

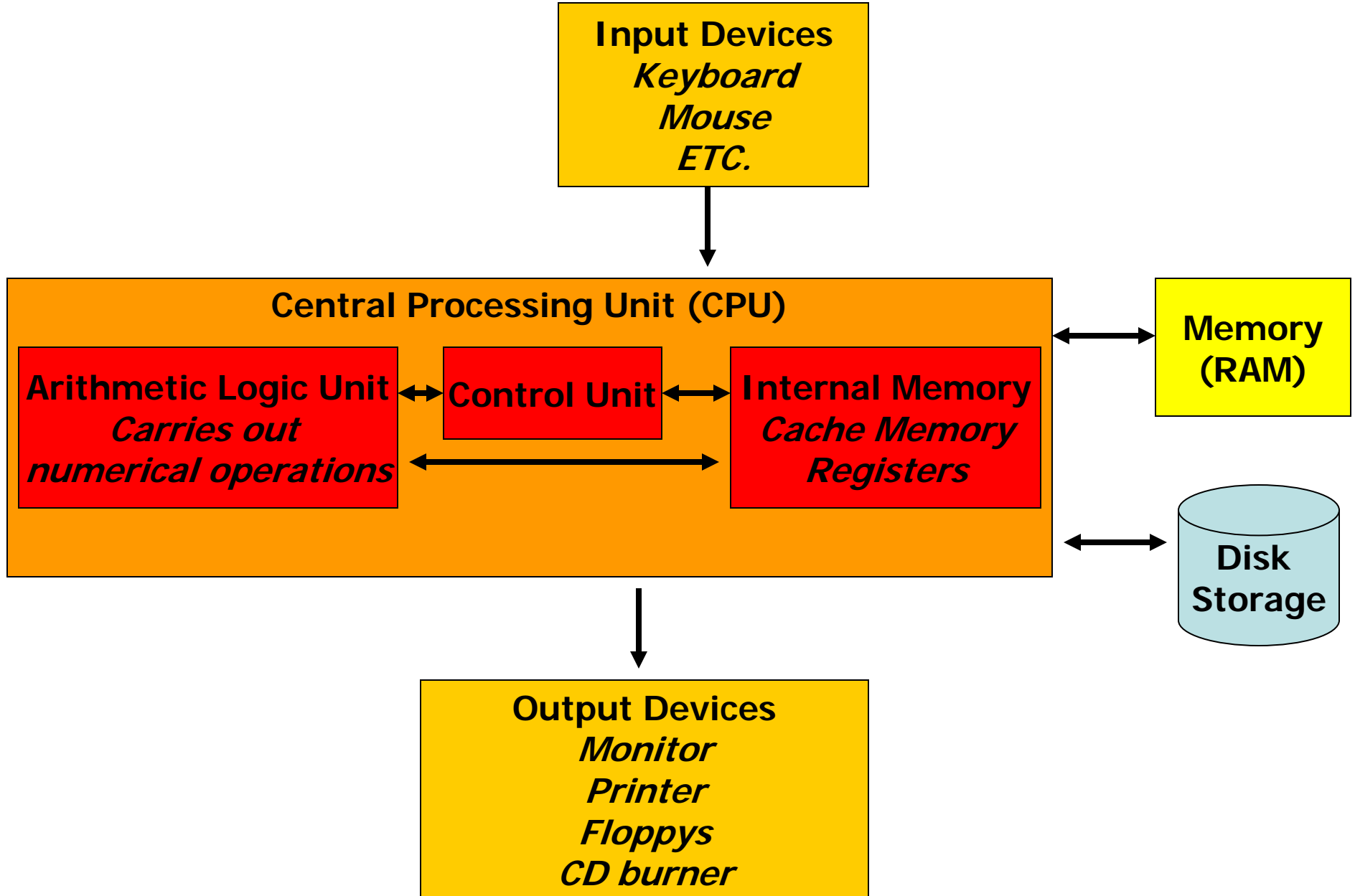
# What do we want to accomplish in this class?

- Learn to use a Linux or Un\*x Computer for Scientific Computing Tasks
  - Important for so many reasons!
- Learn to write programs in FORTRAN-95
  - Important for numerical computing
  - FORTRAN remains the premiere numerical computing language (we will see why later)
- Learn to write programs in C++
  - Important for many non-numerical tasks such as visualization, data acquisition, systems programming, etc. where compatibility with the C programming language and/or object-oriented methods are preferable

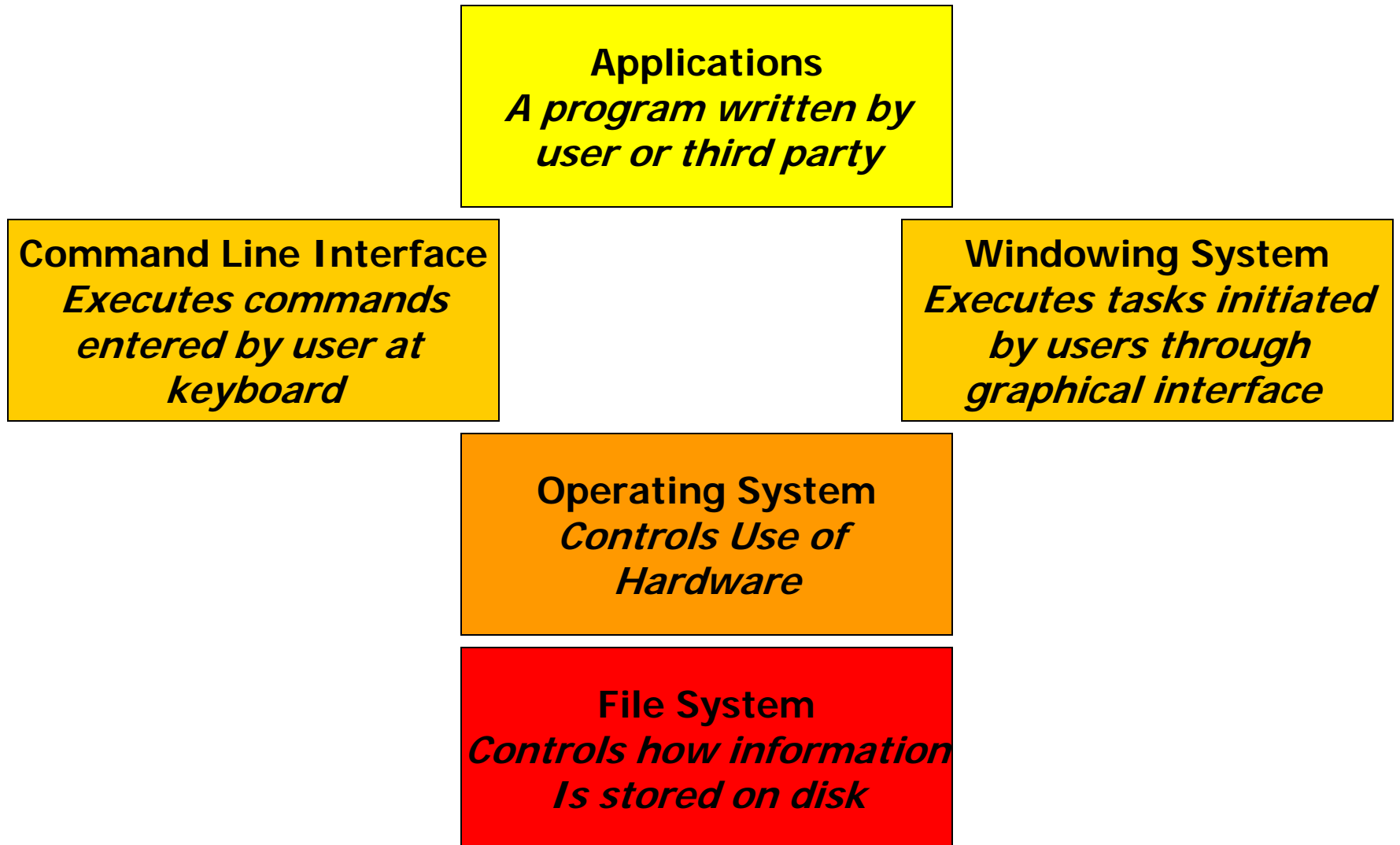
# A How to Begin?

- A Journey of a Thousand Miles Begins with a Single Step -- *Confucius*
- Let's establish a vocabulary to support our endeavor!
- Learn how to use the basic command line interface
- Learn some basic commands
- Learn how to edit a file
- Learn to send & receive a simple email message

# A Typical Computer



# Layers of Software



# Interfacing with the Computer: Windows vs. Command Line Interfaces

- Windowing Interfaces (MS Windows; Apple OS; X11)
  - Offer a Graphical User Interface (GUI; pronounced “Goo-eeey”)
  - Point and Click w/ Mouse to Accomplish tasks
  - Sometimes easy for beginners to use
  - If menu structure of application is well designed
  - Often not flexible
  - Can be difficult to automate repetitive tasks for many applications
- Command Line Interface
  - A.K.A. terminal or shell
    - Can be a window on systems with graphical interfaces
  - Commands typed in via keyboard
  - More difficult for beginners
  - Must remember commands
  - Longer learning curve than windowing
  - More flexible
  - Usually easier to automate many repetitive tasks

# The Linux/Un\*x Command Line Interface

- When you log into a Un\*x system and start typing in commands, whether you are setting in from of the monitor of that system (known as the system console) or logging in from another machine you are actually interacting with a command interpreter program known as the shell
- The shell accepts your commands and submits them for execution. The shell will prompt you with a command prompt that you can customize (we will discuss how to customize this later)
- Note that on Un\*x all commands, filenames, and directory names are case sensitive, i.e. **data.txt** is a different file from **Data.txt**
- For clarity we will denote this command prompt by the symbol ">"
- For example typing the command: `ls` at the command prompt causes the shell to execute the "ls" command which lists the files that are present.

# About the shell

- There are many different shells
  - sh – Bourne shell
  - bash – Bourne again shell
  - csh – C-shell
  - tcsh – T-C-shell
  - ksh – Korn shell
  - zsh – Z-shell
- Each has its own strengths & weaknesses
- We will use the bash shell throughout this class
- bash is now the default linux shell

# What does the shell do?

- Short answer: executes commands that you type in via the keyboard
- Long answer: controls the execution of programs that are stored on disk or which are built into the shell
- Most commands are programs that stored on disk somewhere
- A few may be built into the shell

# Some basic Linux/Un\*x Commands for File Manipulation

- ls – lists files stored on disk
- date – shows the current date
- finger – shows who is logged on
- cp – copies one file to another
- rm – removes a file stored on disk
- mv – moves, i.e. renames a file
- cat – catenates a file, i.e. shows the file contents
- more – displays the contents of a file a page at a time
- pwd – lists the pathname of your present working directory on the disk
- cd – changes your working directory on the disk
- mkdir – makes, i.e. creates, a new directory
- rmdir – removes, i.e. deletes, a directory
- man – displays the manual page for a command

# Command Arguments

- Some commands accept one or more “inputs”
- These “inputs” are called arguments

- Example 1: One argument

```
rm file1.dat
```

Deletes the file named file1.dat

- Example 2: Two Arguments

```
cp file2.dat file3.dat
```

Copies the file named file2.dat to a file named file3.dat

- Example 3: Variable Numbers of Arguments

```
rm file1.dat
```

or

```
rm file2.dat file3.dat
```

Deletes the file named file1.dat (first case) or the files named file2.dat and file3.dat

# Command flags

- Arguments to a command do not have to be filenames
- They can be other inputs that control the execution of the command
- Example 1:  
`ls -l`  
Gives a long-form (detailed) listing of files
- Example 2:  
`rm -i`  
Inquires yes/no before deleting a file

# What arguments and/or flags does a command accept?

- Use the “man” command to read the manual page for the command

- Example

`man ls`

Displays the manual page for the ls command

- A very important acronym: RTFM
  - RTFM = **R**ead **T**he **F**reakin' **M**anual!

# Editing Files on Linux/Un\*x Systems

- Numerous applications exist to help you modify the contents of a text file
- These applications are called editors
- Two popular editors on Linux/Un\*x systems are emacs & vi
- emacs is extremely powerful while remaining easy for a beginner to use
- When used on a graphical interface basic editing can be done entirely with pull-down menus at the top of the emacs window.

# Sending & Receiving Email with the mail command

- There are many ways to send & receive email on most Linux/Un\*x systems
- A crude, but quick, command is “mail”

Example:

`mail phy277@mail.astro.sunysb.edu`

Type in some text

Type a “.” in column 1 to exit the mail command

- Mail a message to yourself
- Execute the mail command without any arguments to display any email you have received

# Reading Assignments

- Unix tutorials linked on class web page:
  - University of Utah Unix Tutorial
  - Tutorial 1 of University of Surrey
- Chapman:
  - Chapter 1
- Learn the basic commands we have introduced in this class
- Be prepared to answer questions in the next class as you are likely to be queried.