Slide 1

Administrivia

- About minute essays: In addition to answering whatever questions I ask, feel free to ask me, well, anything (preferably related in some way to the course or computing).
- About the reading: What may work well is to wait until after class, and then focus on things mentioned in class while still at least skimming other material (might be something that would interest you!).
- Homework 1 on the Web; due a week from Wednesday.

Slide 2

Reading The Fine Manuals

- In times past, testy local experts sometimes responded to questions with "RTFM" (Read The Fine Manual). Well, it's a useful skill, though it's losing ground to STFW (Search The Fine Web).
- One of the most useful things you can learn is how to learn more. Documentation on UNIX systems not always perfect, and not particularly novice-friendly, but usually thorough, and traditionally kept in local files.
- Local documentation:
  - `man` pages.
  - `info` pages.
  - Elsewhere on the system. `locate` on Linux may help.

Slide 3

RTFM — `man` pages

- Reference documentation (as opposed to tutorials).
- Organized into "sections" (user commands, sysadmin commands, library functions, etc.). Can have entry with same name in multiple sections. `-a` option or section number gives access to non-default.
- Of particular interest is the section `SEE ALSO` — sometimes lists other related commands.
- `man -k` (or `apropos`) to search for command names.
- Try `man man` ...
- Now you might want to know about `more`, or `less`.

Slide 4

RTFM — `info` pages

- Also reference documentation, sometimes more current / complete than `man` pages. (Why are there are two systems? Probably historical reasons!)
- Organized in a way somewhat similar to hypertext.
- Try `info info` ...

Slide 5

Other Useful Info-Gathering Commands

- `locate` (fast file-search by name, Linux but maybe not other UNIX).
- `whereis` (finds files associated with a command).
- `file` (tries to identify file contents).
- `which` (looks for command name in search path).

Slide 6

A Little About Files

- A key underlying concept — "everything's a file" (sequence of bytes). Directories are files. Devices are represented as "special files". Many files are text.
- Windows/DOS "extensions" doesn't really apply, though some commands and some graphical programs do make use of filename suffixes.
- Also no notion of "drive letters" — all paths form a single hierarchy. Removable media can be "mounted" (incorporated into the hierarchy) and "unmounted".
- 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 7

A Little About Files – File Permissions

- Security model is simple but fairly flexible — rights (read, write, execute) for owner, group, others; "sticky bit" provides a few more things. (UNIX "groups" provide a way to share files among some but not all users. Linux sets up a group for each user; we sometimes set up additional groups for classes, research projects, etc.)
- `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*.

Slide 8

A Little More About File Permissions

- "Sticky" bit is — interesting? For directories (`t`), means files can be removed only by owner (example `/tmp`). For files (`s`), means program executes with the permissions of the file owner (example `/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 9

A Little About Files – Links

- "Links" (hard or soft) allow non-tree directory structure.
- "Soft" (symbolic) link (`ln -s`) is just a special type of file pointing to another file. Allows access through either name, but can "break" 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 10

A Little About Processes

- Another key concept — process 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", inherited by child processes. Examples — `USER`, `PATH`.
- Processes can be terminated with `kill`; `kill -9` to do equivalent of "force quit".

Slide 11

Processes 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, which you can show with `jobs`. You can use this number with `fg`, `bg`, or `kill`.

Slide 12

Homeworks

- First homework is written problems. Some will be that, others programming.
- 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 what if anything was noteworthy about the assignment.

Slide 13

Minute Essay

- Anything today particularly unclear, or that you want to hear more about?