Slide 1

Administrivia

- Reminder: Homework 1 due Thursday (11:59pm).

  Remember to put course name/number and assignment in subject line of e-mail.

  Also remember to tell me (body of the e-mail is fine) if you worked with another student or got help from someone else.

Slide 2

Minute Essay From Last Lecture

- Several people said “lots of commands to learn” or “lecture moved pretty fast”. Yes. Practice/experiment outside class will help. True for many (most?) things we do in this course. If you have trouble remembering all the commands (which you might at first!): In times past beginners got paper “cheat sheets” of commonly-used commands. Maybe make yourself an electronic equivalent?
- How do you get out of `man`? It’s showing you stuff with the command `less`, and you exit from that by typing `q`.

Slide 3

Command Line Review

- Last time we looked at commands for navigating the file system and working with files (moving, copying, etc.). (Review on next slide.) Other useful/interesting commands in chapter 2. Good to go through the list and try them out for yourself.

  (Yes, if you're sitting in front of the machine you can use the GUI. If you're logged in from somewhere else, the command line may work better.)

- Remember/note that `man` shows you information about a command, and `man -k` shows you a list of commands related to a keyword.

Slide 4

`vi` Tips — Review

- Cut/copy/paste basics:

  `dd` cuts a whole line. `yy` copies a whole line.

  `p` pastes after the current line. `P` pastes before the current line.

- Search by typing , text to search for, Enter. Repeat search with `n`. Search-and-replace using this, `cw`, and `.` (See book.)

- `:help` brings up online help. `:q` to exit.

- For the record — `vi` on our systems is actually `vim` ("vi improved"), much more featureful than "real" `vi`. `:help visual-mode` describes one feature you may like.

Slide 5

More `vi` Tips — Review

- `u` means "undo" the previous action (insertion, deletion, paste). Repeat to undo multiple actions.
- `:q!` exits without saving. Useful if you make a complete mess of things.

Slide 6

`vi` Tips — Errors/Mistakes

- If you type just `q` rather than `:q`, `vi` thinks you want to record a macro. Screen will show "recording". Press `q` to make it stop.
- If you type `q:` rather than `:q`, `vi` thinks you want it to display a history of commands and shows them to you in a subwindow. Type `:q` to make that go away.

Slide 7

`vi` Tips — Errors/Mistakes, Continued

- If you just close the terminal window when running `vi`, that "crashes" `vi`. So what? Well . . .
- `vi` creates a hidden file that saves information that can help with recovery if it crashes. Deleted on normal exit, otherwise not. And then next time you start `vi` on that file — screenful of messages starting "ATTENTION" and "Found a swap file" and finally asking you whether you want to open it anyway or what. If you respond R `vi` will try to recover unsaved changes; otherwise not. To actually delete this hidden file, so you don't get that same screenful of messages next time, respond `D`.

Slide 8

Commands For Working With Files — Review

- Now that we have a way of creating files, we can try out some other basic commands.
- `cat` to show contents of a file. `more` or `less` to show it a screenful at a time.
- `cp` to copy one file to another. `-i` to warn about overwrites.
- `mv` to move or rename a file. `-i` to warn about overwrites.
- `rm` to delete a file. (Note — no recycle bin, so use with caution! or `-i` to prompt.)

Slide 9

UNIX Filesystem Basics

- Unlike in Windows (and Mac?), UNIX filesystems are case-sensitive (so hello and Hello are different files).
- Files have two levels of ownership — "owner" (user) and "group". Groups allow sharing files with some but not all users.
- File access is controlled by "permissions". Three levels (owner, group, and everyone else), three types of access (read, write, execute).
- `ls -l` shows permissions. `chmod` changes them.

Slide 10

Scala

- Scala is short for "scalable programming language". (We may talk more later about what that means.) Relatively new language, but we think good for a first course.
- Various options for running Scala source code. Today we will look at two of them — typing it in interactively, and executing "scripts".

  `scala` starts an interactive environment ("REPL" – "read, evaluate, print" loop), good for trying things out.

  `scala program.scala` runs the program in file `program.scala`.
- By tradition (established by the inventors of the C language, in 1970-something), our first program will just write to the screen "hello, world").

Slide 11

A First Scala Program

- Type `scala` at the command line to start the interpreter.

  Now type

  ```
  println("hello, world")
  ```

  (and press return).

  Try misspellings and other variations. Type `:quit` to end.

- Or we could put that single line in a file `hello.scala` and run it with the command

  ```
  scala hello.scala
  ```

Slide 12

Comparison — Python

- An equivalent program in Python (the language used in some sections of POP I last semester):

  ```
  print "hello, world"
  ```

- Run interactively or as script using command `python`.

Slide 13

Comparison — C

- An equivalent program in C (the language previously used in POP I, when it was PAD I):

```
#include <stdio.h>

int main(void) {
    printf("hello, world\n");
    return 0;
}
```

- No option for running interactively; first compile (command `gcc`) to create executable file, then "run" the file.

Slide 14

Comparison — Java

- An equivalent program in Java (a language often used as a first language in high-school courses):

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("hello, world");
    }
}
```

- No option for running interactively; first compile (command `javac`) to create Java bytecode, then run bytecode using command `java`.

Slide 15

Programming in Scala

- In Scala (as in many, maybe most, programming languages) two of the basic building blocks are expressions (similar to expressions in math) and statements (roughly speaking, complete instructions).
- Before attempting definitions, try a few things . . .
- Start up the Scala interpreter and try typing in a few arithmetic expressions.
- Try making the "hello world" program print a second line.

Slide 16

Programming-Language Terminology

- *token*: set of characters that means something in the language, often separated by whitespace.
- *literal*: token representing a value (e.g., 1).
- *statement*: set of tokens that give a complete action.
- *expression*: set of tokens that together give a value (e.g., 1 + 1).
- *type*: set of values together with operations on them (e.g., integer, (text) string). Every value has a type.

Slide 17

Numeric Literals and Expressions

- Numeric literals and expressions should be fairly familiar. Notice that Scala (like many programming languages) makes a distinction between integers and numbers that have (or might have) a fractional part.
- Use the interpreter's REPL to try out things. Some things may be surprising — integer division, large numbers, calculations using fractions. To understand some of these it helps to know how the computer represents numbers.

Slide 18

Binary Numbers

- We humans usually use the decimal (base 10) number system, but other (positive integer) bases work too. (Well, maybe not base 1.) Binary (base 2) is more widely used in computers because it makes the hardware simpler.
- In base 10, there are ten possible digits, with values 0 through 9.
  In base 2, there are 2 possible digits (*bits*), with values 0 and 1.
- In base 10, 1010 means what? What about in base 2?

Slide 19

Converting Between Bases

- Converting from another base to base 10 is easy if tedious (just use definition).
- Converting from base 10 to another base? Let's try to develop an algorithm (procedure) for that . . .

Slide 20

Decimal to Binary, Take 1

- One way is to first find the highest power of 2 smaller than or equal to the number, write that down, subtract it from the number, and continue:
  1. If $n = 0$, stop.
  2. Find largest p such that $2^p \leq n$.
  3. Write a 1 in the p-th output position.
  4. Subtract 2^p from n.
  5. Go back to first step.
- Is this okay? What's not quite right about it? (We don't say what to put in the positions that don't have ones in them.)
- (Example.)

Slide 21

Decimal to Binary, Take 2

- Another way produces the answer from right to left rather than left to right, repeatedly dividing by 2 (again n will be the number we want to convert):
  1. If $n = 0$, stop.
  2. Divide n by 2, giving quotient q and remainder r.
  3. Write down r.
  4. Set n equal to q.
  5. Go back to first step.
- Is this okay? What's not quite right about it? (We don't say to write down the remainders from right to left.)
- (Example.)

Slide 22

Computer Representation of Integers

- Computers represent everything in terms of ones and zeros. For non-negative integers, you can probably guess how this works — number in binary. Fixed size (so we can only represent a limited range).
- How about negative numbers, though? (To be continued.)

Slide 23

Minute Essay

- Anything today that was particularly unclear?