The UNIX terminal

connection to ec2-54-163-191-2.compute-1.amazonaws.com closed.
Milans-MacBook-Pro:~ milanmalinsky$ ssh ubuntu@ec2-54-161-181-1.compute-1.amazonaws.com
The authenticity of host 'ec2-54-161-181-1.compute-1.amazonaws.com (54.161.181.1)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-54-161-181-1.compute-1.amazonaws.com,54.161.181.1' (RSA) to the list of known hosts.
ubu3@ec2-54-161-181-1compute-1.amazonaws.com's password:
Welcome to Ubuntu 15.10 (GNU/Linux 4.2.0-23-generic x86_64)

* Documentation: https://help.ubuntu.com/

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

Last login: Thu Jan 21 21:11:11 2016 from 90.176.140.61
ubu3@ip-10-144-40-7:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos wpsg_2016
ubu3@ip-10-144-40-7:~$ cd wpsg_2016/
activities/ software/ source/
ubu3@ip-10-144-40-7:~$ cd wpsg_2016/activities/
bayescn/
ubu3@ip-10-144-40-7:~$ cd wpsg_2016/activities/
ubu3@ip-10-144-40-7:~/wpsg_2016/activities$ exit
logout
Connection to ec2-54-161-181-1.compute-1.amazonaws.com closed.
Milans-MacBook-Pro:~ milanmalinsky$ ssh ubuntu@ec2-54-161-181-1.compute-1.amazonaws.com
ubu3@ec2-54-161-181-1compute-1.amazonaws.com's password:
Welcome to Ubuntu 15.10 (GNU/Linux 4.2.0-23-generic x86_64)

* Documentation: https://help.ubuntu.com/

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

Last login: Fri Jan 22 09:42:18 2016 from 90.176.140.61
ubu3@ip-10-144-40-7:~$ 
On the workshop desktop
On a Mac computer
Why use something from the 1960s?
Why use something from the 1960s?

**Scripting:** Write down a sequence of commands to perform a task.

In genomics, a task almost always takes minutes, sometimes hours - not fun to sit and wait this long for the next mouse-click.
Why use something from the 1960s?

**Scripting:** Write down a sequence of commands to perform a task.

In genomics, a task almost always takes minutes, sometimes hours - not fun to sit and wait this long for the next mouse-click.

**Easy remote access:** Running a real world genomics project on your own computer is impossible; you will usually access high performance compute facilities at your university.
Why use something from the 1960s?

**Scripting:** Write down a sequence of commands to perform a task
In genomics, a task almost always takes minutes, sometimes hours - not fun to sit and wait this long for the next mouse-click.

**Easy remote access:** Running a real world genomics project on your own computer is impossible; you will usually access high performance compute facilities at your university.

**GUI for many programs not available:** Genomics is a fast moving field and developing a graphical interface takes time and effort.
Why use something from the 1960s?

**Scripting:** Write down a sequence of commands to perform a task
In genomics, a task almost always takes minutes, sometimes hours - not fun to sit and wait this long for the next mouse-click.

**Easy remote access:** Running a real world genomics project on your own computer is impossible; you will usually access high performance compute facilities at your university.

**GUI for many programs not available:** Genomics is a fast moving field and developing a graphical interface takes time and effort

**Powerful tools available in UNIX:** enabling you to work through large amounts of files, data, and tasks quickly in an automated (programmatic) way
How to survive without ‘seeing’ your data, without pop-out menus, etc.?

1. **Google**: get a cheat-sheet with a list of possible commands:

   ![Google search for unix commands]

   **Basic UNIX commands**
   mally.stanford.edu/~sr/computing/basic-unix.html
   Note that some of these commands are different on non-Solaris machines - see SunOS
   ... But you can also edit the command line (see the guide to More UNIX).
   More UNIX Commands - Emacs - Pathnames

2. **Manual pages for each command**:  

   ![Ubuntu manual page for `ls`]

   What do the `cp`, `vi`, and `awk` commands do? try: `man cp`, `man vi`, `man awk`
How to survive without ‘seeing’ your data, without pop-out menus, etc.?

3. Help: more concise than manual pages; describes different ways to run the program:
   
   **ubuntu@ip-10-144-40-7:**~$ bcftools --help
   or
   **ubuntu@ip-10-144-40-7:**~$ bcftools -h

4. Genomics tools also usually have manual pages:
   **ubuntu@ip-10-144-40-7:**~$ man bcftools

5. Google again:

   ![Google search for bcftools](image_url)
   About 23,700 results (0.37 seconds)

   **bcftools - Samtools**
   https://samtools.github.io/bcftools/bcftools.html
   Nov 16, 2015 - DESCRIPTION. BCFtools is a set of utilities that manipulate variant calls in the Variant Call Format (VCF) and its binary counterpart BCF.
UNIX directory structure

/   
  ├── home
  │    └── ubuntu
  │        └── wpsg_2016
  │            └── activities
  │                  ├── software
  │                          └── source
  │            └── Desktop
  │                    └── ...
  ├── usr
  │    └── ...
  ├── bin
  │    └── ...
  └── ...

UNIX directory structure

Home Directory:  ~

That’s where you are upon login:

ubuntu@ip-10-179-185-48:~$
Go to the ‘activities’ folder and find what is in there:

```
ubuntu@ip-10-179-185-48:~$ cd wpsg_2016/
ubuntu@ip-10-179-185-48:~/wpsg_2016$ cd activities/
ubuntu@ip-10-179-185-48:~/wpsg_2016/activities$ ls -lah
```

total 32K
```
drwxrwxr-x 8 ubuntu ubuntu 4.0K Jan 21 11:53 .
drwxrwxr-x 5 ubuntu ubuntu 4.0K Jan 19 23:24 ..
drwxrwxr-x 4 ubuntu ubuntu 4.0K Jan 21 11:55 bayescan
drwxrwxr-x 6 ubuntu ubuntu 4.0K Jan 21 12:02 bayescenv
drwxrwxr-x 2 ubuntu ubuntu 4.0K Jan 20 17:52 fineSTRUCTURE
drwxrwxr-x 2 ubuntu ubuntu 4.0K Jan 21 11:54 RADseq_R
```
drwxrwxr-x 2 ubuntu ubuntu 4.0K Jan 21 15:31 STRUCTURE
```
drwxrwxr-x 2 ubuntu ubuntu 4.0K Jan 21 17:01 vcftools_plink
```
UNIX directory structure

Go to the ‘activities’ folder and find what is in there:

```
$ cd wpsg_2016/
$ cd wpsg_2016/activities/
$ ls -lah
```

```
total 32K
drwxrwxr-x 8 ubuntu ubuntu 4.0K Jan 21 11:53 .
drwxrwxr-x 5 ubuntu ubuntu 4.0K Jan 19 23:24 ..
drwxrwxr-x 4 ubuntu ubuntu 4.0K Jan 21 11:55 bayescan
drwxrwxr-x 6 ubuntu ubuntu 4.0K Jan 21 12:02 bayescenv
drwxrwxr-x 2 ubuntu ubuntu 4.0K Jan 20 17:52 fineSTRUCTURE
drwxrwxr-x 2 ubuntu ubuntu 4.0K Jan 21 11:54 RADseq_R
drwxrwxr-x 2 ubuntu ubuntu 4.0K Jan 21 15:31 STRUCTURE
drwxrwxr-x 2 ubuntu ubuntu 4.0K Jan 21 17:01 vcftools_plink
```

Two special ‘files’ present in every folder:
1) dot - means ‘this directory’
Go to the ‘activities’ folder and find what is in there:

Two special ‘files’ present in every folder:
1) dot - means ‘this directory’
2) dot dot - mean ‘directory above’
What is the ‘absolute path’ to the activities folder?
UNIX directory structure

What is the ‘absolute path’ to the activities folder?

```
ubuntu@ip-10-179-185-48:~/wpsg_2016/activities$ pwd
/home/ubuntu/wpsg_2016/activities
```
UNIX directory structure

What would be the ‘absolute path’ to the ‘software’ folder?
UNIX directory structure

What would be the ‘absolute path’ to the ‘software’ folder?

/home/ubuntu/wpsg_2016/software
UNIX directory structure

What would be the ‘absolute path’ to the ‘software’ folder?

/home/ubuntu/wpsg_2016/software

And ‘relative path’ from the activities folder to the software folder?
UNIX directory structure

What would be the ‘absolute path’ to the ‘software’ folder?

/home/ubuntu/wpsg_2016/software

And ‘relative path’ from the activities folder to the software folder?

../software
Many ‘programs’ are in the `/bin` folder

You have already seen some in action:

- **cd**  - Change directory (part of the ‘**bash**’ program)
- **ls**  - List directory contents
- **pwd** - Know where you are (‘print working directory’)

```
Many programs are in the /bin folder

You have already seen some in action:
- cd - Change directory (part of the ‘bash’ program)
- ls - List directory contents
- pwd - Know where you are (‘print working directory’)
```
### A few ways to view a text file:

<table>
<thead>
<tr>
<th>less</th>
<th>head</th>
<th>tail</th>
<th>cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>view a text file one screen-full at a time</td>
<td>view the top 10 lines of a file</td>
<td>view the bottom 10 lines of a file</td>
<td>print the whole file at once on screen</td>
</tr>
<tr>
<td>space-bar: scroll</td>
<td>-n num option controls the number of lines</td>
<td>-n num option controls the number of lines</td>
<td></td>
</tr>
<tr>
<td>q: quit</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Exercise 1: Navigating the file system and some UNIX system files

1. Move to the directory /etc
   • What is the first line of the file ‘hosts’ in the directory /etc?
   • What is the relative file path to get to /var/log from here? What is the absolute path?

2. Move to the directory /var/log
   • What is the contents on line 73 of the ‘dpkg.log’ file?
   • Without changing directories, what is the second line of the ‘cpuinfo’ file in the /proc directory?
   • What is the command to read this file with a relative path?
   • An absolute path?

3. Move back to your home folder (remember ~), what directories do you see?
Helpful features to save you typing:

1. Tab completion
   • <tab> once to complete a ‘word’ uniquely
   • <tab><tab> twice to show all possible completions

2. Up-arrow (history)
   • Show previous commands

Try:
1. cd /etc
2. ls
3. cd c<tab><tab>
4. cd cal<tab>
UNIX directory structure

Create a new ‘unix’ folder in activities

(~/wpsg_2016/activities):

```
ubuntu@ip-10-179-185-48:~/wpsg_2016/activities$ mkdir unix
ubuntu@ip-10-179-185-48:~/wpsg_2016/activities$ ls -lар
total 3708
drwxrwxr-x 2 ubuntu ubuntu 4096 Jan 21 17:01 vcftools_plink
-rw-r--r-- 1 ubuntu ubuntu 3750880 Jan 23 10:48 unixExampleData.tar.gz
drwxrwxr-x 2 ubuntu ubuntu 4096 Jan 21 10:55 unix
drwxrwxr-x 2 ubuntu ubuntu 4096 Jan 21 15:31 STRUCTURE
drwxrwxr-x 2 ubuntu ubuntu 4096 Jan 21 11:54 RADseq_R
drwxrwxr-x 2 ubuntu ubuntu 4096 Jan 21 17:52 fineSTRUCTURE
drwxrwxr-x 6 ubuntu ubuntu 4096 Jan 21 12:02 bayescenv
-drwxrwxr-x 4 ubuntu ubuntu 4096 Jan 21 11:55 bayescan
drwxrwxr-x 5 ubuntu ubuntu 4096 Jan 19 23:24 ...
drwxrwxr-x 9 ubuntu ubuntu 4096 Jan 23 10:55 .
```
Extracting example data into the new **unix** folder:

The example data are in the following compressed **tar file**:
~/wpsg_2016/activities/unixExampleData.tar.gz

What is a tar archive?

When you download programs for UNIX, they often come in tar files.

Genomics data is normally stored compressed to save on disk space/costs.

```
tar = tape archive
```
Extracting example data into the new **unix** folder:

The example data are in the following compressed **tar file**:
~:/wpsg_2016/activities/unixExampleData.tar.gz

<table>
<thead>
<tr>
<th>gzip/gunzip</th>
<th>tar -xvzf</th>
<th>tar -xvf</th>
<th>bgzip</th>
</tr>
</thead>
<tbody>
<tr>
<td>compress/decompress a file</td>
<td>extract a gzipped tar archive</td>
<td>extract an uncompressed tar archive</td>
<td>another compression algorithm you will come across in genomics</td>
</tr>
<tr>
<td>like unixExampleData.tar.gz</td>
<td>like unixExampleData.tar</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Extracting example data into the new `unix` folder:

The example data are in the following compressed `tar` file:
```bash
~/wpsg_2016/activities/unixExampleData.tar.gz
```

```bash
ubantu@ip-10-179-185-48:~/wpsg_2016/activities$ cd unix/
ubantu@ip-10-179-185-48:~/wpsg_2016/activities/unix$ tar -xvzf ../unixExampleData.tar.gz
exampleVariants.vcf.gz
examplesGenomeSeqence.fastq.gz
ubantu@ip-10-179-185-48:~/wpsg_2016/activities/unix$ ls -lah
  total 3.6M
drwxrwxr-x 2 ubuntu ubuntu  4.0K Jan 23 11:29  .
drwxrwxr-x 9 ubuntu ubuntu  4.0K Jan 23 11:01  ..
-rw-r----- 1 ubuntu ubuntu  3.6M Jan 22 19:35 examplesGenomeSeqence.fastq.gz
-rw-r----- 1 ubuntu ubuntu  7.1K Jan 22 20:00 exampleVariants.vcf.gz
```

The `unixExampleData` archive contains two compressed files:
1) `examplesGenomeSeqence.fastq.gz`
   - output from a genome sequencer - Illumina type
2) `exampleVariants.vcf.gz`
   - a file with variants (differences) from the reference genome
Working with the sequence file:

The FASTQ format:

@HS22_09582:8:1105:9492:70247#84/1
GAGAATCTCATCCACATCACTACAGGCTATATTGGCCCGCAGCCAGGCAGGCAGGGGTAAAATCTCTTGCATGCCTGATCCACCCCTGGCATGCTACTGAT+
B?CFDFGEFFEEGFEGGGFFIIEGIEGIEECCDDFGJFGFGEFEHFEFEGHEEFGFG?GFFFGGGGDFGEFFHFBGGFFEGFDFFHFBGGGBFFDFGGEEFCE

1. Decompress the file
2. Count the number of reads
   - `wc -l examples GenomeSequence.fastq`
   - `grep "@HS" examples GenomeSequence.fastq`
   - `grep -c "@HS" examples GenomeSequence.fastq`
   - `grep -v "@HS" examples GenomeSequence.fastq`
   - `grep -v -c "@HS" examples GenomeSequence.fastq`
3. Print and count reads with undetermined bases in them:
   - `grep "N" examples GenomeSequence.fastq`
   - `grep -c "N" examples GenomeSequence.fastq`
Working with the VCF variants file:

The VCF file (without a header) is a text file with `<tab>` separated columns:

• columns 1-8: information about a variant (location, alleles, quality scores, filtering, etc.)
• columns 9 onwards: information about the genotypes (variants) present in each individual
Working with the VCF variants file:

The VCF file (without a header) is a text file with `<tab>` separated columns:
• columns 1-8: information about a variant (location, alleles, quality scores, filtering, etc.)
• columns 9 onwards: information about the genotypes (variants) present in each individual

There is a lot of information, especially in the INFO and FORMAT columns: you are going to find what some of these things mean tomorrow and during the rest of the course. But if you can’t wait, the formal specification is here: [http://www.1000genomes.org/wiki/analysis/variant%20call%20format/vcf-variant-call-format-version-41](http://www.1000genomes.org/wiki/analysis/variant%20call%20format/vcf-variant-call-format-version-41)
Working with the VCF variants file:

The VCF file (without a header) is a text file with <tab> separated columns:
• columns 1-8: information about a variant (location, alleles, quality scores, filtering, etc.)
• columns 9 onwards: information about the genotypes (variants) present in each individual

We will use one of UNIX’s cool features: **Pipes**

So far we have been doing this:

```
STDIN → Program 1 → STDOUT
```

Now we are going to learn to link multiple UNIX programs:

```
STDIN → Program 1 → STDOUT
```

```
Program 1 → PIPE → Program 2
```

```
Program 2 → PIPE → Program 3
```

```
STDIN → Program 2 → STDOUT
```
Working with the VCF variants file:

The VCF file (without a header) is a text file with `<tab>` separated columns:

- columns 1-8: information about a variant (location, alleles, quality scores, filtering, etc.)
- columns 9 onwards: information about the genotypes (variants) present in each individual

1. Decompress the file
2. Have a look at the file
   - `less exampleVariants.vcf`
   - `less -S exampleVariants.vcf`
3. Find out how many chromosomes there are:
   - `cut -f 1 exampleVariants.vcf`
   - `cut -f 1 exampleVariants.vcf | uniq`
   - Make sure you know what `cut -f` does! Try `cut -f 2`
   - `man uniq` (see what `uniq` does)
4. Find how many variants have a “T” as the reference allele (fourth column):
   - `cut -f 4 exampleVariants.vcf`
   - `cut -f 4 exampleVariants.vcf | grep "T"`
   - `cut -f 4 exampleVariants.vcf | grep -o "T" | wc -l`
   - `man grep` (see what the `-o` option does)
Working with the VCF variants file:

5. Capture the genotype information for the first three individuals into a different file:
   • `cut -f 10-12 exampleVariants.vcf > threeGenotypes.txt`

6. Remove the newly created file:
   • `rm threeGenotypes.txt`

!!!There is **NO** undo button, no trash can!!!!

Once deleted a file is lost forever (or at least you’d need professional data recovery)
   • `rm -i exampleVariants.vcf` (then press `n` and enter)

```
ubuntu@ip-10-179-185-48:/wpsg_2016/activities/unix$ rm -i exampleVariants.vcf
rm: remove regular file ‘exampleVariants.vcf’? n
```
Using a second set of example data:

The example data are in the following compressed tar file:
~/wpsg_2016/activities/unixExampleData2.tar.gz
Using a second set of example data:

The example data are in the following compressed tar file:
~/wpsg_2016/activities/unixExampleData2.tar.gz

The `unixExampleData2` archive contains 11 uncompressed fasta sequence files:

```
ubuntu@ip-10-179-185-48:~/wpsg_2016/activities/unix$ tar -xvzf ../unixExampleData2.tar.gz
scaffold_600.fa
scaffold_601.fa
scaffold_602.fa
scaffold_603.fa
scaffold_604.fa
scaffold_605.fa
scaffold_606.fa
scaffold_607.fa
scaffold_608.fa
scaffold_609.fa
scaffold_610.fa
```
Using a second set of example data:

The example data are in the following compressed tar file:
~/wpsg_2016/activities/unixExampleData2.tar.gz

Each file contains a header line (starts with >) and then a lot of sequence:

```
ubunto@ip-10-179-185-48:~/wpsg_2016/activities/unix$ less scaffold_600.fa

>scaffold_600
TATATATATATATATATATATATATATATATATCTGACAATACCTAATAATATTGGCCATATTATATTTATACATTACCACGAGTGA
GATGACTTTTAGTCTCATGAACAACATACAAACAAATTATCTAGTTTATTTGCTAATCTAAATATGACTTGCAGCAC
AGAAATGAAACCCCAACATATCAGTTCTACGTCAGGTTCGTTACGCCAGATACCTCAACTATACAGCTAGTGTTCATGA
CAAAATGAATGACCAACAAACATTTTTCTCTTTATTTTCTGGCAAAACAAAGCTGTTATGTAAACGTTCCTGGTTGATAGA
TCATGTTGCTAGGCAACGGAGGCTAGACCATCCATTTCACAAGCGCTCGCACCTCCCGGCCTAGCGTTTTAAGTACG
CGGGCCGTGGAGGACTGCTACGCGGCTGCAAGACCCGTGAATTGGGATAAGACTAGAAGCTTTTGTGGCGTTTTTGATGA
CGTCTTTTCGACTCGAGCTCTGGCAGTCTCGGCAACACCGAGAAAAACAAGATGAAGTGGCCTCTCAAGACTCGCTAAACACTAC
AACCAGTTGAAAACTGCTAACTTGATTGGCTGGCAACAAAAATTTTGTATTTTATTTTGTGCTTCTATTTAATTTTATTTTT
TGATCTACCTTTTTGTCTACGCTCTTGCTCTCCTCTCCTCTCTCGTGTAATCTAAATGTTTAAATGTTTTAAATCTGAC
CCTGTTTTACTGAGCTGTATGCTAAATATGAATAAAATGTGATATAAAAATATCTGGCCATAACACGAGCAAAATT
```

How would you:
1) find the length of the DNA sequence in each file?
2) edit the header line of each file to read “chromosome_” instead of “scaffold_”?
3) What if you had 3000 such files?
Find the length of the DNA sequence in each file:

One solution is to use awk:

```bash
man awk
```

**NAME**
gawk - pattern scanning and processing language

**SYNOPSIS**
gawk [ POSIX or GNU style options ] -f program-file [ -- ] file ...
gawk [ POSIX or GNU style options ] [ -- ] program-text file ...

**DESCRIPTION**

Gawk is the GNU Project's implementation of the AWK programming language. It conforms to the definition of the language in the POSIX 1003.1 Standard. This version in turn is based on the description in *The AWK Programming Language*, by Aho, Kernighan, and Weinberger. Gawk provides the additional features found in the current version of Brian Kernighan's awk and a number of GNU-specific extensions.

The command line consists of options to gawk itself, the AWK program text (if not supplied via the -f or --file options), and values to be made available in the ARGV and ARGV pre-defined AWK variables.
**awk**, a stream programming language

**pattern** {**action**}

1. **Awk** is column (**field**) aware:
   - $0$ - the whole line
   - $1$ - column one
   - $2$ - column two
   ...

2. **pattern** can be any logical statement:
   - $3 > 0$ - if column 3 is greater than 0
   - $1 == 32$ - if column 1 equals 32
   - $1 == 3$ - if column 1 equals column 3
   - $1 == "consensus"$ - if column 1 contains the string, "consensus"

If pattern is true, everything in {...} is executed
awk, a stream programming language

```
pattern {action}
```

- Apply action to every line
  - Execute action once at start:
    - BEGIN {action} pattern {action}
  - Execute action once at end:
    - pattern {action} END {action}

```
BEGIN {action} pattern {action} END {action}
```
awk, a stream programming language

pattern {action1; action2; action3}

1. Built in variables

   NR - number of records seen so far (aka line number)
   NF - number of fields in the current record
   FILENAME - name of the current file being read

2. Built in functions

   length(x) - length of the field
   print(x) - print a field
   rand() - generate a random number
   sqrt(x) - calculate square root of x
   sub(x, y) - substitute s for r in $0

3. User defined variables

   increment: n = n + 1
   multiply: n += $2 * $3
Find the length of the DNA sequence in each file:

Find the length of the DNA sequence in file `scaffold_600.fa`:

```
awk 'NR > 1 { total=total+length($0)} END{print(total)}' scaffold_600.fa
```

- do not count the first line
- always add the line length to `total`

- What is the answer?
- How would you do this for 3000 files?
Find the length of the DNA sequence in each file:

Find the length of the DNA sequence in file `scaffold_600.fa`:

```
awk 'NR > 1 { total=total+length($0)} END{print(total)}' scaffold_600.fa
```

do not count the first line
always add the line length to `total`

• What is the answer?
• How would you do this for 3000 files?

**Answer:** a shell loop
Find the length of the DNA sequence in each file:

Find the length of the DNA sequence in files `scaffold_600.fa` through to `scaffold_610.fa`:

- for i in {600..610}; do echo scaffold_${i}.fa; done

a shell variable using the variable
Find the length of the DNA sequence in each file:

Find the length of the DNA sequence in files *scaffold_600.fa* through to *scaffold_610.fa*:

- `for i in {600..610}; do echo scaffold_${i}.fa; done`

  a shell variable using the variable

The solution is:

```
for i in {600..610}; do
echo scaffold_${i}.fa
awk 'NR > 1 { total=total+length($0)} END{print(total)}' scaffold_${i}.fa
done
```

Or, say if you want to omit `scaffold_605`, one alternative is:

```
for i in 600 601 602 603 604 606 607 608 609 610}; do
echo scaffold_${i}.fa
awk 'NR > 1 { total=total+length($0)} END{print(total)}' scaffold_${i}.fa
done
```

But how do you write it all on one line?
Shell scripting

- Anything you can do on the shell can be placed in a shell script
- Shell scripts often end in the suffix `".sh"`
- Comments can be written in scripts with a `"#"`
- `#!/bin/bash` must be the first line - specifies interpreter
Shell scripting

Emacs
Richard Stallman - 1976
Founded GNU Project

Vi
Bill Joy - 1976
BSD/Sun Microsystems

```
ubuntu@ip-10-144-40-7:/wpsg_2016/activities/unix$ man vi
VIM(1)                                     General Commands Manual                                     VIM(1)
NAME
   vim - Vi IMproved, a programmers text editor
```
Shell scripting

• vi <filename>
• vi /absolute/path/to/file
• vi ../../../relative/path/to/file

Command mode versus Text-entry mode

Your mouse cannot help you!

vi commands start with a colon “ : ”
Shell scripting

Do the following:
1. `vi printLengths.sh`
2. Enter the Text-entry mode by pressing the “i” key
3. Write the script (the comments are optional)
4. exit the text-entry mode by pressing <esc>
5. Use the command `:w` to save (write) the file
6. Use the command `:q` to exit `vi`

Get a cheat-sheet for `vi`, e.g.:
A detailed guide to bash scripting:
http://tldp.org/LDP/abs/html/
# Shell scripting

```
ubuntu@ip-10-144-40-7:~/wpsg_2016/activities/unix$ ls -lah

total 500K
-drwxrwxr-x 2 ubuntu ubuntu 4.0K Jan 24 00:29 ..
drwxrwxr-x 12 ubuntu ubuntu 4.0K Jan 23 22:59 ..
-rw-r--r-- 1 ubuntu ubuntu 238 Jan 24 00:16 printLengths.sh
-rw-r--r-- 1 ubuntu ubuntu 43K Jan 23 19:38 scaffold_600.fa
-rw-r--r-- 1 ubuntu ubuntu 43K Jan 23 19:39 scaffold_601.fa
-rw-r--r-- 1 ubuntu ubuntu 43K Jan 23 19:39 scaffold_602.fa
```

<table>
<thead>
<tr>
<th>file permissions</th>
<th>Owner</th>
<th>Group</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw-</td>
<td>rw-</td>
<td>r--</td>
<td></td>
</tr>
</tbody>
</table>

file permissions
Shell scripting

Shell scripts must be executable: `chmod +x printLengths.sh`

```bash
ubuntu@ip-10-144-40-7:~/wpsg_2016/activities/unix$ ls -lah
```
```
total 500K
drwxrwxr-x  2 ubuntu ubuntu  4.0K Jan 24 00:29 .
drwxrwxr-x 12 ubuntu ubuntu  4.0K Jan 23 22:59 ..
-rw-rw-r--  1 ubuntu ubuntu 238 Jan 24 00:16 printLengths.sh
-rw-r--r--  1 ubuntu ubuntu  43K Jan 23 19:38 scaffold_600.fa
-rw-r--r--  1 ubuntu ubuntu  43K Jan 23 19:39 scaffold_601.fa
-rw-r--r--  1 ubuntu ubuntu  43K Jan 23 19:39 scaffold_602.fa
```
Script/program execution and $PATH

Now you should be able to execute your script:

```
./printLengths.sh
```

executing in this folder (remember, a dot “.” means ‘this folder’)

`printLengths.sh`

will not work

`~/wpsg_2016/activities/unix` is not in your $PATH
Script/program execution and $PATH

Now you should be able to execute your script:
```bash
./printLengths.sh
```

executing in this folder (remember, a dot “.” means ‘this folder’)

printLengths.sh will not work

`~/wpsg_2016/activities/unix` is not in your PATH

```
ubuntu@ip-10-144-40-7:/wpsg_2016/activities/unix$ echo $PATH
/home/ubuntu/wpsg_2016/software/beast/bin:/usr/local/sbin:/usr/local/bin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin
```

```
ubuntu@ip-10-144-40-7:/wpsg_2016/activities/unix$ PATH=$PATH:~/wpsg_2016/activities/unix
```

```
ubuntu@ip-10-144-40-7:/wpsg_2016/activities/unix$ echo $PATH
/home/ubuntu/wpsg_2016/software/beast/bin:/usr/local/sbin:/usr/local/bin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/ubuntu/wpsg_2016/activities/unix
```

Type:
```
PATH=$PATH:~/wpsg_2016/activities/unix
```
and try again:
```bash
printLengths.sh
```
Now you should be able to execute your script:

```
./printLengths.sh
```

executing in this folder (remember, a dot “.” means ‘this folder’)

`printLengths.sh` will not work

`~/wpsg_2016/activities/unix` is not in your PATH

Type the above and try again:

`printLengths.sh`

All the software you use during this workshop has already been put in your PATH so you don’t have to search for the folders where it is stored.