Slide 1

Administrivia

- Reminder: Homework 1 due today.

Slide 2

Basic Organization / Terminology

- Kernel — heart of operating system, manages processes and files and so forth.
- Shell — program that interprets what you enter, calls (“launches”) other programs.

  This being UNIX, there are several, mostly offering similar functionality but maybe with different syntax.

  Several ways to start a shell — next slide.
- Graphical environments, window managers, etc. Also several of these!

Slide 3

Starting a Shell

- From the console, type control-alt-Fn, where n is 1, 2, . . . 6, and log in. (To get back to the graphical virtual console, control-alt-F7.)
- From a graphical environment, start a "terminal emulator" (`xterm`, `gterm`, etc.). If your desktop has a taskbar, might be good to put a "start a terminal" icon on it. (For GNOME, right click on taskbar, then "add to panel", "launcher from menu", etc.)
- From a Windows system, run `putty`.
- Other ways (log in remotely with `ssh`, . . .)

Slide 4

A Little About Shells

- Several choices; most commonly used are probably `bash` and `tcsh`. By default, you get the one in your entry in the password file.
- How to find out what that is? `echo $SHELL`. (This displays the environment variable `SHELL`. More about those later.)
- How to change? `chsh` command on some systems; on others, can only be changed by administrator.

  Or start a different one by typing its name, like any other command.
- Following discussion is about `bash`, but many other shells offer similar functionality.

Slide 5

What Your Shell Does With What You Type

- Shell provides in-place editing (arrow and other keys), command history, tab completion of filenames, etc. — until you press “return”.
- Shell then processes command line — expands wildcards and references to variables, “tokenizes” command into commandname and parameters.
- Shell then either processes command (if a builtin), or locates executable in “search path” (`PATH` environment variable) and forks off a new process.
- Command’s return code then available via shell variable.
- (Aside: Wonder what a simple shell program looks like? Look at first homework from CSCI 4320 . . .)

Slide 6

What `bash` Does With What You Type — In-Place Editing

- Simple editing — left and right arrows; ctrl-a, ctrl-e, etc.
- Command history — move forward/back with up and down arrows, search with ctrl-r.
- Tab completion — for filenames, command names, etc.
- Read about `bash` and/or `readline` — `man` and `info` pages for more info.

Slide 7

What `bash` Does With What You Type — Processing Command Line

- Shell takes completed line and expands filename wildcards, references to variables (more about both in next slides), "tokenizes" command into commandname and parameters, splitting (by default) at whitespace.
- If that's not what you want — e.g., to include a space in a filename, inhibit expansion of filename wildcards, etc. — use escape character (backslash) or quotes. Single quotes inhibit all of this, double quotes all but variable substitution.

Slide 8

What `bash` Does With What You Type — Processing Command Line

- Shell locates command. Two cases:
  - Builtin command — shell executes directly.
  - External command — shell finds an executable by looking in "search path" (`PATH` environment variable) and forks off a new process.

  (Why the distinction? Some things can't reasonably by done in a new ("child") process!)
- Command's return code then available via shell variable.

  (Why would anyone care? Useful in writing scripts.)

  (Where does the return code come from? whatever is returned by program — e.g., from C program's `main`.)

What `bash` Does With What You Type — Miscellaneous

- Notice that some keys have meanings other than what Windows users are used to — ctrl-C, ctrl-D, ctrl-Z, possibly also ctrl-S, ctrl-Q (depending on environment — e.g., which terminal emulator).

Environment Variables

- Associated with a process (e.g., a shell) there can be “environment variables”. Useful as another way (in addition to command-line arguments, input from file/keyboard, etc.) of giving process information.
- Some variables of interest — `PATH`, `SHELL`, `HOME`, `USER`.
- To display current value, `printenv FOO` or `echo $FOO`.
- To set value, `FOO=value` (no spaces) in `bash`.
- To make value available to other commands, `export FOO`.

Filename Expansion

- You probably already know about using `*` as a wildcard for specifying one or more files. Other options too — “filename expansion” section in full `bash` manual or `info` pages.
- `echo` can be used to check what a particular expression expands to.

Slide 11

Minute Essay

- How is the pace of the class so far? too fast (too much new-to-you info), too slow (too little new-to-you info), . . . ?

Slide 12