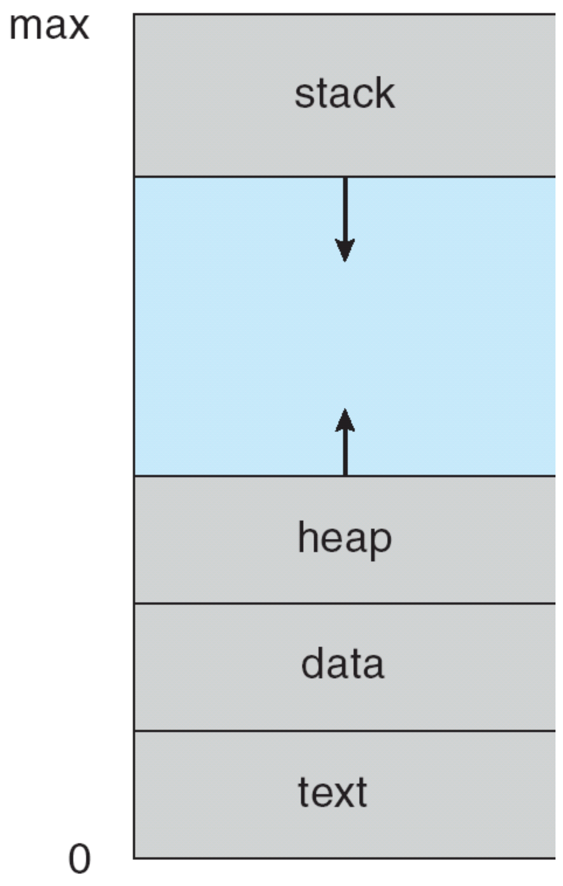
Chapter 3: Processes

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, and user processes execute user code.
- The operating system is responsible for the following activities in connection with process: the creation and deletion of both user and system processes; the scheduling of processes;and the provision of mechanisms for synchronization communication,and deadlock handling for processes.

Process (cntd)

- 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.

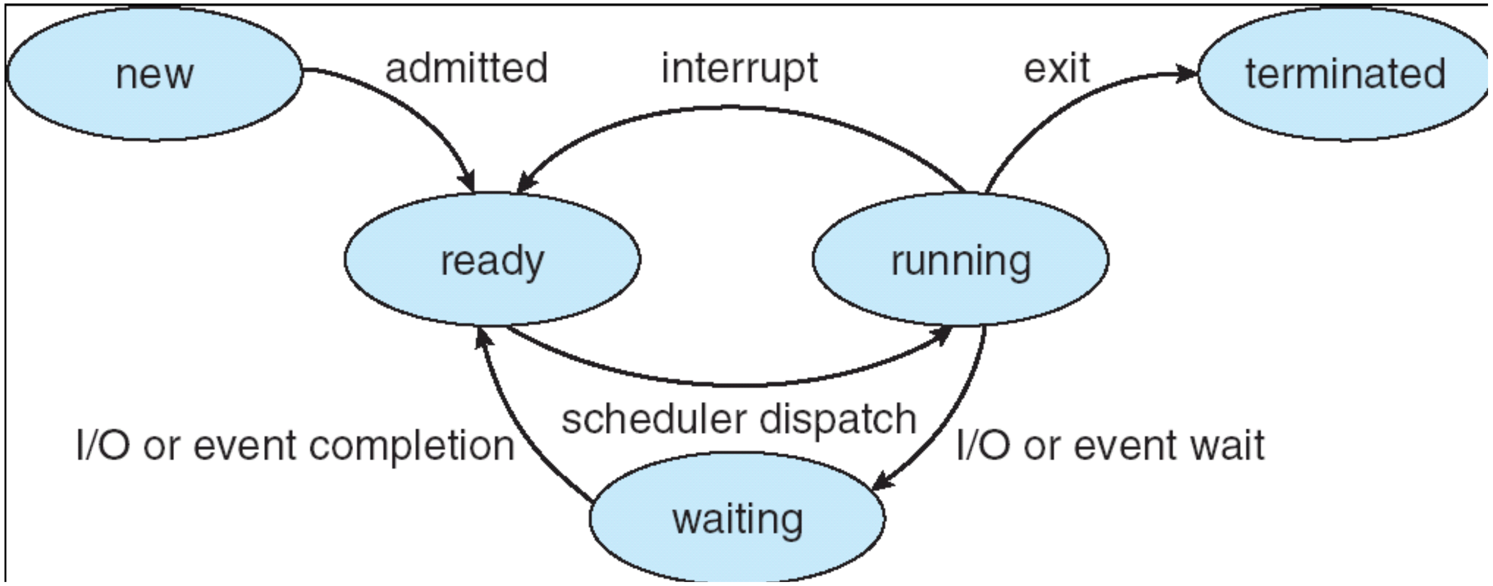


Process (cntd)

- 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:

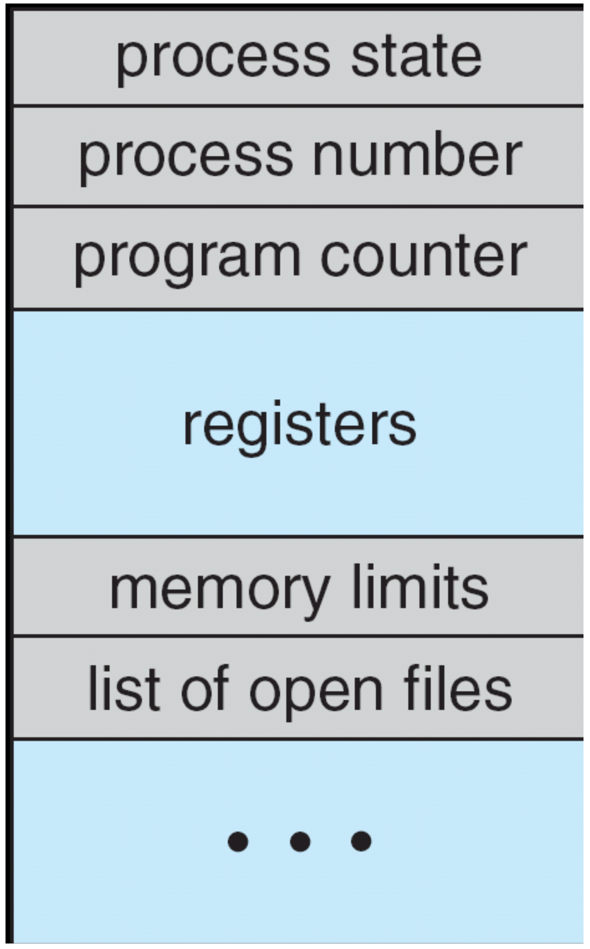


Process states

- **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

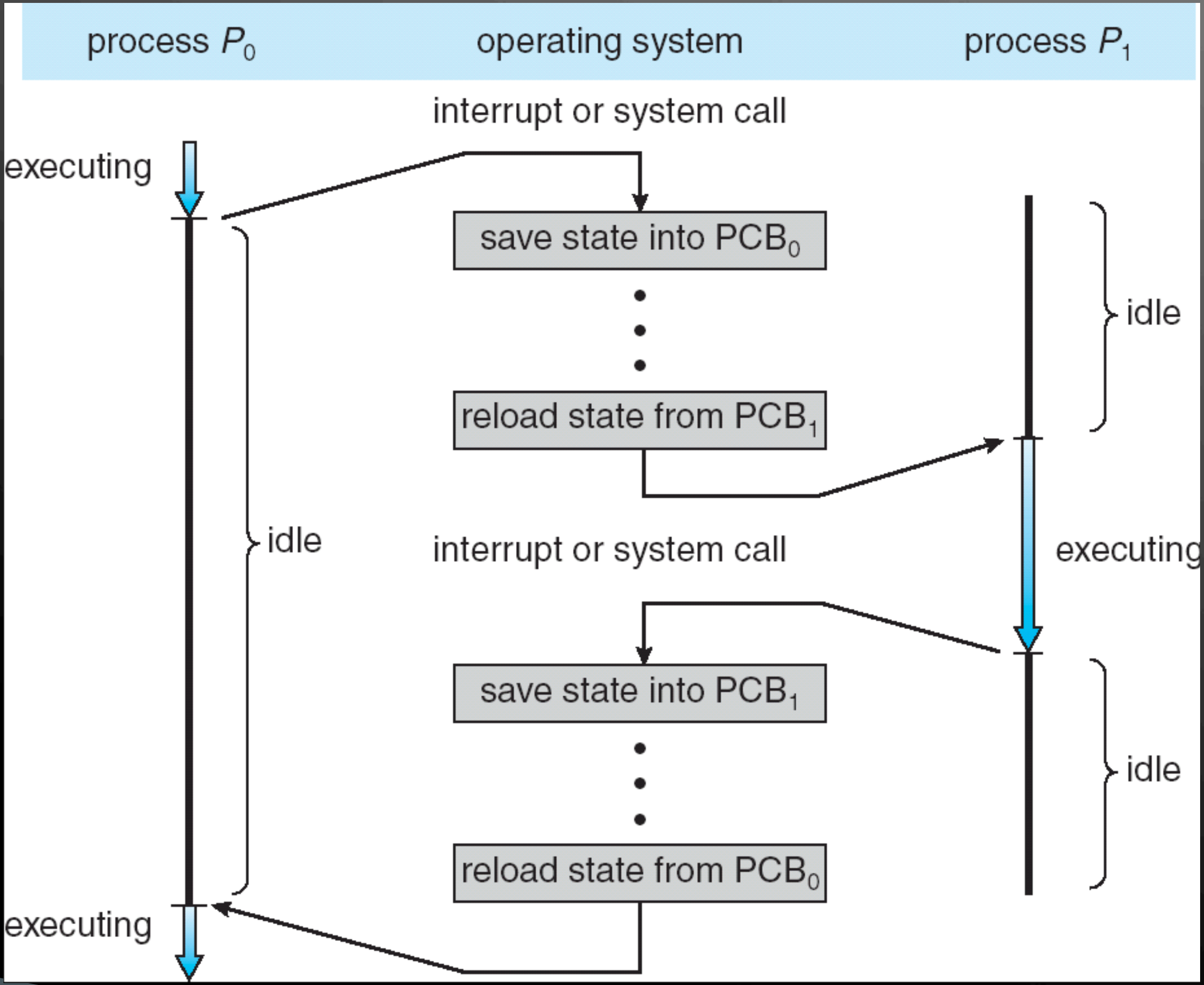
- When processes are swapped out of memory and later restored, additional information must also be stored and restored. Key among them are the program counter and the value of all program registers.
- For each process there is a Process Control Block, PCB, which stores the following (types of) process-specific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



Process Control Block

- **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

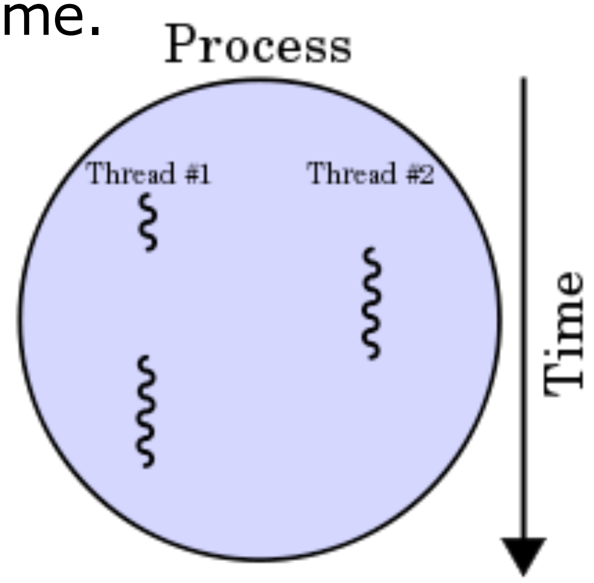


Thread

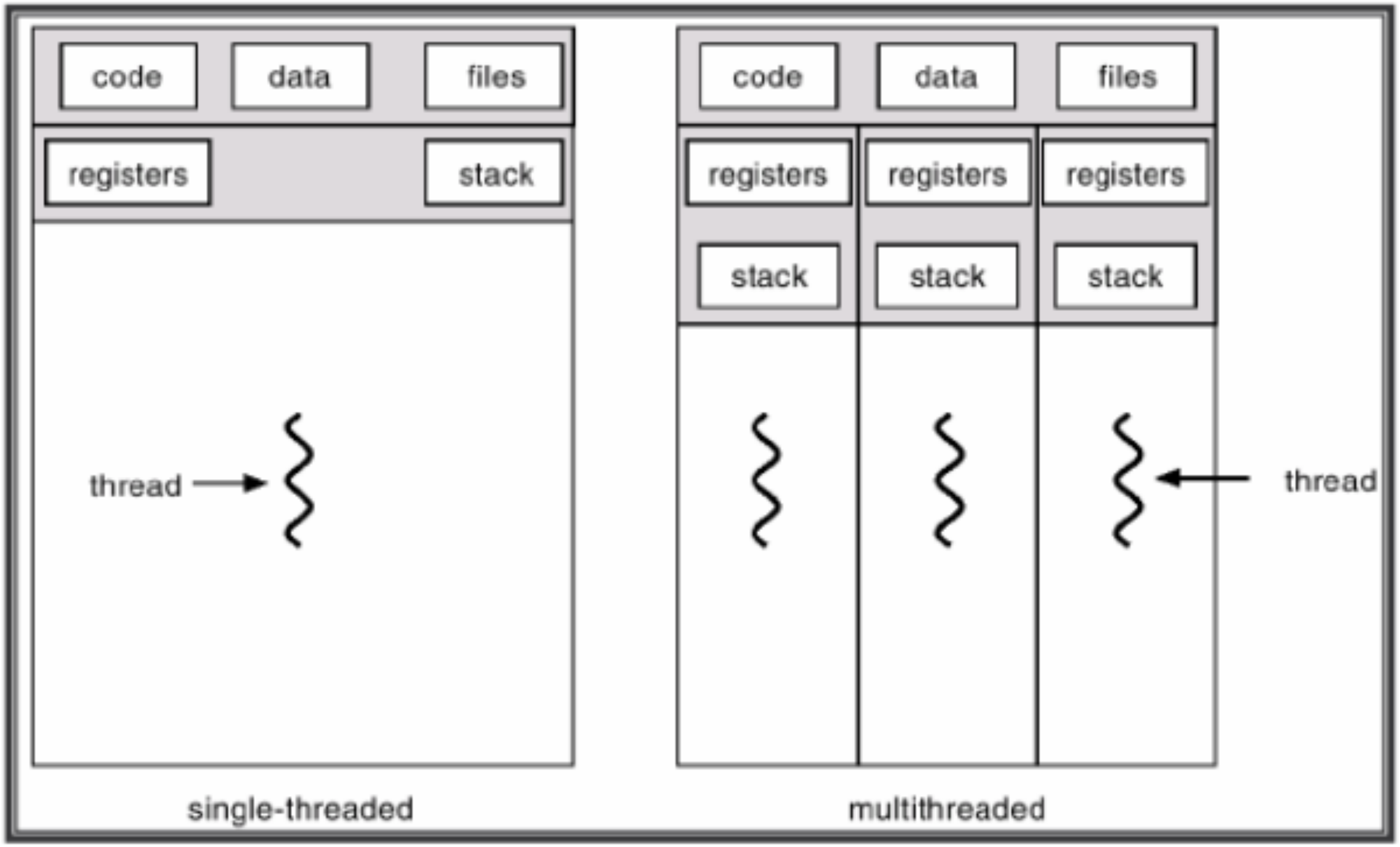
The process model discussed so far has implied that a process is a program that performs a single thread of execution.

- 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.



Single and Multithreaded Processes



Benefits of Threads

- Responsiveness
- Resource Sharing.(Threads in a process share the same address
- space)
- Economy
- Utilization of MP Architectures.(Multiple threads can run on
multiple CPUs in parallel)