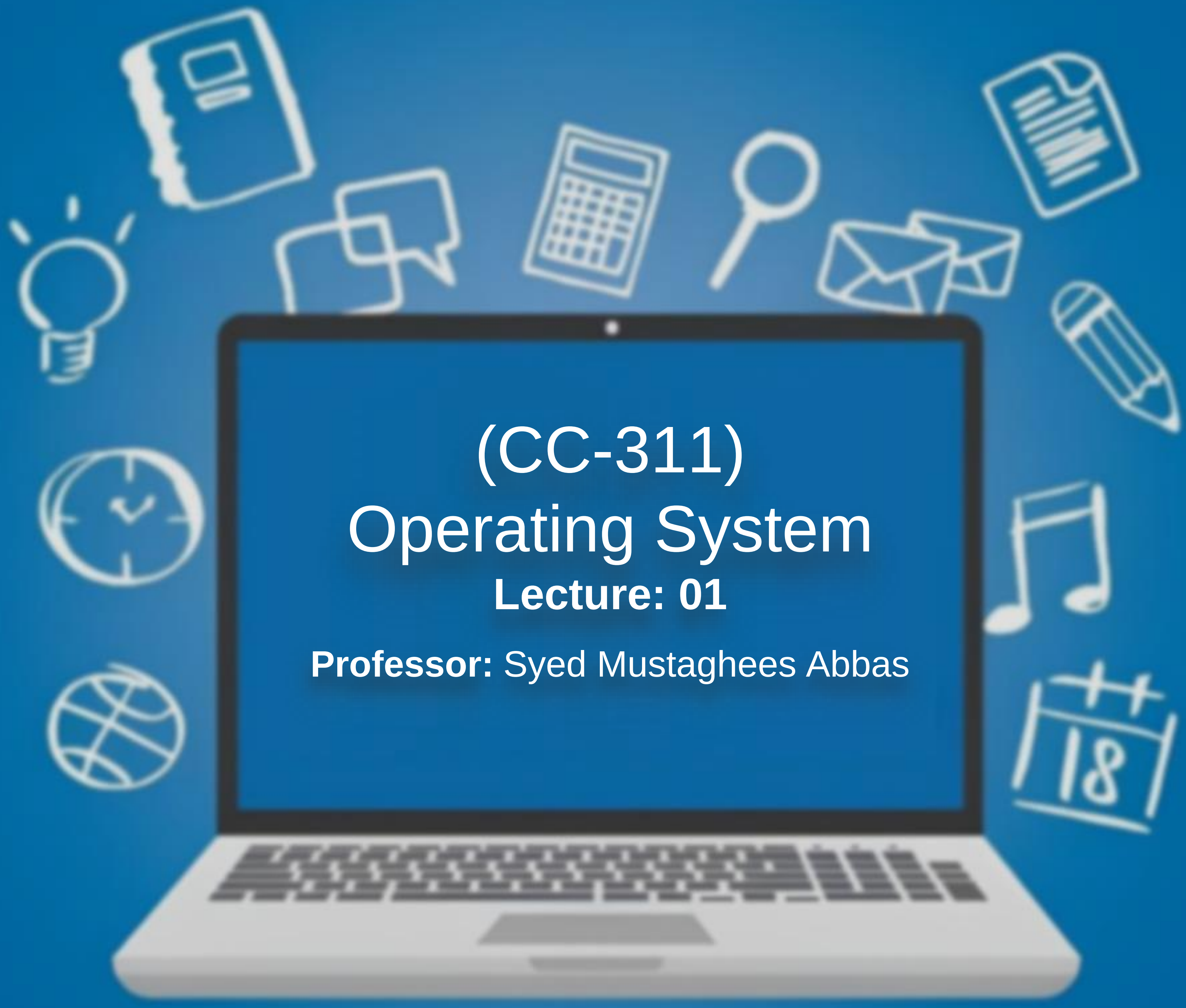


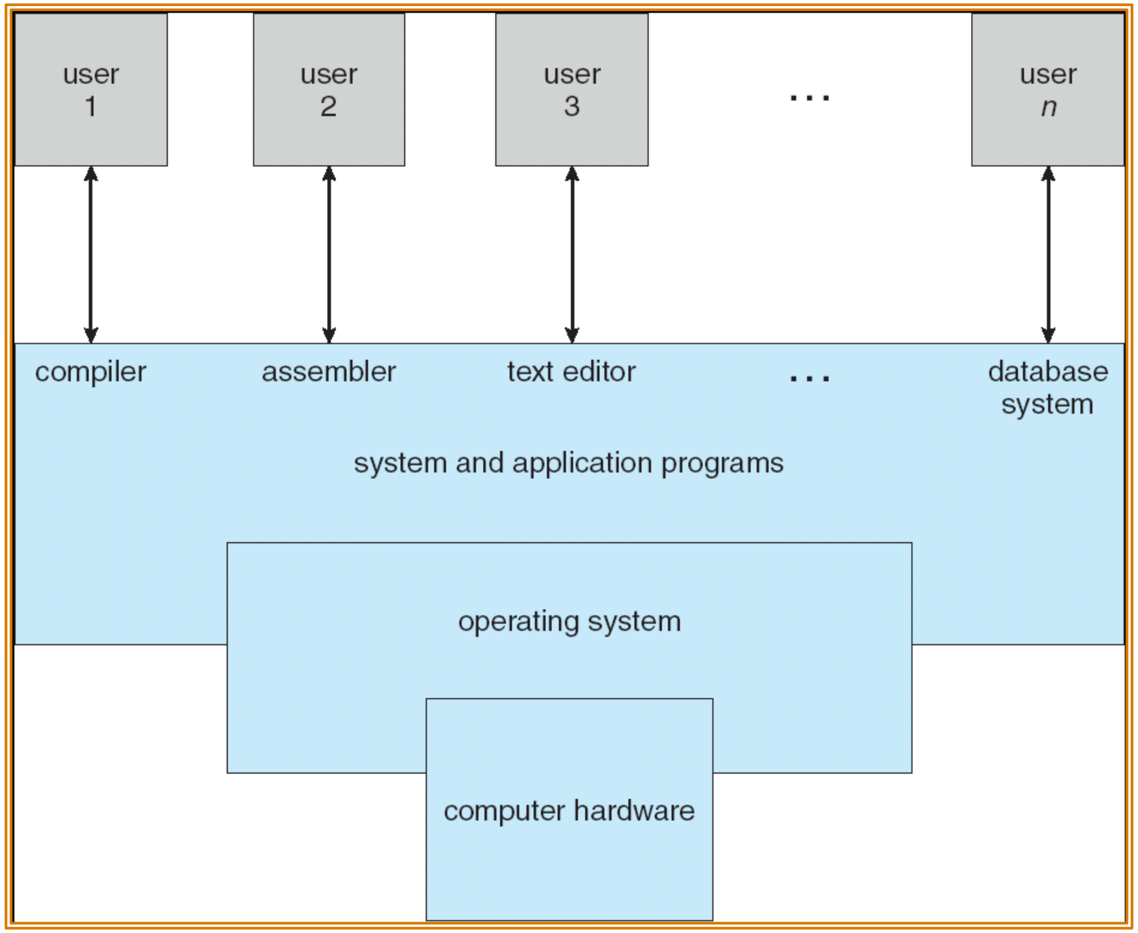
(CC-311) Operating System

Lecture: 01

Professor: Syed Mustaghees Abbas

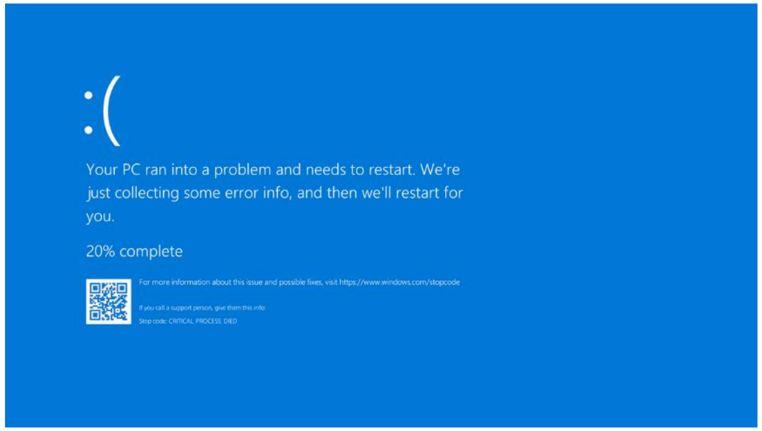
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware.



What is an Operating System?

- ➢ A more common definition is that the operating system is the one program running at all times on the computer (usually called the kernel), with all else being systems programs and application programs. This last definition is the one that we generally follow.
- ➢ **Kernel panic?**
  - ✓ Displays a serious error message when encountering a critical, unrecoverable problem.
  - ✓ System crash, preventing normal operation.
  - ✓ like "blue screen of death" in Windows.



:(

Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

20% complete

For more information about this issue and possible fixes, visit https://www.windows.com/stopcode

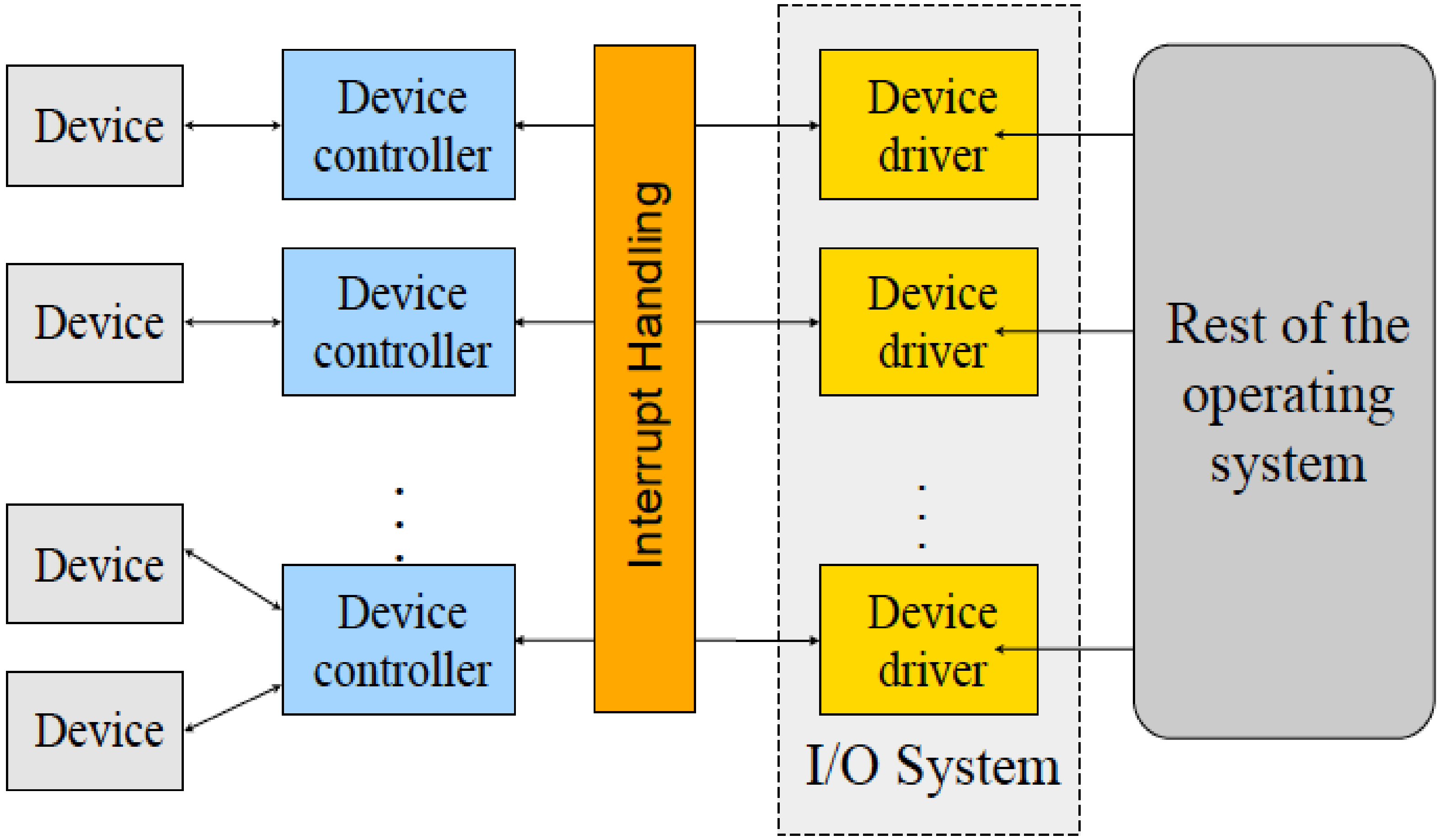
If you call a support person, give them this info:

Stop code: CRITICAL PROCESS DIED

Device Controller and Device Driver

- A device controller is a physical part of a computer system that makes sense of the signals going to, and coming from the CPU for the different input/output devices.
- The Device Controller receives the data from a connected device and stores it temporarily in some special purpose registers (i.e. local buffer) inside the controller. Then it communicates the data with a Device Driver.
- For each device controller there is an equivalent device driver which is the standard interface through which the device controller communicates with the Operating Systems through Interrupts.
- Device controller is a hardware whereas device driver is a software.

Device Controller and Device Driver



Interrupts vs Polling

- When interacting with a device controller, the CPU can wait for a response by polling the status register(s), i.e., by periodically checking whether the status of the device has changed. This is known as **Polling**.
- Problem with polling: The CPU is busy waiting for some event to happen. CPU utilization will be low. So Interrupts are used by devices for asynchronous event notification.
- Interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.

Interrupts vs Polling

- When an interrupt is fired, the CPU jumps to a predefined position in the kernel's address space and executes an interrupt handler. When an interrupt occurs, the CPU can start reading data from the device controller's data registers.
- Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt.
- Device controller informs CPU that it has finished its operation by causing an interrupt.

Hardware vs Software Interrupts

- A hardware interrupt is an electronic alerting signal sent to the processor from an external device, either a part of the computer itself such as a disk controller or an external peripheral.
- A software interrupt (also called a monitor call) is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed.
- Software interrupt is often called a trap or exception and is used for errors or events occurring during program execution that are exceptional enough that they cannot be handled within the program itself.

Dual mode of operation

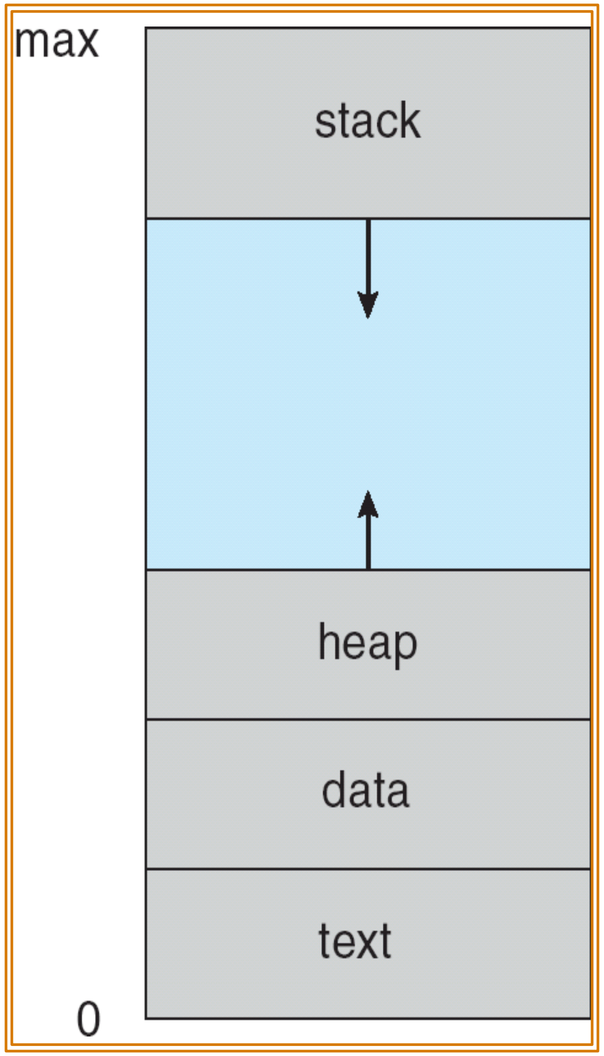
- Dual-mode operation allows OS to protect itself and other system components
- **User mode and kernel mode**
- Mode bit provided by hardware
- Provides ability to distinguish when system is running user code or kernel code
- Some instructions designated as privileged, only executable in kernel mode
- System call changes mode to kernel, return from call resets it to user
- For example, if an I/O instruction is executed under user mode, the hardware does not execute it, although identifies it as an illegal execution and traps it to the operating system.

What is a Process?

- ➢ A process can be thought of as a program in execution.
- ➢ Needs certain resources—such as CPU time, memory, files, and I/O devices—to accomplish its task.
- ➢ It is a unit of work in most systems.
- ➢ Systems consist of a collection of processes: Operating-system processes execute system code or system mode, and user processes execute user code or user mode.
- ➢ The operating system is responsible for the following activities in connection with process:
  - ✓ Creation and Deletion of both user and system processes
  - ✓ The scheduling of processes
  - ✓ Provision of mechanisms for synchronization communication
  - ✓ Deadlock handling for processes.

Process Division

- Process memory is divided into four sections.
  - ✓ The **text section** comprises the compiled program code, read in from non-volatile storage when the program is launched.
  - ✓ The **data section** stores global and static variables, allocated and initialized prior to executing main.
  - ✓ The **heap** is used for dynamic memory allocation, and is managed via calls to new, delete, malloc, free, etc.
  - ✓ The **stack** is used for local variables. Space on the stack is reserved for local variables when they are declared (at function entrance or elsewhere, depending on the language), and the space is freed up when the variables go out of scope. It is also used for function return values.

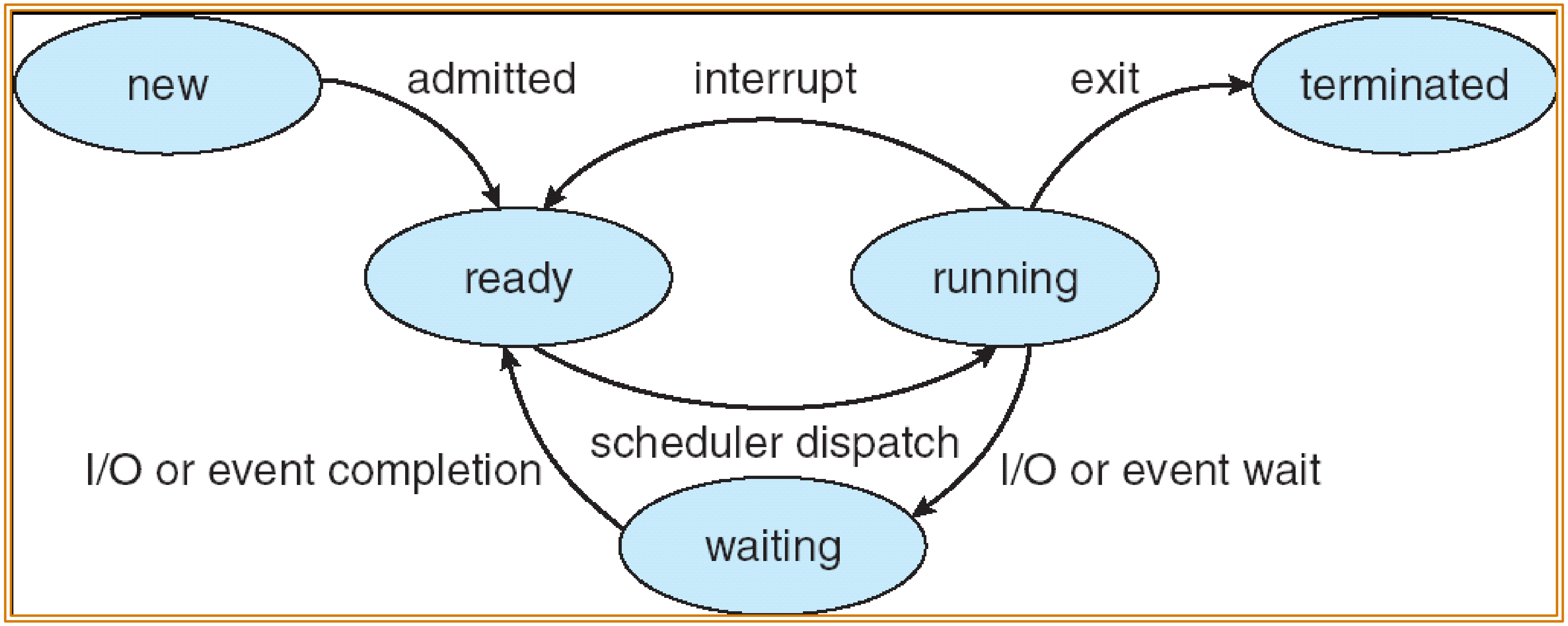


Stack Overflow

- Note that the stack and the heap start at opposite ends of the process's free space and grow towards each other. If they should ever meet, then either a **stack overflow** error will occur, or else a call to new or malloc will fail due to insufficient memory available.

Process states

- As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. Each process may be in one of the following states:



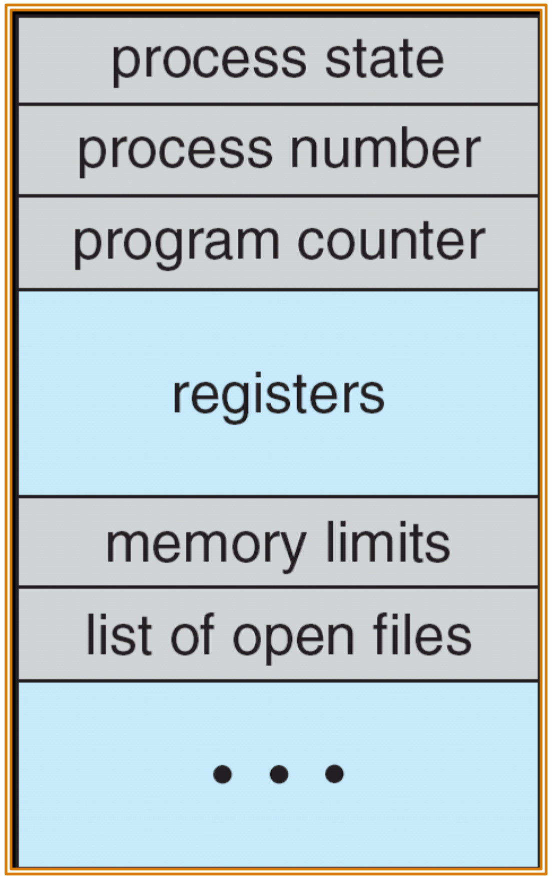
Process states

- **New:** The process is in the stage of being created.
- **Ready:** The process has all the resources available that it needs to run, but the CPU is not currently working on this process's instructions.
- **Running:** The CPU is working on this process's instructions.
- **Waiting:** The process cannot run at the moment, because it is waiting for some resource to become available or for some event to occur.

  For example: The process may be waiting for keyboard input, disk access request, inter-process messages, a timer to go off, or a child process to finish.
- **Terminated:** The process has completed.

Process Control Block (PCB)

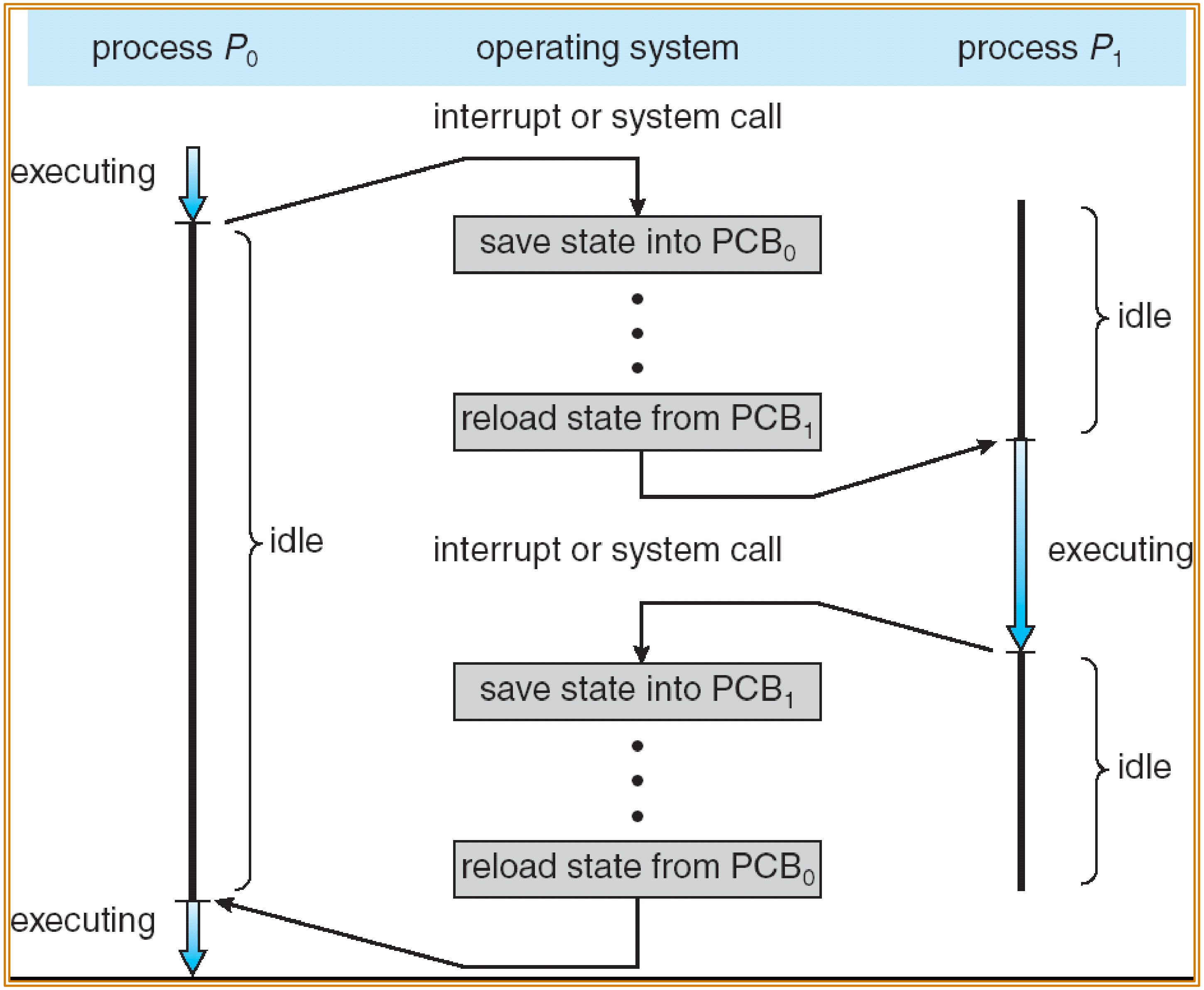
- When processes are swapped out of memory and later restored, additional information must also be stored and restored. Key among them are the program counter and the value of all program registers.
- For each process there is a Process Control Block, PCB, which stores the following (types of) process- specific information.



Process Control Block (PCB)

- **Process State:** Running, waiting, etc., as discussed above.
- **Process ID**, and parent process ID.
- **CPU registers and Program Counter:** These need to be saved and restored when swapping processes in and out of the CPU.
- **CPU-Scheduling information:** Such as priority information and pointers to scheduling queues.
- **Memory-Management information:** *e.g.* page tables or segment tables.
- **Accounting information:** user and kernel CPU time consumed, account numbers, limits, etc.
- **I/O Status information:** Devices allocated, open file tables, etc.

CPU Switch From Process to Process

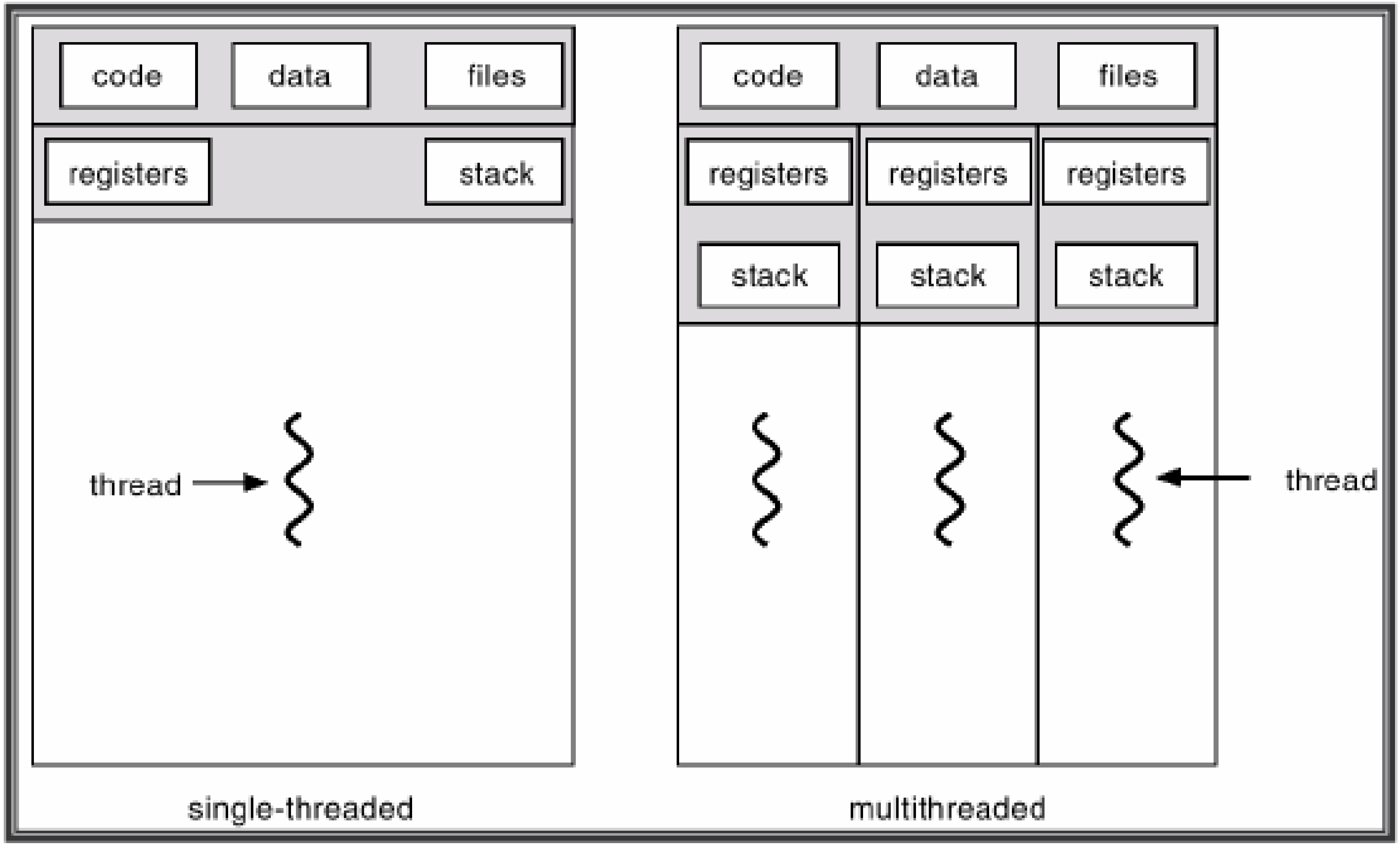


Thread

- The process model discussed so far has implied that a process is a program that performs a single thread of execution.
- For example, when a process is running a word-processor program, a single thread of instructions is being executed. This single thread of control allows the process to perform only one task at one time.
- The user cannot simultaneously type in characters and run the spell checker within the same process.

  For example: Many modern operating systems have extended the process concept to allow a process to have multiple threads of execution and thus to perform more than one task at a time.
- A thread is a light-weight process.
- Thread is a lighter than Process but heavier than the child process
- Thread and Child Process are not same. Thread is a lighter than Process but heavier than the Child Process.

Single and Multithreaded Processes



Benefits of Threads

- **Responsiveness**
- **Resource Sharing.** (Threads in a process share the same address space)
- **Economy.** (Operating System using thread concept are less expensive)
- **Utilization of MP(Micro Processor) Architectures**. (Multiple threads can run on multiple CPUs in parallel)