CSE 11

Intro to Computer Science with Java

- Overview of CSE 11
- Fast review of computers, programming, and UNIX
- Hand out class computing accounts

Overview of CSE 11

- This information can be found on the class web pages, rooted at

  `http://sunpal.ucsd.edu/~cs11f`

  Be sure to read this stuff!... I will hold you responsible for knowing it

Fast review of computers, programming, and UNIX

- ✔ Why do we like computers?
- ✔ Computer hardware and software
- ✔ Computer programming and high-level languages
- ✔ Unix filesystems
- ✔ Getting started in the CSE 11 environment

Why do we like computers?

..... because computers can do *anything!* *

* well, almost

Can computers really do *anything*?

- The idea for a universal computing machine was developed by the British mathematician Alan Turing in 1936. Neal Stephenson writes of this discovery in his book *Cryptonomicon*:

Turing figured out something entirely different, something unspeakably strange and radical.

He figured out that mathematicians, unlike carpenters, only needed to have one tool in their toolbox, if it were the right sort of tool. Turing realized that it should be possible to build a meta-machine that could be reconfigured in such a way that it would do any task you could conceivably do with information. It would be a protean device that could turn into any tool you could ever need.

- Turing's mathematical idea of a universal machine has been realized as the modern digital computer...

But, really... ANYTHING?

- Well, there are some limits to what a computer can do:
  - There are some things no machine can do at all, in principle
  - There are some things that are impossible for machines, in practice

    (you will learn more about the limits of what is computable in CSE 101 and CSE 105)

- However, in an introductory CS course like this, the absolute limits of computers are not really relevant:
  - you'll be limited only by your imagination and your skill at programming

What would you like to do with computers?

- Computers are “universal” machines, with some absolute limits to what they can do
- While learning to program, exercise your imagination, and think about what possibilities you would like to create with computers

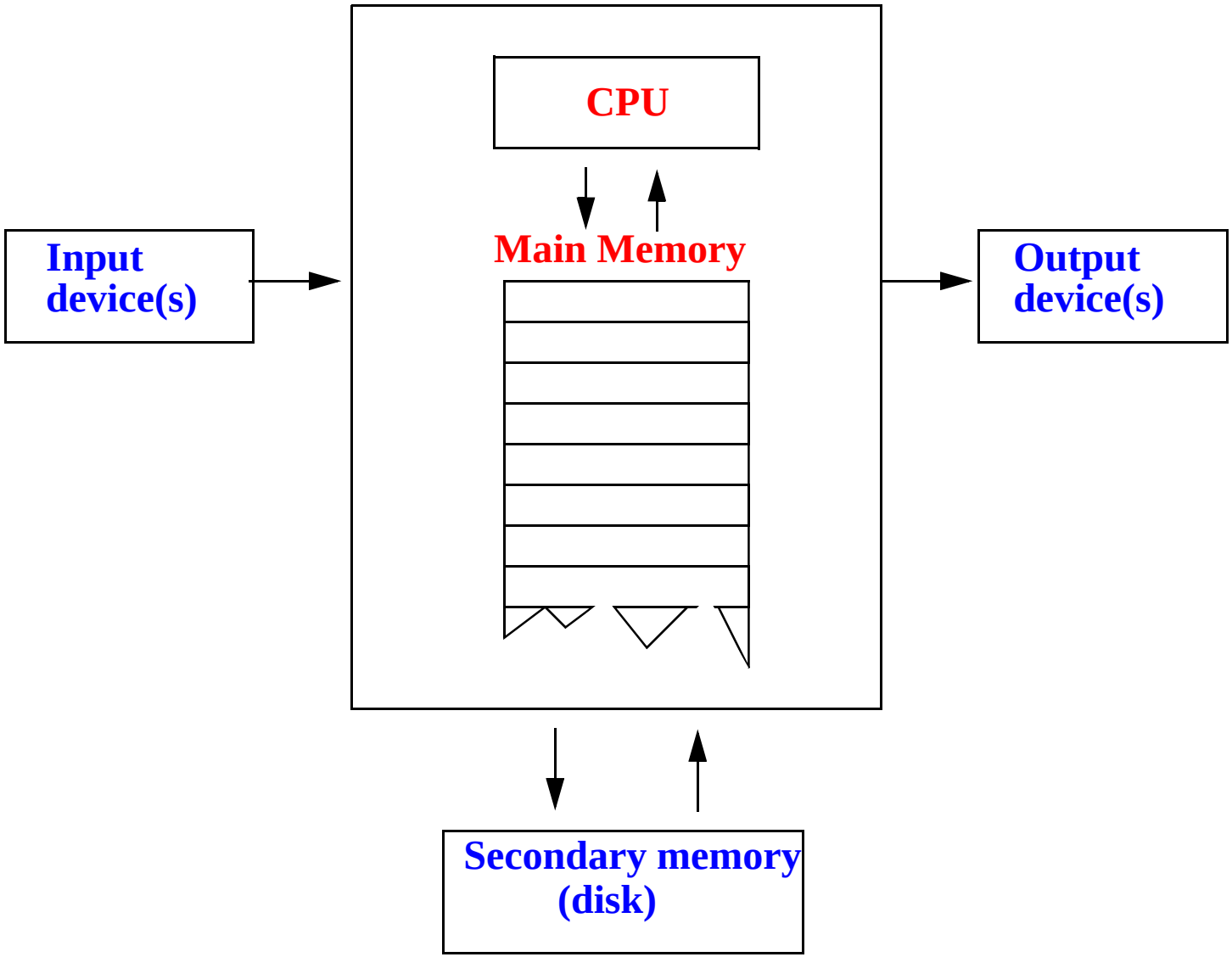
  ... (There will be a question on the final exam!) ...

- As the next generation of computer scientists, you will have your chance to make those possibilities a reality

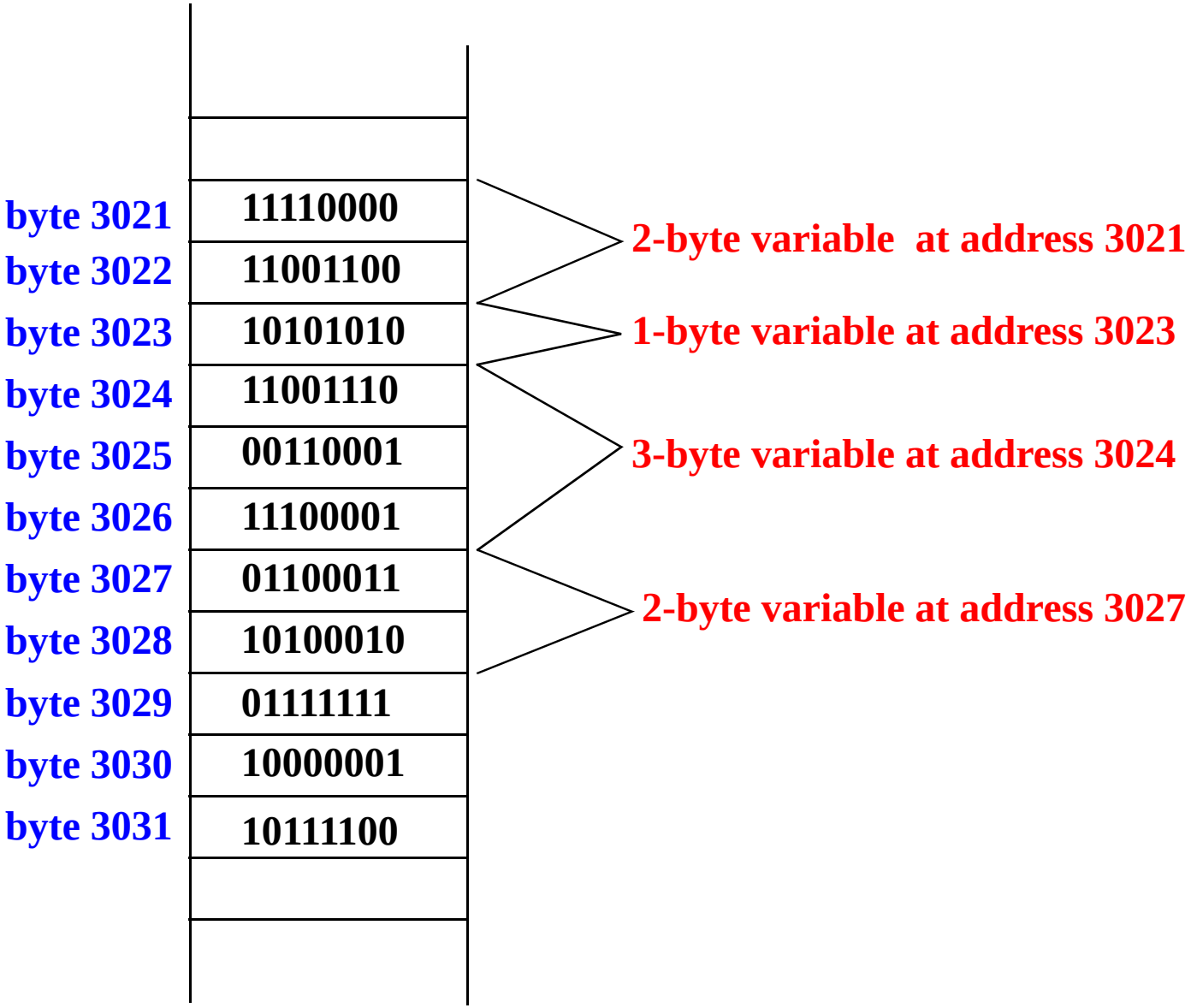
Computer hardware and software

- A major part of computer science is understanding the art of computer programming
  - Note: this is *not* the only part. But it is a major part
- Computer programming is the process of getting a computer to do something useful, something it has never done before
- You do this by giving the computer instructions to do what you want it to do
- Definitions of some basic concepts:
  - Program: a set of instructions for a computer to follow
  - Software: the collection of programs used by a computer
  - Hardware: the physical devices that make up a computer

Basic structure of a typical digital computer



Main memory in a digital computer



Programming a computer

- To get a computer to do something useful, you write a program, a sequence of instructions for it to follow, and turn the program into a sequence of bits --- 1's and 0's --- that the CPU can read in from memory to tell it what to do
- How do you do this?
- Basically, 4 steps:
  - ***Design*** your program (this is really the hardest part, especially for large programs)
  - ***Write*** your program in a "high-level" computer language, using an editor (a piece of software)
  - ***Compile*** your program using a compiler (another piece of software) to produce a "machine language" translation that the computer hardware can understand
  - Tell the operating system (yet another piece of software) to ***run*** this machine language translation of the program

Why write programs in a high-level language?

- "Machine code" is easy for computers to understand, but hard for people to understand:

```
00101000100100100100100000001111010100100010
00001110100100000100101011111111111111111111
```

- "Source code" in a high-level programming language is easy for people to understand, and can be automatically translated into machine code by a compiler

  [...an idea from the 1950's. High-level computer programming was invented at about the same time as rock-n-roll. Coincidence? You decide...]

  Examples of high-level language statements:

```
ADD 2 PLUS 2 AND PRINT THE RESULT

cout <<  2 + 2 ;

System.out.println( 2 + 2 );
```

High-level languages make it a lot easier for people to program computers
.... easier, but still takes time and effort to learn how to do it, and do it well

Programming computers and using computers

- Programming a computer is different from using a computer
- You program a computer to create a program that a user can use (got that?...)
- Some people find that they enjoy using a computer, but really don't enjoy programming
- Warning: You may find this out about yourself this quarter!

Finding out you're not a programmer

- If you decide this quarter that this isn't for you, here are some things to keep in mind:
  - It's not because you're stupid. You didn't get into UCSD by being stupid.
  - It may be a matter of talent (like musical talent...) People do have different talents.
  - In any case it *could* be a sign that you should do something else with your life
    - There are lots of possibilities out there and there's no good reason to be miserable doing something you don't like...
    - But talk to an advisor before you do anything precipitous!

Why use Java as the high-level language?

- Java is a modern programming language, developed in the 1990's as a powerful, portable, high-performance, robust, secure, networkable, dynamic, multithreaded, buzzword-compliant high-level language

- C, C++, and C# are also powerful and popular high-level languages that share a lot of details with Java; once you've learned Java, they (and others) are not very hard to learn

- Java is an object-oriented programming language... the Object-Oriented Programming (OOP) paradigm provides a powerful high-level viewpoint from which to write programs

Programming and fundamental concepts of computer science

- In this course you will learn two kinds of things:
  - fundamental concepts of computer science
  - details about how to program in Java
- These are both important, but the first is -- um, more fundamental

- Some fundamental concepts we will discuss (there will be others--watch for them!):
  - Algorithms: step-by-step recipes for doing computations, and how to implement them in computer programs
  - Variables: names of objects, and how to use them in programs
  - Control contructs: how to correctly structure programs
  - Functions: the encapsulation of behavior in programs
  - Abstract data types: collections of variables and functions that operate on them through an abstraction barrier
  - Object-oriented design and implementation: grouping data and operations to solve a problem

- But first we will start with a quick orientation to get started

Logging in to your account

- From the Sun-PAL lab:
  - Read the “CDE Labs” handout on how to get started
  - Type your login name and password when prompted, and open up a terminal window to use the Unix command line interface

- From your Unix/Linux machine:
  - The best way is to use a secure shell program **`ssh`** to connect to sunpal, and display terminal and graphics windows on your local X display
  - (You could also use **`telnet`** or **`rlogin`**, but these are not secure)

- From your Windows or MacOS machine:
  - Use a **`ssh`** client such as PuTTY for a secure terminal connection
  - (You could also use a telnet client, but this is not secure)
  - You will not be able to display the grapical output of programs running on sunpal unless you have a X display server running on your machine

Using the Unix command line interface (CLI)

- In your telnet or xterm window, you will see a command line ("shell") prompt that looks like

```
sunpal.ucsd.edu%
```

- ... type a command to the prompt, followed by the return key
- ... a command runs a program; when the program exits, you will get another shell prompt
- ... when you are done running programs, close the windows you do not need and exit your login session by typing

  ```
  exit
  ```

  ... or, on a workstation in SUN-PAL lab, clicking on the little EXIT icon on the front panel. (Don't ever leave your workstation logged in!)

The UNIX operating system

- The machines in the instructional labs for this and many other UCSD courses run (a version of) the UNIX operating system
- To do your assignments and turn them in, you need to know some basics of UNIX
- So, let's look at some of those basics related to the UNIX filesystem

The UNIX filesystem: directories and files

- A UNIX filesystem consists of *directories* and *files*
- Directories are objects that contain other directories and files
- Files are objects that contain data
  - e.g. text, Java code, executable programs, etc.
- The filesystem is implemented in secondary storage (one or more hard disk drives)

The UNIX filesystem: pathnames

- The UNIX filesystem is *hierarchical*
  - On a UNIX machine, there is one directory (only one!) that is not contained in any other directory: the "root" directory, named `/`
  - Every other directory and every file has a unique directory that contains it
  - The full pathname of a directory or file consists of the name of the file, and the name of every containing directory on the path to the root directory, separated by forward slashes slashes:

    ```
    /usr/bin
    ```

    ```
    /home/solaris/sunpal/cs11f/cs11faa/.login
    ```

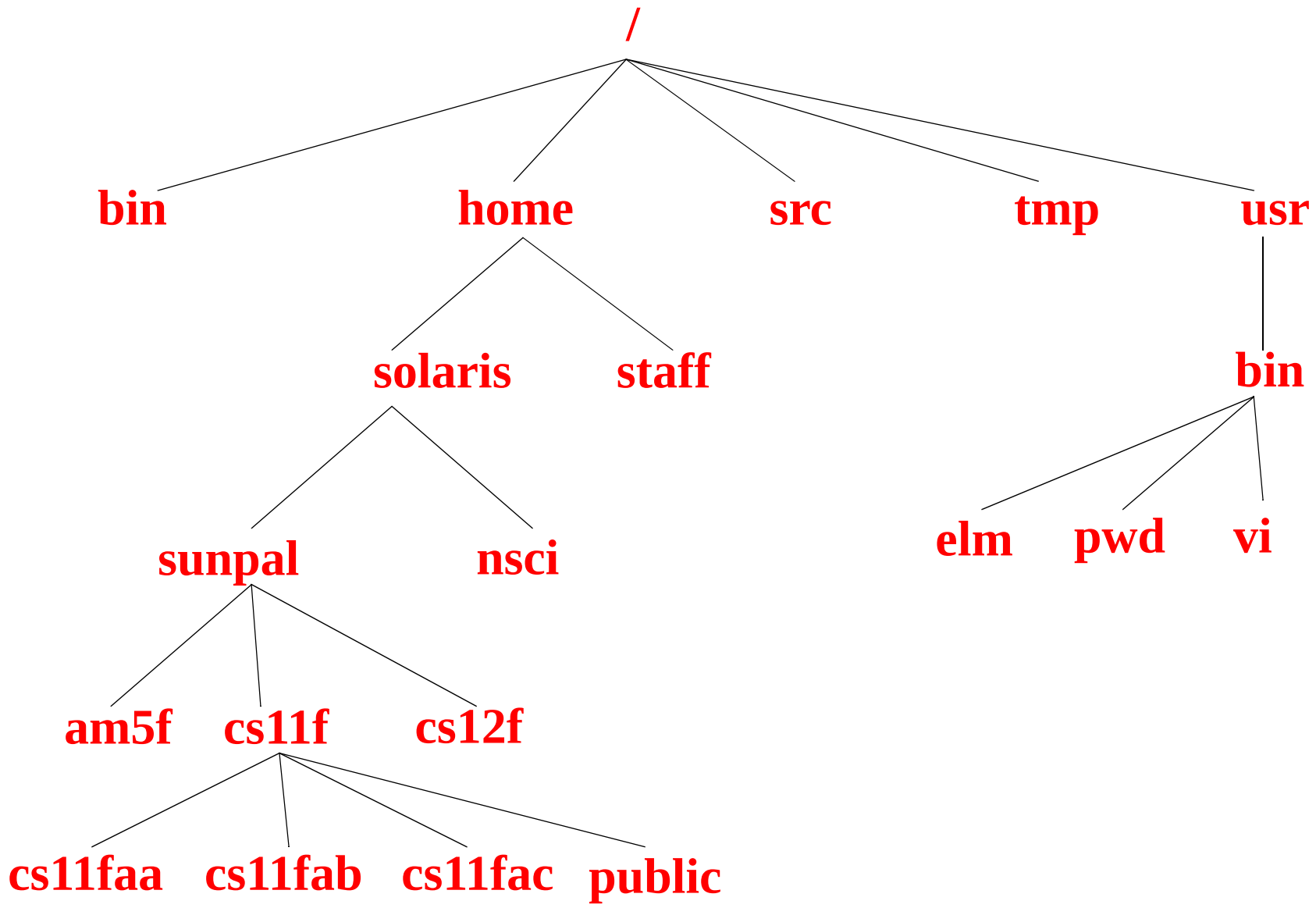
  - In many contexts you do not need the full pathname; some shorthand will do

The current working directory

- When interacting with a UNIX computer, there is always a *current working directory*
  - To print the full pathname of the current working directory to the screen, use the command
    `pwd`
  - To change the current working directory, type the command
    `cd pathname`
    with the pathname of the directory you want to change to
  - To list the files and directories in the current working directory, type the command
    `ls`
  - When you login, the current working directory is your "home directory"
  - Your home directory can be referred to with the shorthand pathname
    `~` (tilde)
  - The directory containing a given directory can be referred to with the pathname
    `..` (double-dot)

    so, **`~/../public`** is a pathname of what directory?...

(Part of) a Unix filesystem hierarchy



Getting information about CSE 11

- All information for the course is linked to from the class web pages:

  `http://sunpal.ucsd.edu/~cs11s`

- To view these pages, fire up Netscape from the CDE front panel if you're using a lab machine, or use your favorite browser from anywhere on the net

- You will want to bookmark that URL, or edit the preferences in your browser to make it your home page

Using an editor

- You will need to know how to use a text editor for creating and modifying the programs you will write in this course, and afterward...
- The two most popular editors in the UNIX world are **`emacs`** and **`vi`**
- **`xemacs`** is a version of **`emacs`** that is especially nice for Java programming
- **`vim`** is a powerful vi-based editor you might want to try
- Tutorials for all these exist online
- (The simple textedit editor on the SunPAL machines is not recommended! Use one of the above instead.)

Using email

- You will receive your programming assignment and exam scores by email sent to your class account, so you should learn how to read email there
- Many mail utilities exist for UNIX
- **`elm`** or **`pine`** are two that are easy to use

Getting help

- For general Unix concepts and commands, see the recommended UNIX textbook

- For online help: type
  `help` ***`topic`***
  to access online help for some topic, or
  `man` ***`command`***
  to access the online manual pages for some command

- Human help:
  CSE 11 tutors are available in the terminal labs; see the web pages for their hours
  Other general-purpose tutors are available: type **`help human`** for more info

- Discussion board:
  We will have a web-based discussion board for asking and answering questions related to CSE 11, linked to from the course home page. Use it!

- Email:
  After you've read the documentation and talked to tutors and consulted the discussion board, email reasonable questions to **`cs11s@sunpal`**

Helpful hints

- Make backups of your files
- Be sure to log out from your lab workstation when you're done
- Change your password to something secure, and don't give anyone your password
- Be aware of lecture, assignment, and exam schedules for the course; CSE 11 moves fast
- Make backups of your files (did I mention making backups of your files?)

For next time...

- ✔ Read chapters 1 and 2 of Savitch
- ✔ Start on Programming Assignment #1
- ✔ Plan to go to a discussion section this week