Slide 1

Administrivia

- Homework 2 on Web; due Monday.
- Notice that answers to non-opinion minute essay questions (e.g., first question from Monday) are in final version of notes online.
- Reading assignments meant to be skimmed — read carefully only parts that we talked about in class, or that interest you.

Slide 2

More Filters

- `sed` — "stream editor" — non-interactive program, by default does *not* edit in place, but works as a filter, transforming input to produce output.
- Some simple uses:
  - Search and replace:
    ```
    sed 's/old/new/g' infile >outfile
    ```
  - Delete lines containing some string:
    ```
    sed '/this/d' infile >outfile
    ```
    (How else could you do this?)
- Especially useful with regular expressions (later).

Slide 3

More Filters, Continued

- `awk` — implementation of programming language AWK — "pattern scanning and processing language".
- Lines of AWK program specify pattern and action. (Can also include function definitions.)
- Basic processing — split each line of input ("record") into "fields", compare to patterns in program, execute actions for any patterns that match.
- Simple uses:
  - Print selected fields from input (as in examples from last time).
  - Print selected lines of input:
    ```
    awk '/this/' infile
    ```
    (How else could you do this?)

Slide 4

More Useful Commands

- `find`. Very powerful/flexible, though if you don't use it often you probably will have to read the man page to remember syntax.
- Simple examples:
  - Find all files in the current directory modified in the last week.
    ```
    find .  -mtime -7
    ```
  - Find all files in your home directory whose name contains `hello`.
    ```
    find $HOME -name "*hello*"
    ```
  - Find all files in the current directory that end in `.bak` and apply `rm -i` to them.
    ```
    find .  -name "*.bak" -exec rm -i {} \;
    ```

Slide 5

More Useful Commands, Continued

- `diff` — compare files or directories. (A good use — "regression testing" of programs.)
- `pushd`, `popd` (actually shell built-ins) — manipulate shell's stack of directories.

Slide 6

More Useful Commands, Continued

- `xargs` — "build and execute command lines from standard input".
  - Find all processes for program `java` and kill them:

```
ps aux | grep java | awk '{print $2}' | xargs
kill
```

Slide 7

Shell Input as a Programming Language

- What `bash` understands is in a sense a programming language, with the shell as its interpreter:
  - Variables (untyped).
  - Expressions (arithmetic and logical).
  - Conditionals (if/then/else) and loops.
  - Functions.
- Can be used interactively, or collected into "scripts".
- I will talk about `bash`, but most shells provide similar functionality, just sometimes with different syntax. If you want to write scripts portable to most Unix systems, probably best to stick to `sh` subset of `bash`.

Slide 8

Shell Scripts

- A "shell script" is just a sequence of things you could type at the shell prompt, collected in a (text) file.
- Normally, first line of script is `#!` followed by path for shell (`/bin/bash`, e.g.), and the file is marked "executable" (with `chmod`). But you can also execute commands in file `anyfile` via `bash anyfile`.
- With the exception of the first line, lines starting with `#` are comments.

Slide 9

Minute Essay

- Write a command to find all the files in the current directory (and subdirectories) that are less than a week old and list them in reverse order by modification time (i.e., newest to oldest).

Slide 10

Minute Essay Answer

- The solution I had in mind was

```
find .  -mtime -7 | xargs ls -lt
```

but there are undoubtedly other ways!