Slide 1

Administrivia

- Homework 7 on the Web. Goal of the assignment is to get you to try out features useful in writing technical/scholarly papers. I tried to make it somewhat open-ended so you can do things *you* might find useful or interesting.
- Right now due date is next Wednesday, but we can extend that if by Monday it's clear you'll need more time.

Slide 2

Minute Essay From Previous Lecture

- I asked last week about topics, and — several responses.
- One prompted today's lecture (though I had planned to probably do this anyway).
- Another asked about system administration. *Maybe*, but this is an area where UNIX-like systems can vary a lot, and even within Linux it depends on "distribution".

Slide 3

A Few More LaTeX Tips

- `\usepackage[utf8]{inputenc}` for non-ASCII input.
- `\usepackage[normalem]{ulem}` for underlining.
- `\usepackage{quotes}` to automatically convert double-quote characters to LaTeX version of "smart quotes"
- `\usepackage{hyperref}` to make all cross-references into hyperlinks and include support for other hyperlinks.

Slide 4

Running Things "In Absentia"

- You already know how to run programs on a Linux (or other UNIX) computer without being physically present — remote login.
- Can you also run programs without being "present" even remotely? Yes . . .

Slide 5

One-Time Batch Work

- `at` and `batch` allow you to put “jobs” (sequences of commands) in a queue for later execution. `batch` says “run when system load permits”. `at` says “run at specified time” (lots of options for that — look at `man` page).
- `atq` shows queued work. `atrm` allows cancelling previously-scheduled work.
- Both of these send stdout and stderr by e-mail. On your own system, this may be straightforward. On the classroom/lab machines, simplest way to make this work may be to forward mail to your TMail account. To do this, make a plain-text file `~/.forward` with the forwarding address.

Slide 6

Scheduled Work

- Background daemon `cron` executes “jobs” at scheduled intervals — every minute, hour, day, etc. (These days it often seems to be `anacron`, which takes into account the fact that systems may not be continuously on).
- What jobs? System-related jobs are those in `/etc/cron.daily` etc. There are also user-specific “tables” listing other jobs.
- To schedule something, as administrator you could put something in one of those `/etc/cron.*` directories. Or . . .

Slide 7

Scheduled Work, Continued

- But `cron` also makes user of those user-defined tables, managed via `crontab`. (**CAUTION:** Don't try this on one of our machines until I check with the "real" sysadmin!)
- Syntax for `crontab` entries is somewhat arcane, but documented in `man 5 crontab`.
- Output of these "`cron` jobs" goes to e-mail, as with `at`.
- The environment (including environment variables) for these jobs may be somewhat different from what you have in a shell. Probably best not to assume too much, for example about `$PATH`.

Slide 8

Work Started Interactively

- You've probably(?) observed that if you start a command and then close the terminal where you started it, the command stops.
- One way around this is with command `nohup`. Type `nohup` followed by the command, which should probably redirect all three standard streams (stdout, stderr, *and* stdin), followed by `&`.
- Another way is to use the command `screen` . . .

Slide 9

`screen` — A Text-Based Window Manager(!)

- `screen` is . . . a "virtual virtual terminal", a "text-based window manager", something that multiplexes a physical terminal betwen several processes, usually interactive shells.
- Supports one or more "windows" (programs, usually shells), plus one or more "regions" (areas on screen).
- Functionality includes
  - Ability to leave programs running even if "real" terminal isn't there — i.e., disconnect/reconnect.
  - Ability to copy and paste text among windows, log stuff, etc.

Slide 10

`screen` Basics

- `screen` starts things up. By default, no visual cues that you're in a `screen` session. Probably a good idea to have a simple configuration file (`~/.screenrc` to change that. (There's one on the "sample programs" page.)
- Commands to `screen` start with `control-a`. (To send an actual `control-a` to a program such as `emacs`, repeat.)
- `control-a d` detaches session. `screen -r` to resume.
- `exit` exits a "window".
- `control-a ?` shows key bindings.

Slide 11

More `screen` Basics

- `control-a c` creates a new window. (`exit` to end.) `control-a "` gives menu for selecting window.
- `control-a S` creates a new region ("splits the screen"). `control-a X` to end. `control-a Tab` switches to next region.
- Fully documented in `man` page, or try `http://www.gnu.org/software/screen`. (Worth noting that this is for the GNU version of the command; some UNIX-like systems (Mac??) have a non-GNU version, which is less featureful.)

Slide 12

Minute Essay

- Can you think of things for which you might use one or more of the tools discussed today?