Text processing commands

Cut command.

cut command selects a list of columns or fields from one or more files.
Option -c is for columns and -f for fields. It is entered as
cut *options* [files]
for example if a file named testfile contains

```
this is firstline
this is secondline
this is thirdline
```

Examples:
cut -c1,4 testfile will print this to standard output (screen)

```
ts
ts
ts
```

It is printing columns 1 and 4 of this file which contains t and s (part of *this*).

```
cut –f1,2 testfile  will print

this is
this is
this is
```

Options:

1. Character column cut

   cut -c list [file_list]

   Option:
   -c list Display (cut) columns, specified in list, from the input data. Columns are counted from one, not from zero, so the first column is column 1. List can be separated from the option by space(s) but no spaces are allowed within the list. Multiple values must be comma (,) separated. The list defines the exact columns to display. For example, the -c 1,4,7 notation cuts columns 1, 4, and 7 of the input. The -c -10,50 would select columns 1 through 10 and 50 through end-of-line (please remember that columns are conted from one)
2. 2. Delimiter-based (parsing) cut

   cut -f list [-d char] [-s] [file_list]

   Options:
   d char The character char is used as the field delimiter. It is usually quoted but can be escaped. The default delimiter is a tab character. To use a character that has special meaning to the shell, you must quote the character so the shell does not interpret it. For example, to use a single space as a delimiter, type -d' '.

   -f list Selects (cuts) fields, specified in list, from the input data. Fields are counted from one, not from zero. No spaces are allowed within the list. Multiple values must be comma (,) separated. The list defines the exact field to display. The most practically important ranges are "open" ranges, were either starting field or the last field are not specified explicitly (omitted). For

example:

Selection from the beginning of the line to a certain field is specified as -N, were N is the number of the filed. For example -f -5

Selection from the certain filed to the end of the line (all fileds starting from N) is specified as N-. For example -f 5-

Specification can be complex and include both selected fields and ranges. For example, -f 1,4,7 would select fields 1, 4, and 7. The -f2,4-6,8 would select fields 2 to 6 (range) and field 8.

Paste Command.

paste command merge the lines of one or more files into vertical columns separated by a tab.

for example if a file named testfile contains

```
this is firstline
```

and a file named testfile2 contains

```
this is testfile2
```

then running this command

paste testfile testfile2 > outputfile

will put this into outputfile

```
this is firstline       this is testfile2
```

it contains contents of both files in columns.

who | paste - - will list users in two columns.

Options:

- -d'char' separate columns with char instead of a tab.
- -s merge subsequent lines from one file.

Sort command.

sort command sort the lines of a file or files, in alphabetical order. for example if you have a file named testfile with these contents

```
zzz
aaa
1234
yuer
wer
qww
```

```
wwe
```
Then running
sort testfile
will give us output of
```
1234
aaa
qww
wer
wwe
yuer
zzz
```

Options:

Option	Description
-d	Sorts via dictionary order, ignoring non-alphanumerics or blanks.
-f	Ignores case when sorting.
-g	Sorts by numerical value.
-M	Sorts by month (i.e., January before December).
-r	Provides the results in reverse order.
-m	Merges sorted files.
-u	Sorts, considering unique values only.

-

Uniq command.

uniq command removes duplicate adjacent lines from sorted file while sending one copy of each second file.

Examples

sort names | uniq -d will show which lines appear more than once in names file.

Options:

- -c print each line once, counting instances of each.
- -d print duplicate lines once, but no unique lines.
- -u print only unique lines.

Translate characters.

Syntax

- *tr [-c] [-d] [-s] [string1] [string2]*

-c	Complement the set of characters specified by string1.
-d	Delete all occurrences of input characters that are specified by string1.
-s	Replace instances of repeated characters with a single character.

string1	First string or character to be changed.
string2	Second string or character to change the string1.

Examples

The `-d` option deletes a range of characters.

```
echo "abcdef"                  # abcdef
echo "abcdef" | tr -d b-d      # aef


tr -d 0-9 <filename
# Deletes all digits from the file "filename".
```

```
tr "A-Z" "*" <filename      # changes all uppercase letters into *
```

The grep Command:

The grep program searches a file or files for lines that have a certain pattern. g/re/p which means "globally search for a regular expression and print all lines containing it." The syntax is:

```
grep pattern file(s)
```

There are various options which you can use along with grep command:

Option	Description
-v	Print all lines that do not match pattern.
-n	Print the matched line and its line number.
-l	Print only the names of files with matching lines (letter "l")
-c	Print only the count of matching lines.
-i	Match either upper- or lowercase.

grep ’text’	find all lines containing string text
grep ’^text’	find all lines beginning with text
grep ’text$’	find all lines ending with text
grep -v ’text’	find all lines except those containing string text
grep -v ’text$’	find all lines except those ending with text

grep character ranges

There is a lot you can do with grep – we’ll just look at some character ranges

grep ’[0-9]’ find lines containing any number
grep ’[A-Z]’ find lines containing any uppercase letter
grep ’^[A-Z]’ find lines beginning with an uppercase
grep ’[a-z]’ find lines ending with a lowercase
grep ’[aeiouAEIOU]’ find lines containing a vowel
grep ’[^aeiouAEIOU]$’ find lines ending with consonant (not a vowel)
grep -i ’[aeiou]’ find lines ending with vowel (ignore case)
grep -i ’^[^aeiou]’ find lines beginning with consonant (ignore case)