Shell Basics

,

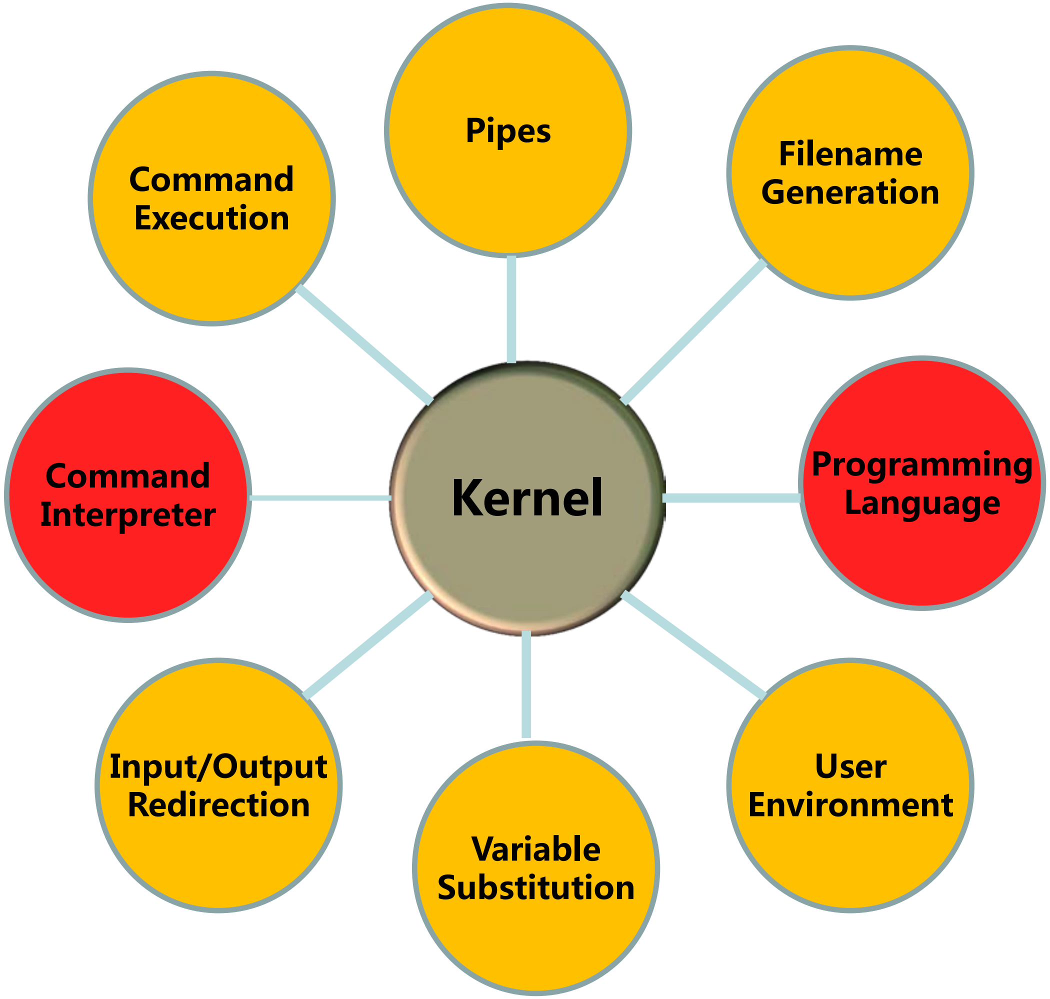
Topics

- The command line syntax
- How the shell interprets command line
- Change file and directory permissions
- Direct and redirect output
- Explain the differences between a stand-alone utility and a shell built-in

Linux Shell

- Shell provides a user interface to the kernel
- Similar to DOS but DOS has only one set of interface while Linux can select different shell: **`bash`**, **`tcsh`**, **`csh`**, **`zsh`**, etc.
- Different shell has similar but different functionality
- Bash is the default for Linux
- GUI is just an application on Linux

The Shell Components



Pipes

Filename Generation

Programming Language

User Environment

Variable Substitution

Input/Output Redirection

Command Interpreter

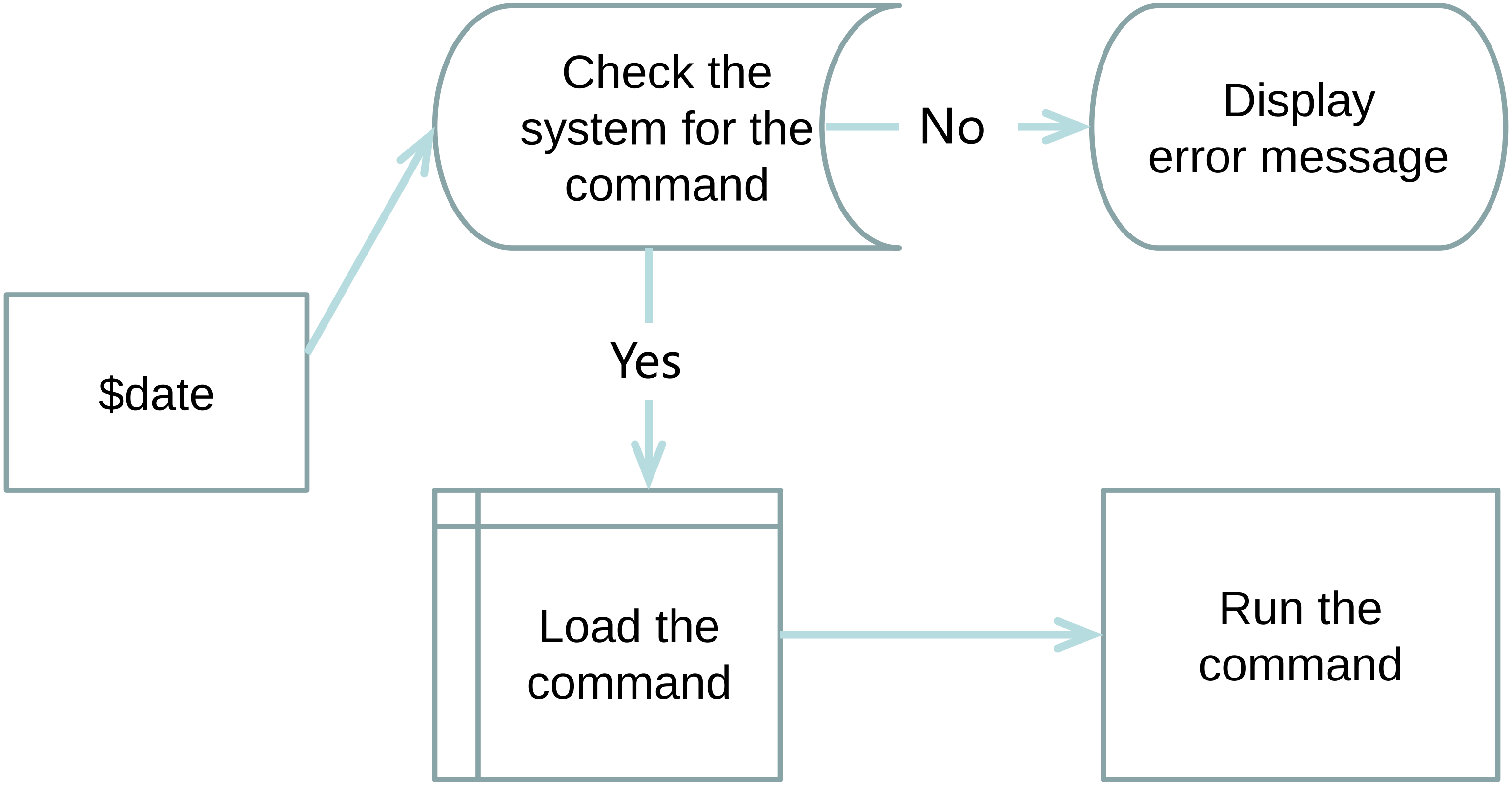
Command Execution

Kernel

Shell Command Forms

Syntax	Effect
`cmd &`	Execute cmd in background
`cmd1;cmd2`	Execute multiple cmds on the same line
`cmd1|cmd2`	Pipe; use output from cmd1 as input to cmd2
`` cmd1 `cmd2` ``	Command substitution; use cmd2 output as arguments to cmd1
`cmd1 $(cmd2)`	Command substitution; nesting is allowed
`cmd1 && cmd2`	AND; execute cmd2 only if cmd1 succeeds
`cmd1||cmd2`	OR; execute cmd2 only if cmd1 fails

Command Interpretation and Execution



Variable and Filename Substitution

- Variable substitution assigns values to variables
- Wildcard (metacharacters) for filename abbreviation

Pipes

- Use the output of one command as the input of another command
- Eliminate the need for temporary files

Redirection Forms

File descriptor	Name	Abbreviation	Typical default
0	Standard input	stdin	Keyboard
1	Standard output	stdout	Screen
2	Standard error	stderr	Screen
/dev/null	Data sink		Screen

Input/Output Redirectors

Redirector	Function
`>file`	Direct standard output to `file`
`<file`	Take standard input from `file`
`>>file`	Direct standard output to `file;` append to file if it already exists
`<>file`	Use `file` as both standard input and standard output
`>&n`	Duplicate standard output to file descriptor n
`<&n`	Duplicate standard input from file descriptor n
`&>`	Direct standard output and standard error file
`<<text`	Read standard input up to a line identical to text (text can be stored in a shell variable). Input is usually typed on the screen or in the shell program. Commands that typically use this syntax include cat, echo, ex, and sed. If text is enclosed in quotes, standard input will not undergo variable susbtitution, etc.

noclober: No Overwriting

- Redirecting output can destroy a file

```
$cat orange pear > orange
cat: orange: input file is output file
```

- By setting noclobber, you prevent a file from being overwritten by redirection:

```
$set –o noclobber
```

Environment Variables

- Environment variables define user interaction behavior.
  - System-wide configuration files (such as /etc/profile)
  - User-specific configuration files (e.g., ~/.profile)
- Discover: **`env or printenv`**
- Add or change: **`set`**

Customization of a Session

- Each shell supports some customization
  - User prompt
  - Email address
  - Shortcuts (alias)
- Customization takes place in startup files
  - Startup files are read by the shell at start
  - Startup can differ for different shell

Bash Startup Files

Startup File	Function
`/etc/profile`	Out-of-the-box login shell settings
`/etc/bash.bashrc`	Out-of-box non-login settings
`/etc/bash.bashrc.local`	Global non-login settings
`~/.bash_profile`	Login shell user customization
`~/.bashrc`	Non-login shell user customization
`~/.bash_logout`	User exits from interactive login shell

Filename Generation

- Wildcards

Builtins

- Utilities built into a shell
- The shell does not fork a new process in running a builtin
- To display a list of bash builtins, give the following command:

```
info bash shell builtin
```

alias and unalias

- alias is a bash builtin:
- Assign a shorthand as a synonym :

```
alias dropbox='cd ~victoryu/cis18a/dropbox‘
```

- Remove aliases with unalias

The Shell Script

- A file that contains Unix commands

```
#!/bin/sh
echo "Hello World!"
```

- To run a shell script:

```
$helloworld.sh
```

- Shell script can perform a complex series of tasks or a repetitive procedure quickly

Positional Parameters

- Command name and arguments are positional parameters
  - $0: the command
  - $1-$9: the arguments
- Positional parameters can be referenced by the commands in the script

```
#!/bin/sh
echo "Hello $1 $2"
echo Today is `date +"%a, %b %d, %Y"`
```

Online Bash Reference

See

http://www.gnu.org/software/bash/manual/bashref.html

Summary

- Shell provides a character-based user interface for the Unix kernel
- A shell is both a command interpreter and a programming language
- The shell also supports the following features:
  - Variable substitution
  - Filename generation
  - Pipes
  - Input/output redirection
  - User environment customization