Process Abstraction

References: OSTEP Ch-4

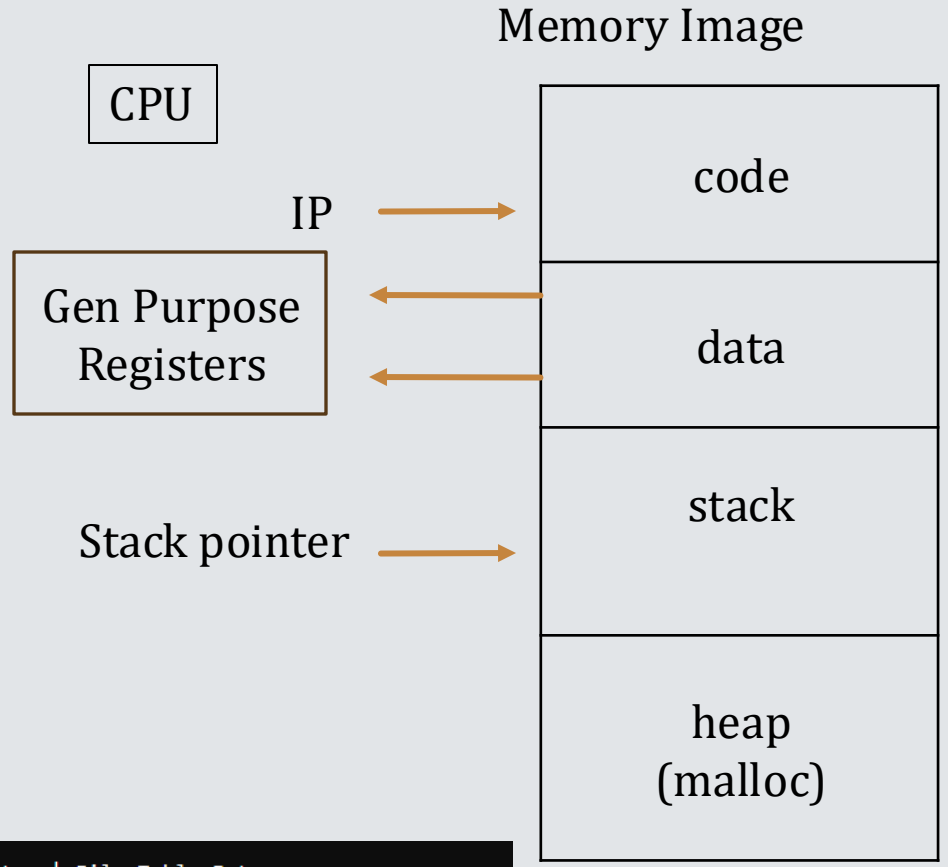
Operating System Process Abstraction

- Definition of a process: The abstraction provided by the OS of a running program.
- OS provide
  - Timesharing → give each process an illusion that it is using CPU/resource independently.
  - Scheduling → method of executing task, in what order, or when to switch.
- Timesharing
  - Virtualization of running multiple process
    - On a CPU: a user listens to music, browse the web, etc.
- Schedulers
  - Selects one of the active process to execute on a CPU
  - Policy : Decides which process to run
  - Mechanism: how to "context switch" among processes
    - → switch betⁿ processes.

A process consist of :-

A.exe will have a unique PID when executed.

- Process Identification Number (PID)- A unique Identifier
- Memory Image of a Process
  - Static Memory- Code and data
  - Dynamic Memory - Stack and Heap
- CPU state or context can be represented by CPU registers
  - Program Counter / Instruction Pointer
  - General Purpose registers to hold the current operands
  - Stack pointers (esp and ebp in x86)
- File descriptor → it is an integer in the FDT
  - Pointers to open files and devices
    - Eg. Standard input/ output (STDIN, STDOUT, STDERR)

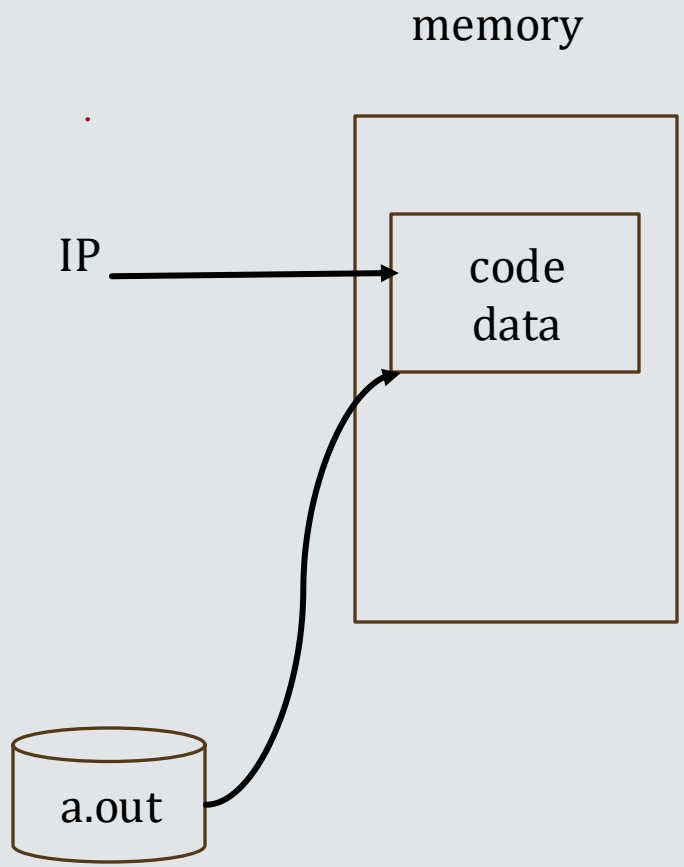



```
File Descriptor | File Table Entry
--------------------------------
      0         | Stdin (keyboard input)
      1         | Stdout (screen output)
      2         | Stderr (error output)
      3         | myfile.txt (opened by the program)
      4         | socket connection
```

→ treated as file.

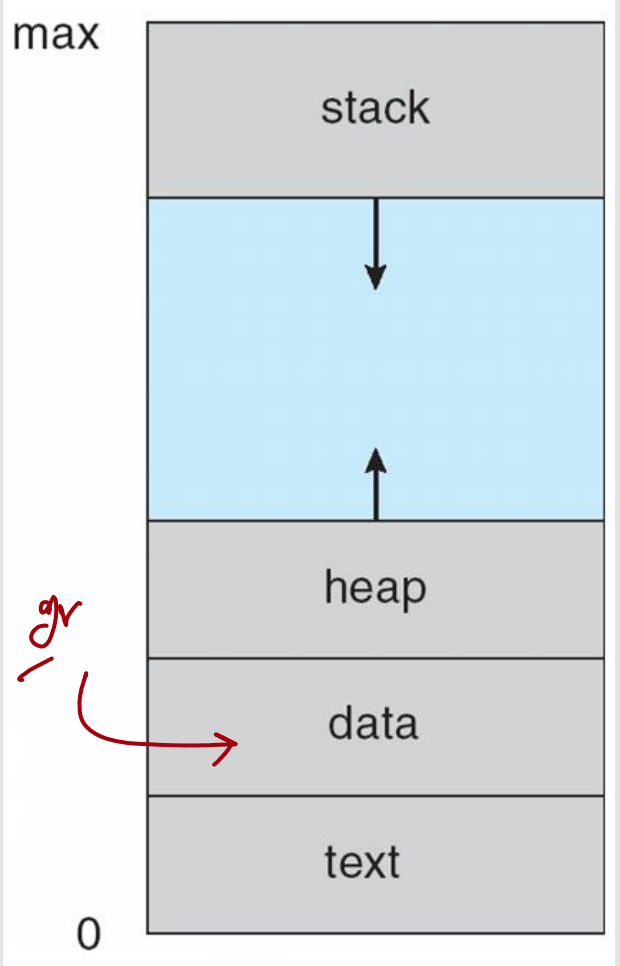
Operating System: Creating a Process

- OS allocates memory and create the memory image of the process
  - The coode and data stored in secondary memory (HDD/SDD) is loaded into the memory (RAM)
  - During execution, the stack and heap are created accordingly
- Open the basic files to facilitates the input output
  - STDIN
  - STDOUT
  - STDERR
- Initialize the CPU Registers
  - PC/IP to point to the first instruction in the process



Summary: Process Concepts

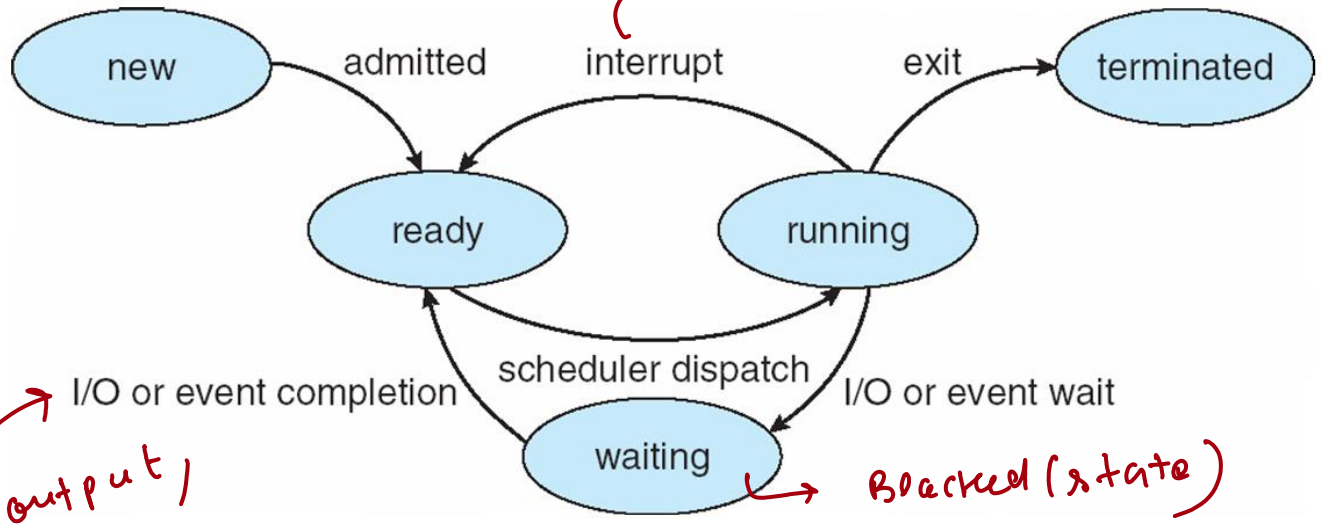
- One program can have several processes
- **Process** has multiple parts
  - The program code, also called **text section**
  - Current activity - **program counter**, registers
  - **Stack** containing temporary data like function parameters, return addresses, local variables
  - **Data section** containing global variables
  - **Heap** -dynamically allocated memory during run time

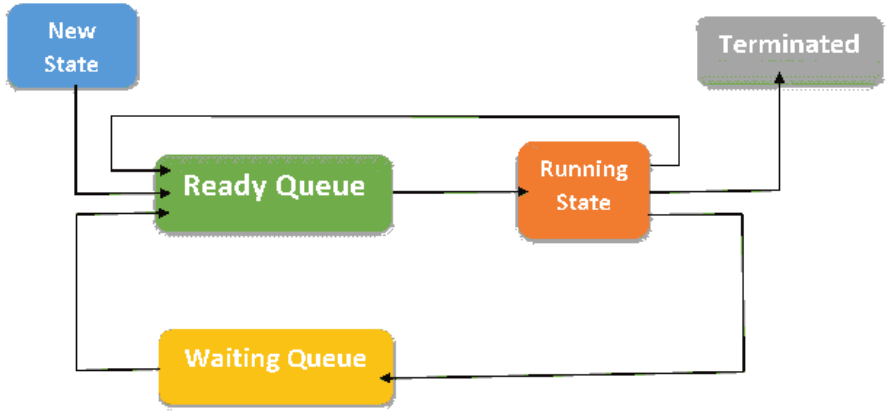


Different state of a Process

- Running: currently executing on CPU
- Ready: waiting to be scheduled
- Blocked/Waiting: suspended, not ready to run
  - Why? Waiting for some event, e.g., process issues a read from disk
  - When is it unblocked? Disk issues an interrupt when data is ready
- New: being created, yet to run
- Dead/Terminated

Process State Diagram





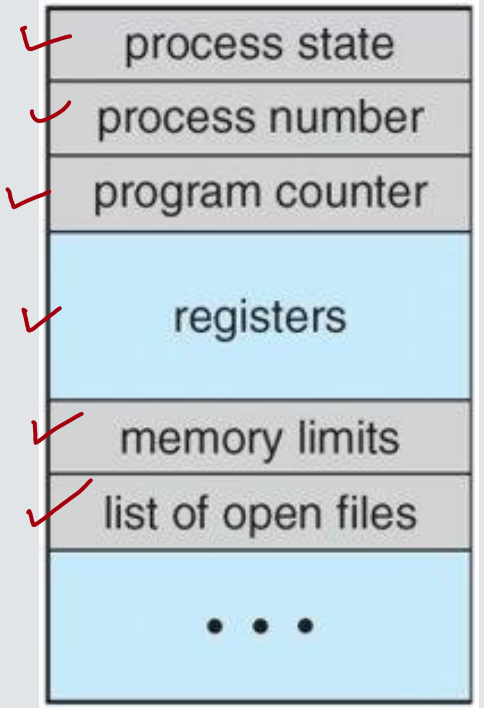
- **new**: The process is being created
- **running**: Instructions are being executed
- **blocked/waiting**: The process is waiting for some event to occur
- **ready**: The process is waiting to processor assignment.
- **dead/terminated**: The process has finished execution

Example: Tracing Process State: CPU and I/O

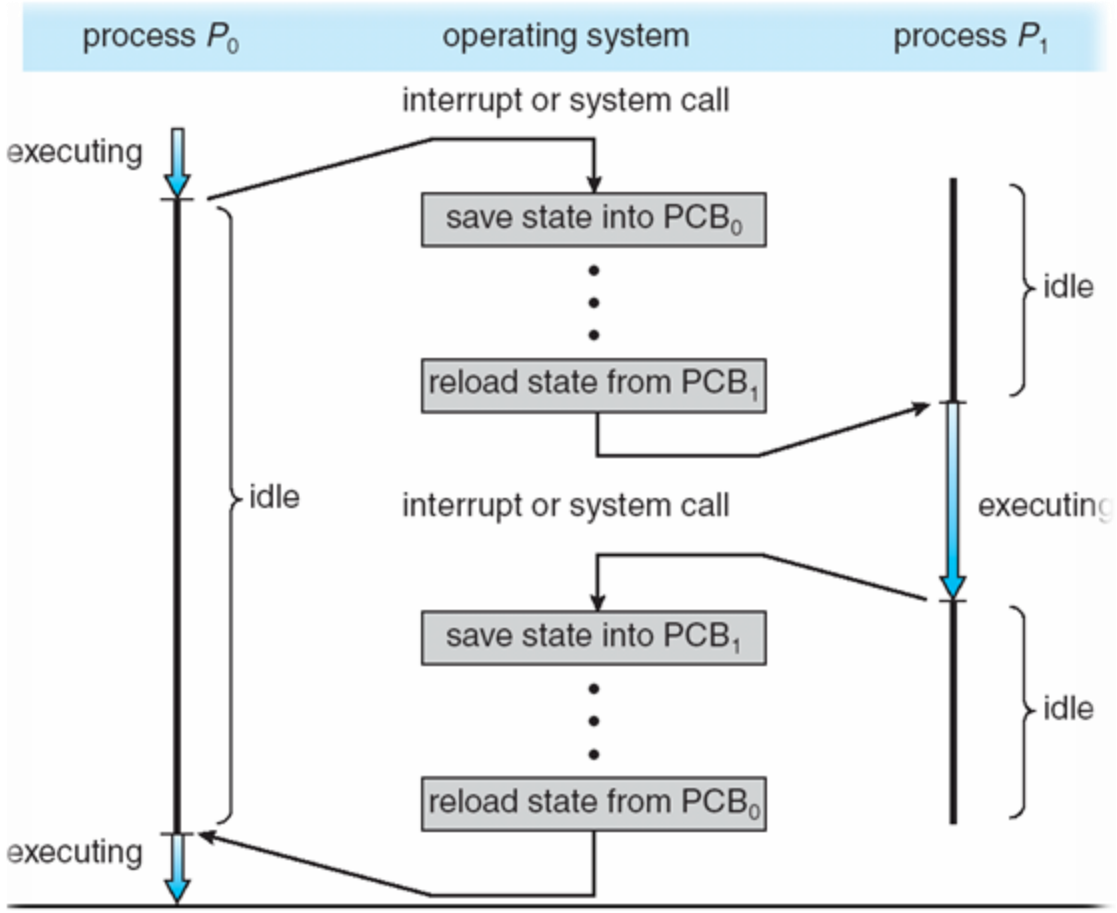
Timesteps	P0	P1	P2	Remarks
1	Running	Ready	Ready	P0 is running, P1 and P2 are wairing in the ready queue
2	Running	Ready	Ready	
3	Running	Ready	Ready	PO initiates I/O
4	Blocked/Waiting	Running	Ready	P0 is blocked or waiting for I/O to complete so P1 is running.
5	Blocked/ Waiting	Running	Ready	P1 initiates I/O
6	Blocked/ Waiting	Blocked/ Waiting	Running	P0 and P1 are both blocked, so P2 is running
7	Ready	Blocked/ Waiting	Running	P0 I/O is done
8	Ready	Blocked/ Waiting	Running	P2 finishes execution
9	Running	Ready	-	P1 I/O done, and P0 finishes execution
10	-	Running	-	

PCB: OS Data Structures

- OS maintains a data structure (e.g., list) of all active processes
- Information about each process is stored in a process control block (PCB)
  - Process state – running, waiting, etc
  - Program counter – location of instruction to next execute
  - CPU registers – contents of all process-centric registers
  - CPU scheduling information- priorities, scheduling queue pointers
  - Memory-management information – memory allocated to the process
  - Accounting information – CPU used, clock time elapsed since start, time limits
  - I/O status information – I/O devices allocated to process, list of open files



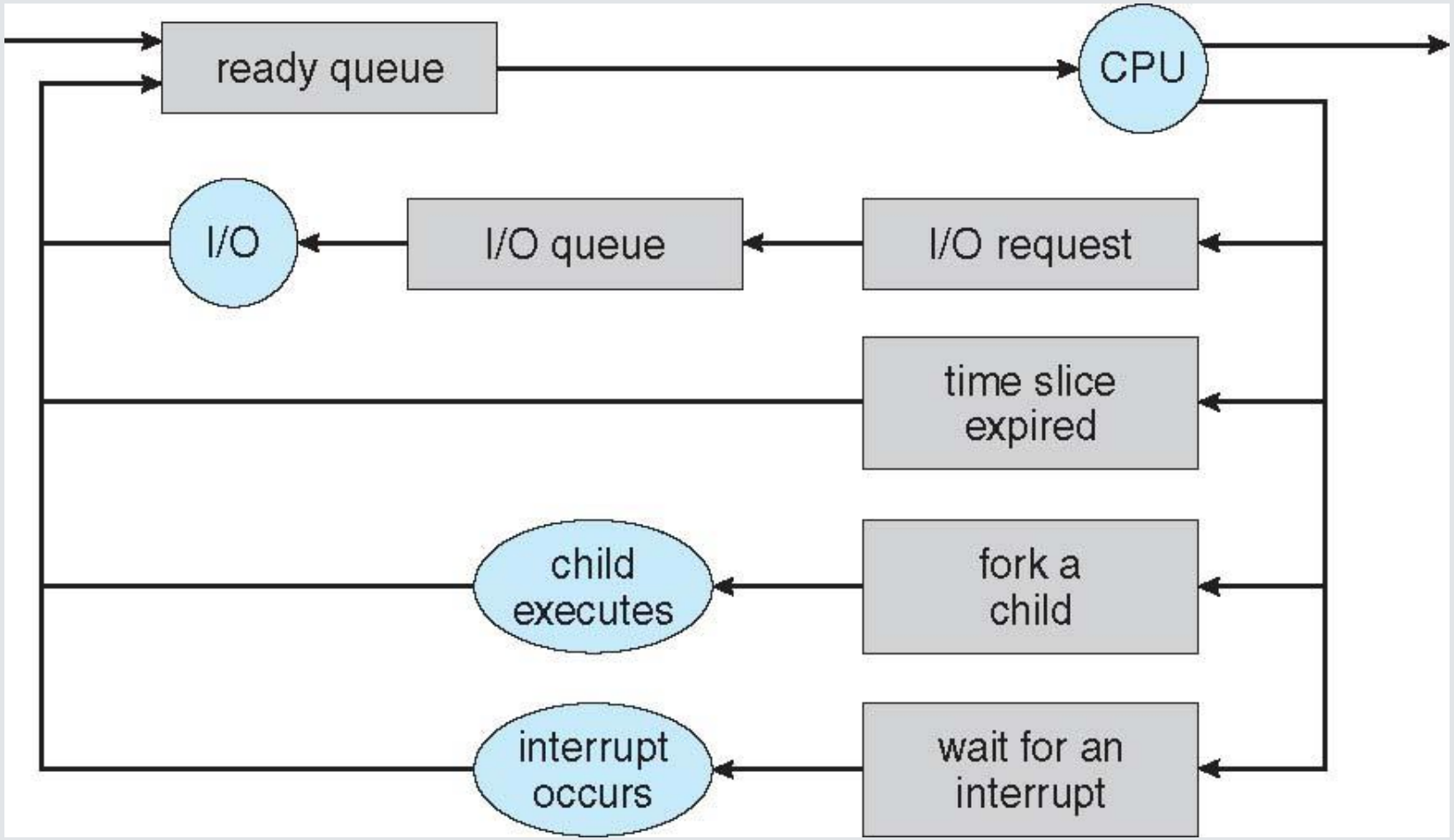
Context Switch From One Process to Another



Process Scheduling

- Maximize CPU use, quickly switch processes onto CPU for time sharing
- **Process scheduler** selects among available processes for next execution on CPU
- Maintains **scheduling queues** of processes
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Device queues** – set of processes waiting for an I/O device
  - Processes migrate among the various queues

Representation of Process Scheduling



Schedulers

- **Short-term scheduler (CPU scheduler)** – selects which process should be assigned to CPU for execution
  - Short-term scheduler is invoked frequently
- **Long-term scheduler (Job scheduler)** – selects which processes should be brought into the ready queue (RAM)
  - Long-term scheduler is invoked less frequently
  - It controls the **degree of multiprogramming**
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts
  - Long-term scheduler strives for good ***process mix***

Thank You