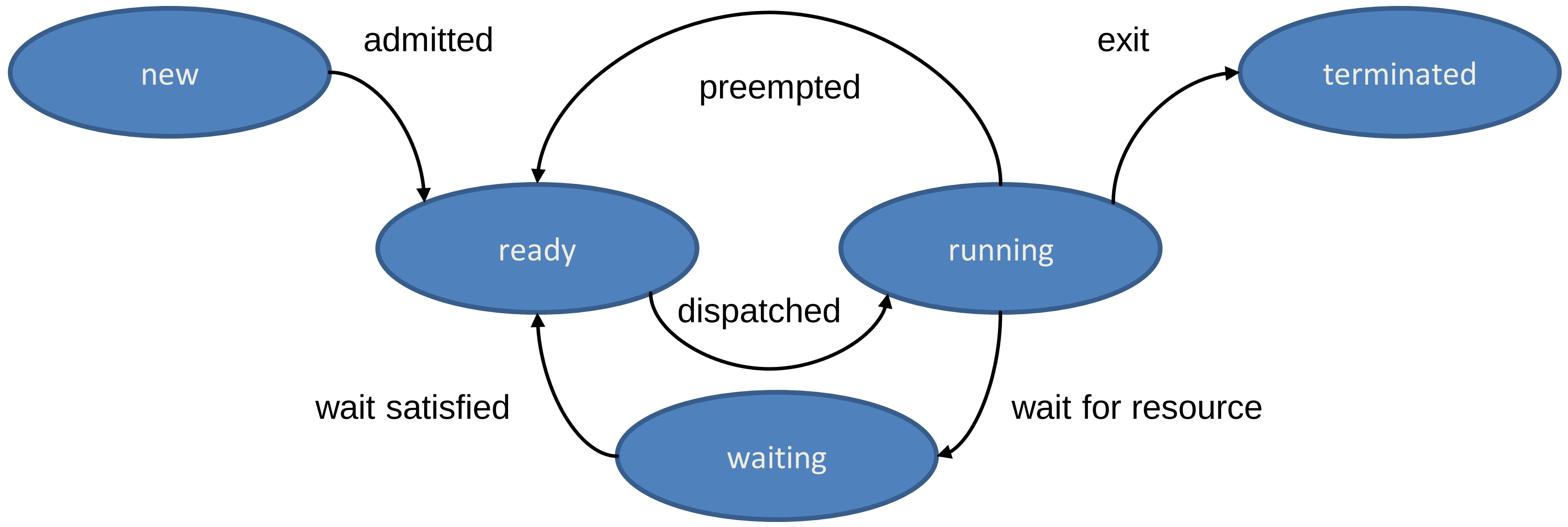
1. What are some reasons that a program cannot execute an instruction immediately?

2. What's a process?

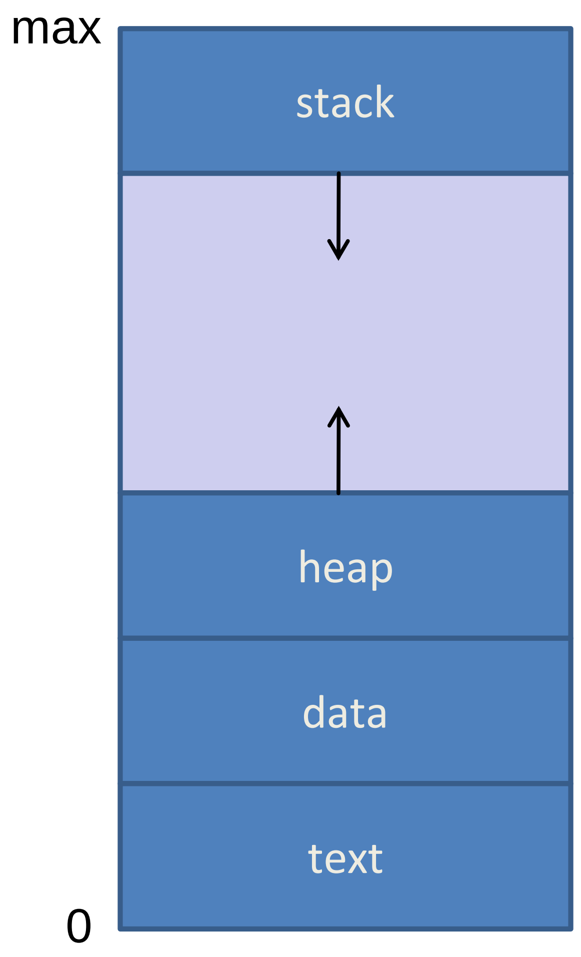
3. How do so many processes execute at the same time?

Process

- A process is a program in execution
- States of a process



Processes in memory



Temporary data: function parameters, return addresses, local variables

Dynamically allocated memory

Global variables

Program code

Questions (2)

1. What's a thread?

2. What are differences between a process and a thread?

3. How are mechanisms for creating threads in Java?

Motivation for threads

- One of the major concepts we learned in programming is that program execution proceeds step by step. Until a step is **completed**, the next step **cannot be started**.

- Consider the following scenarios:
  - Scenario (1): Printing a large file
  - Scenario (2): Loading a web page

Scenario (1): Printing a large file

- You are editing a large file. You ask the document editor to print it for you. While it is being printed, the editor does not allow you to continue editing your file (as the printing needs to be completed first).

- Nowadays, document editors do print jobs in the background - allowing you to continue editing.

Scenario (2): Loading a web page

- You just navigated to a web page and there are many pictures and videos on the web page.
- You have a slow internet connection and have to wait a long while before you can start reading the information on the web page.
- Most good web pages will load text first and allow to start reading while it continues to upload the pictures/videos in the background.

Threads

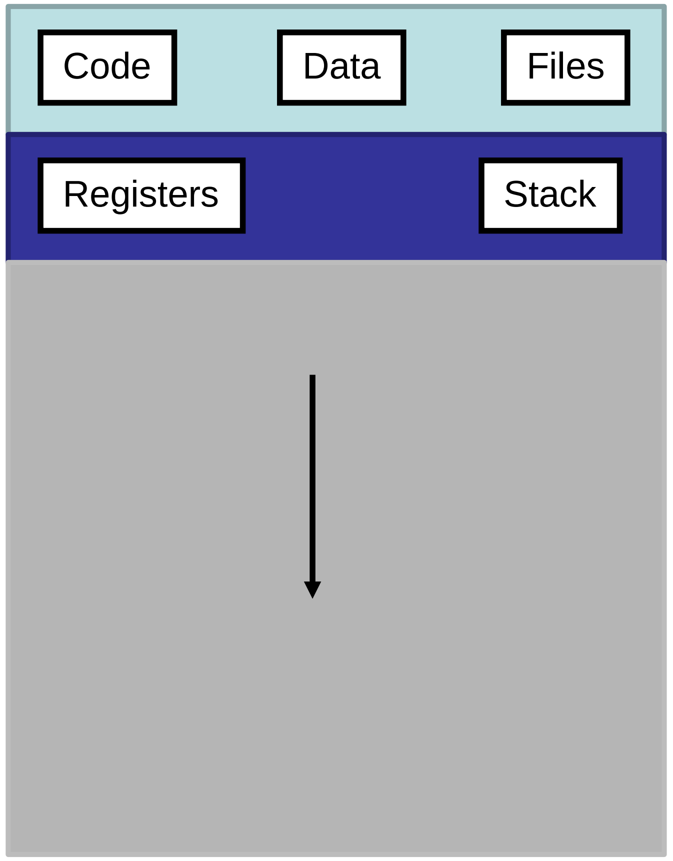
- In both the previous scenarios, we want more than one set of tasks to be going on **at the same time (i.e. concurrently).**
- Typically, there are many programs on a computer (operating system) and these can be executed by the computer at the same time. Executing programs are called PROCESSES.

- However, it is also possible to have **separate sets of tasks** being **concurrently** executed in the **SAME PROGRAM**. These are called THREADS.

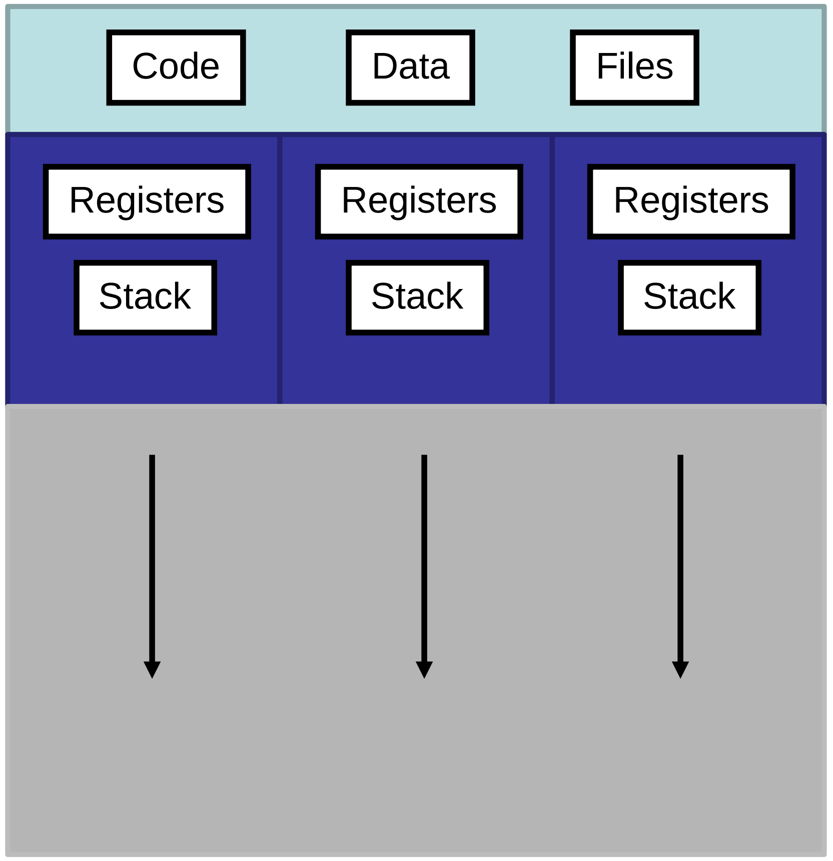
Threads (2)

- Basic unit of CPU utilization
  - Flow of control within a process
- A thread includes
  - Thread ID
  - Program counter
  - Register set
  - Stack
- Shares resources with other threads belonging to the same process
  - Text (i.e., code) section
  - Data section (i.e., address space)
  - Other operating system resources
  - E.g., open files, signals

Single-threaded vs. multi-threaded

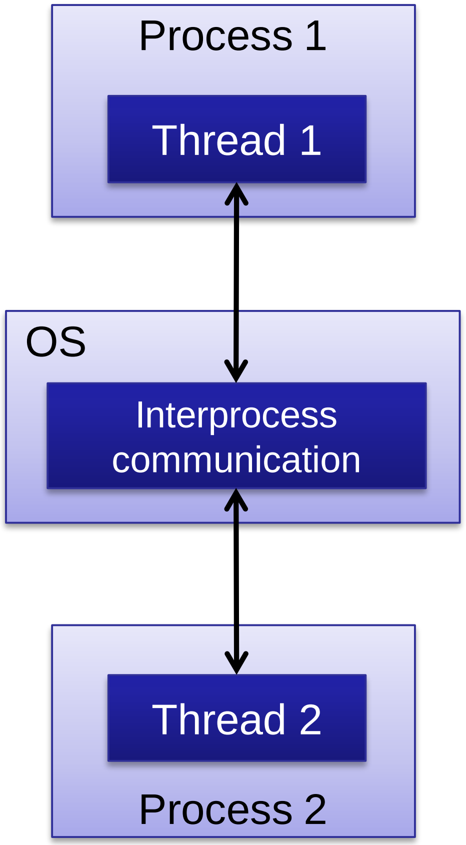


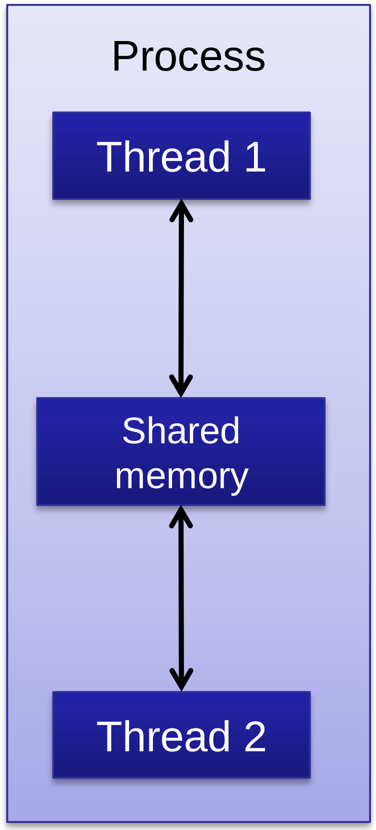
Single-threaded



Multi-threaded

Processes vs. threads





Processes vs. threads (2)

Processes

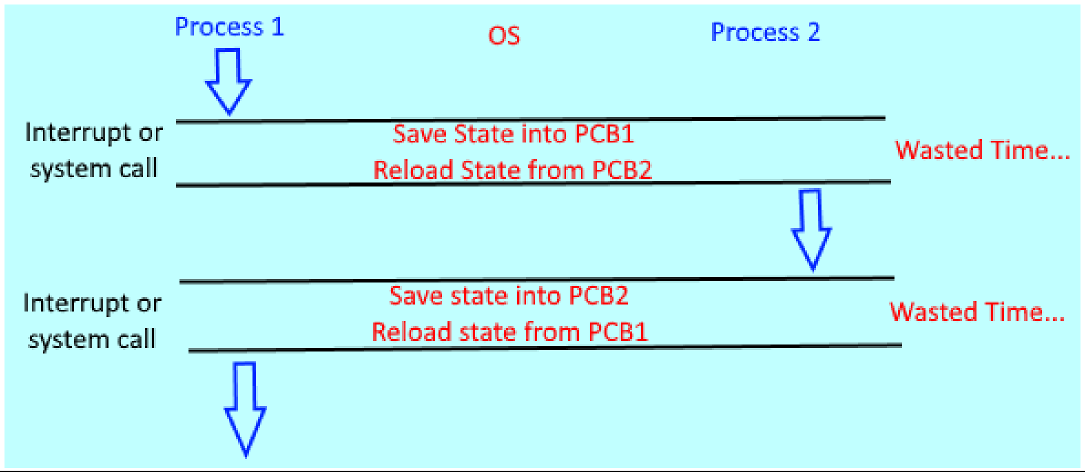
- Communication explicit
- Often requires replication of data
- Address spaces protected
- Writing debuggers is harder
- More scalable

Threads

- Convenient communication via shared variables
- More space efficient - sharing of code and data
- Context switch cheaper
- Harder to debug – race conditions

Context switch between processes

- Switching the CPU to another process is costly (context switch between threads is cheaper)!
- It requires saving the state of the old process and loading the saved state for the new process
    - When a process is suspended/interrupted its state is saved in PCB (Process Control Block) and will be loaded again when it is ready to run



Source: www.stackoverflow.com/

Multithreading models

User-level threads

- Supported above the kernel
- Implemented by a thread library
- Creation, scheduling, and management in user space

Kernel-level threads

- Supported directly by the operating system
- Creation, scheduling, and management in kernel space
- Virtually all contemporary operating systems support kernel-level threads – including Windows, Linux, Mac OS

Questions (3)

1. What are some issues with threads (or concurrency in general)?
    - Data races (race conditions)
    - Deadlocks

2. What is a mechanism provided in Java to handle these issues?
    - **Synchronized** keyword (race condition):

      synchronized (this) { return this.i++; }

Parallelism vs. concurrency
(often used interchangeably)

Parallelism

- Parallel processing of subtasks for the purpose of speedup
- Requires **parallel hardware**

Concurrency

- Overlapping but potentially unrelated activities
- Access to shared resources possible
- Does not require parallel hardware

Concurrent programming (multi-tasking)

Separation of concerns

- Group related code
- Keep unrelated code apart
- Examples
  - Waiting for input vs. processing input in interactive applications
  - Waiting for requests vs. processing requests in server
  - Background tasks such as monitoring the file system for changes in desktop applications
- Beneficial even when using a single CPU

Web browser

DVD player

Literature – Threads

- There is a lot to know about threads.
- Good resource:
  - http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

- COMS 430: good class to take (about concurrency)
- COMS 527: Concurrent systems (graduate course)