

## Intro to Unix

Tom Nichols

October 19, 2001

1

## Outline

---

- Overview/Intro
  - Modifying your environment
  - Simple Aliases
  - Simple Scripts
- 
- This is an very quick intro
- For more, see  
<http://www.sph.umich.edu/~nichols> "Unix Tips"

2

## Intro: Shell Basics

---

- The Shell
  - Just anoter program
  - Accepts your keystrokes
  - Runs programs on your behalf
- More than one!
  - Everything here is about the C-shell (csh)
  - Others include
    - Bourne shell (sh)
    - T-shell (tcsh)
    - Korn-shell (ksh)

3

## Intro: Shell Basics

---

- Most basic shell commands
  - cd - Change directory
  - Ex cd ~  
*Change to home directory*
  - pwd - Print working directory
  - echo - Print some text
  - Ex echo Hello world
- Most basic non-shell command
  - List contents of directory ls

4

## Intro: Wild Cards

---

- Special characters are "expanded" by the shell...
- \*
  - Matches zero or more characters (any)
  - Ex ls \*foo  
*Lists all files ending in foo, including yofoo & foo*
- ?
  - Matches exactly one character (any)
  - Ex ls foo-?ar  
*Lists files such as foo-bar & foo-har  
but not foo-ar*
  - Ex ls foo.???  
*Lists files with three character suffix*

5

## Intro: Variables - Shell variables

---

- You can define variables in the shell
  - Much like in Splus or any language.
  - Unlike Splus, you must mark a variable with a special character... \$
- Define with *set*, evaluate with \$
  - Ex set food = doughnuts  
echo Yummie \$food  
*Sets variable food to have value "doughnuts"  
then prints "Yummie doughnuts"*

6

## Intro: Variables - Shell variables

---

- Only can set to a single “whitespace separated token”  
Ex `set food = doughnuts are good food`  
`echo Yummie $food`  
*Creates an error or just prints “Yummie doughnuts”*  
Ex `set food = "doughnuts are good food"`  
`echo Yummie $food`  
*Prints “Yummie doughnuts are good food”*

7

## Intro: Variables - Shell variables

---

- Some variables are used by the shell  
→ `home, user, prompt, path`  
Ex `echo I am $user and I live at $home`  
*Prints I am nichols and I live at /afs/sph.umich.edu/usr/n/nichols*  
Ex `set prompt = YourWishIsMyShellCommand`  
*The shell will prompt you with that corny string instead of “>”*  
Ex `set path = ($path ~/bin)`  
*Tells the shell to look for programs in your own “bin” directory*

8

## Intro: Variables - Shell variables

---

- You can set variables to save on typing  
Ex `set list = "prog.c dood.c blah.c prog.h"`  
`ls -l $list`  
`cp $list /tmp`  
`lpr $list`

9

## Intro: Variables - Environmental variables

---

- Almost identical to shell variables, but they  
→ Use different definition command, `setenv`  
→ Persist, they get passed to programs you call
  - Shell variables are *local*
  - Environmental variables are *global*
- By convention, always in all CAPITAL LETTERS
- Used by system to control defaults  
Ex `setenv PRINTER student_lp matlab`  
*This sets the default printer; Matlab will know about it*

10

## Modifying Your Environment: .cshrc/.login

---

- Any shell settings we discussed so far are not *persistant*  
→ If you set the `prompt` variable and logout, when you log back in it will be gone
- `.login` & `.cshrc`  
→ Shell *scripts* executed every time you `login (.login)` or start a new shell (`.cshrc`)  
→ Should only have to put things in `.login`  
→ But to keep life simple, put things in `.cshrc`
- To permantly change the prompt  
→ Add this to the end of `.cshrc`  
`set prompt =`  
`"YourWishIsMyShellCommand: "`  
→ Note the quotes; allows the extra space

11

## Modifying Your Environment: .cshrc/.login

---

- If you mess up `.cshrc`, you cannot log in!
- *Always* make backups before making a change  
→ `cp .cshrc .cshrc_save`
- Modify `.cshrc` *very* carefully!  
→ After changing it, use another telnet window to make sure you can still log in.  
→ If you can't log in in the new telnet window, use your old one to do `mv .cshrc_save .cshrc`

12

## Simple Aliases

---

- Aliases most powerful aspect of C-shell
- Allows you replace long, complicated commands with shorthands
- Syntax
  - `alias aliasname aliascommand`
- Common aliases

```
alias rm rm -i
```

*Every time you type `rm, rm -i` is actually used; asks if you're sure about deleting files*

```
alias cdthere cd
```

`/some/really/long/annoying/path/name`

*Just like typing the command.*

13

## Simple Scripts

---

- Scripts are a way to automate repetitive tasks
- To create a script, create an empty file, say, `myscript` that begins

```
#!/bin/csh
```
- And tell UNIX that it's "executable"

```
chmod +x myscript
```
- Then when you type "`./myscript`" it will run
  - The "`./`" is needed to tell the shell that the script is here, in this directory

14

## Simple Scripts

---

- Best to create a "`~/bin`" directory and put it in your path (see above)
- For example
  - Put `myscript` in `~/bin`
  - Then, regardless of what directory you're in, typing `myscript` will run it!

15

## Simple Scripts

---

- Simplest script: `ps2pdfs_v1`

```
#!/bin/csh
#
# Convert Fig1 to PDF
#
#

ps2pdf Fig1.ps
```
- Just runs those commands

16

## Simple Scripts

---

- Simplest script: `ps2pdfs_v2`

```
#!/bin/csh
# Convert several figs to PDF

set list = (Fig1.ps Fig2.ps Fig3.ps)

foreach f ($list)

    ps2pdf $f

end
```
- Uses a list variable and "foreach"

17

## Simple Scripts

---

- Simplest script: `ps2pdfs_v3`

```
#!/bin/csh
# Convert specified figs to PDF

foreach f ($*)

    ps2pdf $f

end
```
- Let's user set the files to use

18

## Simple Scripts

---

- Modifiers
  - Cute ways to modify shell variables
- `set file = "/group/nichols/Fig.eps"`
- `echo $file:r`
  - Removes suffix: `/group/nichols/Fig`
- `echo $file:e`
  - Gives suffix: `eps`
- `echo $file:t`
  - Gives filename: `Fig.eps`
- `echo $file:h`
  - Gives directory: `/group/nichols`

19

## Simple Scripts

---

- Simplest script: `ps2pdfs_v4`

```
#!/bin/csh
# Convert specified PS files to PDF,
# added "_new" to name

foreach f ($*)

    # Take off suffix
    set base = $f:r

    ps2pdf $base.ps ${base}_new.pdf

end
```

20

## Conclusions

---

- Start good habits
  - Write lots of comments
  - Write enough so that you can understand *exactly* what's going on six months from now
- Be very careful about modifying `.cshrc`
- SPH has great UNIX support
  - <http://www.sph.umich.edu/phisa/tech/docs.html>

21