



UNIX shell environments

CS 2204

Class meeting 4



Shell characteristics

- Provides command line as an interface between the user and the system
- Is simply a program that starts automatically when you login
- Uses a command *language*
 - Allows programming (shell scripting) within the shell environment
 - Uses variables, loops, conditionals, etc.
 - Next week



Various UNIX shells

- sh (Bourne shell)
- ksh (Korn shell)
- csh (C shell)
- tcsh
- bash
- ...
- Differences mostly in scripting details



The Korn Shell (ksh)

- We will be using ksh as the standard shell for this class
- This will be important for shell scripting assignments
- Language is a superset of the Bourne shell (sh)



Changing your shell

- On most UNIX machines:
 - `which ksh` (note path)
 - `chsh`
- On the lab machines:
 - `which ksh` (note path `/bin/ksh`)
 - `ypchsh`



Environment variables

- A set of variables the shell uses for certain operations
- Variables have a name and a value
- Current list can be displayed with the `env` command
- A particular variable's value can be displayed with `echo $<var_name>`
- Some interesting variables: `HOME`, `PATH`, `PS1`, `USER`, `HOSTNAME`, `PWD`



Setting environment variables

- Set a variable with `<name>=<value>`
- Examples:
 - `TERM=vt100`
 - `PS1=myprompt>`
 - `PS1=$USER@$HOSTNAME:`
 - `PS1="multiple word prompt> "`
 - `PATH=$PATH:$HOME`
 - `DATE=`date``



Aliases

- Aliases are used as shorthand for frequently-used commands
- Syntax: `alias <shortcut>=<command>`
- Examples:
 - `alias ll="ls -lF"`
 - `alias la="ls -la"`
 - `alias m=more`
 - `alias up="cd .."`
 - `alias prompt="echo $PS1"`



Repeating commands

- Use `history` to list the last 16 commands
- Use `fc -l <m> <n>` to list commands `m` through `n`
- Use `r` to repeat the last command
- Use `r <string>` to repeat the last command starting with `string`



Editing on the command line

- Some command lines can be very long and complicated - if you make a mistake you don't want to start all over again
- You can interactively edit the command line in several ways
 - `set -o vi` allows you to use vi commands to edit the command line
 - `set -o vi-tabcomplete` also lets you complete commands/filenames by entering a TAB



Login scripts

- You don't want to enter aliases, set environment variables, set up command line editing, etc. each time you log in
- All of these things can be done in a script that is run each time the shell is started
- For ksh:
 - `~/ .profile` - is read for a login shell
 - `~/ .shrc` - is read for login and other interactive shells



Example .profile (partial)

```
# set ENV to a file invoked each time sh is
  started for interactive use.
```

```
ENV=$HOME/.shrc; export ENV
```

```
HOSTNAME=`hostname`; export HOSTNAME
```

```
PS1="$USER@$HOSTNAME>"
```

```
alias 'll'='ls -l'
```

```
alias 'la'='ls -la'
```

```
alias 'ls'='ls -F'
```

```
alias 'rm'='rm -i'
```

```
alias 'm'='more'
```

```
set -o vi
```

```
echo ".profile was read"
```



stdin, stdout, and stderr

- Each shell (and in fact all programs) automatically open three “files” when they start up
 - Standard input (stdin): Usually from the keyboard
 - Standard output (stdout): Usually to the terminal
 - Standard error (stderr): Usually to the terminal
- Programs use these three files when reading (e.g. `scanf()`), writing (e.g. `printf()`), or reporting errors/diagnostics



Redirecting stdout

- Instead of writing to the terminal, you can tell a program to print its output to another file using the `>` operator
- `>>` operator is used to append to a file
- Examples:
 - `man ls > ls_help.txt`
 - `Echo $PWD > current_directory`
 - `cat file1 >> file2`



Redirecting stdin

- Instead of reading from the terminal, you can tell a program to read from another file using the < operator
- Examples:
 - `Mail user@domain.com < message`
 - `interactive_program < command_list`



Pipes and filters

- Pipe: a way to send the output of one command to the input of another
- Filter: a program that takes input and transforms it in some way
 - `wc` - gives a count of words/lines/chars
 - `grep` - searches for lines with a given string
 - `more`
 - `sort` - sorts lines alphabetically or numerically



Examples of filtering

- `ls -la | more`
- `cat file | wc`
- `man ksh | grep "history"`
- `ls -l | grep "bowman" | wc`
- `who | sort > current_users`