Slide 1

Administrivia

- No project. Homework 6 on the Web. Should be very easy. Due in a week. We need a “not accepted past” date.
- (Status of grade information.)
- Information about office hours this week and next coming by e-mail soon.

Slide 2

“What Command Do I Use To . . . ”

- You know about `apropos` as a way to discover new commands. You probably also know that it’s not perfect.
- So in the following slides, a tour of some commands I have found useful . . . (We won’t have time to discuss in any depth, but perhaps useful or interesting to review later?)
- (The point of this tour is not to present details of any of the commands, just to make you aware they exist, so you can follow up on those that seem useful.)

Slide 3

Commands for Working With Text and Other Data

- `script` to capture all terminal input/output. (`exit` to stop capturing.)

  Not as useful as it might be because you also get stuff to control terminal, make colors, etc., but could be a good approach if you need to capture both input and output.

- `strings` to search a file for printable strings.

  Can be useful as a quick-and-dirty (i.e., not necessarily 100% reliable) way of scanning non-text files (e.g., files in MS Office formats) for printable text.

Slide 4

Commands for Working With Text and Other Data, Continued

- `ispell` or `aspell` to check/correct spelling.
- `od` to show data in various forms (binary, hexadecimal, etc.). Useful for finding out exactly what's in a non-text file. Examples:

  `od -t c textfile` to show characters including line-end and other control characters.

  `od -t x1 somefile` to show data a byte at a time in hexadecimal format.

Slide 5

Commands for Printing

- `lpr` to print PostScript, PDF, or text. Add `-P` and a printer name to specify the printer (e.g., `lpr -Pstylus foo.pdf`).

  `lpq` to check print queue; `lprm` to cancel a print job (`-P` to specify a printer here too. If these don't seem to work on our systems, try them on Sol, which does the actual printing.)
- `enscript` or `pr` to pretty-print text. Options allow printing in landscape mode with different font sizes, in multi-column format, etc.

Slide 6

Commands to Compress and Archive Data

- `gzip` and `gunzip` to compress/uncompress data. Or try `compress` and `uncompress` (not available on our Linux machines, but found on many UNIX systems).
- `tar` to create UNIX-standard-format "archive" file, a.k.a. "tarball". (Conceptually similar to ZIP archive files — which you can generate, using `zip`.)

  Another way to copy a directory, preserving symbolic links:

```
(cd sourceDir; tar cf - . ) | \
    ( cd target; tar xf - )
```

Text-Mode Calculators (`bc` and `dc`)

- Useful in that both support arbitrary precision. (So, if you want to know *exactly* what 2^{100} is . . .)
- I sometimes use from within `vim`, for quick calculations.

Slide 7

Commands for Accessing Other Machines

- `ssh` to remotely log in / run commands.

  `-Y` flag allows running X-based (GUI) programs. (Also `-X`, but that may be less secure.)

  `ssh user@machine` logs in as a (possibly) different user.

  `ssh user@machine "command"` to execute single command (or commands). *Note* that this may bypass some of normal shell setup (e.g., reading `.bash_profile`.

  Can set up so it doesn't prompt for a password. Link to instructions on "Useful links" page.
- `rsh` and `telnet` to provide similar functionality, but less securely. Often turned off by sysadmins for that reason.

Slide 8

Slide 9

Commands for Copying Files Between Machines

- `scp` to copy file(s) between machines. If you set up `ssh` to not prompt for password, applies to this too.
- `rsync` to "synchronize" a target file/directory with a source. Useful in maintaining backup. Example:

```
rsync -avz --delete /users/yourName/ \
    /directory-for-backup
```

  Precede directory/file with `user@machine:` to copy to/from remote machine. `-e ssh` may be needed in order to use SSH rather than RSH.

Slide 10

Commands for Working with Programs

- `-E` (show preprocessor output) and `-S` (generate assembly-language output) flags on most compilers.
- `gdb` source-level debugger. Semi-graphical version available from within `emacs`.

Slide 11

Web-Related Commands

- `wget` or `curl` to download a Web page or pages. `wget` has options to allow downloading a page and everything it references.
- `lynx`, `elinks` (a.k.a. `links`), or `w3m` to browse in text mode.

  (What's the difference? Slightly different capabilities. Better help (IMO) in `lynx`, but it doesn't do frames. `elinks` does frames but help isn't (IMO) as useful. I haven't tried `w3m`!)

  (Advantage of these commands is that they're lightweight and ignore some things that make graphical Web browsing annoying. Disadvantage is also that they ignore some things . . .)

Slide 12

Miscellaneous Other Command(s)

- `time` to run a command and say how long it took. (Actually there's often more than one thing by this name, e.g., a shell built-in and also a command. Access the latter with a full path name or by preceding the name with a backslash (in `bash` anyway).)
- `top` monitors performance in realtime.

(Mostly-)Text-Mode Plotting (`gnuplot`)

- Usually run in graphical mode, but interface is text-only. Help available from within program by typing `help`. (Help is modeled after online help on VAX VMS operating system, and is — different.)
- Can also be run in batch mode — e.g., if you want to be able to easily regenerate plots when data changes.
- Nice for LaTeX users because it can produce output in various LaTeX-friendly formats (including ones that allow final typesetting to use same fonts as document).
- Examples on sample programs page.

Sending Mail from the Command Line

- Simplest / most primitive program for sending (and reading) mail is `mail`. Pretty reasonable for sending pre-composed text-only messages. Example:

```
echo "this is a test" | mail -s "test" bmassing@cs.trinity.edu
```

- What about attachments? `mail` doesn't really "do" MIME. Next slides . . .

Sending Mail from the Command Line with Attachments

- Use `mail` with an old-style mechanism for encoding files as plain text:
  - Encode files to attach with `shar`. Recipient pipes message body through `unshar`.
  - Encode files to attach with `uuencode`. Recipient pipes message body through `uudecode`.
- Use another text-mode MUA (e.g., `mutt` or `pine`) that's "scriptable" and understands MIME. Example:

```
echo "here is my file" | mutt -a somefile -s "my file" -- bmassing@cs.trinity.edu.
```

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 17

A Text-Based Window Manager, Continued

- You know what to do to find out more . . . (Or try `http://www.gnu.org/software/screen`.)
- Some useful info for getting started:
  - `screen` starts things up.
  - `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.
  - `control-A ?` shows key bindings.
  - `control-A D` detaches session. `screen -r` to resume.

Slide 18

Miscellaneous Fun (?) Stuff

- Some selected text-based games installed in `/users/cs3190u/games/games`. "Caveat user" — installation was quick-and-dirty, so not all games work 100%, but interesting as a sampling of what games were like in the (not-always-so-)Good Old Days.
- And of course there's

  `telnet towel.blinkenlights.nl`

  (If you try this — just type the above, and be patient. Use control-] (possibly followed by pressing return) and then "quit" to interrupt it if you need to.)

Slide 19

Course Wrap-Up — What I Hope You Got From This Class (Details)

- More things in your "bag of tricks" (see later slide).
- Practice in reading man pages and otherwise learning more.
- Exposure to traditional tools you might need, or want, to use sometime. You may not remember details, but I hope you will be less intimidated.

Slide 20

Course Wrap-Up — What I Hope You Got From This Class (Big Picture)

- Exposure to a different operating system / user interface paradigm — many small programs that work together, information kept in text files, emphasis on being expert-friendly and scriptable, etc.

  "A tour of UnixWorld / TextWorld."

  "More than one way to do things."
- Encouragement to find out how to use all your tools as intelligently as possible.

  I like the old tools because I know how to make them work together. But it's worth noting that many "modern" tools (GUI-based programs, graphical file managers, etc.) have their own way of working together — common set of keybindings, cut-and-paste metaphor, drag-and-drop, multiple selections, etc. Compare and contrast!

Slide 21

Course Wrap-Up — Topics

- Shell features — command history, redirecting input and output, scripting features (if/then/else and loops).
- Pipes.
- Filter programs (`awk`, `sed`, `grep`, etc.).
- Text editors.
- LaTeX.
- `make` and makefiles.
- Regular expressions (for text editing, `grep`, etc.).

Slide 22

From First Lecture — Shameless Evangelism/Ranting

- “UNIX is obsolete — history goes back to 1969!”

  You can fix a lot of bugs in 40 years, and the odds are better that what you learn will still be useful years from now.
- “It’s not user-friendly!”

  Sure it is; it’s just choosy about its friends. Designed by programmers for programmers — “expert-friendly” as opposed to “novice-friendly.”
- “Everyone knows GUIs are better!”

  For some things and some people, maybe so. But which is more expressive, pointing and gesturing or speech?
- (You don’t have to agree with me! but in theory now you have more information on which to base on opinion.)

Slide 23

From First Lecture — The UNIX Philosophy

- As stated by one of its developers (Doug McIlroy):

  “Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.”

- There’s more, but the emphasis is on (1) providing a set of lightweight tools that can be put together to do interesting things, and (2) providing choices to users (sometimes almost too many!)

Slide 24

Minute Essay

- None — sign in.