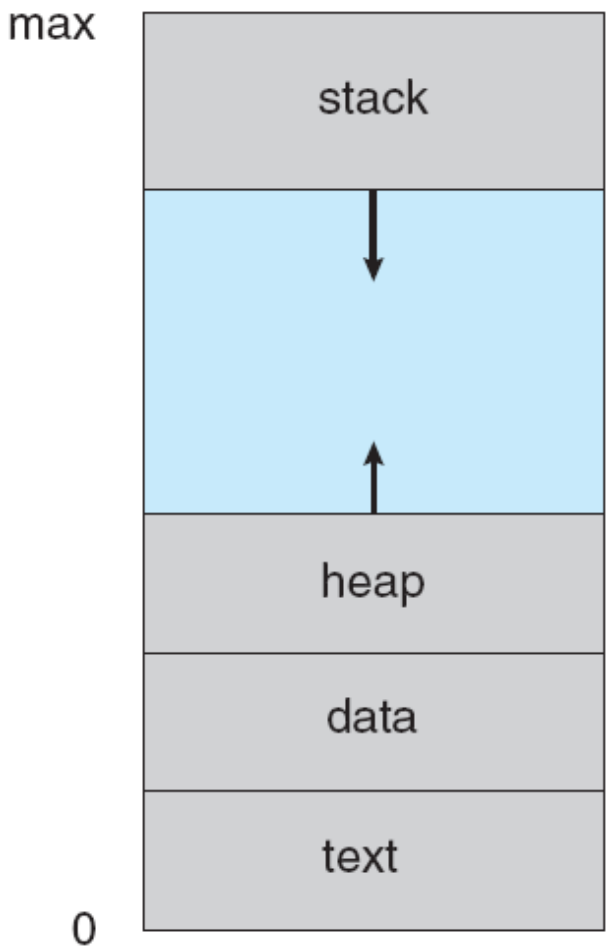
3 Processes

Process Concept

- An operating system executes a variety of programs:
  - Jobs for batch system
  - User program / tasks for time-sharing system
- Textbook uses the terms “job” and “process” almost interchangeably
- Process – a program in execution; process execution must progress in sequential fashion
- A process includes:
  - program counter
  - stack
  - data section

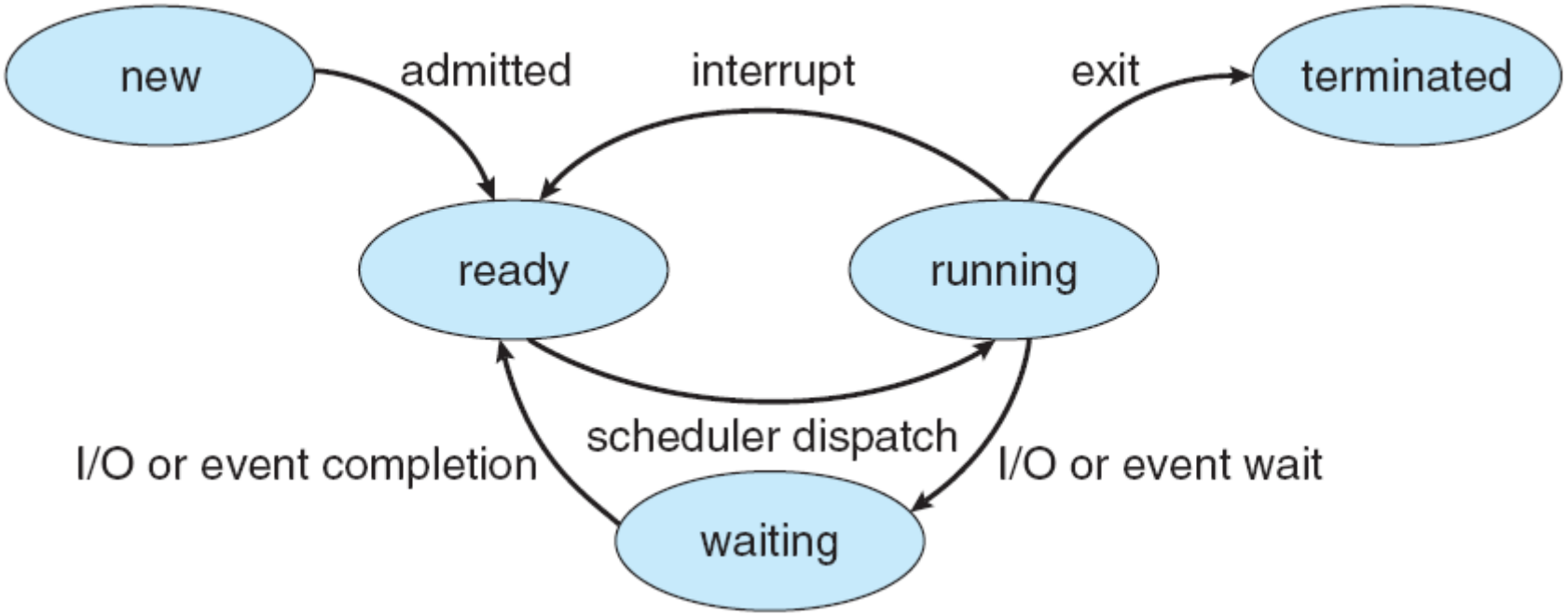
Process in Memory



Process State

- As a process executes, it changes state
  - new: The process is being created
  - running: Instructions are being executed
  - waiting: The process is waiting for some event to occur
  - ready: The process is waiting to be assigned to a process
  - terminated: The process has finished execution

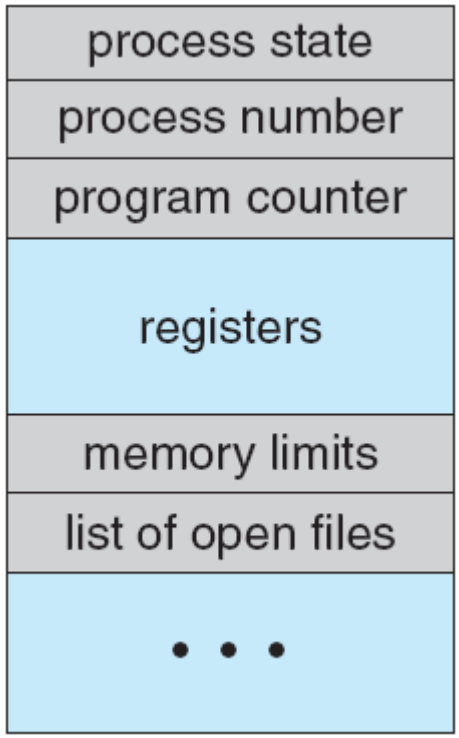
Diagram of Process State



Process Control Block (PCB)

- Information associated with each process
  - Process ID
  - Process state
  - Program counter
  - CPU registers
  - CPU scheduling information
  - Memory-management information
  - Accounting information
  - I/O status information

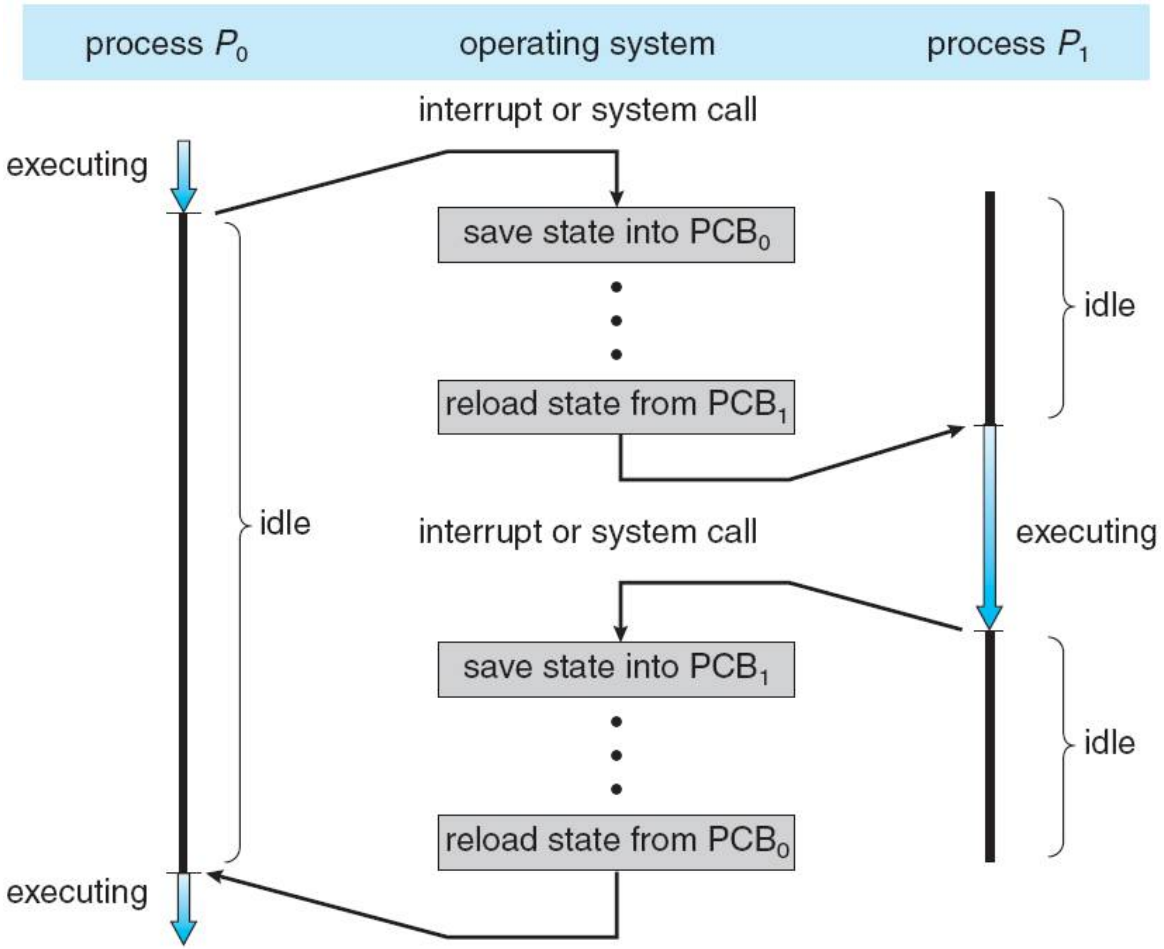
Process Control Block (PCB)



Context Switch

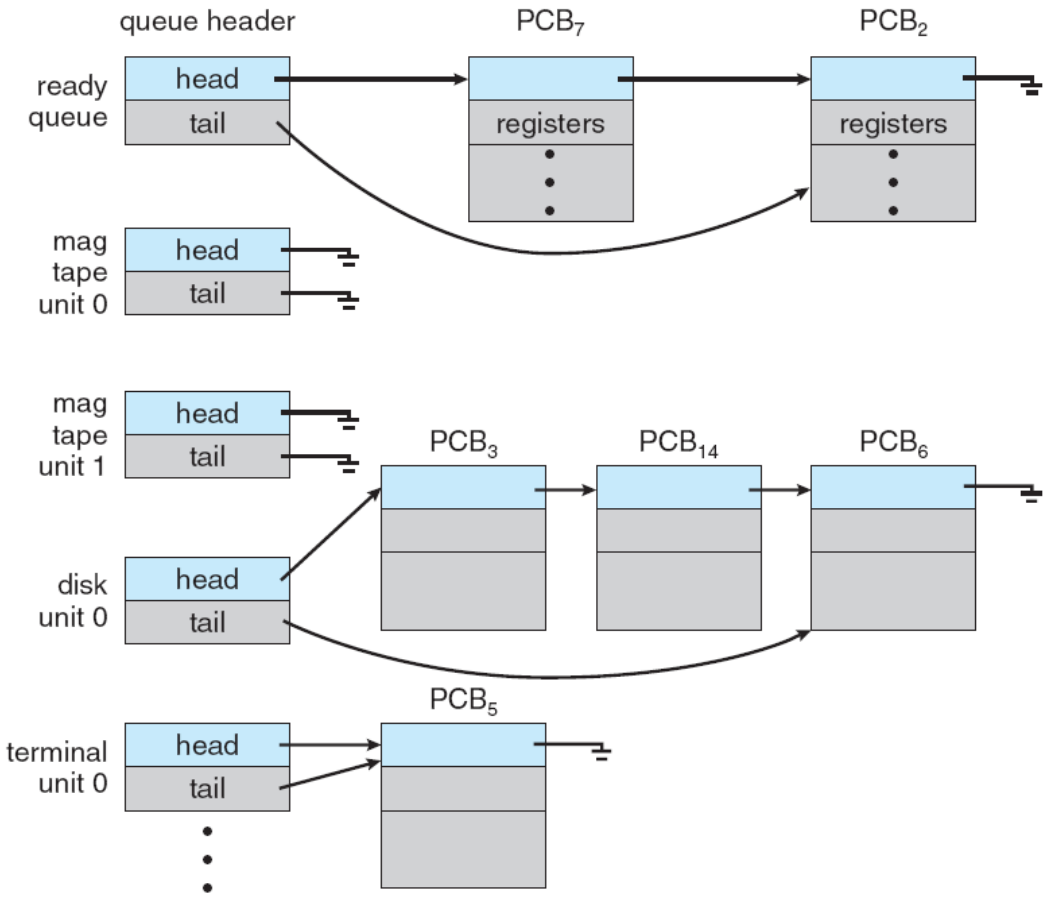
- When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process
- Context-switch time is overhead; the system does no useful work while switching
- Time dependent on hardware support

CPU Switch From Process to Process

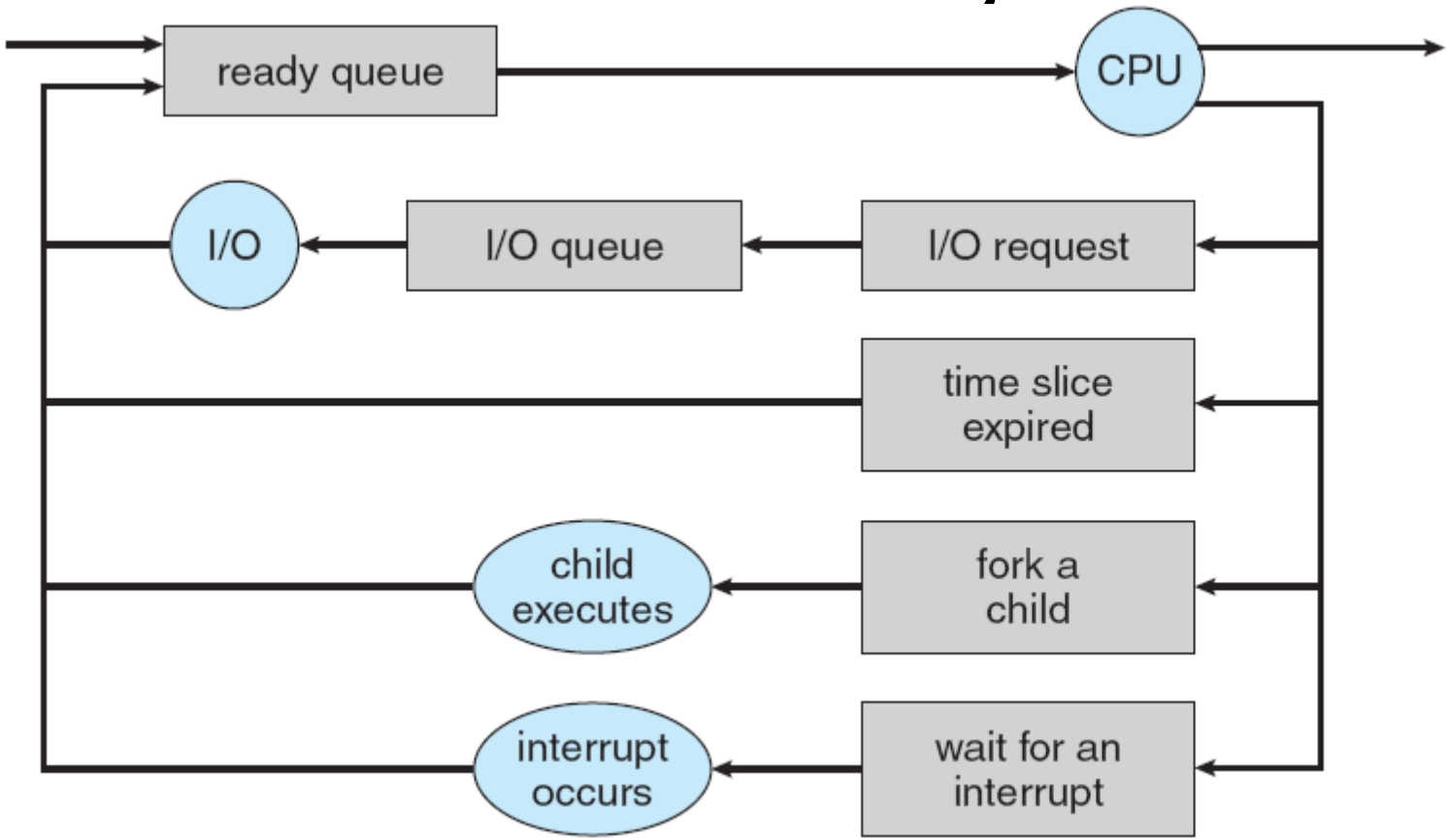


Process Scheduling

- Ready queue and various I/O queues



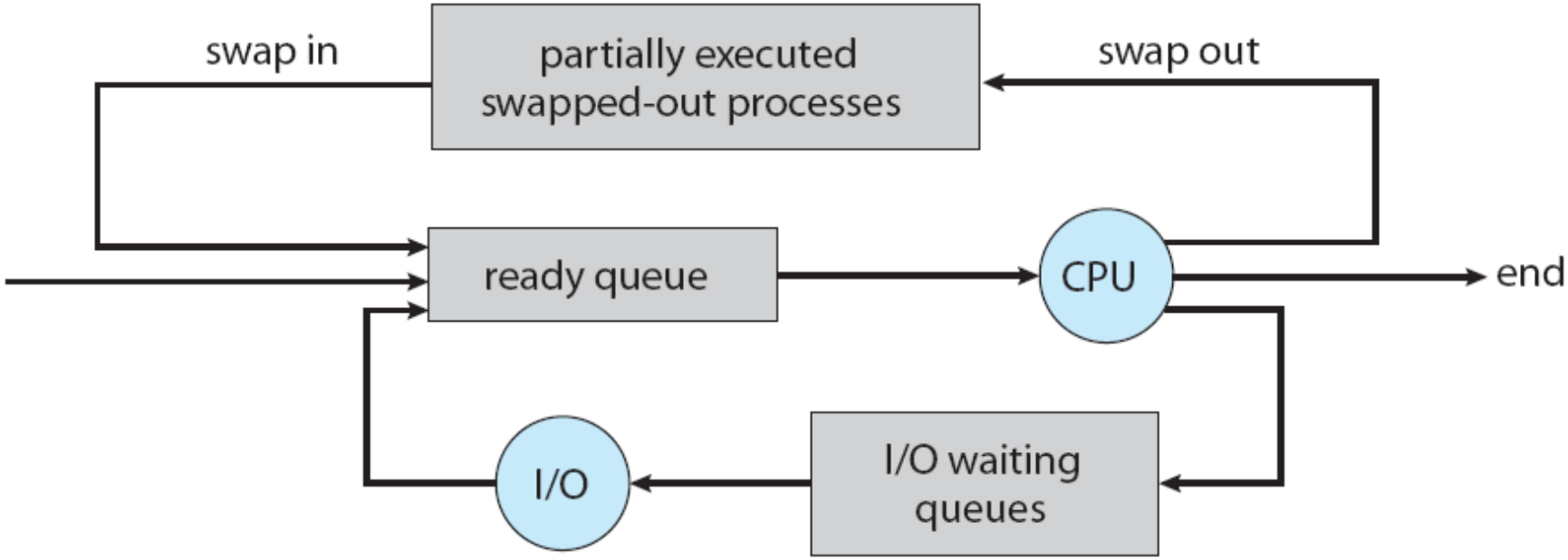
Queueing-diagram Representation of Process Scheduling



Schedulers

- Long-term scheduler
  - Job scheduler
  - Controls degree of multiprogramming
- Medium-term scheduler
  - swapping
- Short-term scheduler
  - CPU scheduler

Addition of medium-term scheduler to the Queueing Diagram

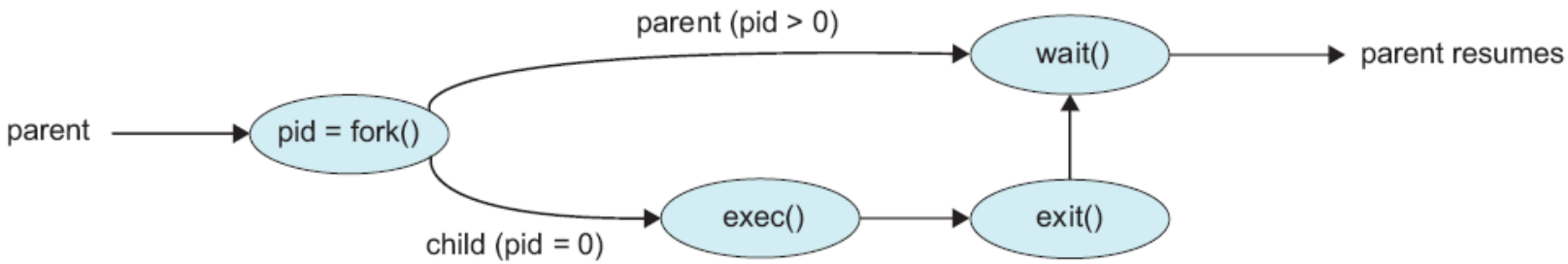


Process Creation

- Parent process create children processes, which, in turn create other processes, forming a tree of processes
- Resource sharing
  - Parent and children share all resources
  - Children share subset of parent's resources
  - Parent and child share no resources
- Execution
  - Parent and children execute concurrently
  - Parent waits until children terminate
- Address space
  - Child duplicate of parent
  - Child has a program loaded into it

Process Creation (Cont.)

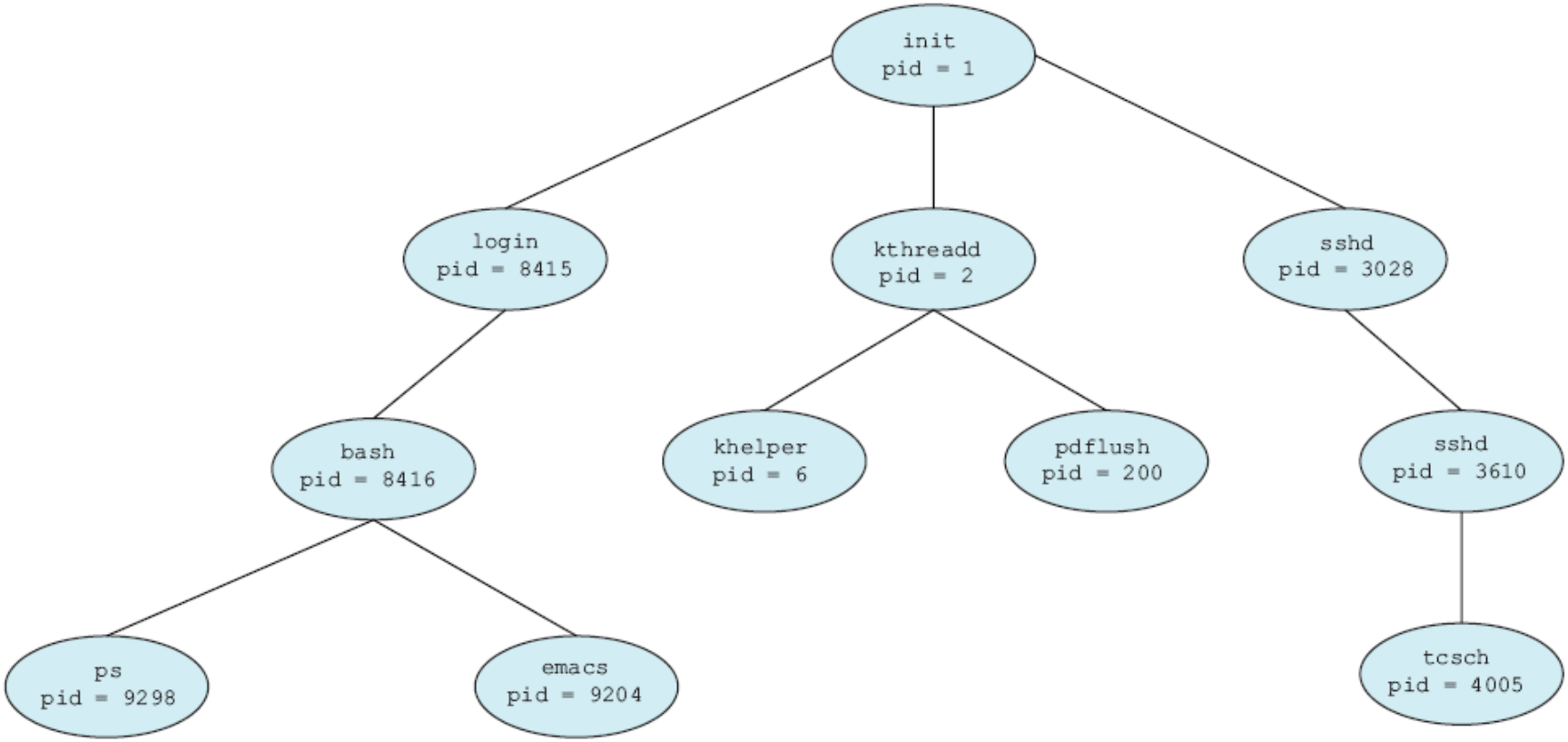
- UNIX examples
  - fork system call creates new process
  - exec system call used after a fork to replace the process' memory space with a new program



C Program Forking Separate Process

```
int main()
{
        pid_t  pid;
        /* fork another process */
        pid = fork();
        if (pid < 0) { /* error occurred */
                fprintf(stderr, "Fork Failed");
                exit(-1);
        }
        else if (pid == 0) { /* child process */
                execlp("/bin/ls", "ls", NULL);
        }
        else { /* parent process */
                /* parent will wait for the child to complete */
                wait (NULL);
                printf ("Child Complete");
                exit(0);
        }
}
```

A tree of processes on a typical Solaris



Process Termination

- Process executes last statement and asks the operating system to delete it (exit)
  - Output data from child to parent (via wait)
  - Process' resources are deallocated by operating system
- Parent may terminate execution of children processes (abort)
  - Child has exceeded allocated resources
  - Task assigned to child is no longer required
  - If parent is exiting, some operating system do not allow child to continue if its parent terminates; all children terminated - cascading termination

Cooperating Processes

- Independent process cannot affect or be affected by the execution of another proces
- Cooperating process can affect or be affected by the execution of another proces
- Advantages of process cooperation
  - Information sharing
  - Computation speed-up
  - Modularity
  - Convenience