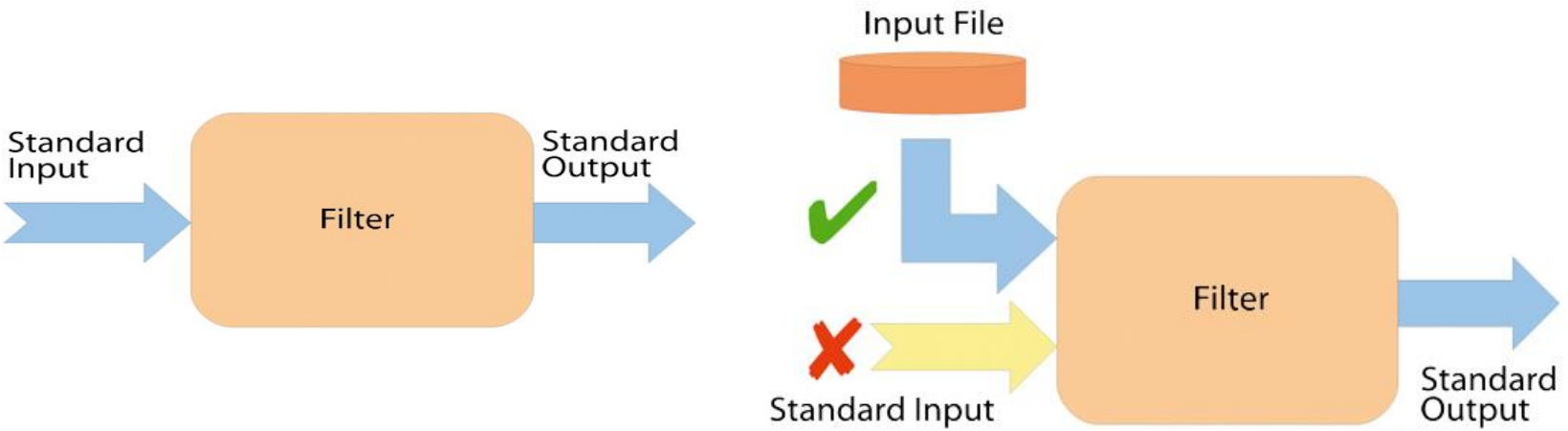
Pipes & Filters

Up close

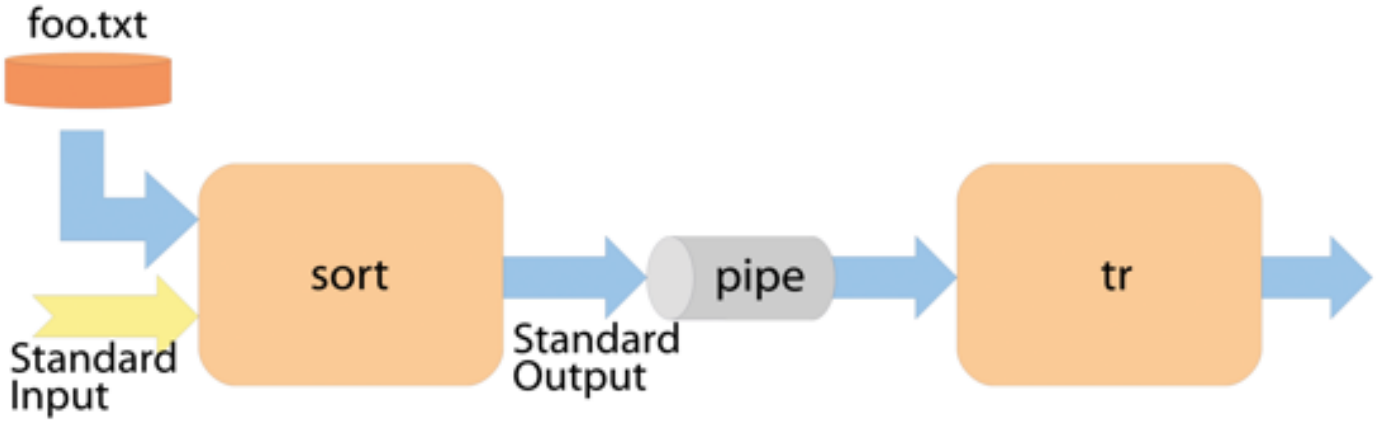
Pipes & filters

- **Filter:** When a **command** performs operations on input and writes the result to the standard output, it is called a ***filter***
  - ***Example: cut, sort, tr, grep, wc, sed, awk(gawk), uniq***



Pipes & filters

- **Pipe**: The standard output (stdout) of one command is sent to the standard input (stdin) of a second command using | (vertical)
  - Example: ls –al | wc –l
  - Example: ls /sbin | grep mk | sort -r | head -3



SORT

SORT filter command:

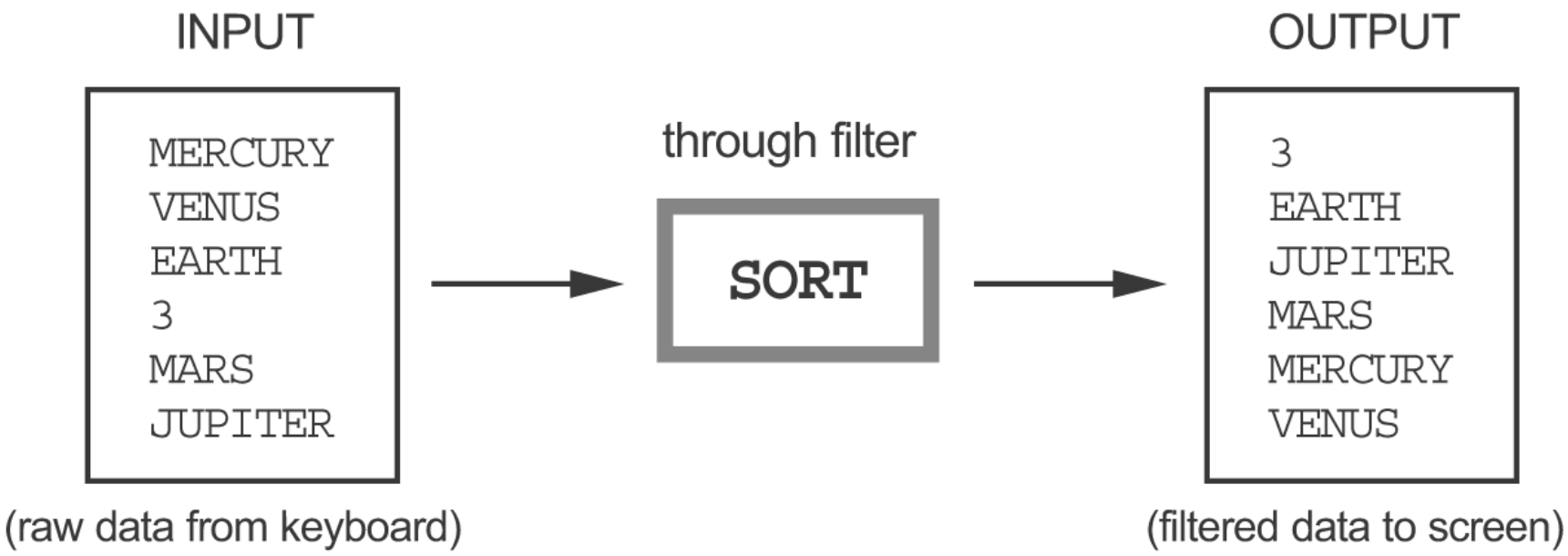
- Arranges lines of input in ascending order
- Sends output to standard output unless redirected

The SORT Command

SORT syntax:

sort [*OPTION*]... [*FILE*]...

Using SORT


INPUT
MERCURY
VENUS
EARTH
3
MARS
JUPITER
(raw data from keyboard)
through filter
SORT
OUTPUT
3
EARTH
JUPITER
MARS
MERCURY
VENUS
(filtered data to screen)

sort options

-M, **month-sort** compare (unknown) < `JAN' < ... < `DEC'

-n, **numeric-sort:** compare according to string numerical value

-r, **reverse:** reverse the result of comparisons

c, **check:** check whether input is sorted; do not sort

-k, **key=*POS1[*,POS2]:** start a key at POS1, end it at POS 2 (origin 1)

-m, **merge** merge already sorted files;

-o, **output=*FILE*** write result to FILE instead of standard output

-t, **field-separator "*SEP*" use SEP instead of non- to whitespace transition**

-u, **unique** with **-c**: check for strict ordering otherwise: output only the first of an equal run

Using SORT

- Numbers are numbers only when mathematical operation performed on them
- Numbers often used as character data
- Character data sorted from left to right
- Numeric data sorted by units

Using SORT

- Sort sequence order:
  - punctuation marks (including spaces)
  - numbers
  - letters (lowercase then uppercase)

Using SORT

- Sort sequence of BB, aa, #, 123, bb, 13, AA
  - # 123 13 aa AA bb BB
- Sort sequence of “Carolyn Smith and Robert Nesler”
  - Carolyn Smith
  - Robert Nesler

Cut filter

- Syntax

  cut *OPTION*... [*FILE*]...

Options for cut

-c, --characters=LIST	Select only the characters from each line as specified in LIST. LIST specifies a character, a set of characters, or a range of characters;
-d, --delimiter=DELIM	use character DELIM instead of a tab for the field delimiter.
-f, --fields=LIST	select only these fields on each line; also print any line that contains no delimiter character, unless the -s option is specified. LIST specifies a field, a set of fields, or a range of fields;
-n	This option is ignored, but is included for compatibility reasons.
--complement	complement the set of selected bytes, characters or fields.
-s, --only-delimited	do not print lines not containing delimiters.
--output-delimiter=STRING	use STRING as the output delimiter string. The default is to use the input delimiter.
--help	Display a help message and exit.
--version	output version information and exit.

To cut selected columns

- To extract only a desired column from a file use -c option

```
[ppk@icttelnet2 sed-dir]$ cat > file1
linux command cut is used for text scriptsing
Cut can be used for extracting portion of text from a file
by selecting columns

[ppk@icttelnet2 sed-dir]$ cut -c2 file1
i
u
y
```

Select Column of Characters using Range

- Range of characters can also be extracted from a file by specifying start and end position delimited with –

```
[ppk@icttelnet2 sed-dir]$ cut -c1-3 file1
lin
Cut
by
```

Select Column of Characters using either Start or End Position (1)

- Either start position or end position can be passed to cut command with -c option

[ppk@icttelnet2 sed-dir]$ cut –c 3 - file1

  - used for text scriptsing
  - It can be used for extracting portion of text from a file selecting columns

Select Column of Characters using either Start or End Position (2)

```
[ppk@icttelnet2 sed-dir]$ cut -c - 8 file1
linux co
Cut can
by selec
```

- Extracts 8 characters from the beginning of each line from file1

Select a Specific Field from a File

```
$ cut -d':' -f1 /etc/passwd
root
daemon
bin
adm
lp
```

- Instead of selecting “n” number of characters, to extract a whole field, combine option -f and -d.
- The option -f specifies which field you want to extract, and
- The option -d specifies what is the field delimiter that is used in the input file.

Select Multiple Fields from a File

```
[ppk@icttelnet2 sed-dir]$ grep "/bin/bash" /etc/passwd |cut -d':' -f1,6
root:/root
administrator:/home/administrator
ppk:/home/ppk
vinay:/home/vinay
1011:/home/1011
1154:/home/1154
1160:/home/1160
1178:/home/1178
1192:/home/1192
1194:/home/1194
```

- Example displays username and home directory of users who has the login shell as “/bin/bash”.

One more example

- To display the range of fields specify start field and end field as shown below. In this example, say field 1 through 4, 6 and 7

```
[ppk@icttelnet2 sed-dir]$ grep "/bin/bash" /etc/passwd |cut -d':' -f1-4,6,7 |more
root:x:0:0:/root:/bin/bash
administrator:x:500:500:/home/administrator:/bin/bash
ppk:x:501:501:/home/ppk:/bin/bash
vinay:x:502:502:/home/vinay:/bin/bash
1011:x:503:100:/home/1011:/bin/bash
1154:x:504:100:/home/1154:/bin/bash
1160:x:505:100:/home/1160:/bin/bash
1178:x:506:100:/home/1178:/bin/bash
1192:x:507:100:/home/1192:/bin/bash
1194:x:508:100:/home/1194:/bin/bash
1246:x:509:100:/home/1246:/bin/bash
1280:x:510:100:/home/1280:/bin/bash
1282:x:511:100:/home/1282:/bin/bash
1294:x:512:100:/home/1294:/bin/bash
```

Change Output Delimiter for Display

```
[ppk@icttelnet2 sed-dir]$ grep "/bin/bash" /etc/passwd |cut -d':' -s -f1, 6,7 --output-delimiter=‘#’
```

- To change the output delimiter use the option –output-delimiter
- The input delimiter is : (colon), but the output delimiter is # (hash).

Filter tr: translate

- Changes a given character or set of characters to another character or set of characters.

  Syntax: tr [options] set1 [set2]

- Examples:
  - tr a b #replaces a with b std i/p
  - tr a-z A-Z <std input entry>

tr: translate

- tr cannot accept the names of files as arguments, it can nevertheless be used to modify copies of their contents. All that is necessary is to use the *input redirection operator*
- *Example:*
  - tr a b < file1
  - tr c d < file1 > file2
  - tr c d < file1 > file1 (incorrect)

tr examples

- cat file5 | tr ‘[efg]' '[xyz]' > file6
- cat file5 | tr '[e-g]' '[x-z]' > file6
- cat file7 | tr A-Z a-z > file8
- cat file7 | tr [:upper:] [:lower:] > file8
- cat file | tr [:space:] ‘\t’ # whitespace to tab

More on tr

- Multiple spaces to single space: -s

  echo “hello there world” |tr –s [:space:] ‘ ’

- Delete specific characters/digits: -d

  echo “institute” | tr –d ‘t’

  echo “digits127889966only” | tr –d [:digit:]

tee command

- Used to store and view (both at the same time) the output of any other command
- Usage:
  - Write output to stdout, and also to a file
  - Write the output to multiple files
- Options: -a (append)

  Example: echo "hello there" | tee -a names.dat

The FIND Filter

- FIND filter command:
  - The **find** command is a powerful *nix utility that allows the user to find files located in the file system via criteria such as the file name, when file was last accessed, when the file status was last changed, the file's permissions, owner, group, size, or even number of inodes.

The FIND Filter

- FIND syntax:

  find <location> <comparison-criteria> <search-term>

Find examples

- **List all files in current and sub directories**

  $ find

  $ find .

  $ find /

- **Search specific directory or path**

  $ find ./shellscripts

  $ find ./shellscripts **–name** “sc?”

  $ find ./shellscripts **–name** “s*.c”

  $ find ./shellscripts **–iname** “s*.c” (ignore case)

Find examples

- **Limit depth of directory traversal**

  $ find ./scripts –maxdepth 2 –name “s*.c”
- **Invert match**

  $ find ./scripts -not –name “s*.c”
- **Combine multiple search criteria**

  $find ./scripts –name “*.c” –o –name “*.txt”
- **Search only files or only directories**

  $ find ./scripts –type f -name “s*” (files)

  $ find ./scripts –type d –name “p*” (directories)

Find examples

- **Search multiple directories together**

  $ find ./scripts ./shellscripts –type f -name “p*” (files)

- **Find hidden files**

  $ find -type f –name “.*”

- **Find files with certain permissions**

  $ find . -type f -perm 0664

- **Find readonly files**

  $ find /etc -maxdepth 1 -perm /u=r

Find examples

- **Find executable files**

  $ find /bin -maxdepth 2 -perm /u=x

Find files modified N days back

$ find ./shellscripts -mtime 50

Find files accessed in last N days

$ find ./shellscripts -atime 50

Find files modified in a range of days

$ find ./shellscripts -mtime +50 –mtime -100

Find empty files and directories

$ find /tmp -type f -empty

grep filter

Syntax:

grep [options] pattern [file]

- The grep command searches either the input or the file you specify for lines that contain characters that match the specified pattern.
- Output from grep is the lines that contain the matching pattern.

grep filter

Examples: $ grep hello file1

hello

$ grep h file1

house

hello

hi

- To find **line numbers** with matching patterns

$ grep –n h file1

2: house

4: hello

5: hi

grep filter

- To find **how many lines with matching** patterns

```
$ grep –c h file1
  3
```

- More than one matching pattern

```
$ grep –e m –e h file1
        mouse
        house
        hello
        hi
```

grep filter

- Use regular expressions in grep search

```
$ grep [mh] file1
        mouse
        house
        hello
        hi
```

- Think of output without square brackets?

Output would look for match for mh

- Variations – egrep & fgrep