1 Basic linux commands

Here will be briefly explained some basic linux commands. For information about more commands, feel free to consult with the internet.

1.1 Getting around

Command	Explanation
`ls`	list contents of current directory
`ls` *dirname*	list contents of directory named *dirname*
`cd` *dirname*	change current directory to *dirname*
`cd ~`	go to home directory (default directory)
`cd ..`	go one directory up
`pwd`	print name of current directory
`mkdir` *dirname*	make a new directory named *dirname*
`man` *command*	show manual for *command*

1.2 File manipulation

Command	Explanation
`cp` *filename1 filename2*	copy file
`cp -r` *dirname1 dirname2*	copy directory
`mv` *name1 name2*	move file or directory (can be used for renaming)
`rm` *filename*	remove file (you won't be able to recover removed file)
`rm -r` *dirname*	remove directory and its contents
`more` *filename*	command for paging through text file one screenful at a time
`less` *filename*	similar as more. Better for viewing large text files
`cat` *filename1 filename2 filenameN*	concatenate text files in print output to terminal window
`head` *filename*	print first 10 lines of a text file to terminal window
`head -n 20` *filename*	print first 20 lines of a text file to terminal window
`tail`	opposite of `head`
`wc` *filename*	show the number of lines, words and characters in a text file
`cut -f 2` *tabDelimitedFilename*	extract 2nd column from a tab delimited file
`cut -f 3 -d ,` *comaSeperatedFilename*	extract 3rd column from a coma seperated file
`nano` *filename*	open *filename* in text editor `nano`

1.3 Archiving and unarchiving of files

Note that for the `tar` utility, option `c` stands for compress, `x` - for uncompress or extract, `z` - for dealing with tar.gz, and `j` - for dealing with tar.bz2

1.3.1 Compressing

Command	Explanation
`tar -cvf` *filename.tar filename*	**compress** file to **.tar** format
`tar -zcvf` *filename.tar.gz filename*	**compress** file to **.tar.gz** format
`tar -jcvf` *filename.tar.bz2 filename*	**compress** file to **.tar.bz2** format
`zip` *filename.zip filename*	**compress** file to **.zip** format
`gzip` *filename*	**compress** file to **.gz** format

ll

1.3.2 Uncompressing

Command	Explanation
`tar -xvf` *filename.tar*	**uncompress** from **.tar** format
`tar -zxvf` *filename.tar.gz*	**uncompress** from **.tar.gz** format
`tar -jxvf` *filename.tar.bz2*	**uncompress** from **.tar.bz2** format
`unzip` *filename.zip*	**uncompress** from **.zip** format
`gzip -d` *file.gz*	**uncompress** from **.gz** format

1.4 Input/Output redirection

Command	Explanation
command `>` *filename*	Output of *command* is saved to *filename*, overwriting it
command `>>` *filename*	Output of *command* is appended at the end of *filename*
command `<` *filename*	*command* reads input from *filename*
command1 `|` *command2*	*command2* takes the output of *command1* and produces result

1.5 Filters

Command	Explanation
`grep` *text filename*	Prints every line in *filename* containing *text*
`sed 's/`*red*`/`*green*`/'` *filename*	Prints every line in *filename* substituting word *red* with word *green*

1.6 Pattern matching

Pattern	Explanation
`*`	matches zero or more characters
`?`	matches one character

1.7 Miscellaneous

Command	Explanation
`echo` *text*	display a line of text
`history`	view your command line history
`wget` *someWebAddress*	download contents of *someWebAddress* to current directory

2 Setup of the working environment

Let's create directories for our data:

```
mkdir programs
mkdir ngs_work
mkdir binaries
```

Many of the open source tools are deposited in the `https://github.com` repository. To download software from `https://github.com` easily, we will use a tool called `git`. `git` is already preinstalled on our servers, however, on your own Ubuntu servers you can install it by typing:

```
sudo apt-get install git
```

3 *De-novo* assembly of sequenced reads

3.1 Installation of *de-novo* assembler

For *de-novo* assembly we will use `mira` assembler. You can download it from `http://sourceforge.net/projects/mira-assembler/`. Click on Files → MIRA → stable. Rightclick on `mira_4.0.2_linux-gnu_x86_64_static.tar.bz2` and `Copy link address`. To download it on our linux server, we will be using command `wget`. In linux terminal type:

```
cd ~/programs
wget -O mira.tar.bz2
```

and paste the copied location. The program will be downloaded in our `~/programs` directory. The downloaded software is archived in .tar.bz2 format, therefore we need to extract it from archive. To extract it from archive, type in terminal:

```
tar -jxvf mira.tar.bz2
```

A new folder named `mira_4.0.2_linux-gnu_x86_64_static` will appear. This is our extracted software. MIRA is already precompiled for us, so we just need to find the compiled binary files in the folder and copy them to appropriate directory:

```
cd mira_4.0.2_linux-gnu_x86_64_static
cd bin
cp mira ~/binaries
```

```
cd ~
```

Let's create a seperate directory for our *de-novo* assembly project and copy the reads in it:

```
cd ngs_work
mkdir denovo
cd denovo
cp ~/data/readsfordenovo.fastq .
```

MIRA needs a manifest file for performing *de-novo* assembly. We will create a basic manifest file. Our manifest file will consist of 5 entries. From MIRA's manual, these entries are:

- **project** - name of our assemblies project. The project name will be used by MIRA in project's directory naming
- **job** - tells the assembler whether
    1. we want to perform *de-novo* assembly or map reads against reference genome
    2. genomic DNA or transcripts were sequenced
    3. we want accurate (slow) or draft (fast) assembly
- **readgroup** - tells assembler which reads can be pooled together when assembling reads from multiple sequencing technologies
- **data** - tells the assembler where are our reads
- **technology** - tells the assembler what sequencing technology was used for generating reads

To create manifest file, we will use text editor `nano`. To start the text editor, type

```
nano
```

and the editor will open. Now, to create the manifest file, in the text editor type:

```
project=readsfordenovo_Assembly
job=denovo,genome,draft
readgroup
data=readsfordenovo.fastq
technology=iontor
```

To save the text file hit `Ctrl o`, enter the name of the file (e.g. `readsfordenovo.mnfst`), hit `Enter` to save and `Ctrl x` to quit `nano`. To launch MIRA, type:

```
~/binaries/mira readsfordenovo.mnfst
```

If you wish to gain finer control of some aspects of the assembling process, then, please, do refer to the MIRA's manual.

4 Building variant calling pipeline

We have *IonTorrent* targeted resequencing data from human chromosomes 1., 2. and 19. Our task is to find all nonsynonymous and stop mutations present in the data and to automatize this process by building data analysis pipeline. To accomplish this task we can divide our work in following subtasks:

- Installation of relevant tools
- Obtaining of reference sequences
- Read mapping against reference genome
- Variant calling
- Variant annotation
- Filtering of nonsynonymous and stop mutation variants
- Pipeline building

4.1 Obtaining reference sequences

We begin our task by obtaining reference genome. We will download reference sequences in a seperate directory to avoid file cluttering. Let's make a new directory in our `ngs_work` named `reseq`, and there we will create a seperate folder `ref` for our reference sequences:

```
cd ~
cd ngs_work
mkdir reseq
cd reseq
mkdir ref
cd ref
```

Our reference sequences can be accessed from a database made by University of California, Santa Cruz. The web address of the database is `http://genome.ucsc.edu/`. To find the necessary references sequences for chromosomes 1., 2. and 19. click on `Downloads` → `human` → `Data set by chromosome`. Right click on `chr1.fa.gz` and choose `Copy link location`. In terminal type:

```
wget
```

and paste the copied location.