H3ABioNet
Pan African Bioinformatics Network for H3Africa

Introduction to Bioinformatics online course: IBT

Linux

Extracting information from files

Learning Objectives

① Learn how to search patterns in files and how to extract specific data

② Learn how to sort files content

③ Learn basic commands to compare files content

④ Learn results redirection

⑤ Learn commands combination

Learning Outcomes

① Be able to search patterns in files extract specific data

②Be able to sort files content

③Be able to use some basic commands to compare files content

④Know how to write commands results into a file

⑤Be able to combine different commands

Part 1

Basic operations on files and data extraction

Some statistics about your file content: wc command

- wc prints newline, word, and byte counts for each file
- syntax: wc <options> <filename>
- Some useful options:
- **-c**: prints the byte counts
- **-m:** prints the character counts
- **-l:** prints the newline counts
- For more info about the different commands use man commandname

Basics operation on files

- **sort**: reorder the content of a file “alphabetically”

  syntax: sort <filename>
- **uniq**: removes duplicated lines

  syntax: uniq <filename>
- **join**: compare the contents of 2 files, outputs the common entries

  syntax: join <filename1> <filename2>
- **diff**: compare the contents of 2 files, outputs the differences

  syntax: diff <filename1> <filename2>

Sorting data

- **sort** outputs a sorted order of the file content based on a specified sort key (default: takes entire input)
- Syntax: sort <options> <filename>
- Default field separator: Blank
- Sorted files are used as an input for several other commands so sort is often used in combination to other commands
- For <options> see man

Sorting data: examples

- Sort alphabetically (default option): sort <filename>
- Sort numerically: sort -n <filename>
- Sort on a specific column (n°4): sort –k 4 <filename>
- Sort based on a tab separator: sort -t $'\t' <filename>
- …

Extracting data from files

- **grep**: to search for the occurrence of a specific pattern (regular expression using the wildcards…) in a file

  Syntax: grep **<pattern>** <filename>

- **cut**: is used to extract specific fields from a file

  Syntax: cut **<options>** <filename>

grep command

- **grep** ("**g**lobal **r**egular **e**xpression **p**rofile") is used to search for the occurrence of a specific pattern (regular expression…) in a file
- Grep output the whole line containing that pattern
- For <options> see man

Example:

Extract lines containing the pattern xxx from a file:

*grep **xxx** <filename>*

Extract lines that do not contain pattern xxx from a file:

*grep **–v** **xxx** <filename>*

grep example

Let's consider a file named "ghandi.txt"

$ cat ghandi.txt

The difference between what we do
and what we are capable of doing
would suffice to solve
most of the world's problems

$ grep what ghandi.txt

*The difference between **what** we do*
*and **what** we are capable of doing*

$ grep -v what ghandi.txt

would suffice to solve
most of the world's problems

cut command

- **cut** is used to extract specific fields from a file
- Structure: cut <options> <filename>
- For <options> see man
- Important options are
  - **-d** (field delimiter)
  - **-f** (field specifier)

Example:

extract fields 2 and 3 from a file having 'space' as a separator

*cut **-d' ' -f2,3** <filename>*

uniq command

- **uniq** outputs a file with no duplicated lines
- Uniq requires a sorted file as an input
- Syntax: uniq <options> <sorted_filename>
- For <options> see man
- Useful option is **-c** to output each line with its number of repeats

Join command

- **join** is used to compare 2 input files based on the entries in a common field (called “join field”) and outputs a merged file
- join requires sorted files as an input
- Lines with identitical “join field” will be present only once in the output
- Structure:

  join <options> <filename1> <filename2>
- For <options> see man

diff command

- **diff** is used to compare 2 input files and displays the different entries
- Can be used to highlight differences between 2 versions of the same file
- Default output: common lines not showed, only different lines are indicated and shows what has been added (**a**), deleted (**d**) or changed (**c**)
- Structure: diff <options> <filename1> <filename2>
- For <options> see man

Part 2

Outputs redirection and combining different commands

Commands outputs

- By **default**, the **standard output** of any command will appear to the **terminal screen**.
- Redirection of the output result to a file is possible.
- This is particularly useful for big files
- Syntax: command options filename.**in** **>** filename.**out**

Outputs redirection

- If the file exists, the result will be redirected to it

```
$ cat ghandi.txt
The difference between what we do
and what we are capable of doing
would suffice to solve
most of the world's problems
$ cut -d' ' -f2,3 ghandi.txt
difference between
what we
suffice to
of the

$ cut -d' ' -f2,3 ghandi.txt > ghandi.txt.out
$ cat ghandi.txt.out
difference between
what we
suffice to
of the
```

- If the file does not exist, it will be automatically created and the result redirected to it.

Commands combination

- The **standard output** of any command will be **one unique output**
- As seen previously, this output can be **printed** in the screen or **redirected to a file**
- However, the **output** result of a command can also be **redirected to another command**
- This is particularly useful when several operations are needed for a file, with no need to store the intermediate outputs

Commands combination: example

- Combining several commands is done thanks to the use of a "|" character

- Structure:

command1 options1 filename1.**in** |command2 options2 **>** filename.**out**

- This can be done for as many commands as needed

Thanks

Shaun Aron & Sumir Panji