Processes

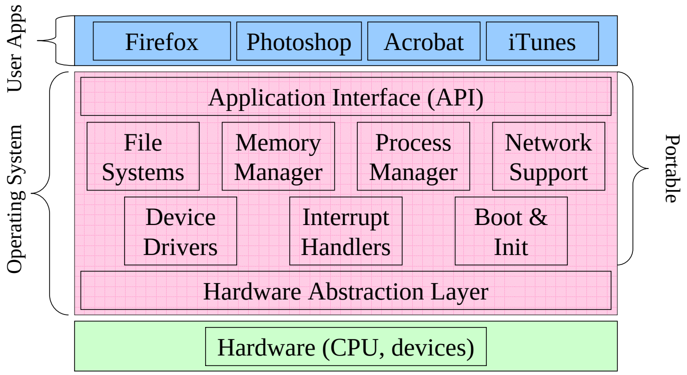
CSE 410, Spring 2007
Computer Systems

http://www.cs.washington.edu/410/

Reading and References

- Reading
  - » Chapter 4 through 4.5.4, *Operating System Concepts*, Silberschatz, Galvin, and Gagne
- Other References
  - » *Microsoft Windows Internals*, 4th ed (previously *Inside Microsoft Windows 2000*, 3rd Edition)

Example OS in operation


User Apps

Firefox | Photoshop | Acrobat | iTunes

Operating System

Application Interface (API)

File Systems | Memory Manager | Process Manager | Network Support

Device Drivers | Interrupt Handlers | Boot & Init

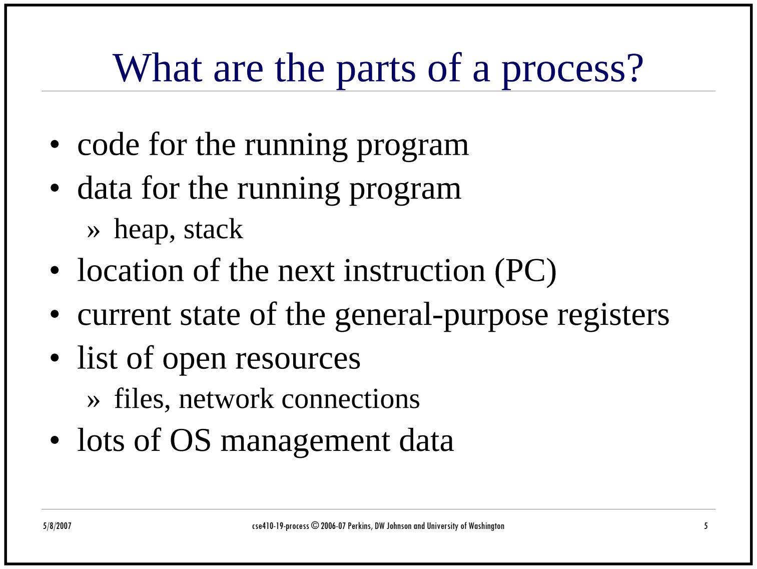
Hardware Abstraction Layer

Portable

Hardware (CPU, devices)

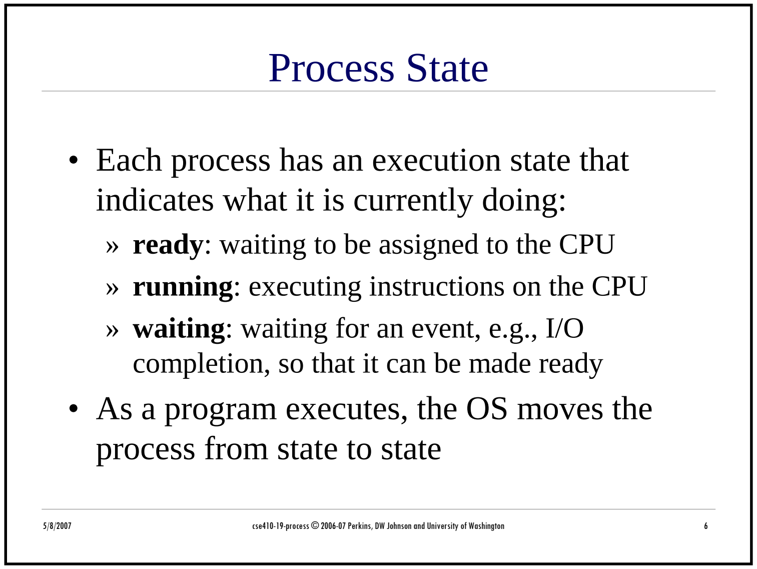
Programs and Processes

- A ***program*** is passive
  - » a file on disk with code that can be run
- A ***process*** is active
  - » an instance of a program in execution
  - » also called *job, task, sequential process*
- There are always many processes running
- Some may be running the same program
  - » but they are still separate and independent processes



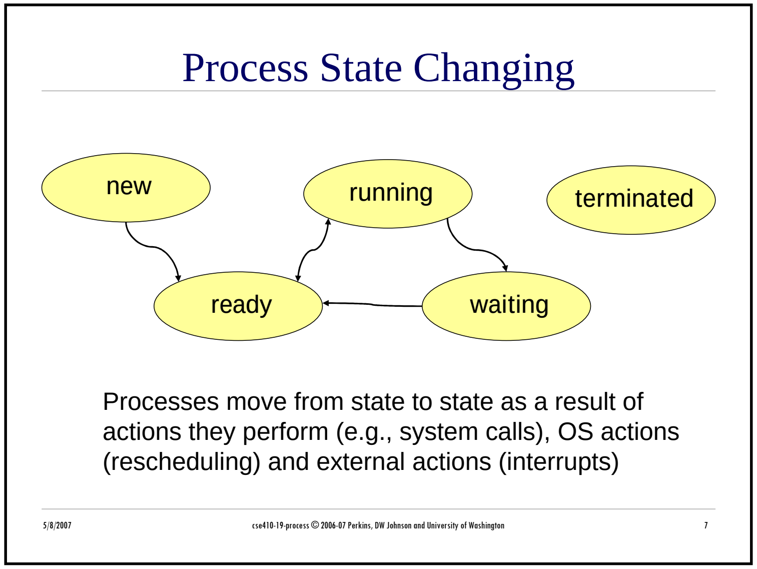
What are the parts of a process?

- code for the running program
- data for the running program
  - » heap, stack
- location of the next instruction (PC)
- current state of the general-purpose registers
- list of open resources
  - » files, network connections
- lots of OS management data



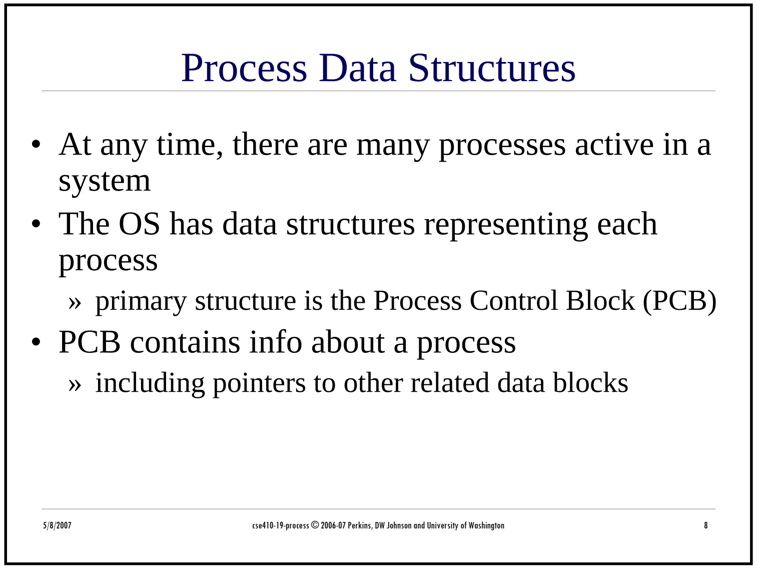
Process State

- Each process has an execution state that indicates what it is currently doing:
  - » **ready**: waiting to be assigned to the CPU
  - » **running**: executing instructions on the CPU
  - » **waiting**: waiting for an event, e.g., I/O completion, so that it can be made ready
- As a program executes, the OS moves the process from state to state



Process State Changing

Processes move from state to state as a result of actions they perform (e.g., system calls), OS actions (rescheduling) and external actions (interrupts)



Process Data Structures

- At any time, there are many processes active in a system
- The OS has data structures representing each process
  - » primary structure is the Process Control Block (PCB)
- PCB contains info about a process
  - » including pointers to other related data blocks

PCBs and Hardware State

- When a process runs, its PC, SP, and registers, are loaded on the CPU
- When the OS switches to a new process, it
  - » saves the current process’s register values to its PCB
  - » loads the next process’s register values from its PCB
- This is called a **context switch**. It occurs 100-1000 times per second
  - » why so often?
  - » why not more often?

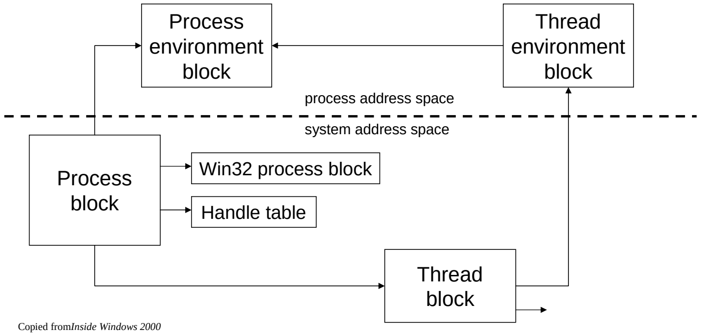
Context switch is pure overhead

- Switching processes can be expensive
  - » register reload
  - » OS data structures
- Lightweight context reduces cost of switch
  - » threads
- Special hardware reduces cost of switch
  - » larger register files with register windows

Simple Process Control Block

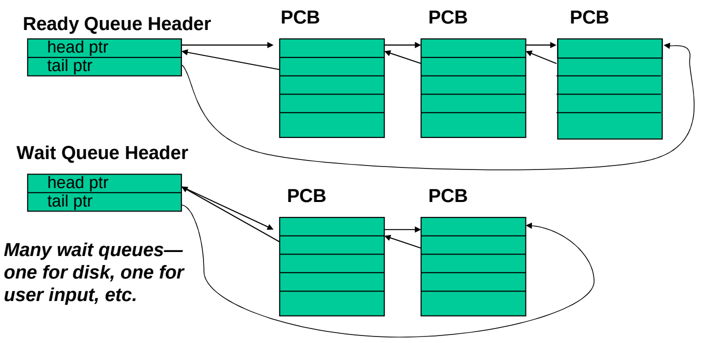
process state
process number
program counter
stack pointer
32 general-purpose registers
memory management info
username of owner
queue pointers for state queues
scheduling info (priority, etc.)
accounting info

Simplified W2K Process Data



Copied from*Inside Windows 2000*

Process State Queues



Many wait queues—one for disk, one for user input, etc.

PCBs and State Queues

- PCBs are data structures in OS memory
- A PCB is created for a process when it starts and put on the ready queue
- While the process is active, PCB is on one of the state queues
- When the process is terminated, its PCB is deallocated *(after a little while)*

Getting control back

- How does the OS get control back from a running process?
  - » The process could explicitly return control to the OS (in many real-time systems)
  - » Generally, we can’t trust the process to do this
- OS sets a timer on the CPU (privileged instruction) and starts a user process
- When the timer expires control passes to OS
  - » impact on “hard real-time” system?

Scheduling a process

- Batch processes tend to be scheduled over a long period by a job scheduler
  - » explicit dollar value on priority
  - » longer time in CPU once loaded and started
- Interactive or soft real time processes are started as needed and compete for CPU
  - » dynamic priorities
  - » rapid context switching of many processes

Creating a process

- The OS creates processes upon request
- The first few processes are all part of the operating system itself
  - » services, sessions, spoolers, network tools, ...
- Further processes created as response to login, user command, scheduled events
  - » winlogin, sshd, navigator, photoshop, ...

create-process

- OS provides create-process system call
  - » parent process creates one or more children
  - » each child can create more children
  - » the result is a process tree
- Parent can wait or continue immediately
  - » create a process and block (synchronous)
  - » create a process and continue (asynchronous)

Processes running on sample Win2K desktop

```
C:\home\finson>pslist -t

Process information for ASH:

Name                             Pid Pri
Idle                               0   0
  System                           8   8
    SMSS                         164  11
      WINLOGON                   184  13
        SERVICES                 236   9
          svchost                428   8
            naPrdMgr             616   8
          spoolsv                456   8
          svchost                488   8
          AppServices            504   8
          FrameworkServic        536   8
          mcshield               568  13
          vstskmgr               584   8
          mdm                    660   8
          nvsvc32                712   8
          regsvc                 780   8
          mstask                 796   8
          SAgentNT               844   8
            EBRR                 808   8
          stisvc                 888   8
          WinMgmt                940   8
          svchost               1084   8
          ADService             1140   8
          mysqld-nt             1332   8
        LSASS                    248   9
      CSRSS                      188  13
```

```
explorer                        1376   8
  firefox                        528   8
    WINZIP32                    1964   8
  CMD                           1272   8
    pslist                      1956  13
  ADUserMon                     1340   8
  LVComS                        1496   8
  point32                       1508   8
  LogiTray                      1532   8
  SOUNDMAN                      1540   8
  Imgicon                       1560   8
  pageant                       1576   8
  UpdaterUI                     1592   8
  rundll32                      1608   8
  MMTray                        1628   8
  jusched                       1648   8
  qttask                        1676   8
  POWERPNT                      1684   8
  shstat                        1696   8
  tbmon                         1732   8
  AcroTray                      1760   8
  WZQKPICK                      1788   8
  ssh_accession.e               1848   8
  xwin32                        1896   8
```

W2K *CreateProcess* function

- Open the program file to be executed
- Create the W2K executive process object
- Create the initial thread (stack, context, ...)
- Notify Win32 subsystem about new process
- Start execution of the initial thread
- Complete initialization (eg, load dlls)
- Continue execution in both processes

Copied from*Inside Windows 2000*