Introduction to Processes

01/21/2021

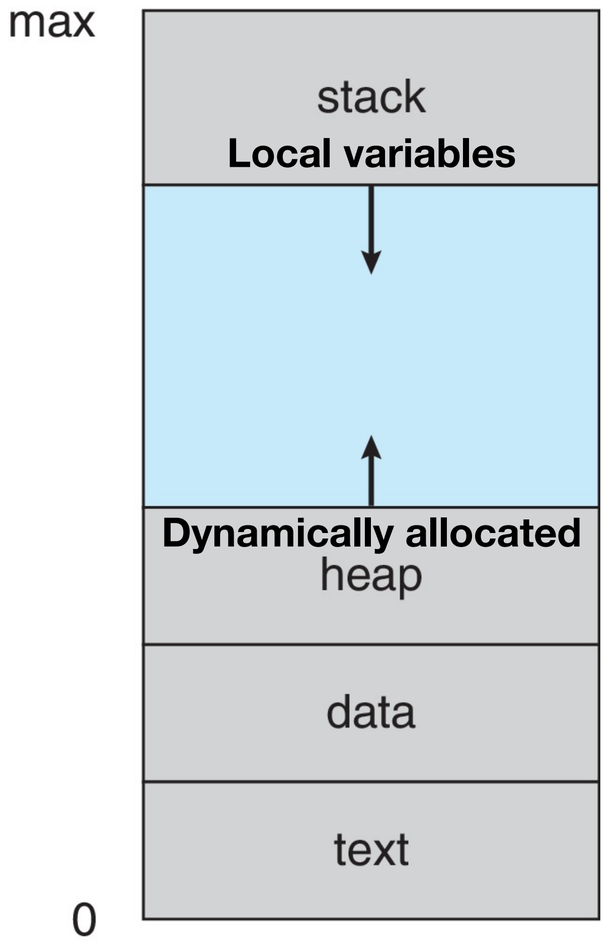
Professor Amanda Bienz

CPU Virtualization

- Provides illusion of many CPUs
- **Time sharing:** Running one process, then stopping it and running another
  - Potential cost is *performance*

A Process

- A process is a **running program**
- A process is comprised of:
  - Virtual memory (address space)
    - Instructions
    - Data section
  - Registers
    - Program counter : register that holds the address of the instruction being executed
    - Stack pointer : register that stores address of last program request in stack



Process API

- **Create:** create a new process to run a program
- **Destroy:** halt a runaway process
- **Wait:** wait for a process to stop running
- **Miscellaneous control:** some method to suspend a process and then resume it
- **Status:** get some status information about a process

Process Creation

- **Load a program code into memory** (into address space of the process)
  - Programs initially reside on disk in an executable format
  - OS performs the loading process **lazily**
    - Loading pieces of code or data one as they are needed during the program
- **The program's run-time stack is allocated**
  - Use the stack for *local variables, function parameters,* and *return address*
  - Initialize the stack with arguments (argc, argv)

Process Creation (cont'd)

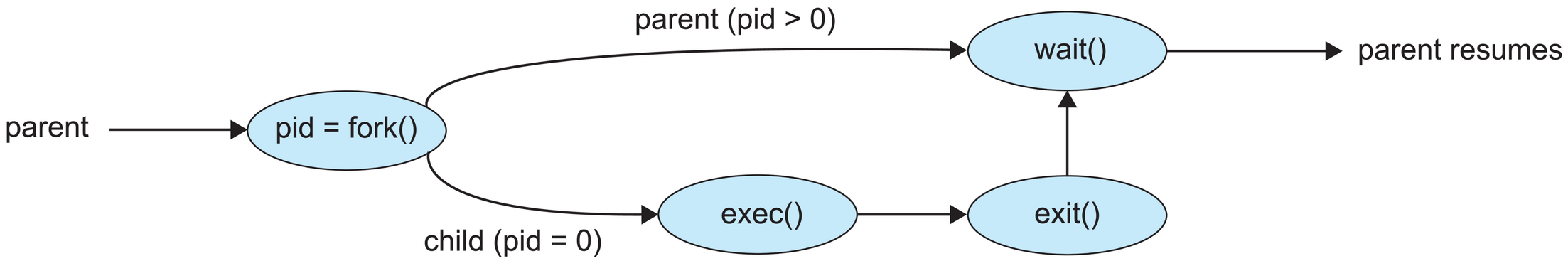
- **The program's heap is created**
  - Used for explicitly requested dynamically allocated data
  - Program request space with malloc() and free space by calling free()
- **The OS does some other initialization tasks**
  - Input/Output setup
    - Each process, by default, has three open file descriptors
    - Standard input, output, error
- **Start the program running at the entry point, namely main()**
  - The OS transfers control of the CPU to the newly-created process

Parent/Child Processes

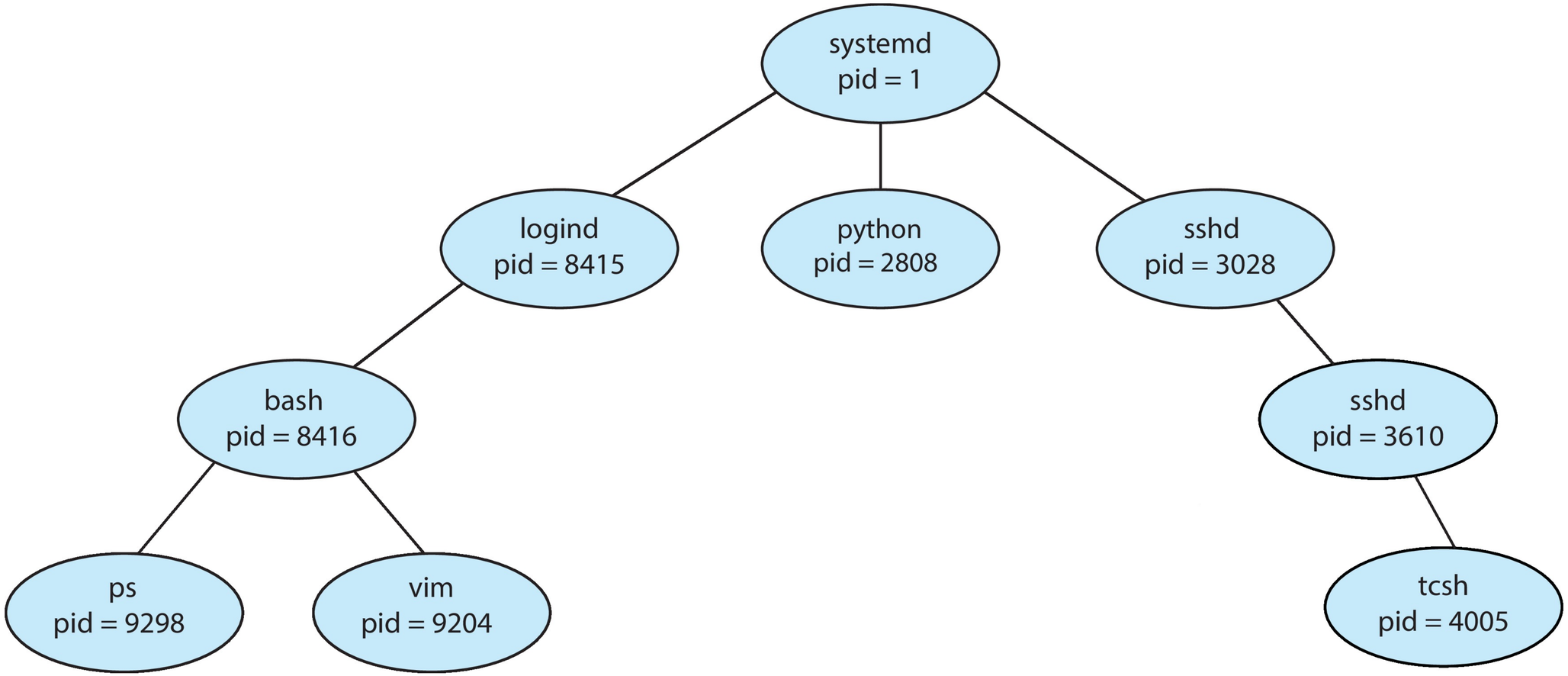
- Parent process creates children processes
  - Children processes create other children, etc
- Each process has a process identifier (pid)
- Resource sharing:
  - Parent and children share all resources, children shared subset, or they share no resources
- Execution options:
  - Parent and children execute concurrently, or parent waits until children terminate

Parent/Children Processes

- **fork()** system call creates new process
- **exec()** system call used after a **fork()** to replace the process' memory space with a new program
- Parent process calls **wait()** waiting for the child to terminate



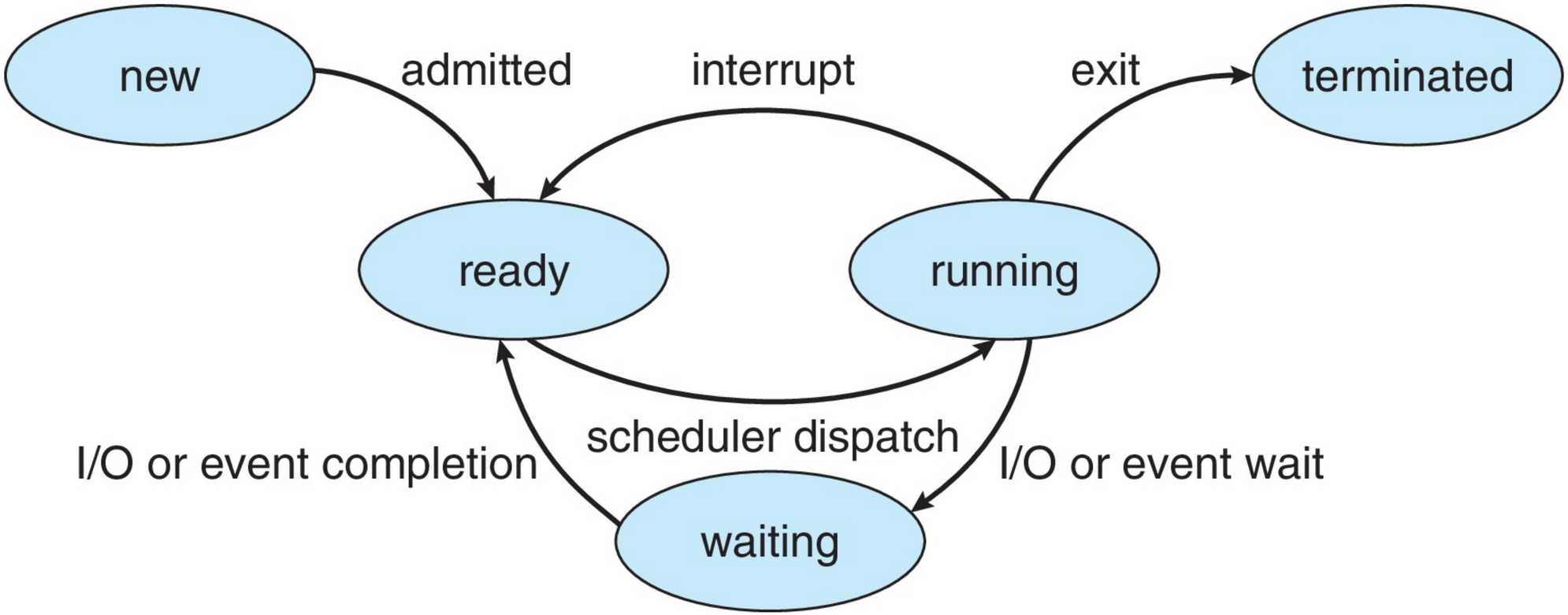
Linux Process Tree



Process State

- The state of a process is defined (in part) by the current activity of that process
  - **New:** process is being created
  - **Running:** instructions are being executed
  - **Waiting (or blocked):** process is waiting for some event to occur (i.e. I/O, completion signal)
  - **Ready:** process waiting to be assigned a processor
  - **Terminated:** process has finished execution

Process State



Process Control Block

Information associated with each process

- 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