Linux: Beyond the Basics

Bingbing Yuan
BaRC Hot Topics – Sep. 2020
Bioinformatics and Research Computing
Whitehead Institute

http://barc.wi.mit.edu/hot_topics/
Logging in to our Linux server

• Our main Unix/Linux server is called tak
• Request a tak account:
  http://bioinfo.wi.mit.edu/bio/software/unix/bioinfoaccount.php

• Connecting to tak:
  http://bioinfo.wi.mit.edu/bio/software/unix/serverConnect.php
  – Windows:
    ➢ MobaXterm
  – Mac
    ➢ Access through Terminal
Log in to tak for Mac

```
ssh -Y username@tak.wi.mit.edu

username@tak.wi.mit.edu's password:
```

Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-50-generic x86_64)

Tak

- System information as of Fri Sep 13 15:53:49 EDT 2019
  System load: 1.31 Memory usage: 72 Processes: 587
  Usage of /: 8.5G of 430.14G Swap usage: 0k Users logged int 16
  * Find genomes at /ifs/genomes
  * Find bioinformatics datasets at /ifs/BaRC_databases

byuan@tak ~$
Connecting to tak from Windows

1) Open MobaXterm

2) Click on the "Start local terminal" button

3) Type: `ssh -Y username@tak.wi.mit.edu`

Note: When you write the password you won’t see any characters being typed.
Hot Topics website:
http://barc.wi.mit.edu/education/hot_topics/

• After you login to tak, create a directory for the exercises within your home directory, and use it as your working directory
  $ mkdir Hot_Topics
  $ cd Hot_Topics

• Copy all files into your working directory
  $ cp -r /nfs/BaRC_training/Linux_Beyond/data_files/* .

• You should have the files below in your working directory:
  – foo.txt, sample1.txt, exercise.txt, Ensembl_info.txt, Gene_exp.txt,
    HumanGenesPlusMinus3kb.bed, peaks.bed, datasets folder
  – You can check they’re there with the ‘ls’ command
Linux Review: Commands

- command [arg1 arg2 ...] [input1 input2 ...]

$ sort -k2,3nr foo.tab

  start end  -n or -g: -n is recommended, except for scientific notation or a leading '+'
  -r: reverse order

$ cut -f1,5 foo.tab

$ cut -f1-5 foo.tab

  -f: select only these fields
  -f1,5: select 1\textsuperscript{st} and 5\textsuperscript{th} fields
  -f1-5: select 1\textsuperscript{st}, 2\textsuperscript{nd}, 3\textsuperscript{rd}, 4\textsuperscript{th}, and 5\textsuperscript{th} fields

$ wc -l foo.txt

  How many lines are in this file?
Linux Review: Common Mistakes

• Case sensitive
  cd /nfs/Barc_Public is different from cd /nfs/BaRC_Public
  `bash: cd: /nfs/Barc_Public: No such file or directory`

• Spaces may matter!
  `rm -f myFiles* vs rm -f myFiles *`

• Office applications can convert text to special characters that Linux won’t understand
  • Smart quotes, dashes
  • Carriage return from DOS
    – Use fromdos to remove carriage return
Linux Review: Pipes

• Stream output of one command/program as input for another
  – Avoid intermediate file(s)
  – Merge multiple commands in to one long command

$ cut -f1 myFile.txt | sort | uniq -c > uniqCounts.txt

pipes
What we will discuss today

- Aliases (to reduce typing)
- sed (for file manipulation)
- awk (to filter by column)
- join (merge files)
- groupBy and intersect from bedtools (not typical Linux)
- loops (one-line and with shell scripts)
- scripting (to streamline commands)
Aliases

• Add a one-word link to a longer command

• To get current aliases (from ~/.bashrc)

    alias

• Create a new alias (two examples)

    alias sp='cd /lab/solexa_public/Reddien'
    alias CollectRnaSeqMetrics='java -jar /usr/local/share/picard-tools/CollectRnaSeqMetrics.jar'

• Make an alias permanent

    – Paste command(s) in ~/.bashrc
sed: stream editor for filtering and transforming text

- Print lines 10 - 15:
  \[\texttt{sed -n '10,15p' bigFile > selectedLines.txt}\]

- Delete 5 header lines at the beginning of a file:
  \[\texttt{sed '1,5d' file > fileNoHeader}\]

- Remove all version numbers (eg: '.1') from the end of a list of sequence accessions: eg. NM_000035.2
  \[\texttt{sed 's/\.[0-9]\+///g' accsWithVersion > accsOnly}\]

  - \textit{s: substitute} \hspace{1cm} g: global modifier (change all)

- Get good examples from “sed cheat sheet”:
  – [https://www.pement.org/sed/sed1line.txt](https://www.pement.org/sed/sed1line.txt)
Join files together

With Linux join

\$ join -1 1 -2 2 --nocheck-order -t $'\t' sorted_File1 sortedFile2

Join files on the 1st field of FILE1 with the 2nd field of FILE2, only showing the common lines.

FILE1 and FILE2 must be sorted on the join fields before running join -t $'\t' : Both input and output use tab as separator --nocheck-order: good for sorted files with header line.

Sorted sample tables to join:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Heart</th>
<th>Skeletal Muscle</th>
<th>Skin</th>
<th>Smooth Muscle</th>
<th>Spinal cord</th>
</tr>
</thead>
<tbody>
<tr>
<td>HHAT</td>
<td>8.15</td>
<td>7.7</td>
<td>5</td>
<td>6.55</td>
<td>6.4</td>
</tr>
<tr>
<td>INPP5D</td>
<td>19.65</td>
<td>5.95</td>
<td>4.55</td>
<td>5.25</td>
<td>14.5</td>
</tr>
<tr>
<td>NDUFA1</td>
<td>441.8</td>
<td>160.2</td>
<td>24.9</td>
<td>188.85</td>
<td>158.75</td>
</tr>
<tr>
<td>RPS6KA1</td>
<td>85.2</td>
<td>47.75</td>
<td>46.45</td>
<td>35.85</td>
<td>44.55</td>
</tr>
<tr>
<td>RYBP</td>
<td>20.45</td>
<td>13.05</td>
<td>11.95</td>
<td>20.7</td>
<td>17.75</td>
</tr>
<tr>
<td>SLC16A1</td>
<td>15.45</td>
<td>20.45</td>
<td>12.2</td>
<td>248.35</td>
<td>27.15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Heart</th>
<th>Skeletal Muscle</th>
<th>Skin</th>
<th>Smooth Muscle</th>
<th>Spinal cord</th>
</tr>
</thead>
<tbody>
<tr>
<td>HHAT</td>
<td>8.15</td>
<td>7.7</td>
<td>5</td>
<td>6.55</td>
<td>6.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ensembl Gene ID</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000280680</td>
<td>HHAT</td>
</tr>
<tr>
<td>ENSG00000280820</td>
<td>LCN1P1</td>
</tr>
<tr>
<td>ENSG00000280584</td>
<td>OBP2B</td>
</tr>
<tr>
<td>ENSG00000280775</td>
<td>RNA5SP136</td>
</tr>
<tr>
<td>ENSG00000252303</td>
<td>RNU6-280P</td>
</tr>
<tr>
<td>ENSG00000280963</td>
<td>SERTAD4-AS1</td>
</tr>
</tbody>
</table>
Regular Expressions

• A sequence of characters defining a search pattern
• Powerful, but syntax is often non-intuitive

• Examples

List all txt files: **ls *.txt**

Replace CHR with Chr at the beginning of each line:

$ sed 's/^CHR/Chr/' myFile.txt

Delete a dot followed by one or more numbers

$ sed 's/\.[0-9]+//g' myFile.txt

<table>
<thead>
<tr>
<th></th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>All characters</td>
</tr>
<tr>
<td>*</td>
<td>Zero or more; wildcard</td>
</tr>
<tr>
<td>+</td>
<td>One or more</td>
</tr>
<tr>
<td>?</td>
<td>One</td>
</tr>
<tr>
<td>^</td>
<td>Beginning of a line</td>
</tr>
<tr>
<td>$</td>
<td>End of a line</td>
</tr>
<tr>
<td>[ab]</td>
<td>Any character in brackets</td>
</tr>
</tbody>
</table>

• Note: regular expression syntax may slightly differ between sed, awk, Linux shell, and Perl
  – Ex: \+ in sed is equivalent to + in Perl
awk

• A simple programing language to process files
• Good for filtering and manipulating multiple-column files
• “awk” comes from the original authors:
  Alfred V. Aho, Peter J. Weinberger, Brian W. Kernighan
awk

- By default, awk splits each line by spaces

- Print the 2nd and 1st fields of the file:
  $ awk '{ print $2"\t"$1 }' foo.tab

- Convert sequences from tab delimited format to fasta format:

  $ head -1 foo.tab
  Seq1   ACTGCATCAC
  $ awk '{ print "">" $1 "\n" $2 }' foo.tab > foo.fa
  $ head -2 foo.fa
  >Seq1
  ACGCATCAC
awk: field separator

• Issues with default separator (white space)
  – one field is gene description with multiple words
  – consecutive empty cells

• To use tab as the separator:

\$ awk -F "\t" '{ print NF }' foo.txt

or

\$ awk 'BEGIN {FS="\t"} { print NF }' foo.txt

BEGIN: action before read input
NF: number of fields in the current record
FS: input field separator
OFS: output field separator
END: action after read input

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n</td>
<td>newline</td>
</tr>
<tr>
<td>\r</td>
<td>carriage return</td>
</tr>
<tr>
<td>\t</td>
<td>horizontal tab</td>
</tr>
</tbody>
</table>
awk: arithmetic operations

Add average values of 4\text{th} and 5\text{th} fields to the file:

$ awk '{ print $0 "\t" ($4+$5)/2 }' foo.tab$

$0: all fields

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td>%</td>
<td>Modulo</td>
</tr>
<tr>
<td>^</td>
<td>Exponentiation</td>
</tr>
<tr>
<td>**</td>
<td>Exponentiation</td>
</tr>
</tbody>
</table>
awk: making comparisons

Print out records if values in 4th or 5th field are above 4:
$ awk '{ if( $4>4 || $5>4 ) print $0 } ' foo.tab

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
<tr>
<td>~</td>
<td>Matches</td>
</tr>
<tr>
<td>!~</td>
<td>Does not match</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Logical AND</td>
</tr>
</tbody>
</table>
More awk examples

• Conditional statements:

Display expression levels for the gene NANOG:
$ awk '{ if(/NANOG/) print $0 }' foo.txt
or
$ awk '/NANOG/ { print $0 }' foo.txt
or
$ awk '/NANOG/' foo.txt

Add line number to the above output:
$ awk '/NANOG/ { print NR"\t"$0 }' foo.txt

NR: line number of the current row

• Looping:

Calculate the average expression (4th, 5th and 6th fields in this case) for each transcript
$ awk '{ total= $4 + $5 + $6; avg=total/3; print $0"\t"avg}' foo.txt
or
$ awk '{ total=0; for (i=4; i<=6; i++) total=total+$i; avg=total/3; print $0"\t"avg }' foo.txt
intersect from bedtools
( intersectBed )

• Find overlaps between two sets of genomic features

https://bedtools.readthedocs.io/en/latest/content/tools/intersect.html
intersectBed: Examples

$ head -2 HumanGenesPlusMinus3kb.bed

| chr1 | 50899700 | 50905978 | ENSG00000271782_RP5-850O15.4 |
| chr1 | 103814769 | 103831355 | ENSG00000232753_RP11-347K2.1 |

$ head -2 peaks.bed

| chr1 | 19921 | 20016 | MACS_peak_1 | 50 |
| chr1 | 20025 | 20890 | MACS_peak_2 | 568 |

$ intersectBed -wa -wb -a HumanGenesPlusMinus3kb.bed -b peaks.bed | head -2

| chr1 | 45956538 | 45968751 | ENSG000000236624_CCDC163P | chr1 | 45955389 | 45956863 | MACS_peak_3385 | 1192.24 |
| chr1 | 45956538 | 45968751 | ENSG000000236624_CCDC163P | chr1 | 45957202 | 45957380 | MACS_peak_3386 | 121.87 |

$ intersectBed -a HumanGenesPlusMinus3kb.bed -b peaks.bed | head -2

| chr1 | 45956538 | 45956863 | ENSG000000236624_CCDC163P |
| chr1 | 45957202 | 45957380 | ENSG000000236624_CCDC163P |

$ intersectBed -a HumanGenesPlusMinus3kb.bed -b peaks.bed | cut -f4 | sort -u | head -2

ENSG00000000003_TSPAN6
ENSG00000000419_DPM1
Summarize by Columns: groupBy (from bedtools)

Input file must be pre-sorted by grouping column(s)!

```
$ sort -k1,1 Ensembl_info.txt | groupBy -g 1 -c 3,2 -o distinct,collapse
```

Partial output

<table>
<thead>
<tr>
<th>Ensembl Gene ID</th>
<th>Ensembl Transcript ID</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000281518</td>
<td>ENST00000627423</td>
<td>FOXO6</td>
</tr>
<tr>
<td>ENSG00000281518</td>
<td>ENST00000630406</td>
<td>FOXO6</td>
</tr>
<tr>
<td>ENSG00000280680</td>
<td>ENST00000625523</td>
<td>HHAT</td>
</tr>
<tr>
<td>ENSG00000280680</td>
<td>ENST00000627903</td>
<td>HHAT</td>
</tr>
<tr>
<td>ENSG00000280680</td>
<td>ENST00000626327</td>
<td>HHAT</td>
</tr>
<tr>
<td>ENSG00000281614</td>
<td>ENST00000629761</td>
<td>INPP5D</td>
</tr>
<tr>
<td>ENSG00000281614</td>
<td>ENST00000630338</td>
<td>INPP5D</td>
</tr>
</tbody>
</table>

Input file must be pre-sorted by grouping column(s)!
Shell Flavors

• Syntax (for scripting) depends on the shell
  ```
  echo $SHELL
  # /bin/bash (on tak)
  ```
• bash is common and the default on tak.
• Some Linux shells (incomplete listing):

<table>
<thead>
<tr>
<th>Shell</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>sh</td>
<td>Bourne</td>
</tr>
<tr>
<td>bash</td>
<td>Bourne-Again</td>
</tr>
<tr>
<td>ksh</td>
<td>Korn shell</td>
</tr>
<tr>
<td>csh</td>
<td>C shell</td>
</tr>
</tbody>
</table>
Shell script advantages

• Automation: avoid having to retype the same commands many times
• Ease of use and more efficient
• Outline of a script:
  
  ```bash
  #!/bin/bash  # shebang: interprets how to run the script
  commands...  # set of commands used in the script
  #comments  # write comments using “#”
  ```

• Commonly used extension for script is .sh (eg. foo.sh), file must have executable permission
Bash Shell: ‘for’ loop

- Process multiple files with one command
- Reduce computational time with many cluster nodes

```bash
for mySam in `\(\text{/bin/ls *sam}\)`
do
    bsub wc -l $mySam
done
```

*When referring to a variable, \$ is needed before the variable name ($mySam), but \$ is not needed when defining it (mySam).*

Identical one-line command:
```bash
for samFile in `\(\text{/bin/ls *sam}\)`; do bsub wc -l $samFile; done
```
Shell script example

#!/bin/bash

# 1. Take two arguments: the first one is a directory with all the datasets, the second one is for output
# 2. For each file, calculate average gene expression, and save the results in a file in the output directory

inDir=$1                # 1st argument
outDir=$2               # 2nd argument; outDir must already exist
                        # Define variables: no spaces on either side of the equal sign
for i in `/bin/ls $inDir` \  # refer to variable with $
    do
        # output file name
        outFileName="${i}_avg.txt"            # {}: $i_avg is not valid;
        # calculate average gene expression
        # NM_001039201    Hdhd2   5.0306    5.3309   5.4998
        bsub "sort -k2,2 $inDir/$i | groupBy -g 2 -c 3,4,5 -o mean,mean,mean >| $outDir/$name"
    done

# You can use editors such as nedit, gedit, emacs, or pico to create shell scripts
Accessing Shared Resources at Whitehead

- **Linux**
  - /nfs/BaRC_Public
  - /nfs/BaRC_training
  - /lab/solexa_public
- **Windows** (access using Start Menu → Search)
  - \wi-files1\BaRC_Public
  - \wi-files1\BaRC_training
  - \wi-bigdata\solexa_public
- **Macs** (access using Go → Connect to Server...)
  - smb://wi-files1/BaRC_Public
  - smb://wi-files1/BaRC_training
  - smb://wi-bigdata/solexa_public

Where’s my lab’s share?

- [http://it.wi.mit.edu/systems/file-storage/lab-share-paths](http://it.wi.mit.edu/systems/file-storage/lab-share-paths)
Further Reading

• BaRC one liners:
  – http://bioinfo.wi.mit.edu/bio/bioinfo/scripts/#unix

• Linux Info for Bioinfo:

• Online books via MIT:
  – http://proquest.safaribooksonline.com/
  – Bioinformatics Data Skills by V.Buffalo (2015)

• Bash Guide for Beginners:
Upcoming Hot Topics

• An Introduction to R - October
• An Introduction to R Graphics - October
• Python: An Introduction - November
• Python: Advanced Topics – November

http://barc.wi.mit.edu/education/hot_topics/upcoming/