Welcome to CSCI-E26:
Intro to Programming in C/Unix with Web Interfaces

Bruce Molay molay@fas.harvard.edu
Brandon Williams bowill@g.harvard.edu

Tonight: Intro to topics, goals, and methods of the course
Agenda:

a. brief summary of course
b. demonstration of Unix programming
c. detailed description of course
d. demonstration of Unix as operating system
e. demonstration of CGI Unix programming
f. summary and wrapup - The big idea

News:

1. Unix intro: Thu 7:30-9 in Bruce's Office Hours
2. Please do infoform web form tonight
3. Sunday workshop: 2-5 pm
   in Bruce's Office Hours
   on Zoom page of Canvas site.

A. Brief Summary of Course

This course is an introduction to C and Unix programming with web programming as an example of how to use C/Unix. The time will be roughly 60% C programming, 25% Unix/Shell programming, and 15% HTML/CGI programming. These numbers are not exact because I sometimes teach two topics in combination at once.

This is an intro to C/Unix, not an intro to programming. If you do not know how to write computer programs, you are not prepared for the course. An intense desire to learn the material or friends who know how to program won't help. Please take an intro to programming course first, then come back. I have been teaching this course since 1987, I am likely to be here next year.

B. Demonstration of Unix Programming

How many of you folks ride the commuter rail to work or to get here? Say you wanted to find the trains you could take after class. We can look at the MBTA web site to see.

Notice how you can select a line, a direction, and a day to see a schedule. You can examine the schedule to figure out which train to take. The site also has a trip planner. There, you can type in the staring point and ending point and when you want to arrive or leave and the computer tells you which trains and connections to take. You've probably seen that sort of thing for air travel and for driving directions. In fact, a lot of web usage is travel related. How does that work? Could you build a travel planning web site? By the end of this course you should be able to.

B. Demo of Unix Programming - continued

A Data Set

Say it was several years ago and this website was being planned.
In preparation, someone typed the printed scheduled into a spreadsh
and produced this plain-text file:

TR=002;dir=i;day=m-f;TI=05:37;stn=campello;Line=middleborough
TR=002;dir=i;day=m-f;TI=05:41;stn=brockton;Line=middleborough
TR=002;dir=i;day=m-f;TI=05:44;stn=montello;Line=middleborough

We look at the sched file: a list of all arrivals/departures
Compare these events to entries in the schedule.
A trip is sequence of these events.
We can see all trains stopping in brockton by searching for
stn=brockton
We can see all stops on train 017 by searching for TR=017

B. Demo of Unix Programming - continued

Operations on this Data Set

- clean it (find typos, missing data, format errors..)
- analyze it: get stats and info
- report: generate schedules
- search : trip planning
- webify : build remote access to reports, stats, and searches

How Would You Do it?

Some of you have worked with datasets. What tools would you use, given this plain text file, to clean, analyze, report, and provide web access to this data set? How long would it take?

3. Unix as Programming Language: 12 Questions to Answer

In particular, if you had to answer these 12 questions, how would you do it and how long would it take:

1. When do trains leave from Braintree going to Boston?
2. What is the time of the earliest train from Ashland to Boston?
3. How many trains stop at West Medford on a weekday?
4. List the stations on the Fitchburg line.
4.5 List the stops with the most trains
5. List all the lines in the system.
6. List the train numbers of all trains passing through Beverly Depot.
7. What station has the most trains on Sunday?
8. Which line has the most stations?
9. During what hour does the greatest number of trains arrive at South Station?
10. When does the last train to Worcester leave Boston?
11. What is the most common train stop time?
12. What is the longest train trip (time, not distance) on the system?

solutions f23 in text file

B. Unix as Programming Language: 12 Questions to Answer

Here is a Unix programming solution to this problem. We use exactly six simple programs, all available on any Unix system:

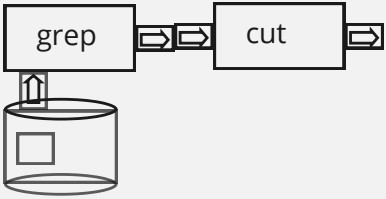
Simple Tools (programs that perform an action)

---

grep - find and print (line oriented)
cut - extract fields from lines
sort - sorts based on any selection of fields
uniq - removes (and/or counts) duplicate lines
head - selects first n lines
wc - counts chars, words, lines

Combined into Pipelines

Demo grep: search for stn=brockton
Demo cut: select the time field
Idea of pipeline: assembly line processing



[temp]

pipeline syntax: grep stn=brockton sched | cut -d";" -f4

Now Answer the 12 Questions by using six tools and pipelines

Note: grep vs python comparison (see 5_Code for lect01)

C. Detailed description of course

We go through the handout

course structure, homeworks, help, exams, academic honesty
workload - can be heavy
focus: C programming to build tools, scripts/pipelines to use them
HTML, CGI to run them over the Net

D. Unix for Users

Unix is an operating system that manages files, directories, programs, and users. The basic operations are:

Big Idea:

login to the system
run commands (programs) to
manage files, process data
logout

login/logout
directories and files
ls, mkdir, rmdir, cd, mv
files
cat, more, rm, mv, cp
programs
i/o redir
IDE - vi/vim/emacs, cc, gdb
Unix at home; Linux, BSD,

File Transfer

see scp handout
vs-code can do it
filezilla -- click to get non-adware versionv

Programming Environments

classic: editor/compiler/debugger
IDEs - vs-code, CLion,

Backup: Your responsibility

git
keep original at home, backup there

More Unix Programming:
tools, pipelines**, scripts**

Storing pipelines in files to create new tools

E. Scripts: Building New Tools and Putting them on the Web

Once we figure out a pipeline to answer a questions we want to save that pipeline so we can reuse it. We do that by putting the pipeline in a script:

```
#!/bin/sh
# train-times for a station and direction
grep "stn=brockton" sched | grep "dir=i" | cut -d";" -f4
```

But that only works for inbound through brockton. To make it more general, we would like to have a tool that works like

```
train-times salem o
```

To do that, we put in parameters for the things to vary:

```
#!/bin/sh
# train-times for a station and direction
# usage: train-times station-name {i|o}
grep "stn=$1" sched | grep "dir=$2" | cut -d";" -f4
```

TODO: modify this so it also allows you to specify m-f, sa, or su

E: Putting a New Tool on the Web

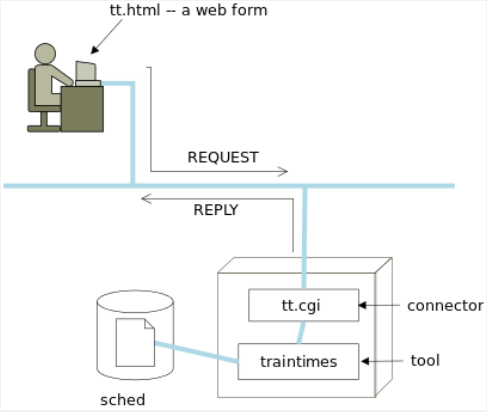
---

train-times works well from the Unix prompt but most people use the web. How do we allow a user to specify the arguments to this script and get the results?

Answer: (1) Web form and (2) back-end connector

web form: train-times.html

connector: train-times.cgi



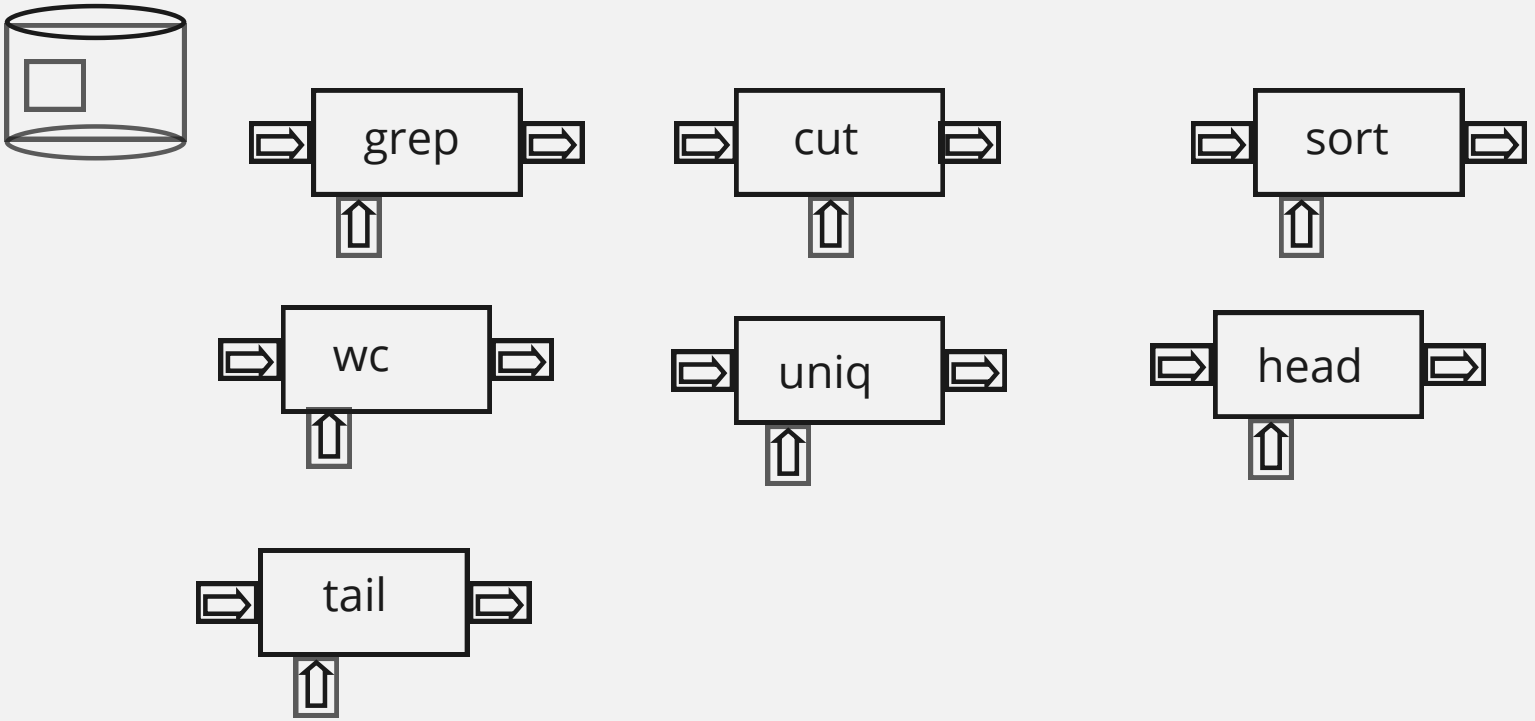
Remarks: The system works, but the report format is crude. We need to process the output to make it look nicer. Therefore we need to find or write some tools to manage the output format.

F. The big picture

In this class we've seen what Unix/C/CGI programming is about

a. there are tools to perform specific operations
b. tools are often written in C
c. pipelines and scripts combine those tools to solve problems
d. HTML forms and CGI connector scripts allow remote access
e. The skills to learn:

C programming to build tools to process text
shell scripts/pipelines to combine tools into programs
HTML pages to accept user input
shell scripts to pass user input to Unix programs


grep
cut
sort
wc
uniq
head
tail