CSE 303:
Concepts and Tools for Software Development

Dan Grossman

Spring 2005

Lecture 4— Shell Variables, More Shell Scripts

Where are We

We understand almost all of the C-shell and its “programming language”.

Final pieces:

- Shell variables
- Environment variables
- More programming constructs
  - Loops
  - Word lists

Shell variables

We already know a shell has state: current working directory, users, aliases, history.

Its state also includes *shell variables* and *environment variables*.

Features:

1. Change variables' values
2. Add new variables
3. Remove variables
4. See if a variable exists
5. Variables "set" but without values

Nonfeature: Local variables

Only (1) is similar to "real" programming languages

Variables

```
set
set i = 17
set
echo $i
set | grep i
set i
echo $i
unset i
echo $i
```

Word Lists

Finally, there is something sorta like an array, but more flexible.

With a word lists, you can use the whole lists, extract ranges, extract individual words, etc.

Generalizing an earlier concept: `argv` is just a variable set to a wordlist and `$`i is just shorthand for `$argv[`i`]`.

Two Essential Variables

There are two predefined shell variables we have been using on every line of every shell interaction:

- `path`
  - How the shell finds what program to run
  - The first thing to check when a command does not do what you think
  - The `which` built-in
- `prompt`
  - Have fun specializing yours
  - Not set in scripts, so `$?prompt` is 0 iff the current shell is noninteractive.

Another you have seen me use: `history`

Examples

- `make_thumb` using a variable to hold the outfile
- `makenfiles` using a variable for a loop
- `limittmp` using a variable to hold a temp-file name
- `mypushd`, `mypopd`, `mydirs` via aliases and a shared variable

Quoting and Variables

- Normal expansions will happen before setting a variable unless you quote.
- Variables get expanded inside double-quotes
- Variables do not get expanded inside single-quotes
- Variables get expanded and then filename metachars get expanded

For our mystack aliases, single quotes were crucial.

Environment Variables

Remember that scripts run in their own shell.

- They get environment variables from their “parent shell”.
- They can set them, but parent will not see the effect.
  - Child shells will.

Most useful thing:

- shell var `path` initialized to environment variable `$PATH`.
- So set `PATH` when you log in and every shell will see it.

Note syntax for `setenv` is different than for `set` (sigh).

Shell Programming Revisited

How do Java programming and C-shell programming compare?

The shell:

- “shorter”
- convenient file-access, program-execution, pipes
- crazy quoting rules and syntax
- also interactive

Java:

- local variables, modularity, typechecking
- real data structures, libraries, regular syntax

Rule of thumb: Don’t write shell scripts over 100 lines?

More on Shell Programming

Metapoint: Computer scientists automate and end up accidentally inventing (bad) programming languages. It's like using a screwdriver as a pry bar.

HW2 in part, will be at the limits of what one should do with a shell script (and we'll end up cutting corners as a result)

There are plenty of attempts to get "the best of both worlds" in a scripting language: Perl, Python, Ruby, ...

Personal opinion: it raises the limit to 1000 or 10000 lines? Get you hooked on short programs.

Picking the C-shell was a conscious decision to emphasize the interactive side and "how bad programming can get".

Next: Regular expressions, `grep`, `sed`, `find`.