Fall2015 COMP 3511 Operating Systems

Tutorial and Lab #2

What we have discussed so far

- What Operating Systems Do
- Computer-System Organization and Architecture
- Operating-System Structure
- Operating System Services
  - System Calls
  - System Programs

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and computer response to users

- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which jobs (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - file
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data -transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control determines who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a "long" period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications
  - Varies between WORM (write-once, read-many-times) and RW (read -write)

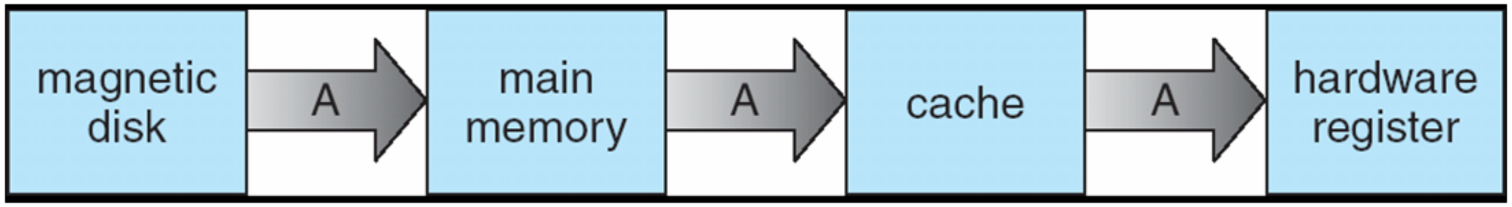
Performance of Various Levels of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

- Movement between levels of storage hierarchy can be explicit or implicit

Migration of Integer A from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a data can exist

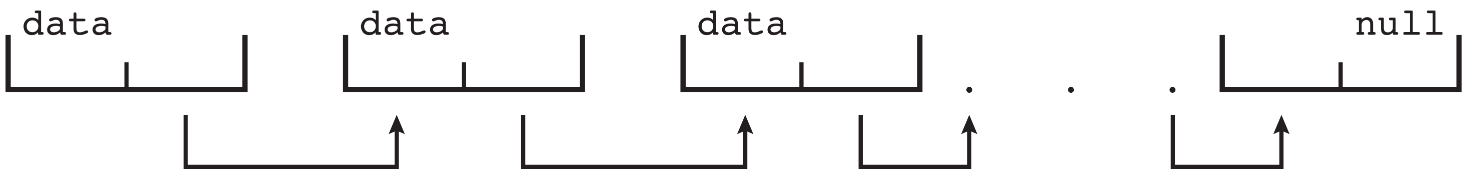
I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user

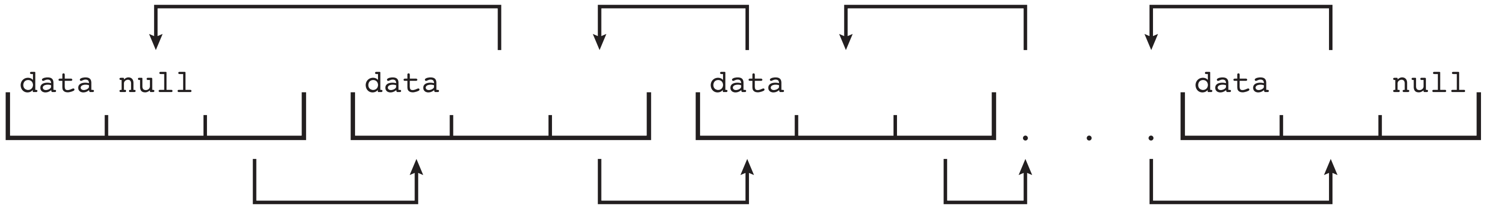
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

Kernel Data Structures

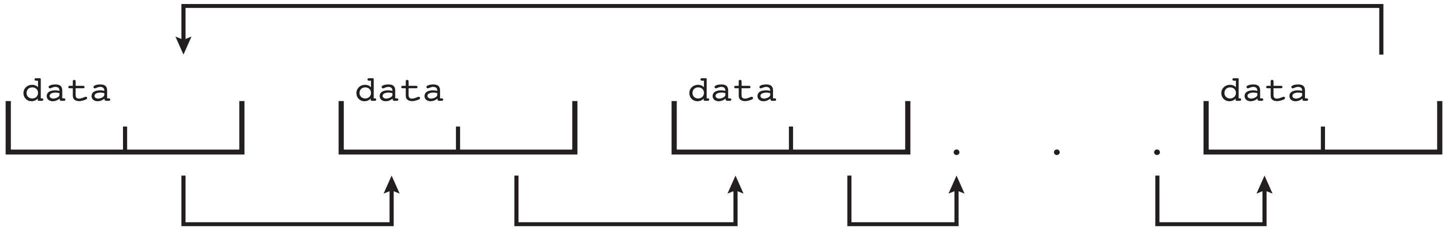
- Many similar to standard programming data structures
- ***Singly linked list***



- ***Doubly linked list***

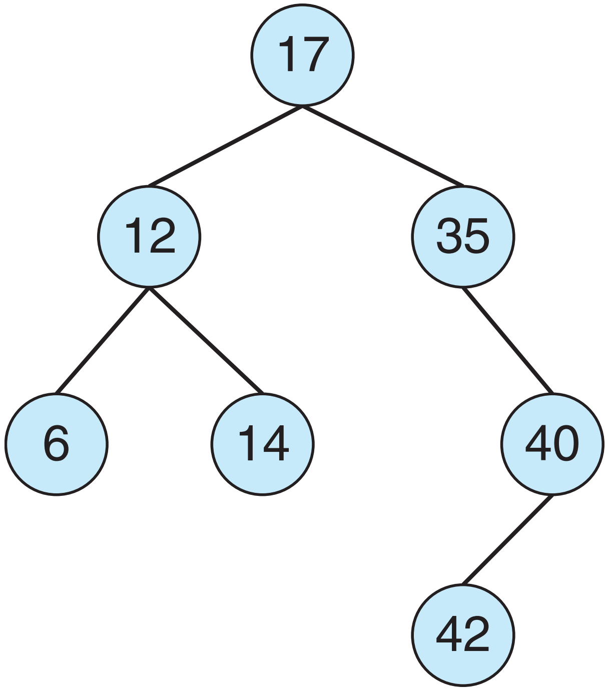


- ***Circular linked list***



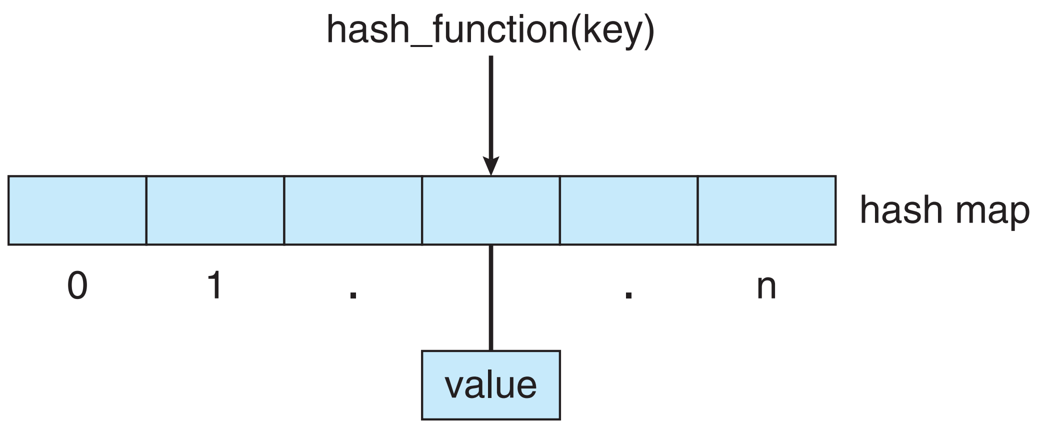
Kernel Data Structures

- **Binary search tree**
  left <= right
  - Search performance is *O(n)*
  - **Balanced binary search tree** is *O(lg n)*



Kernel Data Structures

- **Hash function** can create a **hash map**



- **Bitmap** – string of *n* binary digits representing the status of *n* items
- Linux data structures defined in ***include*** files `<linux/list.h>`, `<linux/kfifo.h>`, `<linux/rbtree.h>`

Q. 1

- What is the main purposes of an operating system?
  - User convenience: to provide an environment for a computer user to execute programs on computer hardware in a convenient and efficient manner
  - Resource allocation: to allocate the separate resources of the computer in a fair and efficient manner (CPU, main memory and etc.). You can also view an OS as a control program, which controls the program execution, prevent users from misusing the resources and handling I/O

Q. 2

- What is an **interrupt** and what is a **trap**?

An interrupt is a hardware-generated change of flow within the system

An interrupt handler is invoked to deal with the interrupt when it occurs; control is then returned to the interrupted context and instruction

An interrupt can be used to signal the completion of an I/O operation

A trap is a software-generated interrupt.
A trap can be used to call the OS routines or to catch arithmetic errors

Q. 3

- List six **services** provided by an operating system that are designed to make it more convenient for users to use the computer system.

User interface

File-system manipulation

Program execution

Communications

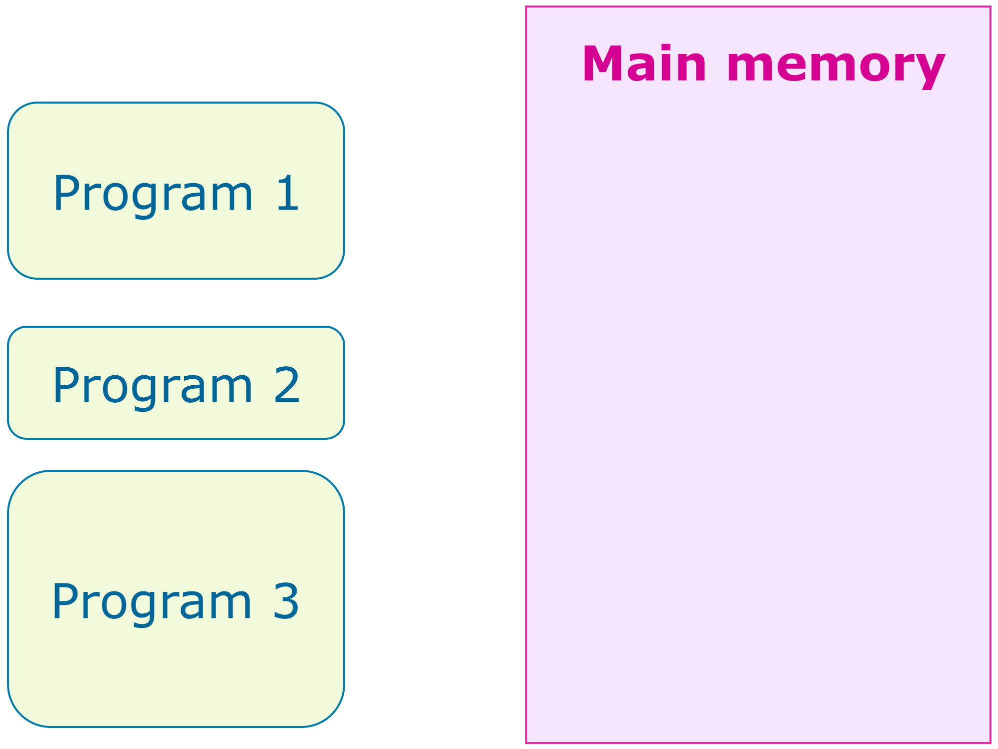
I/O operations

Error detection

Q. 4

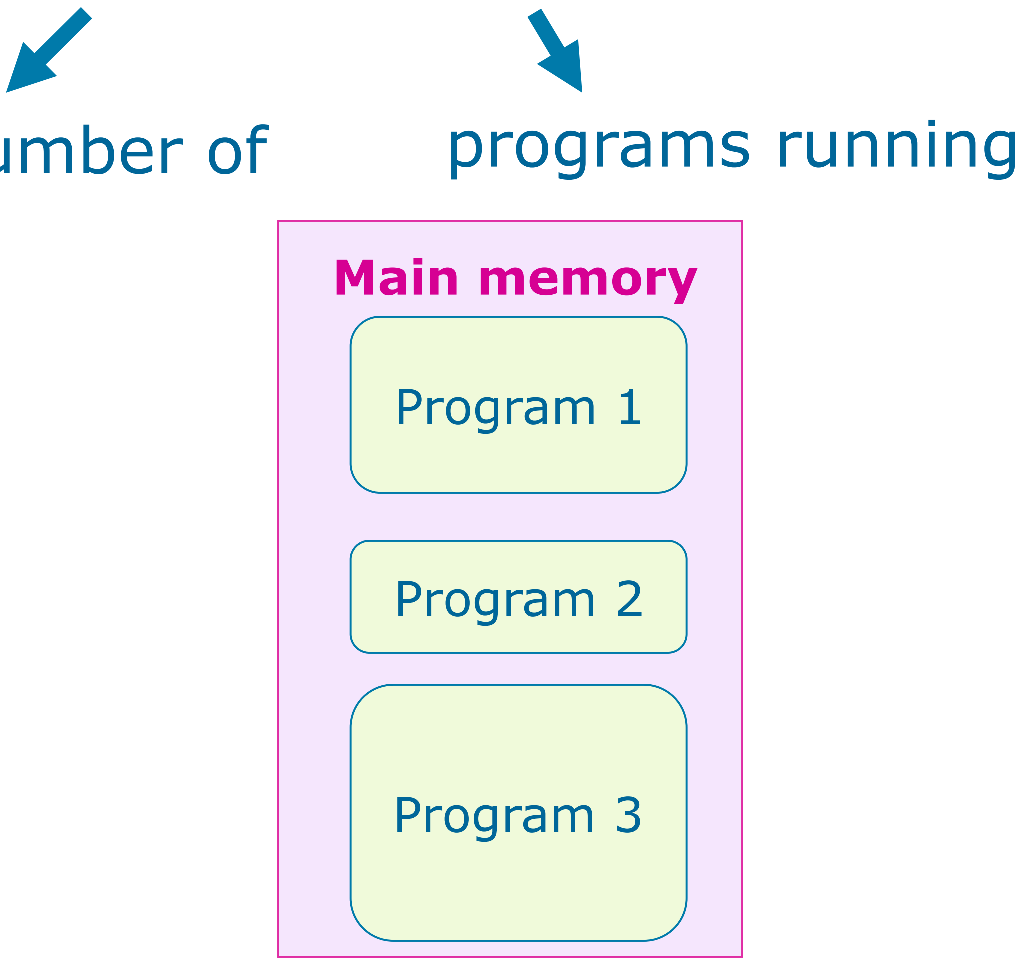
- What is **multiprogramming**?

A number of programs running



Cont.

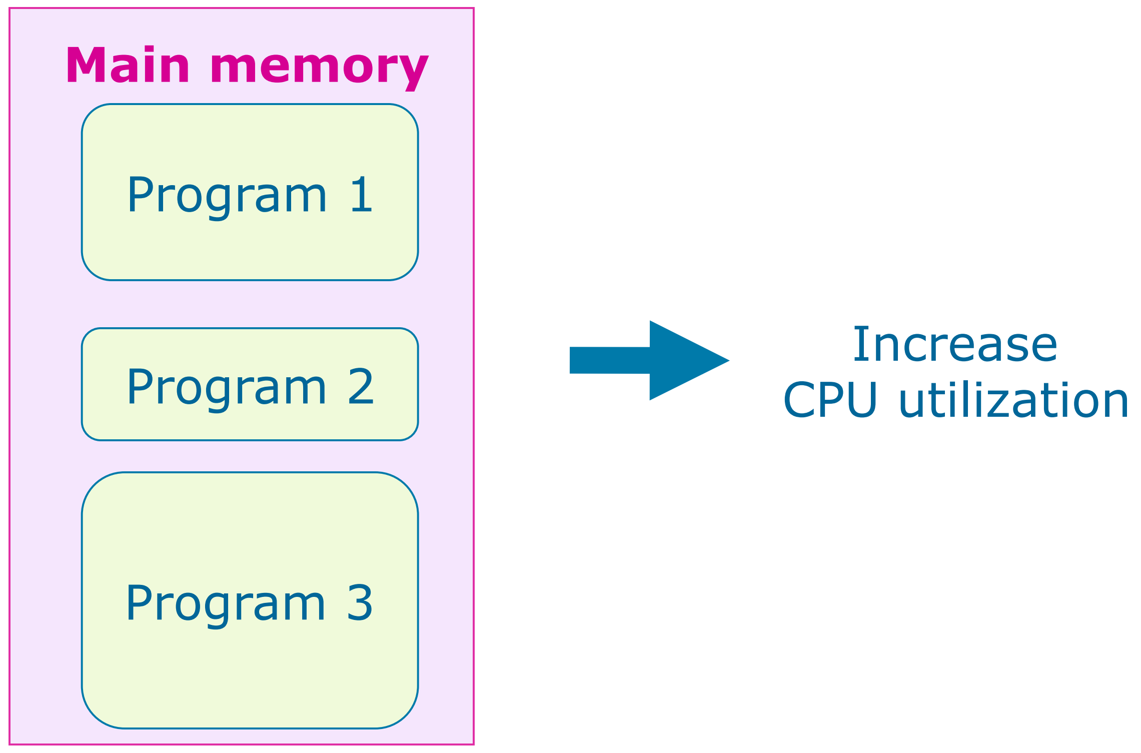
- What is **multiprogramming**?



Cont.

- What is **multiprogramming**?

A number of programs running



Cont.

- What is the main advantage of multiprogramming?
  - It makes efficient use of the CPU by overlapping the demands for the CPU and its I/O devices from various users or/and processes.
  - It attempts to increase CPU utilization by always having a job for the CPU to execute.

Q. 5

- Why OS needs to distinct **kernel mode** and **user mode**?
  - Such dual-mode operation allows OS to protect itself and other system components
  - prevent users from misusing the resources

Q. 6

- What are the three general methods of parameter passing in a system call?
  - Passing parameters in registers
  - Registers passing starting address of blocks of parameters
  - Parameters can be placed, or pushed onto a stack by a program, and popped off by the operating system