Slide 1

Administrivia

- My plan to teach class asynchronously Monday went awry. (My semester is really not off to a good start! But there's still time to improve.) Instead I plan to record one more lecture for this week, for you to watch before next Monday. To be available by Friday (I hope earlier); I'll send e-mail.
- Homework 1 posted; due in a week.
- Notes on options for doing homework added to course Web site under "links".

Slide 2

Files in Linux/UNIX — System Basics

- No notion of "drive letters" — all paths form a single hierarchy, with directory / as its root. (Try `ls -l /`.)
- A key underlying concept — "everything's a file" (sequence of bytes). Directories are files. Devices are represented as "special files" (`ls -l /dev`). Many files are text, including configuration files. (Contrast with Windows registry.) Some "files" are constructed on the fly by the O/S (`ls -l /proc`).
- Removable media can be "mounted" (incorporated into the hierarchy) and "unmounted". Graphical environments may do this automatically when you insert or remove, e.g., a USB drive.
- Windows/DOS "extensions" doesn't really apply, though some commands and some graphical programs do make use of filename suffixes.

Slide 3

Files in Linux/UNIX — User Basics

- We won't review basic commands for navigating and manipulating the filesystem, but you should if you've forgotten (`cd`, `ls`, `cp`, `mv`, `rm`, `mkdir`, `rmdir`).

Slide 4

Files in Linux/UNIX — File Permissions

- Security model is simple but fairly flexible — rights (read, write, execute) for owner, group, others; a few other bits that provide other things.

  UNIX "groups" provide a way to share files among some but not all users. Linux sets up a group for each user; sysadmins can set up additional groups for, e.g., classes or research projects.
- `r` and `w` have obvious meanings; `x` means "can execute" for files, "can `cd` to" for directories. Normally creating or removing files requires write access to *directory*. (Makes sense in a way but may seem surprising.)
- `chmod` to change permissions. Old-style way is use 16-bit value (e.g., `chmod 0777 foo` to give all rights to everyone); newer syntax is symbolic (e.g., `chmod ugo+rwx foo`). Can be applied recursively to directory with `-R`.

Slide 5

Files in Linux/UNIX — File Permissions, Continued

- "Sticky" bit (symbolic name `t`):

  Applies to directories only; means files can be removed only by owner

  Example of use: `tmp`.

- "Set ID"" bits (set user ID and set group ID, symbolic name `s`):

  Applies to files only; means program executes with the permissions of the file/group owner.

  Example of user: `/usr/bin/passwd`.

- `chmod` to change permissions. Old-style way is use 16-bit value (e.g., `chmod 0777 foo` to give all rights to everyone); newer syntax is symbolic (e.g., `chmod ugo+rwx foo`). Can be applied recursively to directory with `-R`.

Slide 6

Files in Linux/UNIX — Links

- "Links" (hard or soft) allow non-tree directory structure. (Analogous to Windows short-cuts.)
- "Soft" (symbolic) link (`ln -s`) is just a special type of file pointing to another file. Allows access through either name, but can "breaks" if pointed-to file isn't there.
- "Hard" (non-symbolic?) link (`ln`) only works within a filesystem but creates a second directory entry to the same underlying file. File itself exists until all (hard) links to it are gone.

Slide 7

Processes in Linux/UNIX

- A key concept in pretty much all operating systems is “process”, loosely defined as one of a set of “concurrently executing” entities (users, applications, etc.)
- Processes can spawn “child” processes. (This happens, e.g., every time the shell runs a command!) Child process cannot change anything in parent (so, e.g., if you `cd` in a script, it only affects the script, not the caller).
- Processes can have “environment variables”, which can be inherited by child processes. Examples — `USER`, `PATH`.
- `ps` to see current process and its children. `ps aux` to see list of all processes. (Marvel at how many!)
- Processes can be terminated with `kill`; `kill -9` to do equivalent of “force quit”.

Slide 8

Processes in Linux/UNIX and “Job Control”

- “The” shell (okay, there are several, but all that I know of) starts a new process for each command. Normally runs “in the foreground” (of the login session).
- Or you can start it “in the background” by putting a `&` after the command. You can also suspend the foreground process with ctrl-Z. (Useful if you want to get back to a command prompt.) Restart a suspended process with `fg`, or put it in the background with `bg`.
- Background and suspended processes get a number; show with `jobs`. Can use this number with `fg`, `bg`, or `kill`.

Slide 9

Homeworks

- First homework is written problems. Some will be that, others programming. E-mail me your answers (TMail address or the one with `@cs`). For written problems *please* send me either plain text or PDF. I'll convert to one of those forms to grade anyway.
- You will also be asked for two more things:
  - An explicit honor-code pledge and a statement about any collaboration or help (if none, say so).
  - A sentence or two reflecting on the assignment.

Slide 10

Minute Essay

- Questions?