Shell Programming (Part 1)

Devin J. Pohly <djpohly@cse.psu.edu>

Clarification

Vim + Make

- Vim integrates with make
  - Write file with `:w`
  - Type `:make target` (or just `:make`) to build
  - Vim will read the output and jump you to errors
- Next error: `:cnext`
- Previous error: `:cprev`

Common errors

- MakeFile is different from Makefile!
- "With multiple files" error from pattern rule: try $<
- Parity sort is even and odd, not + and -

Shell programming

- aka “shell scripting,” “Bash scripting”
- What is it?
  - Series of commands
  - Programming with programs
- What for
  - Automating
  - System administration
  - Prototyping

A sample script: shello

- First line: interpreter
  - The `#!` is important!
- Comment: `#` to end-of-line
- Give the file execute permission
  - `chmod +x shello`
  - Not determined by file extension
  - Typical: `.sh` or none
- Run it
  - `./shello`

```
#! /bin/bash

# Greetings!
echo Shello world

# Use a variable
echo Shello "$USER"

# Set a variable
greetz=Shellutations
echo "$greetz world"
```

Shell variables

- Setting/unsetting
  - `export var=value`
  - No spaces!
  - `unset var`
- Using the value
  - `$var`
- Untyped by default
  - Behave like strings
  - (See `help declare` for more)

Special variables

- Change shell behavior or give you information
  - `PWD`: current directory
  - `USER`: name of the current user
  - `HOME`: the current user's home directory
    - Can often be abbreviated as a tilde (`~`)
  - `PATH`: where to search for executables
  - `PS1`: Bash prompt (will see later)

Exercise

- Make a directory `~/bin`
- Move `shello` script there
- Prepend the directory to your `$PATH`
  - `PATH=~/bin:$PATH`
- Change to home dir
- Run by typing `shello`

Shell initialization

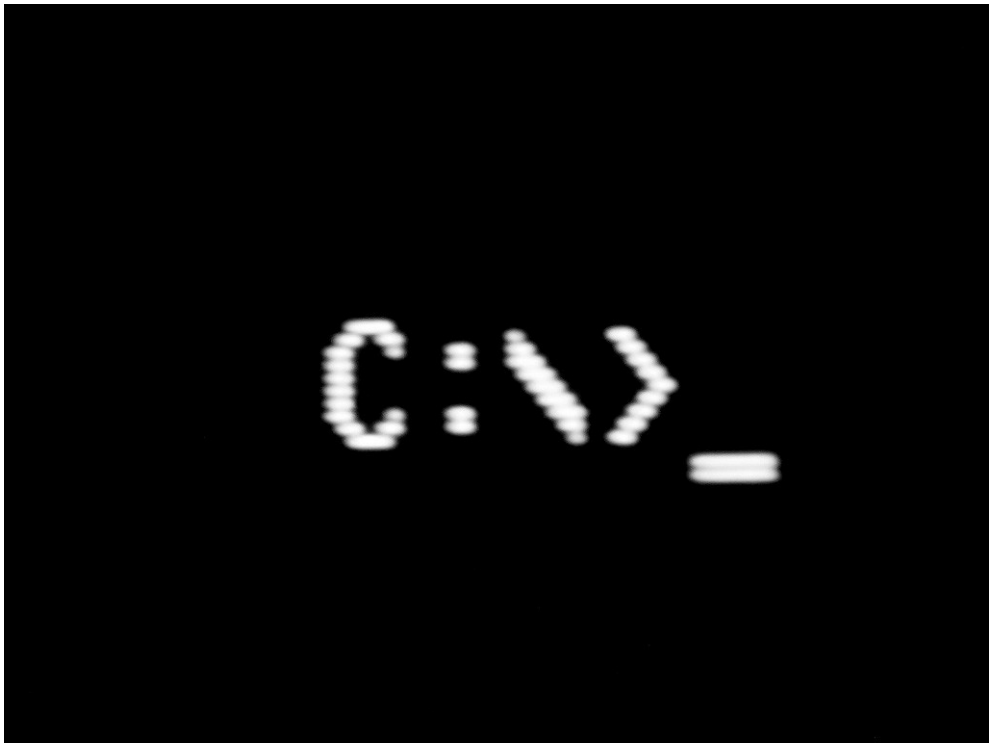
- Set custom variables
- At startup, bash runs shell commands from `~/.bashrc`
  - Just a shell script
- Can do anything you want

- Note: edit this and add:

```
export LD_LIBRARY_PATH=.
```

Fun with prompts

- Main prompt: `$PS1`
- Special values
  - `\u`: username
  - `\h`: hostname
  - `\w`: working dir
  - `\e`: escape (for colors)
  - many others
- This is something you can set in your `.bashrc`



Very detailed treatment: http://www.tldp.org/HOWTO/Bash-Prompt-HOWTO/

Aside: Vim initialization

- At startup, Vim reads ':' commands from `~/.vimrc`
  - Color scheme
  - Indenting preferences
  - Remapping keys
  - Appearance preferences
    - Line numbers
    - Highlighting matched parentheses, braces, etc.
    - Titlebar in terminal

```
colorscheme koehler
set title
set autoindent
set showmatch
set autowrite
filetype plugin indent on
syntax on
```

Special characters

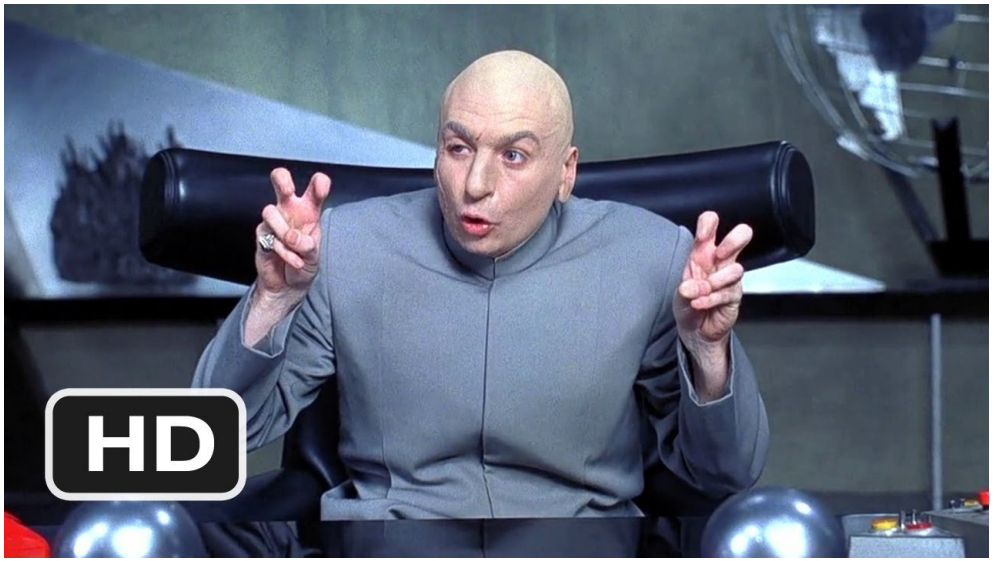
- `echo Penn State is #1`
- `echo Micro$oft Windows`
- `echo Steins;Gate`
- What happened?

Special characters

- `echo Penn State is #1`
- `echo Micro$oft Windows`
- `echo Steins;Gate`
- What happened?
- Many special characters
  - Whitespace
  - #$*&^?!~'`"\{}[]<>()|;
- What if we want these characters?

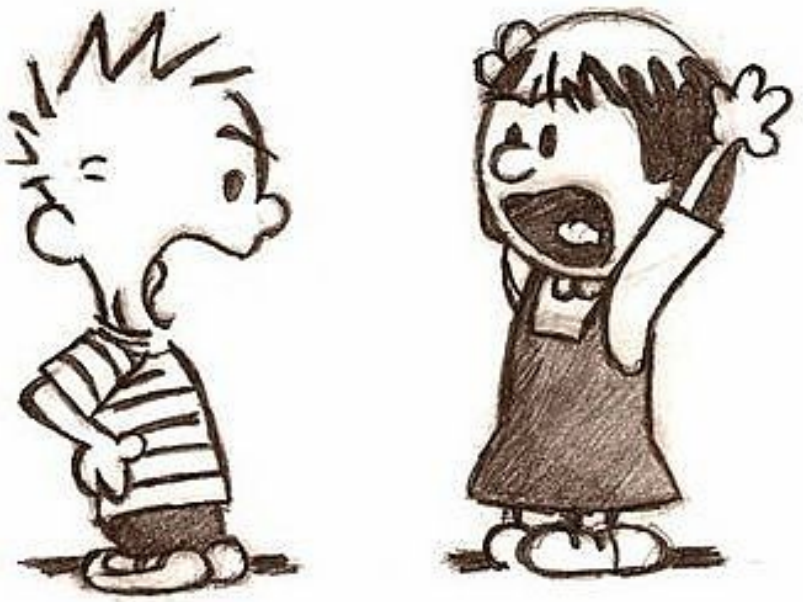
Quoting

- Removes specialness
- Hard quotes: `'...'`
  - Quote everything except closing `'`
- Soft quotes: `"..."`
  - Allow variables (and some other things)
  - Good practice: `"$var"`
- Backslash (escaping)
  - Quotes next character



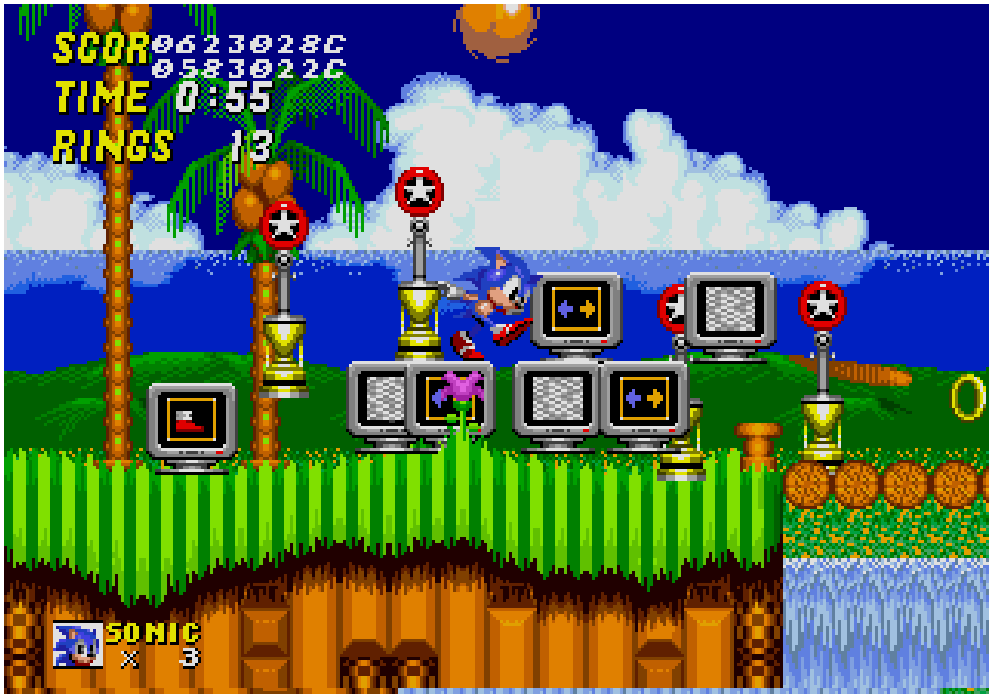
Arguments

- In C: `argc` and `argv[]`
- Split at whitespace
  - How can we override this?
- Arguments to a script
  - `./script foo bar`
  - `$#` is the same as `argc`
  - `"$@"`: all args
  - `"$1"` to `"$9"`: individual



Debug mode

- Shows each command
  - Variables expanded
  - Arguments quoted
- Run with `bash -x`
  - Temporary – just for that run
  - `bash -x shello`
  - Use `-xv` for even more info



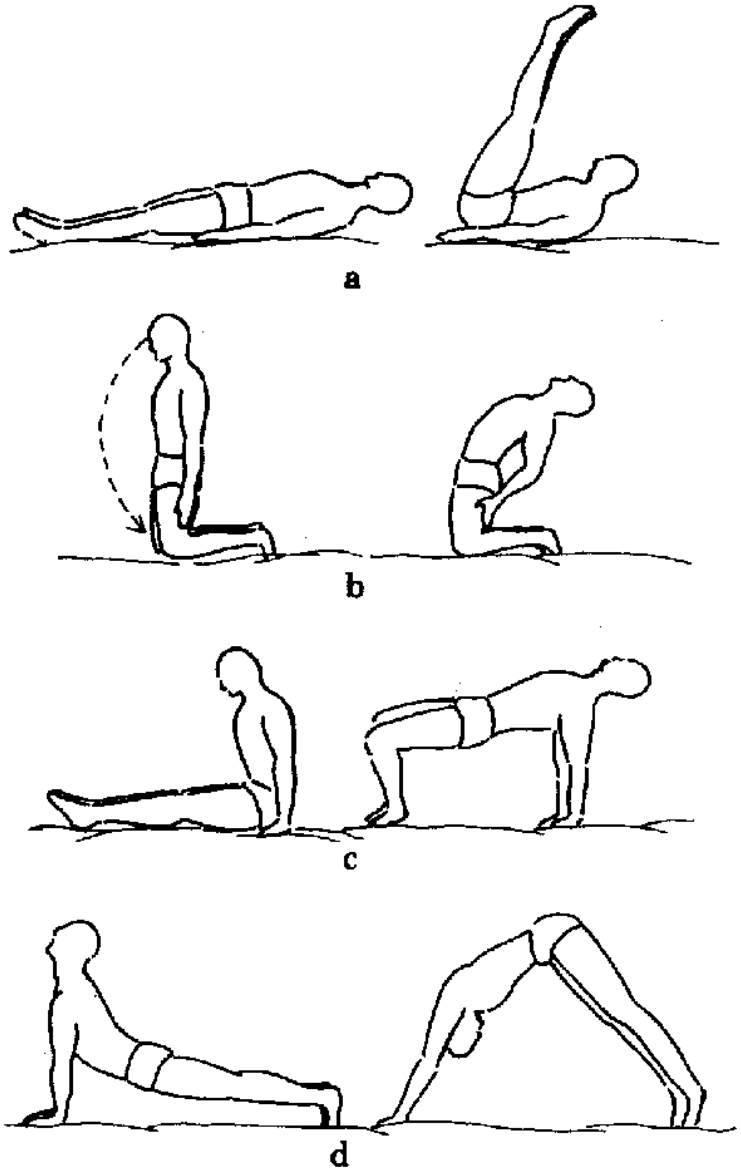
Exercises

- Make a script that:
  - Prints its first argument doubled

```
./script1 foo
foofoo
```

  - Prints its first four args in brackets, one per line

```
./script2 "foo bar" baz
[foo bar]
[baz]
[]
[]
```



Redirecting input/output

- Assigns stdin and/or stdout to a file
- `echo hello > world`
- `echo hello >> world`
- `tr h j < world`

Pipelines

- Connect stdout of one command to stdin of the next
- `echo hello | tr h j`
- `... | rev`
- `... | hexdump -C`
- `... | grep 06`
- UNIX philosophy at work!