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

Quick Review of Passing Vars and Arrays
a. passingdemo.c -- arrays always by ref, others by value

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 overrun 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

How Could We Store All These Records?
Ans: an array of trainstops
We know about arrays, but what about storing a trainstop?

Storing Sched Events: structs
Each sched event is a record with several values
How can we store a combination of values?
C has the struct to store a collection of different types
Struct basics
   type definition, instance creation, member reference
   passing and returning, assigning
Example: structdemo.c

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