CS2043 - Unix Tools & Scripting

Cornell University, Spring 2014[1]

Instructor: Bruno Abrahao

February 10, 2014

[1] Slides evolved from previous versions by Hussam Abu-Libdeh and David Slater

The Stream Editor

`sed`

We will use sed as a *stream editor*, but 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.

sed and tr

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

- `sed` can match regular expressions!
- `sed` can replace more than characters

Basic Example:

Example:

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

Replaces not guilty with guilty using the input and send results to `stdout`.

What happens if we don't have the g?

Without the g, it will only do one substitution per line.

sed deletion

Just like with `tr` we can do deletion with `sed`

sed deletion

- `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!

```
sed ’s/[[:alpha:]]\{1,3\}[[:digit:]]*@cornell\.edu//g’
```

Question: what does this do?

use -r on Linux (-E on OS X) to use extended regular expressions.

sed is greedy!

sed matches a given regular expression to the **longeset** string as possible:

```
$ echo filename1
a b col d e f lapse h i j k lapse m n
$ sed 's/col.*lapse/collapse/g' filename1
a b collapse m n
```

```
$ echo filename2
a b c col d e f col g h i lapse
sed 's/col.*lapse/collapse/g' filename2
a b c collapse
```

sed can save matches

- Sed can store matches in registers.
- We use parenthesis to store matches.
- The first match saved will go to register \1, the second to \2, and so on.

Example

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

- Searches for "words" starting with capital letters at the beginning of the line and separated by comma and one space character.
- Placing an expression inside ( ) tells the editor to save whatever string matches the expression
- Since ( ) are special characters we escape them.
- We access the saved strings as `\1, \2`.
- This script for example could convert a database file from

`Lastname, Firstname` to `Firstname Lastname`

Selecting lines

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

`sed '1,20s/john/John/g' file` - checks lines 1 to 20

`sed '/^The/s/john/John/g' file` - checks lines that start with The

Selecting lines

& corresponds to the pattern found:

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

replaces words with words in quotes

Even more sed

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

```
#! /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!

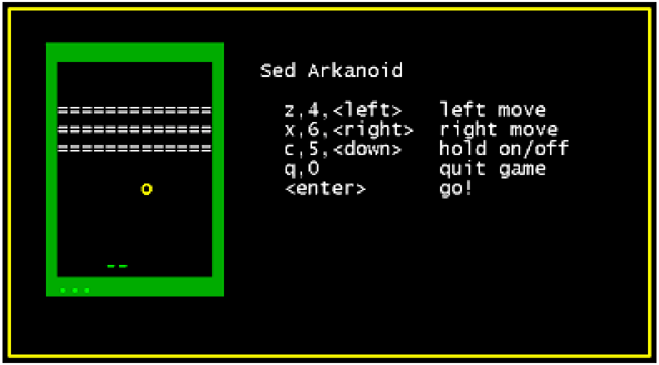
sed in Vim

Replace

Replace pattern matched with regexp with string:
`:%s/regexp/string/[options]`

Sed Arkanoid

Sed is a complete programming language. In fact people have written entire games as sed scripts.



```
Sed Arkanoid

   z,4,<left>     left move
   x,6,<right>    right move
   c,5,<down>     hold on/off
   q,0            quit game
   <enter>        go!
```

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

Selecting lines

For more on sed check out

http://www.grymoire.com/Unix/Sed.html

cron

cron

`cron` is a program that enables unix users to execute commands or scripts automatically at a specified date/time

- `cron` is a daemon, which means it only needs to be started once and will lay dormant until it is required
- On most Linux distributions is automatically installed and entered into the start up scripts so you don't have to start it manually:
  - Check by tying `ps -e | grep cron`
  - Depending on your system, it may show up as `cron` or `crond`
- We can control the cron daemon in a few different ways...

cron and root

If you have a look in your `/etc` directory you will find sub directories called

- `cron.hourly`
- `cron.daily`
- `cron.weekly`
- `cron.monthly`

- If you place a script in any of these directories, it will be run either hourly, daily, weekly or monthly depending on the name of the directory.
- Note: If we did this with our backup script, we would need to replace $\sim$ with /home/hussam since the script would be run as `root`.

cron flexibility

If you want more flexibility in scheduling you can edit a `crontab` file

`crontab`

`crontab` files are cron's config files.

- The main config file is normally `/etc/crontab`
- You can create your own crontab files without root access!

Type `cat /etc/crontab` to have a look at the file:

main crontab

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc cron.monthly )
#
```

Crontab Syntax

crontab

Syntax:

```
a.    b.    c.    d.    e.    command to be executed
```

a. min (0-59)

b. hour (0-23)

c. day of month (1-31)

d. month (1-12)

e. day of week (0-6) (Sunday = 0)

Values can be * (all legal values), a range separated by a hyphen, a single value, a set of values separated by commas or a step value (i.e. */2 could be every two hours).

multiuser crontab

- To edit your crontab file type `crontab -e`
- To view your crontab file type `crontab -l`
- To delete your crontab file type `crontab -r`

A sample line:

```
30   18   *   *   *   ./home/hussam/backup.sh
```

This runs the backup script everyday at 6:30PM.