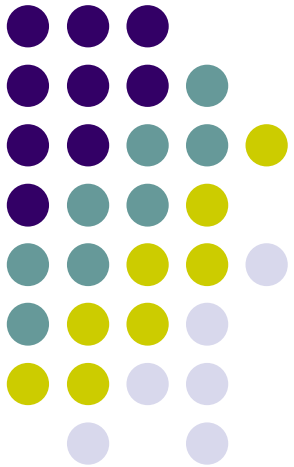
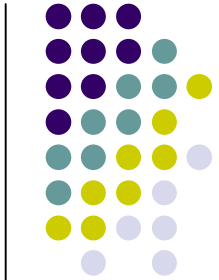
The Shell

Intermediate System Administration

Spring 2010



What is the shell?

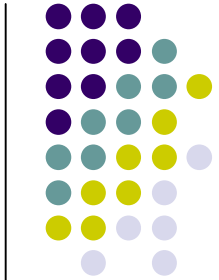


- A program; your gateway into any Unix system
- Allows you to inspect, manipulate, add/remove files and parts of the system
- Allows you to run other programs and control how they run
- A compact programming environment to allow automation of many tasks (sort of like DOS batch files)

In the beginning, there was sh. . .

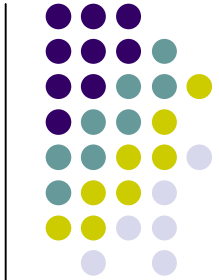
- sh (what would become the Bourne shell) is the original shell first developed for UNIX in 1971
- More commonly used today are derivatives of the Bourne shell (such as bash and ksh) and csh (the C shell, written by Bill Joy, at the time a masters student at Cal) and its derivative tcsh
- Most examples in this course will use bash syntax, but nothing we do will be impossible in any other shell

Diving in: Basic Commands



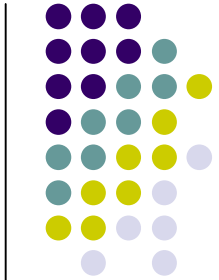
- Demonstration of:
  - cd - change directory
  - ls - list directory
  - pwd - print working directory
  - mkdir - make (empty) directory
  - cp - copy a file
  - mv - move/rename a file
  - rmdir - remove an empty directory
  - rm - remove a file

Looking for Things



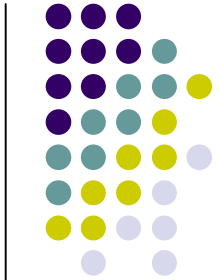
- **grep**: used to search within files for certain patterns (called *regular expressions*, or *regexes*)
- **find**: used to find files within a directory structure
- **locate**: quick but not necessarily up to date (indexed roughly every 24 hours)

Working with Files



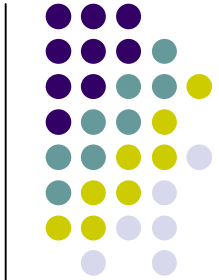
- **cp**: used to copy files from one location to another (think copy/paste)
- **mv**: used to move a file to another location, or to rename it (think cut/paste)
- **rm**: used to delete a file

More Programs



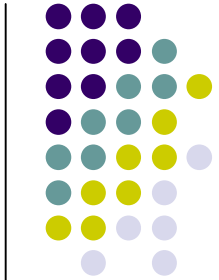
- You will often have to make use of programs you haven't seen before
- When you don't know what a program does or how to use it, *man* it!
- **man**: the single most useful program on your hard drive. Invoking "man programname" will show the *manual page* (i.e."user manual") for that program. The manual page will tell you the correct syntax and detail command line options

Wildcards and Questionmarks



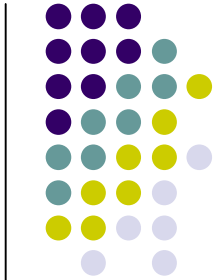
- You've probably heard of "rm -rf *", but what does it do?
- "*" is a special character in the shell. "*" represents *anything*
- "ABC*" matches "ABC1" and "ABC42"
- "?" is another special character. "?" matches *any single character*
- "ABC?" matches "ABC1" but not "ABC42"

The Unix Paradigm



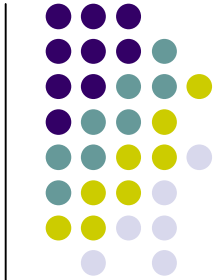
- Similar to the RISC vs CISC ideology
- Write small programs with small purpose and chain them together vs writing huge programs that do just one thing
- The shell makes this chaining possible with its most important feature: the **pipe**

Pipes



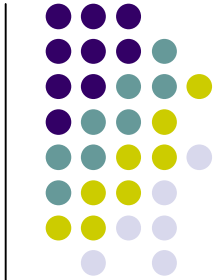
- Pipes are a way to chain the output of one command into the input of another
- For example, you can *grep* the output of *ls* to find all the files that match a particular pattern
- Or. . . you can *ls* the output of *grep*. Anything goes. It just won't do anything if the output of the first command doesn't match the input format of the second
- This is the basic idea behind Automator, LabView, etc.

Pipe Examples



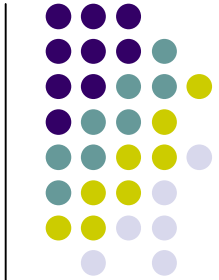
- Convert all WAV files in a directory to OGG:
  - `find | grep '.wav$' | xargs oggenc`
- Count how many lines a text file has
  - `cat jonathan.txt | wc -l`
- Get the file size of every file in a directory by using ls verbose options
  - `ls -l | awk '{print $5 $8}'`

Output Redirection



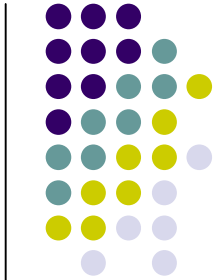
- Running programs can relay output to the screen two ways: via standard output (stdout) and standard error (stderr)
- The shell lets you control the flow of these using > and 2>
- Log the output of a program to prog.log and errors to error.log:
  - `someprogram >prog.log 2>error.log`

Input Redirection



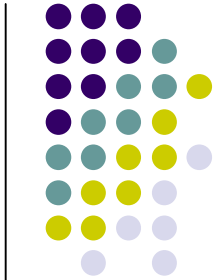
- Many UNIX programs will wait for input from the “standard input” (stdin); by default stdin is the keyboard
- We’ve already seen one example of input redirection (pipes)
- You can also redirect input from a file using “<“
  - `frobnicate < foobar.txt`

Into the Black Hole



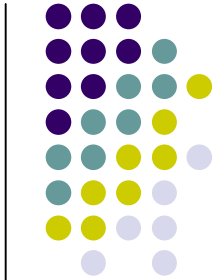
- /dev/null is a special file in UNIX systems
- it contains no data, and ignores anything you write to it
- to discard the output of a program, redirect it to /dev/null
- to explicitly pass no input to a program that expects it, redirect its stdin from /dev/null

Shell Programming



- Many shells support some basic programming constructs- for, while, if-then, etc.
- When combined with other shell features, you can do some pretty complex things with shell scripts
- We will have a shell scripting lecture later in the course

Substitution



- Sometimes, you want to substitute the output of one program into the command line of another
- Alternately, you may want to substitute the contents of a file
- You can do this with substitution:
  - rm $(locate .avi)
  - rm $(<files-to-delete.txt)
- Be careful! The output might not be what you expect and you could delete the wrong thing. . . rm -rf $(echo /)