Instructor: Nicolas Savva

February 4, 2015

---

1 based on slides by Hussam Abu-Libdeh, Bruno Abrahao and David Slater over the years
Course drop deadline today
A2 is out (due Saturday 02/07)
CSUGLab accounts will be activated sometime this week
Today

- Regular expressions (grep recap)
- More Unix tools: cut, paste, split, join
- Stream Editor: sed
- vim
see Lecture 6
WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.

OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!

BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!

IT'S HOPELESS!

EVERYBODY STAND BACK.

I KNOW REGULAR EXPRESSIONS.
**cut**

cut extract sections from each line of the input.

```
cut [-b] [-c] [-d delim] [-f list] [-s] [file]

- delim is a delimiter that separates fields
- list consists of one of N, N-M, N-
```

**Options**

- `-b`: extracts using range of bytes
- `-c`: extracts using range of characters
- `-d`: specifies a delimiter (tab by default)
- `-f`: specifies a range of fields separated by a delimiter
- `-s`: suppresses line if delimiter is not found
Cut examples

employee.txt

This line doesn’t have a demiliter

Examples

- cut -d : -f 1 -s employee.txt: Prints the names
- cut -d : -f 3,4 -s employee.txt: Prints the address and the zip code
- cut -d : -f 2 employee.txt: Prints phone numbers plus the last line
- cut -c 1 employee.txt: Prints their first initial plus the first character of the last line
paste

paste concatenate files side-by-side.

**cut**

`paste [options] [file1 ...]`

**Options**

- `-d`: specify a delimiter to separates fields (instead of tab)
- `-s`: concatenates serially instead of side-by-side
Example 1:
```
paste names.txt phones.txt
Alice 607-233-2464
Bob 607-257-2884
Charlie 605-987-7886
```
### names.txt

| Alice  
| Bob   
| Charlie |

### phones.txt

| 607-233-2464  
| 607-257-2884  
| 605-987-7886 |

### Examples

- `paste -d : names.txt phones.txt`
  - Alice:607-233-2464
  - Bob:607-257-2884
  - Charlie:605-987-7886
## names.txt

- Alice
- Bob
- Charlie

## phones.txt

- 607-233-2464
- 607-257-2884
- 605-987-7886

## Examples

```
paste -s names.txt phones.txt
Alice Bob Charlie
607-233-2464 607-257-2884 605-987-7886
```
Splitting files

split

Splits a files into pieces, i.e., files named xaa, xab, ...

```
cut
split [options][file1] [prefix]
```

**Options**
- `-l`: how many lines in each file
- `-b`: how many bytes in each file
- `prefix`: name prefix of each file produced
Joining files

join

Join lines that contain the same keys between two different files

```
cut
join [options] file1 file2
```

Options

- `-1 field`: join by the \( n \)-th field of file 1
- `-2 field`: join by the \( n \)-th field of file 2
- `-a file_number`: displays unpaired lines of file file_number
Join examples 1/2

age.txt
Alice 12
Bob 30
Charlie 23

salaries.txt
Bob 129,000
Charlie 75,000

Examples
- join age.txt salaries.txt
  Bob 30 129,000
  Charlie 23 75,000
Join examples 2/2

age.txt
Alice 12
Bob 30
Charlie 23

salaries.txt
Bob 129,000
Charlie 75,000

Examples
- join -a1 age.txt salaries.txt
  Bob 30 129,000
  Charlie 23 75,000
  Alice 12
Basic Calculator

bc

Performs arithmetic and logical calculations

Options

- `-l field`: increase the precision to 20 decimal places (default 0)

Examples

- `echo "1/3" | bc`
  0

- `echo "1/3" | bc -l`
  0.33333333333333333

- `echo "1>3" | bc -l`
  0

- `echo "1<3" | bc -l`
  1
sed is a *stream editor*. We will only cover the basics, as it is a completely programming language!

**Stream Editor**

```
sed [options] [script] [file]
```

- Stream editor for filtering and transforming text
- We will focus on `sed 's/<regexp>/<text>’ [file]`
- This form replaces anything that matches `<regexp>` with `<text>`.
- `sed` goes line by line searching for the regular expression.

What is the difference between `sed` and `tr`?

- `sed` can match regular expressions!
- `sed` also does lots of other stuff
Basic Example:

Example:

```
sed 's/not guilty/guilty/g' filename
```

Replaces not guilty with guilty everywhere in the file

What happens if we don’t have the g?

Without the g, it will only do one substitution per line.
Just like with `tr` we can do deletion with `sed`:

- `sed '/regexp/d'` - deletes all lines that contain `regexp`

**Example**

- `sed '/[Dd]avid/d' filename > filename2`
  - deletes all **lines** that contain either David or david and saves the file as `filename2`.
sed understands regular expressions!

The power of sed is that it treats everything between the first pair of /’s as a regular expression. So we could do

```bash
sed 's/\[[[:alpha:]]\{1,3\}\[[[:digit:]]\]*@cornell\.edu/cornell email
removed/g' file
```

to print a file with all cornell email addresses removed.

use `-r` on Linux (`-E` on OS X) to use extended regular expressions.
sed can save the string

Another Example:

```
sed 's/\(~([A-Z][A-Za-z]*), \([A-Z][A-Za-z]*\)\)/\2 \1/\2 \1/' filename
```

- Searches for an expression at the beginning of the line of the form \textit{e1, e2} where \textit{e1} and \textit{e2} are "words" starting with capital letters.
- Placing an expression inside ( ) tells the editor to save whatever string matches the expression.
- Since ( ) are special characters we escape them; i.e. by using \\( \)\.
- We access the saved strings as \texttt{\1, \2}.
- This script for example could convert a database file from \texttt{Lastname, Firstname} to \texttt{Firstname Lastname}.
More sed

You can specify which lines to check by numbers or with regular expressions:

```
sed '1,20s/john/John/g' filename - checks lines 1 to 20
sed '/^The/s/john/John/g' filename - checks lines that start with The
```

& corresponds to the pattern found:

```
sed 's/[a-z]\+/"&"/g' filename
```

replaces words with words in quotes

For more on sed check out http://www.grymoire.com/Unix/Sed.html
How could we use sed to remove a specific regular expression? sed 's/regexp/ /g’ file

Example:

sed 's/[:alnum:]]/ /g’ Frankenstein.txt
Examples:

Let’s strip the directory prefix from our pathnames (i.e. convert `/usr/local/src` to `src`)

Example:

```
pwd | sed ’s/.*//’
```

- Translates anything preceding (and including) a frontslash to nothing
- Note the backslash-escaped frontslash
sed scripting

sed is a complete programming language and we can write sed scripts.

- Any file that begins with `#!` is a script file (we will talk more about this next week).

**Example**

Create a new text file named trim.sed

```bash
#!/usr/bin/sed -f
s/^ //
s/ //
```

You can run this script from the shell like any other program:

- `echo " this is a test " | ./trim.sed`

this is a test

We now have a script that trims leading and trailing whitespace!
Sed is a complete programming language. In fact people have written entire games as sed scripts.

http://aurelio.net/soft/sedarkanoid/