

Processes

CPEN333 – System Software Engineering
2021 W2
University of British Columbia

© *Farshid Agharebparast*



Introduction

- A process is a basic unit of execution.
- Normally, many processes run on a computer.
 - ❖ We discussed that process management is one of the responsibilities of an operating system.
- We are going to discuss some essential concepts related to processes here, and will focus on implementing multi-processing applications next.

Objectives

- To introduce the notion of a process
 - ❖ that is a program in execution, which forms the basis of all computation
- To describe the a few features of processes, including
 - ❖ creation,
 - ❖ termination,
 - ❖ scheduling

Process Concept

- Current-day computer systems allow multiple programs to be loaded into memory and executed concurrently, as opposed to earlier computers which would allow only one program to be executed.
- An operating system executes a variety of programs, and potentially, many processes can execute concurrently, with the CPU(s) multiplexed among them.
- This evolution required firmer control and more compartmentalization of the various programs, resulting in the notion of a **process**.

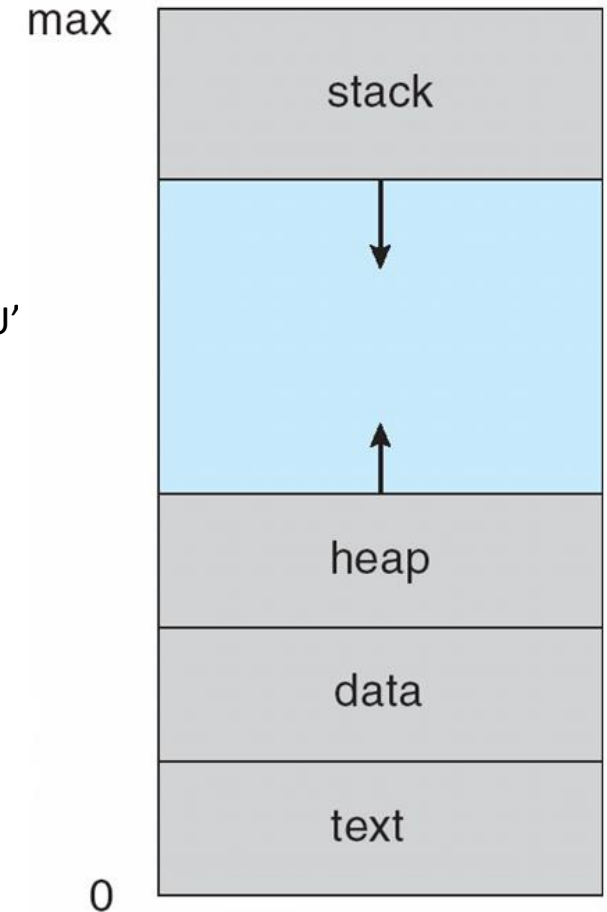
Process

- A **Process** is
 - ❖ a program in execution
 - ❖ the unit of work in a modern system (To be compared with threads later)
- A program by itself is not a process
 - ❖ a program is a passive entity (e.g. a file on disk), whereas a process is an active entity.
- A process execution progresses in sequential fashion (following the usual flow of a program)

Process Concept (cont)

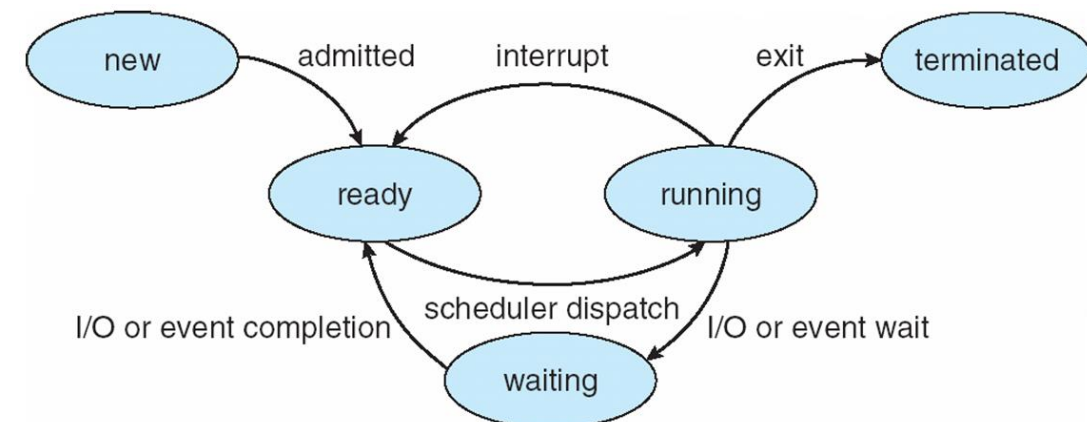
- A process is more than the program code (text section)
- A process also includes:
 - ❖ the current activity
 - **program counter** (indicates the address of the next instruction)
 - contents of the **processor's registers** (A register is a unit of the CPU's fast small internal memory)
 - ❖ generally the **stack** (temporary data such as function parameters, return addresses, ...) and a **data section** (e.g. global variables)
 - ❖ and it may also include a **heap** (dynamically allocated memory at run-time)

Note: Historically, a C program has been composed of: text segment (code), data segment (initialized and uninitialized), stack and heap.



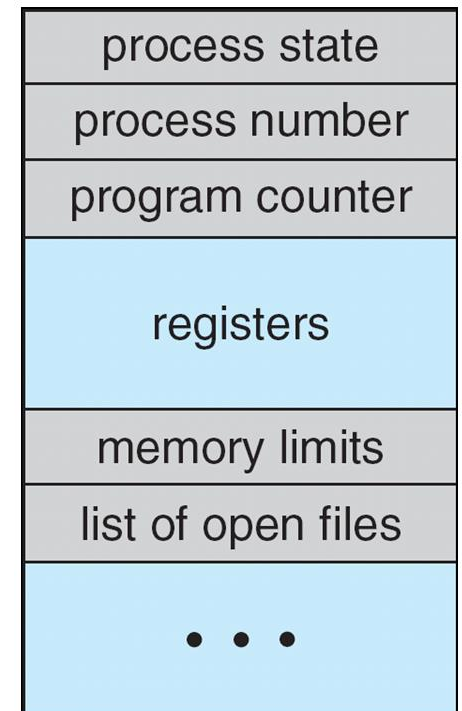
Process State

- The state of a process is defined in part by the current activity of that process. The states can be shown by a process state diagram.
- As a process executes, it changes state (note that these names are generic)
 - ❖ **new**: The process is being created
 - ❖ **ready**: The process is waiting to be assigned to a processor
 - ❖ **running**: Instructions are being executed
 - ❖ **waiting**: The process is waiting for some event to occur
 - ❖ **terminated**: The process has finished execution



Process Control Block (PCB)

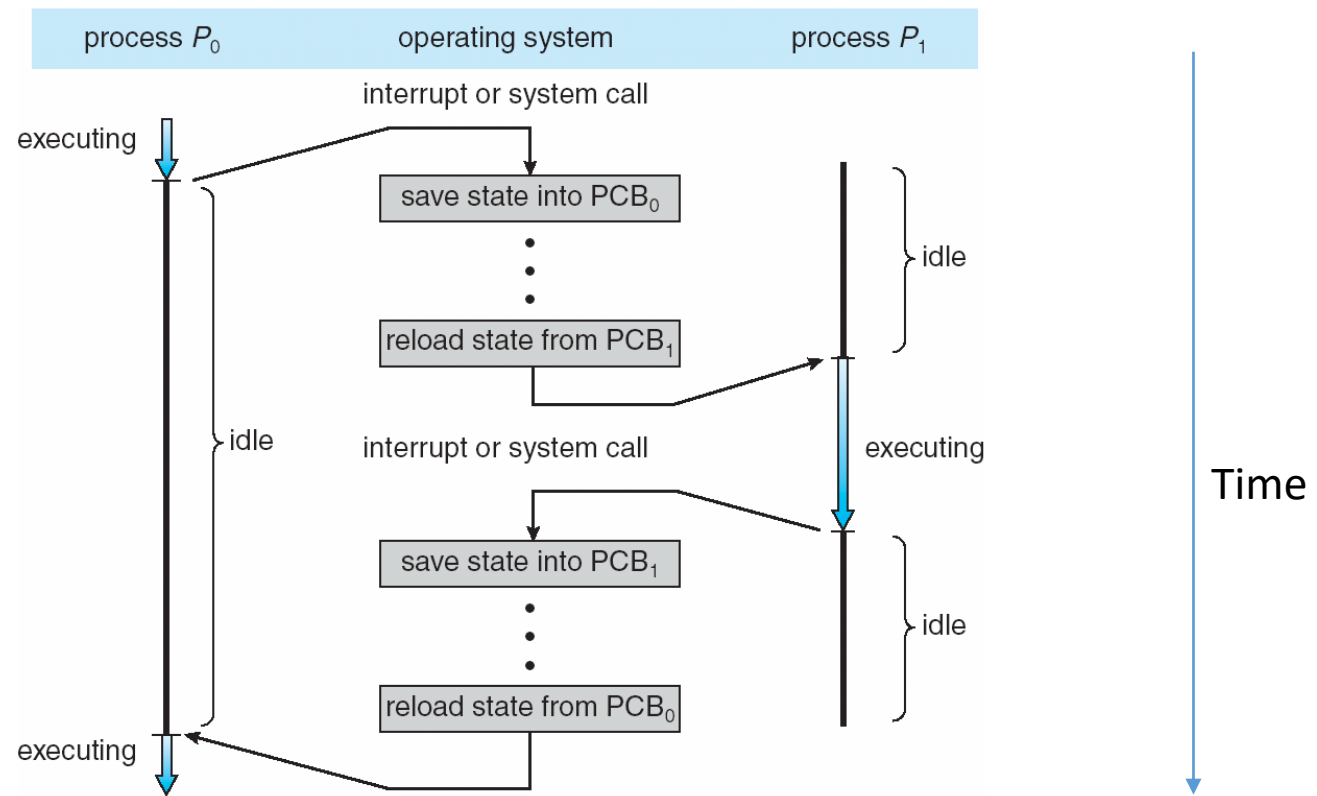
- Each process is represented in the OS by a process control block (PCB).
- The PCB contains many pieces of information associated with a specific process:
 - ❖ Process state (e.g. new, ready, running, halted, ...)
 - ❖ Process number (a unique id number for each process)
 - ❖ Program counter (address of the next instruction to be executed)
 - ❖ CPU registers
 - ❖ CPU scheduling information (e.g. priority)
 - ❖ I/O status information
 - e.g. list of open files, I/O devices allocated
 - ❖ ...



CPU Switch From Process to Process

- When the CPU switches from a process to another one, the state information (CPU registers, PC, ...) must be saved to allow the process to be continued correctly afterward.

A diagram showing one CPU (or CPU core) switching from one process to another process:



Context Switch

- When CPU switches to another process, the system must save the state of the old process (*state save*) and load the saved state for the new process (*state restore*) via a **context switch**
 - ❖ **Context** of a process represented in the PCB
 - including CPU registers, process state, and memory management information
- Context-switch time is overhead
 - ❖ The system does no useful work while switching
 - ❖ Time dependent on hardware support

Threads



- The process model discussed implies that a process is a program that performs a single **thread** of execution.
- Many modern OSs have extended the process concept to allow a process to have multiple threads of execution.

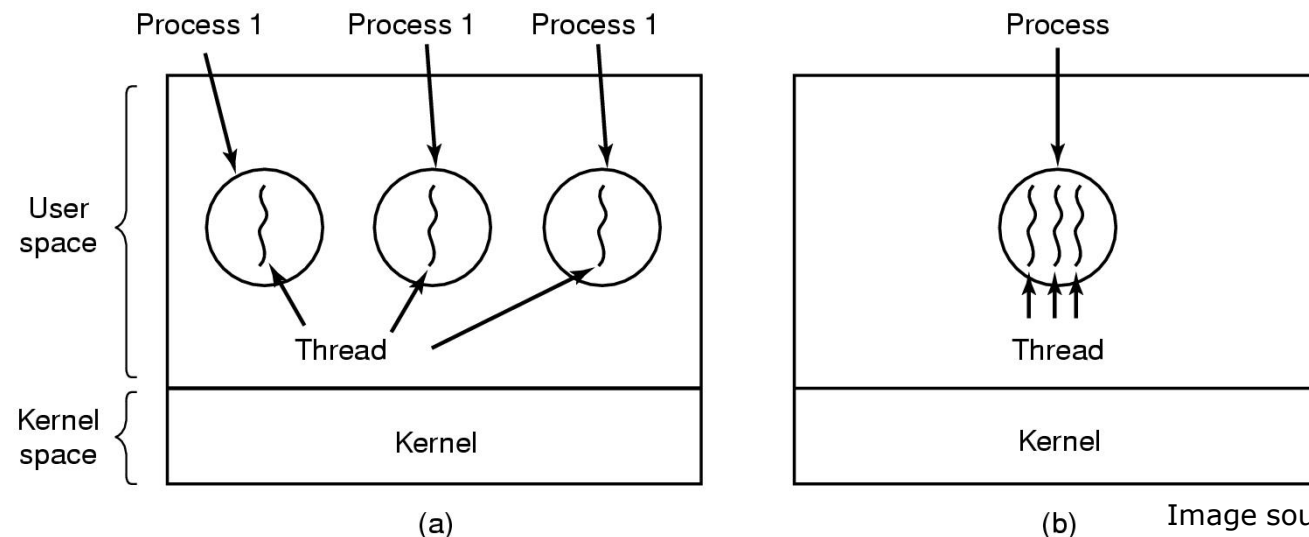
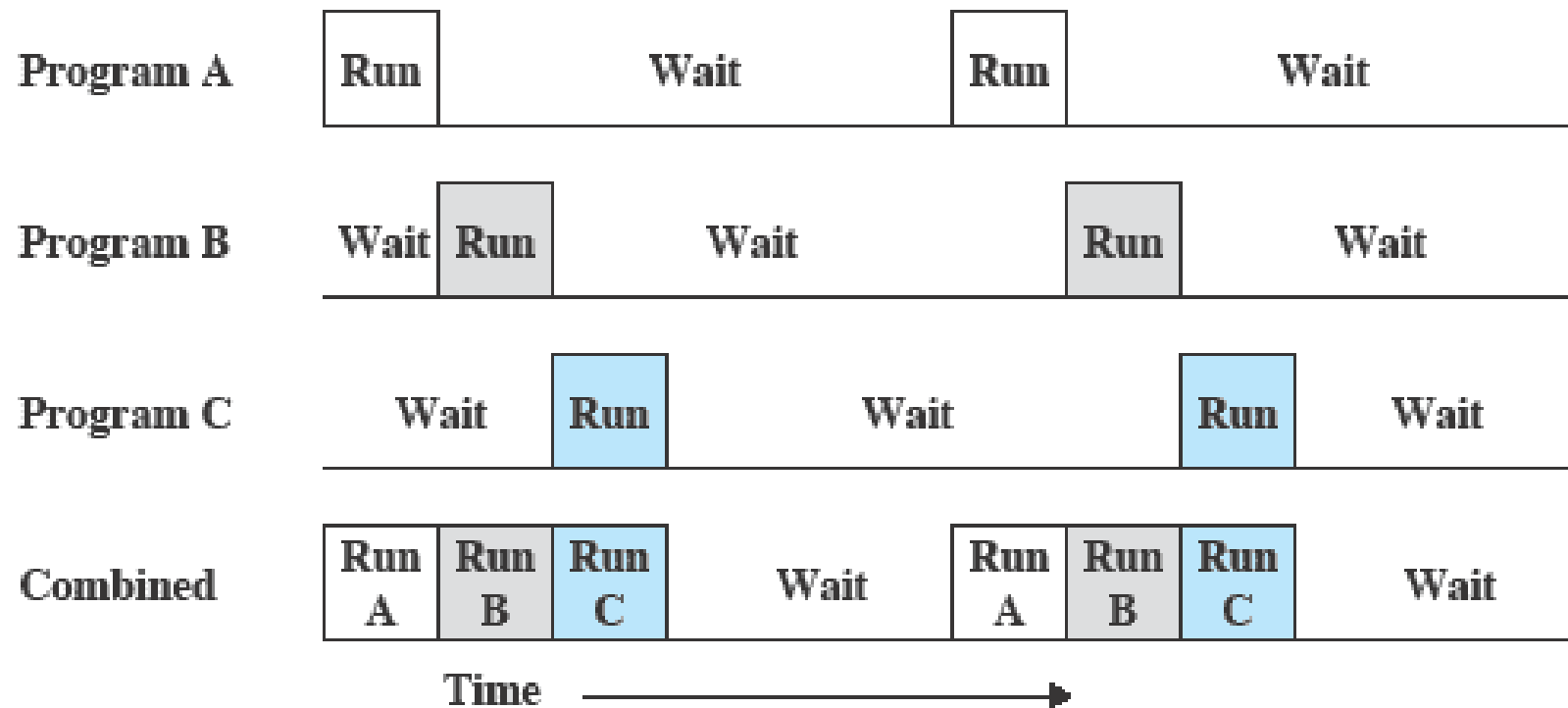


Image source: Tanenbaum's operating systems

- We postpone further discussions on threads to next.

Process Scheduling and time-sharing

- *Objective of time sharing:* to switch a CPU among processes so frequently that the users can interact with each program.



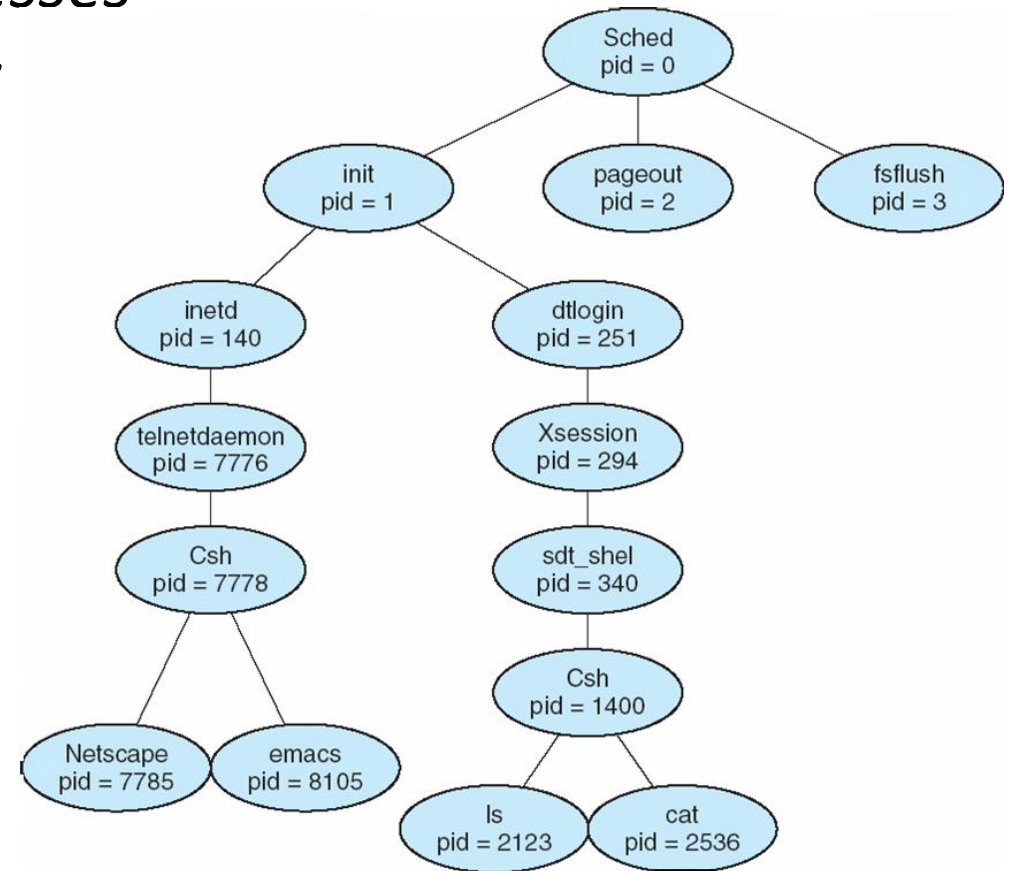
(c) Multiprogramming with three programs

source: Stallings'

Process Creation

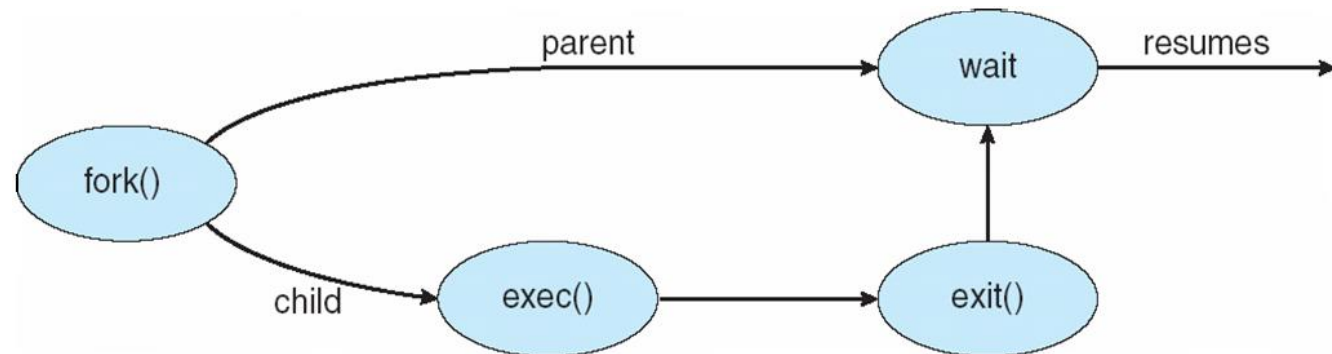
- **Parent** process create **children** processes, which, in turn may create other processes, forming a *tree of processes*
 - ❖ *e.g. a tree of processes on a typical Solaris:*

e.g. use “ps -el” command in macOS or UNIX



Process Creation (cont)

- Generally, a process is identified and managed via a unique **process identifier (pid)**
 - ❖ is an integer number
 - ❖ e.g. used in Windows and UNIX-based OS
- A process will need certain resources (CPU time, memory, files, I/O devices) to accomplish its tasks.
- UNIX's way: two steps
 - ❖ **fork** system call creates new process
 - ❖ **exec** system call is used after a fork to replace the process' memory space with a new program
 - e.g.:



A **system call** is the programmatic way of a program requesting a service from the kernel.

Process Termination

- A process normally terminates when it finished executing its last statement and asks the operating system to delete it (**exit**)
 - ❖ All process' resources are deallocated by the OS
- Parent may terminate the execution of children processes (**abort**), e.g. when:
 - ❖ Task assigned to child is no longer required
 - ❖ If parent is exiting, an operating system may not allow child to continue if its parent terminates (All children terminated - **cascading termination**)

Multiprocess Architecture – Chrome Browser

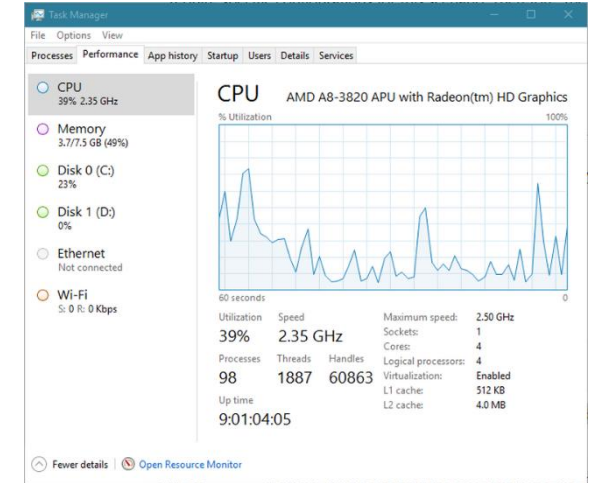
- Web browsers run multiple processes so that if one web site causes trouble, entire browser does not hang or crash
- Google Chrome Browser is multiprocess with 3 categories
 - ❖ **Browser** process manages user interface, disk and network I/O
 - ❖ **Renderer** process renders web pages, deals with HTML, Javascript, new one for each website opened
 - Runs in **sandbox** restricting disk and network I/O, minimizing effect of security exploits
 - ❖ **Plug-in** process for each type of plug-in



OS Tools

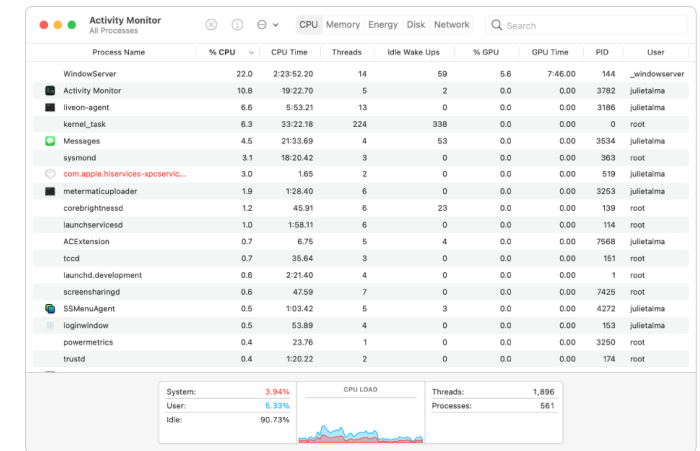
➤ Windows **task manager**

- ❖ use *ctrl+shift+esc* or run *task manager* from the start menu
- ❖ <https://docs.microsoft.com/en-us/cpp/atl/using-task-manager>



➤ macOS **activity monitor**

- ❖ use spotlight to find and run activity monitor
- ❖ <https://support.apple.com/en-ca/guide/activity-monitor/welcome/mac>



References

➤ Windows task manager

- ❖ <https://docs.microsoft.com/en-us/cpp/atl/using-task-manager>
- ❖ [https://en.wikipedia.org/wiki/Task_Manager_\(Windows\)](https://en.wikipedia.org/wiki/Task_Manager_(Windows))

➤ macOS activity monitor

- ❖ <https://support.apple.com/en-ca/guide/activity-monitor/welcome/mac>

➤ Some sections in chapter 3 of Operating Systems Concepts book

Acknowledgement: This set of slides is partly based on the PPTs provided by the Wiley's companion website for the operating system concepts book (including textbook images, when not explicitly mentioned/referenced).

