Programming for Bioinformatics | BIOL7200

Week 4 Exercise

September 14, 2021

Exercises

Regardless of your specific areas of research, file manipulations techniques will always be useful. If there is only one thing that you can learn from this class, let it be this. We'll be happy to go over any concept that you are still struggling with in Thursday's discussion session.

xargs

1. Passing arguments with `xargs`
    1. List the files in the directory, and pass them to the head command using `xargs` and the `-I` flag

    ANS: ls | xargs -I head

    2. Use `xargs` and `touch` to make 10 files by the name of: file1.fna, file2.fna, …, file10.fna

    ANS: seq 1 10 | xargs -I X touch fileX.fna

    3. Use `sed` and `xargs` to change the extension of the all the `.fna` files to `.fasta`

    ANS: ls *.fna |sed 's/.fna//' |xargs -I ASH mv ASH.fna ASH.fasta

    4. Get the first two columns of the UCSC gene file using `cut`

    ANS: cut -f1,2 knownGene.txt
2. Multiple arguments with `xargs`
    1. Print the numbers 1 to 12

    ANS: echo 1 2 3 4 5 6 7 8 9 10 11 12 | xargs

    2. Pass these to `xargs` in multiple of 3 and print them as:

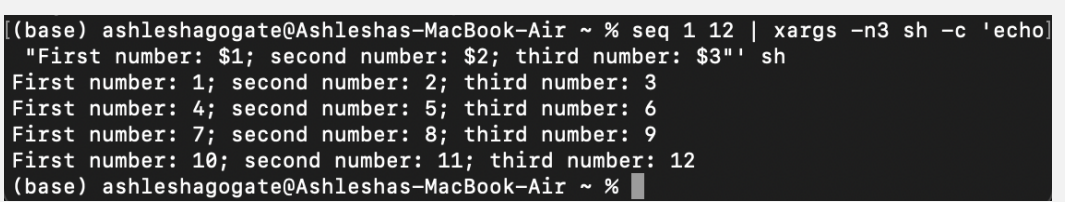
    ANS: seq 1 12 | xargs -n3 sh -c 'echo "First number: $1; second number: $2; third number: $3"' sh

    First number: 1; second number 2; third number 3;

    First number: 4; second number 5; third number 6;

    First number: 7; second number 8; third number 9;

    First number: 10; second number 11; third number 12;



General programming

3. What is the value of `a_number` after each line? (Write the value of `a_number` after each statement; these statements are in continuation)
    1. `a_number = 6`

    ANS: 6
    2. `a_number -= 12`

    ANS: -6
    3. `a_number += 15`

    ANS: 9
    4. `a_number /= 3`

    ANS: 3.0
    5. `a_number *= 2`

    ANS: 6.0
    6. `a_number = a_number + a_number / 2`

    ANS: 9.0

4. Evaluate these Boolean statements. (Answer is TRUE or FALSE)
    1. `1 > 2`

    ANS: FALSE
    2. `2 > 1`

    ANS: TRUE
    3. `!(2 > 1)`

    ANS: FALSE
    4. `(2 > 1) || (1 > 2)`

    ANS: TRUE
    5. `(2 > 1) && (1 > 2)`

    ANS: FALSE
    6. `!(2 > 1) || !(1 > 2)`

    ANS: TRUE
    7. `!((2 > 1) || (1 > 2))`

    ANS: FALSE
    8. `!((2 > 1) && (1 > 2))`

    ANS: TRUE

Biologically-inspired problem

5. Format recognition

A large part of data analysis deals with cleaning and processing data. There are a lot of data formats – some of which might just be different formal standards for storing the same data. Given the large number of formats that exist, it is almost impossible to remember each of them – feel free to look up their specifications when you need to use them.

We will deal with four standard (and common) bioinformatics file formats this week: EMBL, FASTQ, GenBank, and MEGA. Here are their specifications:

Format and link to description

- EMBL
  https://www.genomatix.de/online_help/help/sequence_formats.html#EMBL
- FASTQ
  https://www.genomatix.de/online_help/help/sequence_formats.html#FASTQ
- GenBank
  https://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html
- MEGA
  https://www.megasoftware.net/mega1_manual/DataFormat.html

Let's write simple **awk** one liners to check the format for input files. Only the first line of the input file will be piped into your **awk** one-liner.

Notes:

- You can send the first line using the **head** command. Formats can often be recognized by using only the first column.
- The solution can be very clean and simple to something extremely elaborate. It all depends on how you think.
- There are files provided on Canvas for you to check your one-liner.

ANS: head -1 file.fastq | awk '{if ($1 == "ID") {print "This is an EMBL file"} else if ($1 == "LOCUS") {print "This is a Genbank file"} else if ($1~/#/) {print "This is a MEGA file"} else if ($1~/@/) {print "This is a fastq file"} else {print "Type of file not known"}}'

```
(base) ashleshagogate@Ashleshas-MacBook-Air Downloads % head -1 file.embl | awk '{if ($1 == "ID") {print "This is an EMBL file"}
} else if ($1 == "LOCUS") {print "This is a Genbank file"} else if ($1~/#/) {print "This is a MEGA file"} else if ($1~/@/) {pri
nt "This is a fastq file"} else {print "Type of file not known"}}'
This is an EMBL file
(base) ashleshagogate@Ashleshas-MacBook-Air Downloads % head -1 file.gb | awk '{if ($1 == "ID") {print "This is an EMBL file"} }
else if ($1 == "LOCUS") {print "This is a Genbank file"} else if ($1~/#/) {print "This is a MEGA file"} else if ($1~/@/) {print
 "This is a fastq file"} else {print "Type of file not known"}}'
This is a Genbank file
(base) ashleshagogate@Ashleshas-MacBook-Air Downloads % head -1 file.fastq | awk '{if ($1 == "ID") {print "This is an EMBL file
"} else if ($1 == "LOCUS") {print "This is a Genbank file"} else if ($1~/#/) {print "This is a MEGA file"} else if ($1~/@/) {pr
int "This is a fastq file"} else {print "Type of file not known"}}'
This is a fastq file
(base) ashleshagogate@Ashleshas-MacBook-Air Downloads % head -1 file.mega | awk '{if ($1 == "ID") {print "This is an EMBL file"}
} else if ($1 == "LOCUS") {print "This is a Genbank file"} else if ($1~/#/) {print "This is a MEGA file"} else if ($1~/@/) {pri
nt "This is a fastq file"} else {print "Type of file not known"}}'
This is a MEGA file
(base) ashleshagogate@Ashleshas-MacBook-Air Downloads % head -1 knownGene.txt | awk '{if ($1 == "ID") {print "This is an EMBL f
ile"} else if ($1 == "LOCUS") {print "This is a Genbank file"} else if ($1~/#/) {print "This is a MEGA file"} else if ($1~/@/)
{print "This is a fastq file"} else {print "Type of file not known"}}'
Type of file not known
(base) ashleshagogate@Ashleshas-MacBook-Air Downloads %
```

Final installation problems

This will be the final installation problem for this course. These two problems will test your understanding of the subject matter. If you succeed here, you will be able to tackle all sorts of installations in the field of bioinformatics. There are three problems here that will be solved using the following mechanisms:

- sudo apt-get
- Simple make
- Version control (git)

For the first problem, we will be using version control. We will talk more about this later in the course. To get you started, version control software help you keep track of changes in your code. This is especially important when you're working on a large project or when you are working on a shared codebase. cvs and git are two of many utilities which do just that.

6. Install git using apt-get
   Keep this one simple and figure out what command(s) you need to run to install apt-get on sudo privileges.

ANS: sudo apt-get install git

```
ashlesha@ashlesha:~$ sudo apt-get install git
[sudo] password for ashlesha:
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.25.1-1ubuntu3.2).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 10 not upgraded.
```

7. Straight-forward install
   Next up is SAMtools. SAMtools is a program for manipulating mapping files from the mapping of NGS short read sequence data to a genome. It is very fast and great for file format manipulation. Installation of SAMtools is also one of the easiest installations in bioinformatics.
   1. Download the latest version of the SAMtools *github* source code (here is the homepage: http://www.htslib.org/, github repos are linked within this page). Don't download the pre-compiled binaries.
   2. Unpack the source and ensure you can find the Makefile. If it doesn't exist, how can you generate it?

   ANS: tar -xf samtools-1.13.tar.bz2

   The makefile exsists. If it doesn't, you can use cmake command to generate it.

```
ashlesha@ashlesha:~/Downloads/samtools-1.13$ ls
amplicon_stats.c     bam_markdup.c       bedidx.h        padding.c
AUTHORS              bam_mate.c          ChangeLog.old   phase.c
bam2bcf.c            bam_md.c            config.h.in     README
bam2bcf.h            bam_plbuf.c         config.mk.in    sam.c
bam2bcf_indel.c      bam_plbuf.h         configure       sam.h
bam2depth.c          bam_plcmd.c         configure.ac    sam_opts.c
bam_addrprg.c        bam_quickcheck.c    coverage.c      sam_opts.h
bam_ampliconclip.c   bam_reheader.c      cut_target.c    sample.c
bam_ampliconclip.h   bam_rmdup.c         dict.c          sample.h
bam_aux.c            bam_rmdupse.c       doc             samtools.h
bam.c                bamshuf.c           examples        sam_utils.c
bam_cat.c            bam_sort.c          faidx.c         sam_view.c
bam_color.c          bam_split.c         htslib-1.13     stats.c
bam_endian.h         bam_stat.c          INSTALL         stats_isize.c
bam_fastq.c          bamtk.c             install-sh      stats_isize.h
bam_flags.c          bam_tview.c         LICENSE         test
bam.h                bam_tview_curses.c  lz4             tmp_file.c
bam_import.c         bam_tview.h         m4              tmp_file.h
bam_index.c          bam_tview_html.c    Makefile        version.sh
bam_lpileup.c        bedcov.c            misc
bam_lpileup.h        bedidx.c            NEWS
```

3. Build samtools and its associated programs with make.

ANS: ./configure –prefix=/home/ashlesha/Downloads/samtools

sudo apt-get install gcc

sudo apt-get install libncurses5-dev libncursesw5-dev zlib1g-dev libbz2-dev liblzma-dev

```
config.status: creating config.mk
config.status: creating htslib.pc.tmp
config.status: creating config.h
config.status: linking htscodecs_bundled.mk to htscodecs.mk
ashlesha@ashlesha:~/Downloads/samtools-1.13$
```

make

```
gcc -Wall -g -O2 -I. -Ihtslib-1.13 -I./lz4  -c -o test/split/test_filter_header_rg.o test/split/test_fil
ter_header_rg.c
gcc  -L./lz4  -o test/split/test_filter_header_rg test/split/test_filter_header_rg.o test/test.o libst.a
 htslib-1.13/libhts.a -lpthread -lz -lm -lbz2 -llzma -lz  -lpthread
gcc -Wall -g -O2 -I. -Ihtslib-1.13 -I./lz4  -c -o test/split/test_parse_args.o test/split/test_parse_arg
s.c
gcc  -L./lz4  -o test/split/test_parse_args test/split/test_parse_args.o test/test.o libst.a htslib-1.13
/libhts.a -lpthread -lz -lm -lbz2 -llzma -lz  -lpthread
gcc -Wall -g -O2 -I. -Ihtslib-1.13 -I./lz4  -c -o test/vcf-miniview.o test/vcf-miniview.c
gcc  -L./lz4  -o test/vcf-miniview test/vcf-miniview.o htslib-1.13/libhts.a -lpthread -lz -lm -lbz2 -llz
ma -lz  -lpthread
ashlesha@ashlesha:~/Downloads/samtools-1.13$ ls
```

make install

```
ashlesha@ashlesha:~/Downloads/samtools-1.13$ make install
mkdir -p -m 755 /home/ashlesha/Downloads/samtools/bin /home/ashlesha/Downloads/samtools/bin /home/ashles
ha/Downloads/samtools/share/man/man1
install -p samtools /home/ashlesha/Downloads/samtools/bin
install -p misc/ace2sam misc/maq2sam-long misc/maq2sam-short misc/md5fa misc/md5sum-lite misc/wgsim /hom
e/ashlesha/Downloads/samtools/bin
install -p misc/blast2sam.pl misc/bowtie2sam.pl misc/export2sam.pl misc/fasta-sanitize.pl misc/interpola
te_sam.pl misc/novo2sam.pl misc/plot-ampliconstats misc/plot-bamstats misc/psl2sam.pl misc/sam2vcf.pl mi
sc/samtools.pl misc/seq_cache_populate.pl misc/soap2sam.pl misc/wgsim_eval.pl misc/zoom2sam.pl /home/ash
lesha/Downloads/samtools/bin
install -p -m 644 doc/samtools*.1 misc/wgsim.1 /home/ashlesha/Downloads/samtools/share/man/man1
```

```
ashlesha@ashlesha:~/Downloads/samtools$ ls
bin  share
ashlesha@ashlesha:~/Downloads/samtools$ cd bin
ashlesha@ashlesha:~/Downloads/samtools/bin$ ls
ace2sam             interpolate_sam.pl  novo2sam.pl         samtools              wgsim_eval.pl
blast2sam.pl        maq2sam-long        plot-ampliconstats  samtools.pl           zoom2sam.pl
bowtie2sam.pl       maq2sam-short       plot-bamstats       seq_cache_populate.pl
export2sam.pl       md5fa               psl2sam.pl          soap2sam.pl
fasta-sanitize.pl   md5sum-lite         sam2vcf.pl          wgsim
ashlesha@ashlesha:~/Downloads/samtools/bin$
```

4. Place the created binaries somewhere on your **PATH**, e.g. in your **bin** directory.

ANS: export PATH=/home/ashlesha/Downloads/samtools/bin:$PATH

5. Run them to see what happens. You should get some help information or errors as you didn't give them any input.

ANS: ./bowtie2sam.pl

./export2sam.pl

```
ashlesha@ashlesha:~/Downloads/samtools/bin$ ./bowtie2sam.pl
Usage: bowtie2sam.pl <aln.bowtie>
ashlesha@ashlesha:~/Downloads/samtools/bin$ ./export2sam.pl

export2sam.pl converts GERALD export files to SAM format.

Usage: export2sam.pl --read1=FILENAME [ options ] | --version | --help

  --read1=FILENAME  read1 export file or '-' for stdin (mandatory)
                      (file may be gzipped with ".gz" extension)
  --read2=FILENAME  read2 export file or '-' for stdin
                      (file may be gzipped with ".gz" extension)
  --nofilter        include reads that failed the basecaller
                      purity filter
  --qlogodds        assume export file(s) use logodds quality values
                      as reported by OLB (Pipeline) prior to v1.3
                      (default: phred quality values)
```

8. Harder install: the Kent source tree

The Kent source tree is a huge bundle of bioinformatics utilities named for James Kent, the brain behind the UCSC Genome Browser. The Source Tree has come in handy many times for many people. Although you can get binaries nowadays, you wouldn't learn how to install things which have similar dependencies as not everything is available precompiled. So, do not download the precompiled binaries for this exercise.

1. Obtain the source code for the Kent source tree using git as described at http://genome.ucsc.edu/admin/git.html

```
ashlesha@ashlesha:~$ git clone git://genome-source.soe.ucsc.edu/kent.git
Cloning into 'kent'...
remote: Counting objects: 1008090, done.
remote: Compressing objects: 100% (208478/208478), done.
remote: Total 1008090 (delta 802549), reused 998107 (delta 792775)
Receiving objects: 100% (1008090/1008090), 454.58 MiB | 12.34 MiB/s, done.
Resolving deltas: 100% (802549/802549), done.
Updating files: 100% (23960/23960), done.
ashlesha@ashlesha:~$ ls
Desktop  Documents  Downloads  kent  Music  Pictures  Public  Templates  Videos
ashlesha@ashlesha:~$ cd kent
ashlesha@ashlesha:~/kent$ git checkout -t -b beta origin/beta
Branch 'beta' set up to track remote branch 'beta' from 'origin'.
Switched to a new branch 'beta'
ashlesha@ashlesha:~/kent$ git pull
Already up to date.
ashlesha@ashlesha:~/kent$
```

2. The README file, or some variation thereof, is often very useful for installation. Read it to figure out what you need to do to install the source tree. If you aren't setting up your architecture type, you aren't doing it right.

ANS: mkdir ~/bin/$MACHTYPE

export PATH="$HOME/bin/$MACHTYPE:$PATH"

export PATH="$HOME/bin/aarch64:$PATH"

export MACHTYPE= aarch 64

cd kent/src/lib

make

```
ashlesha@ashlesha:~/kent/src/lib$ make
/bin/sh: 1: libpng-config: not found
/bin/sh: 1: libpng-config: not found
/bin/sh: 1: mysql_config: not found
/bin/sh: 1: mysql_config: not found
cc -O -g -std=c99 -Wall -Wformat -Wimplicit -Wreturn-type -Wuninitialized -D_FIL
E_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -D_GNU_SOURCE -DMACHTYPE_aarch64 -DUSE_HIC
  -Wall -Wformat -Wimplicit -Wreturn-type -Wuninitialized -I../inc -I../../inc -
I../../../inc -I../../../../inc -I../../../../../inc -I../htslib  -I/include   -
o aliType.o -c aliType.c
cc -O -g -std=c99 -Wall -Wformat -Wimplicit -Wreturn-type -Wuninitialized -D_FIL
E_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -D_GNU_SOURCE -DMACHTYPE_aarch64 -DUSE_HIC
  -Wall -Wformat -Wimplicit -Wreturn-type -Wuninitialized -I../inc -I../../inc -
I../../../inc -I../../../../inc -I../../../../../inc -I../htslib  -I/include   -
o annoAssembly.o -c annoAssembly.c
```

sudo apt-get install libpng-dev libmysqlclient-dev uuid-dev

make

cd ..

cd jkOwnLib

make

```
ashlesha@ashlesha:~/kent/src$ cd jkOwnLib/
ashlesha@ashlesha:~/kent/src/jkOwnLib$ make
cc -O -g -std=c99 -Wall -Wformat -Wimplicit -Wreturn-type -Wuninitialized -D_FILE_OFFSE
T_BITS=64 -D_LARGEFILE_SOURCE -D_GNU_SOURCE -DMACHTYPE_aarch64 -DUSE_HIC   -Wall -Wform
at -Wimplicit -Wreturn-type -Wuninitialized -I../inc -I../../inc -I../../../inc -I../..
/../../inc -I../../../../../inc -I../htslib  -I/include -I/usr/include/libpng16  -o ban
dExt.o -c bandExt.c
cc -O -g -std=c99 -Wall -Wformat -Wimplicit -Wreturn-type -Wuninitialized -D_FILE_OFFSE
T_BITS=64 -D_LARGEFILE_SOURCE -D_GNU_SOURCE -DMACHTYPE_aarch64 -DUSE_HIC   -Wall -Wform
at -Wimplicit -Wreturn-type -Wuninitialized -I../inc -I../../inc -I../../../inc -I../..
/../../inc -I../../../../../inc -I../htslib  -I/include -I/usr/include/libpng16  -o cru
dealI.o -c crudeali.c
```

cd ..

cd htslib

make

```
ashlesha@ashlesha:~/kent/src$ cd htslib/
ashlesha@ashlesha:~/kent/src/htslib$ make
cc -O -g -std=c99 -Wall -Wformat -Wimplicit -Wreturn-type -Wuninitialized -DKNETFILE_HO
OKS -DUCSC_CRAM -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -D_GNU_SOURCE -DMACHTYPE_aar
ch64 -DUSE_HIC   -Wall -Wformat -Wimplicit -Wreturn-type -Wuninitialized -I../inc -I../
../inc -I../../../inc -I../../../../inc -I../../../../../inc -I../htslib  -I/include -I
/usr/include/libpng16  -o kfunc.o -c kfunc.c
cc -O -g -std=c99 -Wall -Wformat -Wimplicit -Wreturn-type -Wuninitialized -DKNETFILE_HO
OKS -DUCSC_CRAM -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -D_GNU_SOURCE -DMACHTYPE_aar
ch64 -DUSE_HIC   -Wall -Wformat -Wimplicit -Wreturn-type -Wuninitialized -I../inc -I../
../inc -I../../../inc -I../../../../inc -I../../../../../inc -I../htslib  -I/include -I
/usr/include/libpng16  -o knetfile.o -c knetfile.c
```

3. Attempt to compile the source tree; it should work. If you get stuck, how can you resolve the issue? (Perhaps look into the environment variables). You don't need to install the whole thing. Individual programs can be built by going into their directories and using make to compile the programs as you need them.

ANS: cd kent/src/gfServer
make
kent/src/gfClient
make
kent/src/blat
make
kent/src/utils/faToNib
make
The blat executable is found in the ~/bin/$MACHTYPE folder. It can be run from anywhere since the folder is added to the path.