Today

- More file system commands
- Processes

1

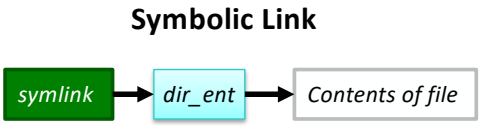
Review: Unix Commands

- What are pipes? What is the syntax for them? How do they work?
- How is the file system organized?
- What is a path?
  - What is the difference between an *absolute* and *relative* path?
- What are some common file system commands?
  - What do they mean? How do you use them?
- What is a link? What is the difference between a "regular" link and a symbolic link?

2

Review: Symbolic Links

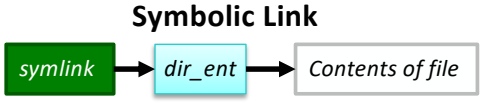
- ***Symbolic*** links are different than regular links (often called **hard links**)
  - Created with `ln -s target dest`
- Can be thought of as a directory entry that points to the ***name*** of another file
- Does not change link count for file
  - When original deleted, symbolic link remains
- They exist because
  - Hard links don't work across file systems
  - Hard links only work for regular files, not directories



Create a symbolic link to the course's directory
Check if that worked by cd-ing to the link

Symbolic Links – in practice

- I created a symbolic link for cs397 in /csci/courses
- Do a *long listing* of /csci/courses
  - How are links distinguished from regular files?
- Go into cs397's handouts directory
  - Do a long listing



Displaying File Contents

- `cat` can be used to display the contents of a file in the terminal
  - When invoked with a list of file names, it concatenates them

```
cat file*
ls | cat -n
```

- Some options:
  - `-n` number output lines (starting from 1)
  - `-v` display non-printing characters in visible form (e.g. ^M)

```
cat ~/.bash_history
```


Understanding a Command with a Pipe

- What does `ls | cat -n` do?
- Break it down:
  - Just run `ls`
  - Then, the whole command

Displaying File Contents

- Interactive commands `more` and `less` show a page at a time
  - To search, use /
  - To quit, hit q
- To view the beginning of a file
  - `head`
  - Use -# to view more or fewer lines
- To view the end of a file
  - `tail`
  - Use -# to view more or fewer lines
  - Use -f to "watch" the file as it grows

```
whatis more
whatis less
more ~/.bash_history
less ~/.bash_history
```


head and tail examples

```
cat /etc/passwd
head /etc/passwd
tail -1 /etc/passwd
ls -lt | tail -3
head –10 /etc/passwd | tail -5
head *.py
tail –f /var/log/httpd/access_log
```

What is the "net effect" of these commands?

Useful Shortcuts: {}

- Examples:
  - `cp index.{html,php}`
    - Expands to `cp index.html index.php`
  - `mv file{,.bak}`
    - Expands to `mv file file.bak`
  - `tar cfz myDir{.tar.gz,}`
    - Expands to `tar cfz myDir.tar.gz myDir`

PROCESSES

Unix Processes

- **Process**: An entity of execution
- UNIX can execute many processes simultaneously
- Creation of a process
  - A unique process id (pid) is assigned to the new process
  - Inherit, create, or initialize other data structures (e.g., file tables, I/O table, etc.)

Background Jobs

- By default, when executing a command in the shell, the shell will wait for the command to exit before printing out the next prompt
- Trailing a command with & allows the shell and command to run simultaneously

```
[sprenkle@fred ~]$ jedit &
[1] 7001
```

pid

Ending a process

- When a process ends, there is a *return* or *exit status* code (an integer) associated with the process
  - 0 means success
  - >0 represent various kinds of failure, up to process

13

Process Information Maintained

- Working directory
- File descriptor table
- Process id
  - number used to identify process
- Process group id
  - number used to identify set of processes
- Parent process id
  - process id of the process that created the process
- Umask
  - Default file permissions for new file

14

Process Information Maintained

We haven't talked about these yet:

- Effective user and group id
  - The user and group this process is running with permissions as
- Real user and group id
  - The user and group that invoked the process
- Environment variables

15

ps

- Report a snapshot of the current processes
- By default, just displays processes in the current terminal
  - Columns by default: PID, TTY, TIME, and CMD
- Accepted options:
  - UNIX options, which may be grouped and must be preceded by a dash
  - BSD options, which may be grouped and must not be used with a dash
  - GNU long options, which are preceded by two dashes

16

ps Examples

Command	Meaning
`ps -e`	See every process on the system
`ps -ef`	See every process on the system, in full listing
`ps ax`	See every process on the system
`ps -ejH`	See a process tree
`ps u`	See in user form

Pipe through `more`

Process Subsystem Utilities

Utility	Functionality
`top`	Monitors tasks
`kill <pid>`	Terminate a process Use `-9` if bugger won't die
`nohup <cmd>`	Makes a command immune to hangup and terminal signal
`sleep <#>`	Sleep in seconds
`nice <cmd>`	Run processes at a certain priority

PROCESS ENVIRONMENT

19

Environment of a Process

- A set of key-value pairs associated with a process
- Keys and values are strings
- Passed to children processes
- Cannot be passed back up
  - i.e., what you do in the child doesn't affect parent
- Common examples:
  - **PATH**: Where to search for programs
  - **TERM**: Terminal type

20

Shell Variables

- Shells have several mechanisms for creating variables
- A ***variable*** is a name representing a string value. Example: PATH
  - Shell variables can save time and reduce typing errors
- Allow you to store and manipulate information
  - Ex: `ls $DIR > $FILE`
- Two types: local and environmental
  - Local are set by the user or by the shell itself
  - Environmental come from the operating system and are passed to children

Shell Variables

- Syntax varies by shell
  - `varname=value  # sh, ksh, bash`
  - `set varname = value  # csh`
- To access the value: **`$varname`**
- Turn local variable into environment:
  - All child processes from this terminal
  - **`export varname  # sh, ksh, bash`**
  - **`setenv varname value  # csh`**

Environmental Variables

Name	Meaning
$HOME	Absolute pathname of your home directory
$PATH	A list of directories to search for
$MAIL	Absolute pathname to mailbox
$USER	Your user name
$SHELL	Absolute pathname of login shell
$TERM	Type of terminal
$PS1	Prompt

To view ***all*** shell variables, `set`

The PATH environment variable

- Colon-separated list of directories
- Executables (without a specified path) are executed if found in a directory in the list
  - Searched left to right
- Example:

```
$ example.sh
-bash: example.sh not found
$ PATH=$PATH:.
$ example.sh
hello!
```

Examples of Impact of PATH

Not having . in your path:

```
$ ls
foo
$ foo
sh: foo: not found
```

```
$ ./foo
Hello, foo.
```


Examples of Impact of PATH

Not having . in your path:

```
$ ls
foo
$ foo
sh: foo: not found
```

```
$ ./foo
Hello, foo.
```

- What **not** to do:

```
$ PATH=.:$PATH
$ ls
foo
$ cd /tmp/
$ ls
Congratulations! Your files have been removed
and you have just sent email to Professor Watson
challenging him to a fight.
```

Examples of Impact of PATH

Not having . in your path:

```
$ ls
foo
$ foo
sh: foo: not found
```

```
$ ./foo
Hello, foo.
```

- What **not** to do:

```
$ PATH=.:$PATH
$ ls
foo
$ cd /tmp/
$ ls
Congratulations! Your files have been removed
and you have just sent email to Professor Watson
challenging him to a fight.
```