Slide 1

Administrivia

- Reminders: Homework 7 due today, Homework 8 next week.
- LaTeX graphics examples added to “sample programs” page.

Slide 2

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 3

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, may be straightforward. (Or it may not. The traditional setup, with mail delivered locally, probably not the norm these days.)

  On the classroom/lab machines, simplest way to make this work probably to forward mail to your TMail or other account. To do this, make a plain-text file `~/.forward` in your home directory with the forwarding address.

Slide 4

Scheduled Work

- Background daemon `cron` executes "jobs" at scheduled intervals — every minute, hour, day, etc. (These days sometimes really `anacron`, which takes into account 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 5

Scheduled Work, Continued

- But `cron` also makes user of those user-defined tables, managed via `crontab`. (`-l` to list, `-e` to edit). (**CAUTION:** If you try this one of our machines, be sure you try out the job interactively first. Runaway jobs could make trouble for other users, and you might not notice right away.)
- 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`.
- Environment (including environment variables) for these jobs may be somewhat different from interactive shell. Probably best not to assume too much, e.g., about `$PATH`.

Slide 6

Work Started Interactively

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

Slide 7

`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.
- (Dr. Fogarty uses this in class I believe? one of you asked about something he does to show a program and execution in a single terminal window?)

Slide 8

`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`, `control-a a`.)
- `control-a ?` shows key bindings.

Slide 9

`screen` — Windows and Regions

- `control-a c` creates a new window running your default shell. Exiting the shell (`control-d` or `exit`) closes it. Closing the last window exits `screen`.

  `control-a` twice switches windows, or `control-a "` if more than two.
- `control-a S` splits a window into two regions. `control-a TAB` switches between them. (Second region seems to start out blank. Switch to it and select a window as above.) `control-a X` exits a region.

Slide 10

`screen` — Detaching and Reattaching

- `control-a d` detaches session. `screen -r` to resume.
- Session also automatically detached if you're logged in remotely and the login times out. Note however that if the machine is rebooted the session ends.
- *Note also* that if you do this from ITS's VDI Linux virtual desktop, the screen session ends when your VDI session ends. *But* you can work around this by first using `ssh` to connect to one of our Linux-only machines (e.g., the DIAS cluster) and running `screen` there.

Slide 11

`screen` — Copying and Pasting

- `control [` to initiate copy-to-buffer. Move cursor to start of area to copy, press space, move to end, press space again.
- `control ]` to paste copied text.

Slide 12

`screen` — Where to Learn More

- `man` page (long but maybe thorough?), 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 sadly is much less featureful.)
- From the `man` page:

  "A weird imagination is most useful to gain full advantage of all the features."

Slide 13

Minute Essay

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