Topic: Strings, CGI, and Structs

Approach: Explore Basics, Improve Website

Overview and Intro:
We have seen four levels of code: form, script, tools, connector
Today focus on C data structures and form improvements

Warnings about Strings
Much Unix programming involves strings. Be careful with strings.
a. Operations =, ==, <, + are not supported, Use functions
   see string.ops.c
b. All string ops require enough memory.
   Serious problems may occur if you over-run memory.
   see memdemo1.c and memdemo2.c

Improving the Train Website
The website has two parts:
Requests: from forms, Replies: from scripts
1. Nicer, More Robust User Input
   train-times with select, just like at mbta.com
2. Nicer, HTML Output
   train-times with html replies
      content-type, echo for html, pipelines to process
      learn about sed
   trainsched with html replies
      content-type, echo for html, pipelines to process
      learn about case
      learn about using scripts as functions - footer
      To do: the html reports need better column format (need tables)
3. Integrated pages: find list of trains, look up schedule

Long-Term Project
Trip Planner - computer figures out trains, times, durations
We can make a nice form, but what about the back-end?
No obvious Unix tools for solving it. We might need new tools:
some C programs to analyze train trips. To do so,
we need data structures to represent train trips.

Data Structures for Train Stops: structs
Basic Structs: just like classes in Java or C++
structdemo.c
How to declare a new type:
Creating variables
Referring to members of the structs
Note: structs may be assigned and passed by value, unlike arrays

Data Structure for Train Trips: array of structs
Creating an array
referring to elements
sample: readtrip.c