Scripting & Computer Environments

Basic Filters

IIIT-H

Aug 10, 2013

...Previously & Today...

Previously: Some more file operations:

1. Securing files/permissions

   `chmod, chgrp, chown`

2. Compressing/archiving files

   `tar, gzip/gunzip, bzip/bunzip, zip/unzip...`

3. Remote-accessing files

   `ssh, scp, sftp ...`

4. Editing files

   `Vi/Vim` editor

Today:

- Redirection & Piping
- Simple Filters
- Shell Wildcards
- Basic File Searching

...Previously & Today...

Previously: Some more file operations:

1. Securing files/permissions

   `chmod, chgrp, chown`

2. Compressing/archiving files

   `tar, gzip/gunzip, bzip/bunzip, zip/unzip...`

3. Remote-accessing files

   `ssh, scp, sftp ...`

4. Editing files

   `Vi/Vim` editor

Today:

- Redirection & Piping
- Simple Filters
- Shell Wildcards
- Basic File Searching

Brainstorm

- How do we secure files?
- Notion of permission for regular files and directories

  Assume a file with permission of 644 and a directory with 755.

  1. case 1: -w for the directory only
  2. case 2: -w for the file only
  3. case 3: -w for both

  What happens if you attempt to delete/move the file?

- `STDIO` or `IOSTREAM` in C/C++?

Brainstorm

- How do we secure files?
- Notion of permission for regular files and directories

  Assume a file with permission of 644 and a directory with 755.

  1. case 1: -w for the directory only
  2. case 2: -w for the file only
  3. case 3: -w for both

  What happens if you attempt to delete/move the file?

- STDIO or IOSTREAM in C/C++?

Brainstorm

- How do we secure files?
- Notion of permission for regular files and directories

  Assume a file with permission of 644 and a directory with 755.

  1. case 1: -w for the directory only
  2. case 2: -w for the file only
  3. case 3: -w for both

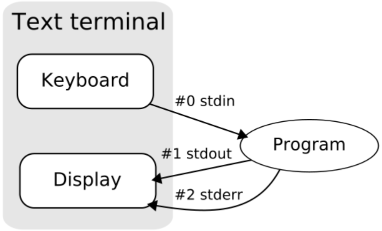
  What happens if you attempt to delete/move the file?

- `STDIO` or `IOSTREAM` in C/C++?

I/O Redirection

- The shell reads input & writes output as a stream of characters. (stream = sequence of bytes)
- Command outputs: result or status/error messages. Sent to?
- The shell provides 3 special files @ login, each with a unique file descriptor value
- Each associated with a default device
  1. `#0 - Standard Input stream (STDIN)`: input to commands.
  2. `#1 - Standard Output stream (STDOUT)`: output from commands.
  3. `#2 - Standard error stream (STDERR)`: errors from commands.

Sources and destinations of a command??



I/O Redirection

- A way of unhooking a stream from its default device
- Changing where input comes from/output goes to
- Operators:
  1. Input: `0<` or just `<`
  2. Output: `>` or `>>`
  3. Error: `2>` or `2>>`

Example

```
wc -l < /usr/share/dict/words

wc < input.txt > output.txt

ls -l -y >> output.txt 2>> log.txt          (ls doesn't take -y)

ls -l input.txt IDoNotExist.txt &> output.txt

ls -l input.txt IDoNotExist.txt 2>&1              (any difference?)

cat /etc/passwd > /dev/null
```

Pipes

Piping (|)

`command1 | command2 | ...| command n`

- Output of a command piped into input for another.
- `STDOUT` $\longrightarrow$ `STDIN`
- Length of the pipe can be "indefinite"
- Redirection vs Piping (`> vs |`)?

Example

```
who | wc -l

history | head -10 | tail -5

ls -l | sort -k 8 > output.txt
```

Pipes

Piping (|)

`command1 | command2 | ...| command n`

- Output of a command piped into input for another.
- `STDOUT` → `STDIN`
- Length of the pipe can be “indefinite”
- Redirection vs Piping (`> vs |`)?

Example

```
who | wc -l

history | head -10 | tail -5

ls -l | sort -k 8 > output.txt
```

Filters

- Simply, commands that use both the `STDIN` and `STDOUT`
- Read input stream $\rightarrow$ [transform it] $\rightarrow$ output the result.
- Example application: text filtering

  e.g. **`cat`**, **`wc`**, **`tr`**, **`grep`**, **`sed`**, **`awk`**, etc

cat (concatenate) & split

`cat [option] [file]`

- `cat` displays contents of file on `STDOUT`.
- Keeps reading from `STDIN` until `EOF` or `ctrl+d`
- `split` splits a file into pieces.

Example

```
cat file1.txt
cat file1.txt file2.txt

split -l 2 file1.txt
split -b 5 file2.txt

cat x*
```

cat (concatenate) & split

`cat [option] [file]`

- `cat` displays contents of file on `STDOUT`.
- Keeps reading from `STDIN` until `EOF` or `ctrl+d`
- `split` splits a file into pieces.

Example

```
cat file1.txt
cat file1.txt file2.txt

split -l 2 file1.txt
split -b 5 file2.txt

cat x*
```

`wc` word count

- `-l` line count
- `-w` word count
- `-m` character count

sort & uniq

- Sorting by collating sequence
- Also merges already-sorted files and checks if sorted or not
- Options: `-r` (reverse order), `-k` (column), `-n` (numerical order), etc
- `uniq` discards identical lines

```
cat file1 file2 | sort | less
cat file1 file2 | sort | unique | less
```

`wc` word count

- `-l` line count
- `-w` word count
- `-m` character count

sort & uniq

- Sorting by collating sequence
- Also merges already-sorted files and checks if sorted or not
- Options: `-r` (reverse order), `-k` (column), `-n` (numerical order), etc
- `uniq` discards identical lines

```
cat file1 file2 | sort | less
cat file1 file2 | sort | unique | less
```

cmp, diff, comm (file comparison)

- cmp - byte-by-byte comparison
- comm - line-by-line comparison of *sorted* files
- diff - which lines be changed to make them identical

head & tail, cut & paste

- cut - slits a file vertically (unlike...?)
- paste - merges lines of a file

e.g. Display only the permission characters of `ls -l`?

```
ls -l | cut -c1-10 > permissions.txt

paste permissions.txt file2.txt > merged.txt
```

`cmp, diff, comm` (file comparison)

- `cmp` - byte-by-byte comparison
- `comm` - line-by-line comparison of *sorted* files
- `diff` - which lines be changed to make them identical

`head` & `tail`, `cut` & `paste`

- `cut` - slits a file vertically (unlike...?)
- `paste` - merges lines of a file

e.g. Display only the permission characters of `ls -l`?

```
ls -l | cut -c1-10 > permissions.txt

paste permissions.txt file2.txt > merged.txt
```

`cmp`, `diff`, `comm` (file comparison)

- `cmp` - byte-by-byte comparison
- `comm` - line-by-line comparison of *sorted* files
- `diff` - which lines be changed to make them identical

`head` & `tail`, `cut` & `paste`

- `cut` - slits a file vertically (unlike...?)
- `paste` - merges lines of a file

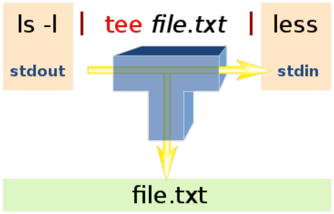
  e.g. Display only the permission characters of `ls -l`?

```
ls -l | cut -c1-10 > permissions.txt

paste permissions.txt file2.txt > merged.txt
```

The tee filter

- An external command
- Reads input from STDIN and duplicates it
- One copy goes to STDOUT
- Another copy goes to a file



```
who | tee file.txt
```

The `tr` filter

`tr [options] set1 set2`

- Translates/deletes each character in set1 to set2
- a.k.a. search and replace
- Receives input only from `stdin`. From files?

Example

```
tr ‘A-Z’ ‘a-z’

cat input.txt | tr ‘ABCD’ ‘1234’

tr ‘ ’ ‘\t’ < input.txt

echo "hello there" | tr -d ‘e’
```

The `tr` filter

```
tr [options] set1 set2
```

- Translates/deletes each character in set1 to set2
- a.k.a. search and replace
- Receives input only from `stdin`. From files?

Example

```
tr ‘A-Z’ ‘a-z’

cat input.txt | tr ‘ABCD’ ‘1234’

tr ‘ ’ ‘\t’ < input.txt

echo "hello there" | tr -d ‘e’
```

Shell Metacharacters (a.k.a. Wildcards)

- Characters with special meaning to the shell

  `* ? < > [ ] ' " ; { } ( ) ! & ^ | \n ...`

- Expanded by the shell first.
- `?` matches any single character
- `*` matches 0+ number of characters (but '.' and '/')
- `[]` matches any element in the set.
- Some characters with special meaning inside `[]`: `-` (hyphen), $\wedge$, !
- \ turns off the special meaning

Example

```
ls -l ?????

rm -i *.c

cp [A-Z]* MyDir

ls -l file[^A-Z]*   or  ls -l file[!A-Z]*

echo \\
```

- One of the basic operations of any OS
- Linux offers some commands: `locate`, `whereis`, `find` ...

```
find <where> -name <search criteria>

find / -name ‘file[^12].c’

find ~ -name ‘My??*’

find . -name ‘*199[0-9]*’

find . -size -2000b -mtime 1 -name ‘*.html’
```

Putting It All Together...

1. Extract only lines 10 through 20 of `/etc/passwd`
2. The first 10 largest files in the current directory?
3. Remove all numbers from all C programs in the current directory
4. What does the following command do?

```
tr ‘a-z’ ‘0-9’ < file.txt | sort -rn | uniq -c | head -n 5
```

Next...

RegEx
Regular Expression
/h[a4@](([c<]([k]|(\|<)))|([k]|(\|<))(x))\s+\
((d)|([t\+]h))[3ea4@]\s+p[l1][a4@]n[3e][t\+]/i
(C)2006 FTS Conventures - www.ftsconventures.com