The Shell - Chapter 5

Monday, February 2, 2015 11:36 AM

Objectives

- Understand Meta Characters
- Describe command line syntax
- Understand how the shell interprets the command line
- Redirect I/O
- Understand "piping" data

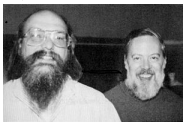
Review

- When you need to find out how a particular command runs? What "helper" command would you place in front?
- How would you "read" this command?
  ls -la /etc
- Name 8 common linux commands
- Fedora 21 is using what Desktop Environment?
- When and why was the "sudo" command created?
- What is a "distro spin"?
- What is a partition?
- What is a file system?
- Name 3 basic partitions every Linux system

requires:

- Name the 3 major design philosophies of Linux/UNIX
- Name these people and their contributions





- Name the two major Linux Distro Families

Shell

↳ This chapter is important

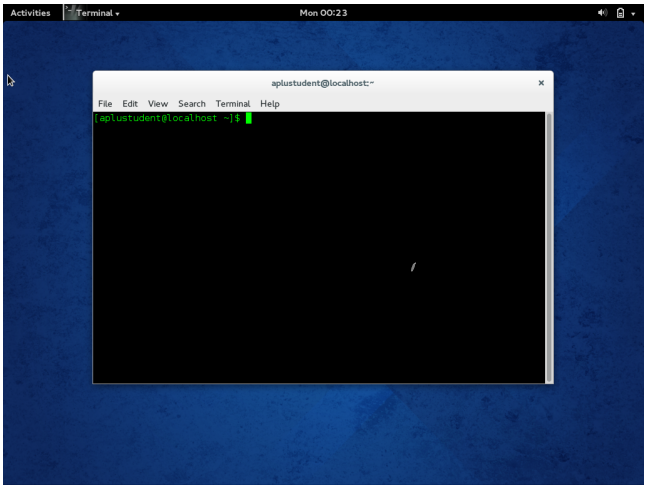
↳ this is the core of Linux

↳ Reason for this class

↳ Here you will learn the shell syntax and how commands are processed by the system.

Shell. PII

- Shell is how you interact with the kernel.
- You never issue commands directly to the kernel.
- When you open a "terminal"



- It is it's own memory space
- You can have multiple terminals/shells open
- "Terminals" came from the days of mainframes, where you didn't have PCs just terminals w/remote

- Variables declared in one shell are not accessible in other shells

are not accessible in other shells. access.

- Shells are Interpreters

what ever you type is parsed for known keywords
- cp, mv, cat, ls etc etc
- Also special characters
  - $; | * ? ' " ` [ ] ( ) $ < > { } # & ! ~
- Whitespace too
- Some characters are "escape"
  - ↳ this tells interpretter to "skip" special meaning and treat it as a printed character.
  - ↳ example echo "Is this thing on?"
    and
    echo " Is this thing on?"
    what happens?

---

ordinary Files + Directory Files

↳ Every thing in Linux is a file? T or F
(true)

↳ From the shell you can always find your location in the filesystem by typing: pwd present working directory

↳ As soon as you login as a user - you are taken to a place under the filesystem tree called your home directory:
/home/aplusstudent
also ~ (tilde) is a shortcut for your home directory

↳ This leads to something called system PATH (chapter 6)

↳ Your home directory is the only place you

→ [illegible] your home directory [illegible] place on the system you own... (like your home)

---

ommand Line

→ A command Line comprises a simple command, a pipeline, or a list (P. 158) (P. 162)

→ Remember whitespace seperates elements

→ Some commands are helpful
  ↳ If you type a command without a required element
    ↳ For example: mkdir [return]
    ↳ What happens?
    ↳ What argument is required?

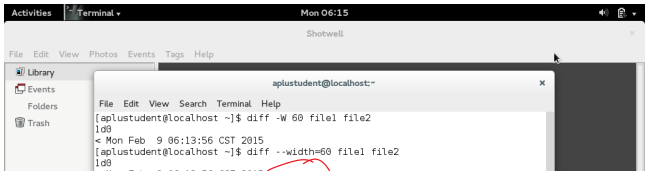
options are always prefaced with a single '-'
  ↳ or by a double '--' that is followed by an englishword.
  ↳ For instance: ls -r (lists a directory content inverse sorting)
    Verbose : ls --reverse

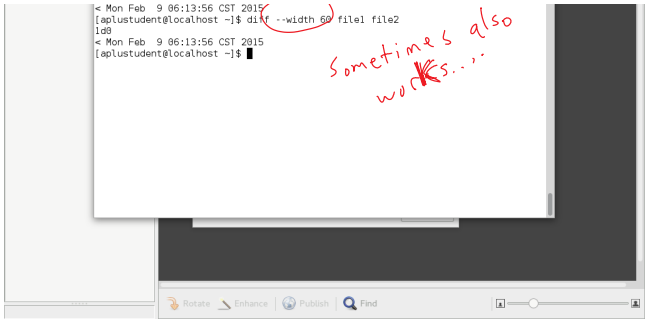
Same

↳ ls -xr, ls -rx, ls -r -x ← SAME too

↳ Most Linux utilities have a --help builtin
  (Those that are GNU tools ← that is 95%)

↳ The '-' and '--' have a significant difference
  ↳ If ever an option requires a value
    ↳ '--' need sto be followed by "="
    ↳ '-' just white space





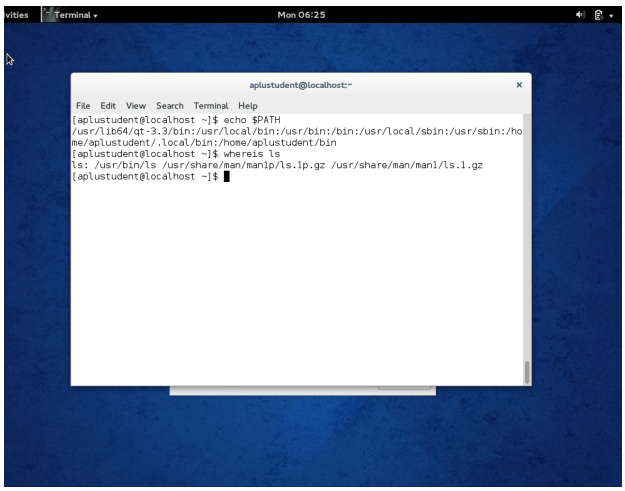
rsing the command line

↳ what happens when you type `ls`?

↳ PATH

- Type echo $PATH (All caps, lets us know it's a system variable)
- Note! Important! System cannot function without it!

What does it say?



↳ When you execute a command `ls` for instance, the OS goes down the line to these simple locations checking for a binary called `ls`

↳ whereis (one word) will tell you of the first occurance of a binary.

typing: abc
it happens? Why?

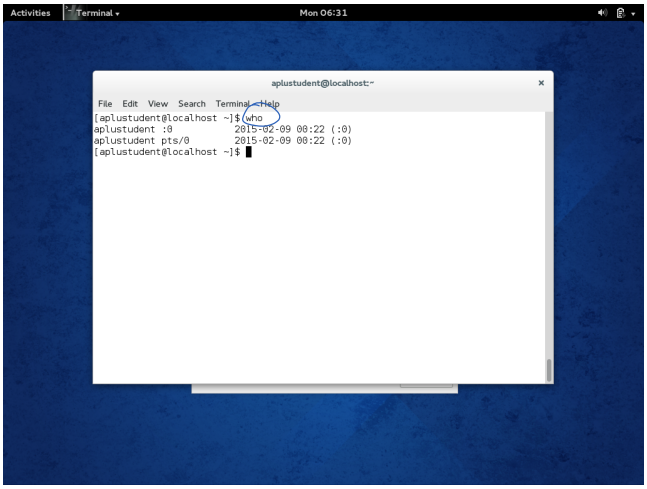
Try it: whereis cat whereis ls whereis vi

---

dard In, Standard Out, Standard Error

By default there are 3 default devices used for I/O
↳ Standard In is what? (Meaning where input comes from by default)
↳ Standard Out is what? (Same as where output goes)

Standard Out is the screen
↳ (who) ← command



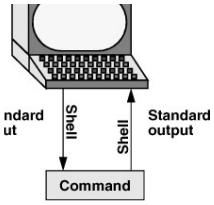
Standard out in this case is a file called /dev/pts/0

So to write to the screen programs just write to /dev/pts/0

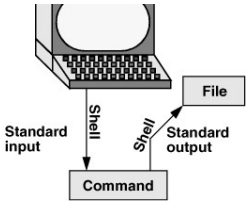
(It's a bit more complicated)

ince the system handles this
↳ Binaries don't even need to know this → just the alias standard out
↳ This is the job of the shell to redirect output to standard out
↳ So what is Standard IN?
↳ keyboard + Mouse

-/O Redirection



Default



Power of Linux to redirect I/O

> is the redirect character

ype date

date > today

ype
ut happens?

Why? Type ls - what happens?

> means append (non-destructive)

has an implicit create if the file does not exist.

th > >> actually is used to concatenate multiple files into

ne
↳ new one. Can you technically "cat" a file and nothing to $std out

dev/null is a "black hole" you can redirect unwanted data there.
↳ also can be used or empty out a file
↳ cat /dev/null > to "zero" messages

line (Important!)

Core

- Unix philosophy was many small tools doing one thing well.
  Then you in order to accomplish complex tasks need a pipeline
- The "pipe" "|" or "¦" usually above [Enter]
- Key concept → results of one command

(std out) becomes std In for the next command.
Can now chained together forever.

see it in action

Let's say I want to find the process id's of my text editor (gedit) so I can kill process. (open gedit or firefox)

ps is show process command

ps -ef shows everything (grep is a "search tool") (will become your best friend)

ps -ef | grep (grep it....)

what is shown?

You can redirect the screen output to a file.

ps -ef | grep gedit > processes.txt

ps -ef | grep gedit | wc → what does this do?

ps -ef | grep gedit | sort > processes.txt

ps -ef | grep gedit | awk '{print $2}' | sort >> processes.txt

(All assumes you have gedit installed)

---

e Command

... No not like that tea command!

- Tee acts a "T" shaped pipe.
- Allowing us to save output in a pipeline without stopping the pipeline.

ps -ef | grep gedit | sort | tee proc.txt | awk {print $2}

What happens?

Execution of commands can be chained with a ";"

Also a && (two "ands")

Smarter as if the command fails the second

- This command won't execute first
- With a ';' it will happily execute.

---

shell meta characters

↳ Unix mantra → let the computer do the small stuff

'?' finds any filename before the "?"

- ls memo?
- would memo5, memo9, and
- but not newmemo5 memos

↳ echo may?report prints

- may4report mayqreport may.report

* (asterik or star)

↳ similar to '?' but can be used positionally and can match zero items as well.

[ ] (brackets) square braces

↳ or

give a range or a set of values

↳ touch file1 file2 file3 file4

↳ touch file[1-4]

echo [aeiou]* → what does this display?

[!-] → immediately inside

[^tsq]* → match any word NOT starting with t, s, or q

[^b-d]* → match files that do not begin with b, c, or d

---

summary

- Shell is Linux command Interpreter
  - Scans proper syntax
  - Executed commands system PATH for you
- Shell delivers output to Std out (which can be redirected)
  - | < > redirection characters
    - tee
- Shell has metacharacters to assist in location and execution of files