CS2043 - Unix Tools & Scripting

Cornell University, Spring 2014[1]

Instructor: Bruno Abrahao

February 3, 2014

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

Cut and paste without mouses and windows

cut

cut extract sections from each line of the input.

cut

```
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`: especifies a delimiter (tab by default)
- `-f`: especifies a range of fields separated by a delimiter
- `-s`: supressses line if delimiter is not found

Cut examples

employee.txt

```
Alice:607-233-2464:15 Sunny Place, Ithaca, NY:14850:female
Bob:607-257-2884:504 Brown St, Ithaca, NY:14850:male
Charlie:605-987-7886:99 Berry Lane, Palo Alto, CA:94304:male
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 -d :  -c 1 employee.txt`: Prints their first initial plus the first character of the last line

Cut and paste without mouses and windows

`paste`

`paste` concatenate files side-by-side.

cut

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

Options

- `-d`: speicfy a delimiter to separates fields (instead of tab)
- `-s`: concatenates serialy instead of side-by-side

Paste examples 1/3

names.txt

```
Alice
Bob
Charlie
```

phones.txt

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

Examples

- `paste names.txt phones.txt`

  ```
  Alice 607-233-2464
  Bob 607-257-2884
  Charlie 605-987-7886
  ```

Paste examples 2/3

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
```

Paste examples 3/3

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 field-th field of file 1
- `-2 field`: join by the field-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
```

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`
  0.33333333333333333333
- `echo "1>3" | bc -l`
  0
- `echo "1<3" | bc -l`
  1

Looking for things

- `find` : Searching for files/directories by name or attributes
- `grep` : Search contents of files

Find it!

`find`

- used to locate files or directories
- search any set of directories for files that match a criteria
- search by name, owner, group, type, permissions, last modification date, and other criteria
- search is recursive (will search all subdirectories too)

Syntax looks like this:

```
find [where to look] criteria [what to do]
```

Simple usage

- display pathnames of all files in current directory and subdirectories
  ```
  find .  -print
  find -print
  find .
  ```
  (all equivalent)
- search for a file by name
  ```
  find .  -name my_awesome_file.txt
  ```

Find options

- `-name` : name of file or directory to look for
- `-maxdepth num` : descend at most *num* levels of directories while searching
- `-mindepth num` : descend at least *num* levels of directories while searching
- `-amin n` : file last access was *n* minutes ago
- `-atime n` : file last access was *n* days ago
- `-group name` : file belongs to group *name*
- `-path pattern` : file name matches shell pattern *pattern*
- `-perm mode` : file permission bits are set to *mode*

... for more: `man find`

More on find

- normally all modifiers for `find` are evaluated in conjunction (i.e. AND). We can find files matching a pattern *OR* another by using the `-o` flag.
- executes a command on found files by using the `-exec command '{}' +` flag.
- executes a command on found files by using the `-exec command '{}' \;` flag.
- The difference between \; and + is that with \; a single grep command for each file is executed whereas with + as many files as possible are given as parameters to grep at once.

Find examples

Find all files accessed at most 10 minutes ago

```
find .  -amin -10
```

Find all files accessed at least 10 minutes ago

```
find .  -amin +10
```

Display all the contents of files accessed in the last 10 minutes

```
find .  -amin -10 -exec cat ‘{}’ +
```

Search by keyword

grep

The purpose of grep is to print the lines that match a particular pattern.

grep

```
grep <string> [file]
```

- searches file for all lines containing <string>
- grep stands for global / regular expression / print

Examples:

```
grep password file
```

- prints all lines that contain the word password in the file file.

What lines contain the word monster in Frankenstein?

```
grep 'monster' Frankenstein.txt
```

More Simple Examples

Two simple ways to use grep are on a file and on piped input:

grep on a file

```
grep "chromium" /var/log/dpkg.log
```

- Shows when I have updated chromium-browser

grep piped input

```
history | grep grep
```

- When have I used `grep` recently?

Grep options

- `grep -i` - ignores case
- `grep -A 20 -B 10` - prints the 10 lines before and 20 lines after each match
- `grep -v` - inverts the match
- `grep -o` - shows only the matched substring
- `grep -n` - displays the line number

Example:

```
grep -v # bashscript
```

- Prints all noncommented lines