Practical Linux Examples

- Processing large text file
- Parallelization of independent tasks

Qi Sun & Robert Bukowski
Bioinformatics Facility
Cornell University

http://cbsu.tc.cornell.edu/lab/doc/linux_examples_slides.pdf
http://cbsu.tc.cornell.edu/lab/doc/linux_examples_exercises.pdf

Estimate the percentage of sequencing reads in the FASTQ file that contains the adapter "AGATCGGAAGAGC".

- **Read file:**

  cat D7RACXX.fastq

- **Select lines that contain the string "AGATCGGAAGAGC"**

  grep AGATCGGAAGAGC

- **Count the selected lines**

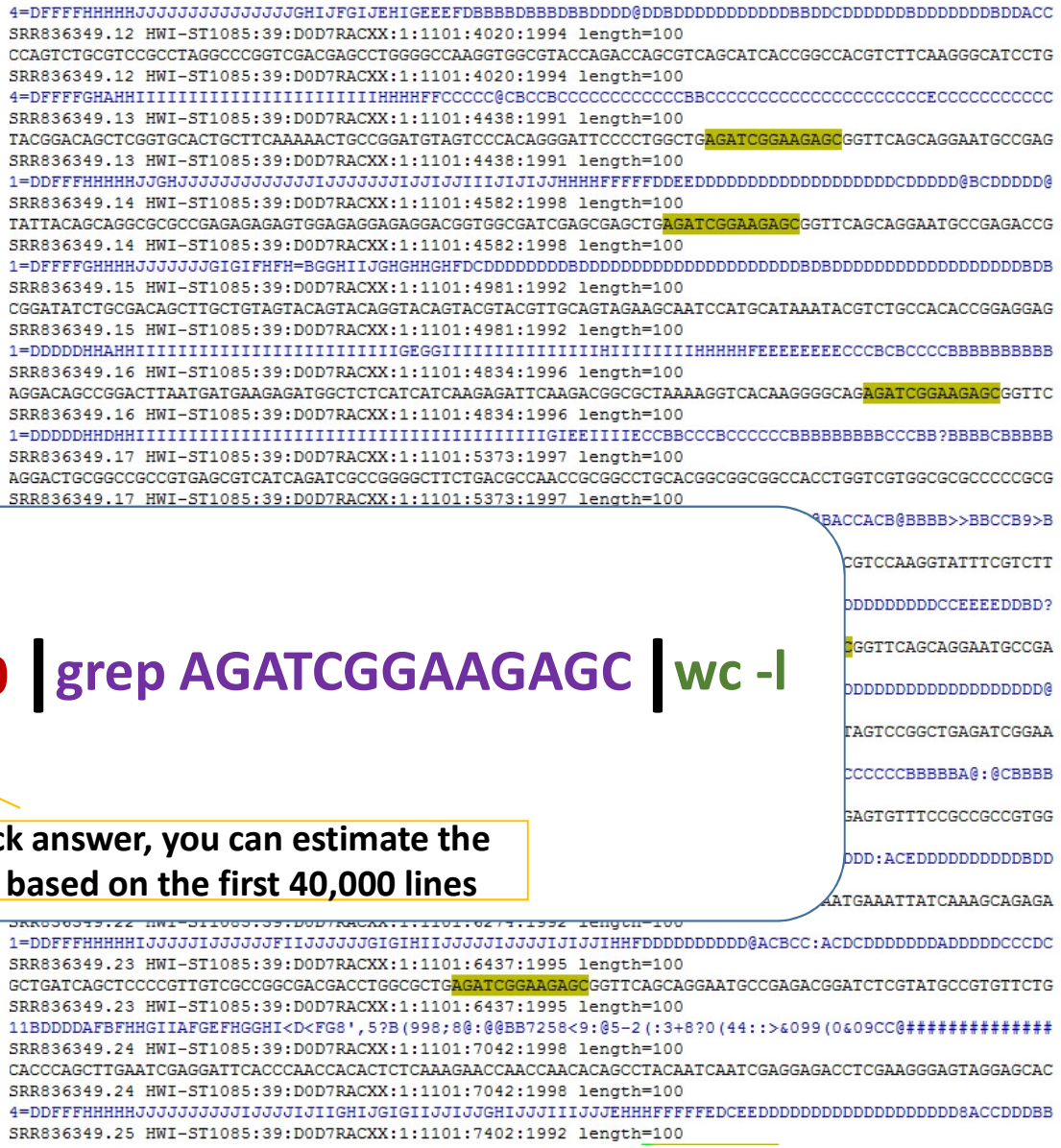
  wc -l

```
4=DFFFFHHHHHJJJJJJJJJJJJJJGHIJFGIJEHIGEEEFDBBBBDBBBDBBDDDD@DDBDDDDDDDDDDDBBDDCDDDDDDBDDDDDDDBDDACC
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
CCAGTCTGCGTCCGCCTAGGCCCGGTCGACGAGCCTGGGGCCAAGGTGGCGTACCAGACCAGCGTCAGCATCACCGGCCACGTCTTCAAGGGCATCCTG
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
4=DFFFFGHAHHIIIIIIIIIIIIIIIIIIIIIIIHHHHFFCCCCC@CBCCBCCCCCCCCCCCCBBCCCCCCCCCCCCCCCCCCCCCECCCCCCCCCCC
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
TACGGACAGCTCGGTGCACTGCTTCAAAAACTGCCGGATGTAGTCCCACAGGGATTCCCCTGGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAG
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
1=DDFFFHHHHHJJGHJJJJJJJJJJJJJJIJJJJJJJIJJIJJIIIIJIJIJJHHHHFFFFFDDEEDDDDDDDDDDDDDDDDDDDCDDDD@BCDDDDD@
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
TATTACAGCAGGCGCGCCGAGAGAGAGTGGAGAGGAGAGGACGGTGGCGATCGAGCGAGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCG
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
1=DFFFFGHHHHJJJJJJJGIGIFHFH=BGGHIIJGHGHHGHFDCDDDDDDDDBDDDDDDDDDDDDDDDDDDDDDBDBDDDDDDDDDDDDDDDDDBDB
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
CGGATATCTGCGACAGCTTGCTGTAGTACAGTACAGGTACAGTACGTACGTTGCAGTAGAAGCAATCCATGCATAAATACGTCTGCCACACCGGAGGAG
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
1=DDDDDHHAHHIIIIIIIIIIIIIIIIIIIIIIIIIIIGEGGIIIIIIIIIIIIIIIIHIIIIIIIIHHHHHFEEEEEEEECCCBCBCCCCBBBBBBBBBB
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
AGGACAGCCGGACTTAATGATGAAGAGATGGCTCTCATCATCAAGAGATTCAAGACGGCGCTAAAAGGTCACAAGGGGCAGAGATCGGAAGAGCGGTTC
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
1=DDDDDHHDHHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGIEEIIIIECCBBCCCBCCCCCCBBBBBBBBBCCCBB?BBBBCBBBBB
SRR836349.17 HWI-ST1085:39:D0D7RACXX:1:1101:5373:1997 length=100
AGGACTGCGGCCGCCGTGAGCGTCATCAGATCGCCGGGGCTTCTGACGCCAACCGCGGCCTGCACGGCGGCGGCCACCTGGTCGTGGCGCGCCCCCGCG
SRR836349.17 HWI-ST1085:39:D0D7RACXX:1:1101:5373:1997 length=100
1=DFFFFHHAHHIIIIIIIIIIIIIIIIIIIIGIIIIHFFCCCCCCCCECCCCBCBCCCCCCCCCCCCCBBCCBBBB>5@BACCACB@BBBB>>BBCCB9>B
SRR836349.18 HWI-ST1085:39:D0D7RACXX:1:1101:5308:1997 length=100
TAGCCAGCGACCGCTACGTGGCAGTCGTCGTGAGGACACCTAATCCTCAATTCGCATAAATATTCACCTACACCCCTTGCGTCCAAGGTATTTCGTCTT
SRR836349.18 HWI-ST1085:39:D0D7RACXX:1:1101:5308:1997 length=100
4=DDFFFHHHHHJJJJJJIJJJJJJJJJJJIHIGIIGGIJHGHHFFFFFFEEEEDDDDDDDEEEFDEDDDDDACCBBDDDDDDDDDDDDCCEEEEDDBD?
SRR836349.19 HWI-ST1085:39:D0D7RACXX:1:1101:5732:1994 length=100
ATCTCTGCTTGATTAGAAAAAGGAGATCCACGGCCTGTGAGTAACTACGCCATGTTACGAATCACAGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGA
SRR836349.19 HWI-ST1085:39:D0D7RACXX:1:1101:5732:1994 length=100
1=DDFFFHHCACFHIJJJJJJJJJJJJJJJJJJJJJJJIJIIFHIIJIJJJJJJJJJJJJJHHHGFFFFFEEEEDDDDDDDDDDDDDDDDDDDDDDDD@
SRR836349.20 HWI-ST1085:39:D0D7RACXX:1:1101:5623:1995 length=100
TCCTGGACTGCACCCTTAGGCGACTTCTGCGCGGATAGTCTTCGCGATAGACATTGGATTTTCTCGCACTTGTTGTGCTTAGTCCGGCTGAGATCGGAA
SRR836349.20 HWI-ST1085:39:D0D7RACXX:1:1101:5623:1995 length=100
4=DDDDDHHBHHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIHEECCCCCCCCFFCCCCCCECCBBBBBCCCCBCBCCCCCCCCBBBBBA@:@CBBBB
SRR836349.21 HWI-ST1085:39:D0D7RACXX:1:1101:6484:1991 length=100
GAAGACTGCGACTTACCGTGTTGGAGTTGTGTAGCCACCGTGAAACACCACACTGCGCGCCGCCGCCACCACCAAAAGAGAGTGTTTCCGCCGCCGTGG
SRR836349.21 HWI-ST1085:39:D0D7RACXX:1:1101:6484:1991 length=100
4=DDFFFGHHGHJJIJJJJJJJJJJIJJIJJJJJIJJJJJJJJJJJJHIIJJJIJJIJHHFDDDDDDDDDDDDDDDDDDDDDDDD:ACEDDDDDDDDDDBDD
SRR836349.22 HWI-ST1085:39:D0D7RACXX:1:1101:6274:1992 length=100
ATATGCCAGCTAATCGTGTGGCTTTAGAAGCCTGTGTACAAGCTCGTAACGAAGGGCGCGATCTTGCTCGTGAAGGTAATGAAATTATCAAAGCAGAGA
SRR836349.22 HWI-ST1085:39:D0D7RACXX:1:1101:6274:1992 length=100
1=DDFFFHHHHHIJJJJJIJJJJJJFIIJJJJJJGIGIHIIJJJJJIJJJJIJIJJIHHFDDDDDDDDDD@ACBCC:ACDCDDDDDDDADDDDDCCCDC
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
GCTGATCAGCTCCCCGTTGTCGCCGGCGACGACCTGGCGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACGGATCTCGTATGCCGTGTTCTG
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
11BDDDDAFBFHHGIIAFGEFHGGHI<D<FG8',5?B(998;8@:@@BB7258<9:@5-2(:3+8?0(44::>&099(0&09CC@###############
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
CACCCAGCTTGAATCGAGGATTCACCCAACCACACTCTCAAAGAACCAACCAACACAGCCTACAATCAATCGAGGAGACCTCGAAGGGAGTAGGAGCAC
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
4=DDFFFHHHHHJJJJJJJJJJIJJJJIJIIGHIJGIGIIJJIJJGHIJJJIIIJJJEHHHFFFFFEDCEEDDDDDDDDDDDDDDDDDD8ACCDDDBB
SRR836349.25 HWI-ST1085:39:D0D7RACXX:1:1101:7402:1992 length=100
```

Estimate the percentage of sequencing reads in the FASTQ file that contains the adapter "AGATCGGAAGAGC".


```
cat D7RACXX.fastq | grep AGATCGGAAGAGC | wc -l
```

Estimate the percentage of sequencing reads in the FASTQ file that contains the adapter "AGATCGGAAGAGC".



cat DeD7RACXX.fastq | head -n 40000 | grep AGATCGGAAGAGC | wc -l

To get a quick answer, you can estimate the percentage based on the first 40,000 lines

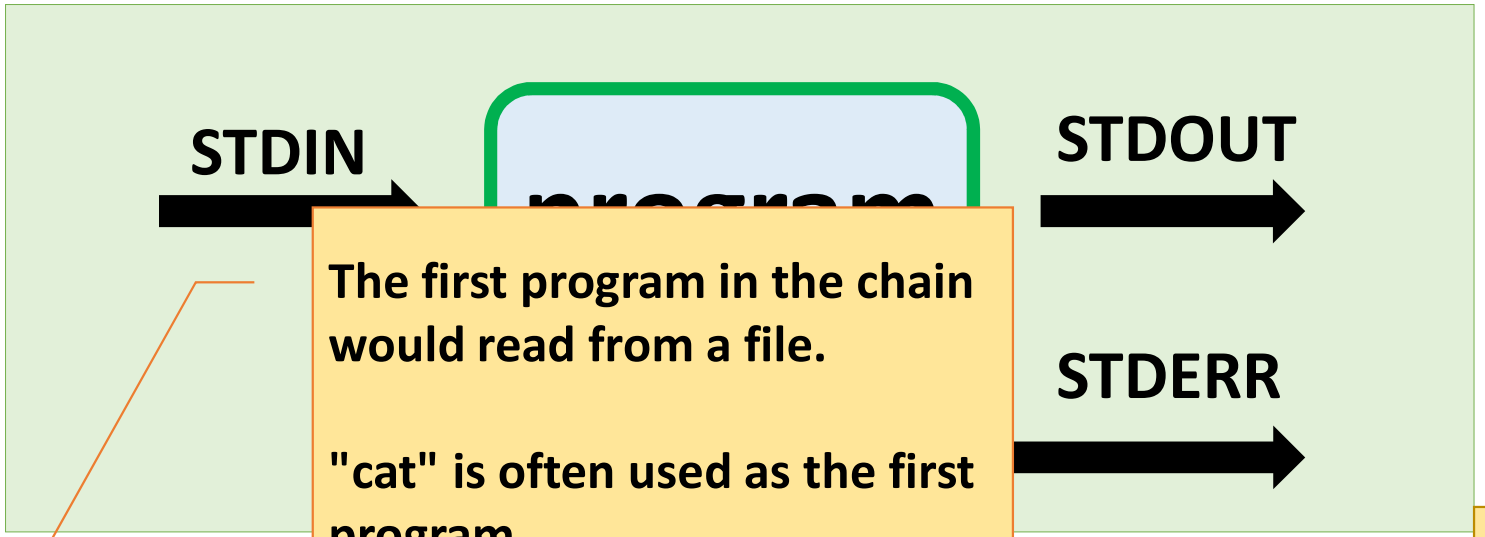
Estimate the percentage of sequencing reads in the FASTQ file that contains the adapter “AGATCGGAAGAGC”.

```
cat D7RACXX.fastq | \
head -n 40000 | \
grep AGATCGGAAGAGC | \
wc -l
```

Use “\” to separate the command into multiple lines

```
4=DFFFFHHHHHJJJJJJJJJJJJJJJGHIJFGIJEHIGEEEFDBBBBDBBBDBBDDDD@DDBDDDDDDDDDDDBBDDCDDDDDDBDDDDDDDBDDACC
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
CCAGTCTGCGTCCGCCTAGGCCCGGTCGACGAGCCTGGGGCCAAGGTGGCGTACCAGACCAGCGTCAGCATCACCGGCCACGTCTTCAAGGGCATCCTG
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
4=DFFFFGHAHHIIIIIIIIIIIIIIIIIIIIIIIIHHHHFFCCCCC@CBCCBCCCCCCCCCCCCBBCCCCCCCCCCCCCCCCCCCCCECCCCCCCCCCC
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
TACGGACAGCTCGGTGCACTGCTTCAAAAACTGCCGGATGTAGTCCCACAGGGATTCCCCTGGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAG
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
1=DDFFFHHHHHJJGHJJJJJJJJJJJJJJIJJJJJJJIJJIJJIIIJIJIJJHHHHFFFFFDDEEDDDDDDDDDDDDDDDDDDDCDDDD@BCDDDDD@
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
TATTACAGCAGGCGCGCCGAGAGAGAGTGGAGAGGAGAGGACGGTGGCGATCGAGCGAGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCG
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
1=DFFFFGHHHHJJJJJJJGIGIFHFH=BGGHIIJGHGHHGHFDCDDDDDDDDBDDDDDDDDDDDDDDDDDDDDBDBDDDDDDDDDDDDDDDDDBDB
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
CGGATATCTGCGACAGCTTGCTGTAGTACAGTACAGGTACAGTACGTACGTTGCAGTAGAAGCAATCCATGCATAAATACGTCTGCCACACCGGAGGAG
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
1=DDDDDHHAHHIIIIIIIIIIIIIIIIIIIIIIIIIIGEGGIIIIIIIIIIIIIIIHIIIIIIIIHHHHHFEEEEEEEECCCBCBCCCCBBBBBBBBBB
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
AGGACAGCCGGACTTAATGATGAAGAGATGGCTCTCATCATCAAGAGATTCAAGACGGCGCTAAAAGGTCACAAGGGGCAGAGATCGGAAGAGCGGTTC
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
1=DDDDDHHDHHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGIEEIIIIECCBBCCCBCCCCCCBBBBBBBBBCCCBB?BBBBCBBBBB
SRR836349.17 HWI-ST1085:39:D0D7RACXX:1:1101:5373:1997 length=100
AGGACTGCGGCCGCCGTGAGCGTCATCAGATCGCCGGGGCTTCTGACGCCAACCGCGGCCTGCACGGCGGCGGCCACCTGGTCGTGGCGCGCCCCCGCG
SRR836349.17 HWI-ST1085:39:D0D7RACXX:1:1101:5373:1997 length=100
1=DFFFFHHAHHIIIIIIIIIIIIIIIIIIIGIIIIHFFCCCCCCCCECCCCBCBCCCCCCCCCCCCCBBCCBBBB>5@BACCACB@BBBB>>BBCCB9>B
SRR836349.18 HWI-ST1085:39:D0D7RACXX:1:1101:5308:1997 length=100
TAGCCAGCGACCGCTACGTGGCAGTCGTCGTGAGGACACCTAATCCTCAATTCGCATAAATATTCACCTACACCCCTTGCGTCCAAGGTATTTCGTCTT
SRR836349.18 HWI-ST1085:39:D0D7RACXX:1:1101:5308:1997 length=100
4=DDFFFHHHHHJJJJJJIJJJJJJJJJJJIHIGIIGGIJHGHHFFFFFFEEEEDDDDDDDEEEFDEDDDDDACCBBDDDDDDDDDDDDCCEEEEDDBD?
SRR836349.19 HWI-ST1085:39:D0D7RACXX:1:1101:5732:1994 length=100
ATCTCTGCTTGATTAGAAAAAGGAGATCCACGGCCTGTGAGTAACTACGCCATGTTACGAATCACAGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGA
SRR836349.19 HWI-ST1085:39:D0D7RACXX:1:1101:5732:1994 length=100
1=DDFFFHHCACFHIJJJJJJJJJJJJJJJJJJJJJJJIJIIFHIIJIJJJJJJJJJJJJJHHHGFFFFFEEEEDDDDDDDDDDDDDDDDDDDDDDD@
SRR836349.20 HWI-ST1085:39:D0D7RACXX:1:1101:5623:1995 length=100
TCCTGGACTGCACCCTTAGGCGACTTCTGCGCGGATAGTCTTCGCGATAGACATTGGATTTTCTCGCACTTGTTGTGCTTAGTCCGGCTGAGATCGGAA
SRR836349.20 HWI-ST1085:39:D0D7RACXX:1:1101:5623:1995 length=100
4=DDDDDHHBHHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIHEECCCCCCCCFFCCCCCCECCBBBBBCCCCBCBCCCCCCCCBBBBBA@:@CBBBB
SRR836349.21 HWI-ST1085:39:D0D7RACXX:1:1101:6484:1991 length=100
GAAGACTGCGACTTACCGTGTTGGAGTTGTGTAGCCACCGTGAAACACCACACTGCGCGCCGCCGCCACCACCAAAAGAGAGTGTTTCCGCCGCCGTGG
SRR836349.21 HWI-ST1085:39:D0D7RACXX:1:1101:6484:1991 length=100
4=DDFFFGHHGHJJIJJJJJJJJJJIJJIJJJJJIJJJJJJJJJJJJHIIJJJIJJIJHHFDDDDDDDDDDDDDDDDDDDDDDDD:ACEDDDDDDDDDDBDD
SRR836349.22 HWI-ST1085:39:D0D7RACXX:1:1101:6274:1992 length=100
ATATGCCAGCTAATCGTGTGGCTTTAGAAGCCTGTGTACAAGCTCGTAACGAAGGGCGCGATCTTGCTCGTGAAGGTAATGAAATTATCAAAGCAGAGA
SRR836349.22 HWI-ST1085:39:D0D7RACXX:1:1101:6274:1992 length=100
1=DDFFFHHHHHIJJJJJIJJJJJJFIIJJJJJJGIGIHIIJJJJJIJJJJIJIJJIHHFDDDDDDDDDD@ACBCC:ACDCDDDDDDDADDDDDCCCDC
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
GCTGATCAGCTCCCCGTTGTCGCCGGCGACGACCTGGCGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACGGATCTCGTATGCCGTGTTCTG
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
11BDDDDAFBFHHGIIAFGEFHGGHI<D<FG8',5?B(998;8@:@@BB7258<9:@5-2(:3+8?0(44::>&099(0&09CC@###############
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
CACCCAGCTTGAATCGAGGATTCACCCAACCACACTCTCAAAGAACCAACCAACACAGCCTACAATCAATCGAGGAGACCTCGAAGGGAGTAGGAGCAC
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
4=DDFFFHHHHHJJJJJJJJJJIJJJJIJIIGHIJGIGIIJJIJJGHIJJJIIIJJJEHHHFFFFFEDCEEDDDDDDDDDDDDDDDDDD8ACCDDDBB
SRR836349.25 HWI-ST1085:39:D0D7RACXX:1:1101:7402:1992 length=100
```

Three streams for a standard Linux program

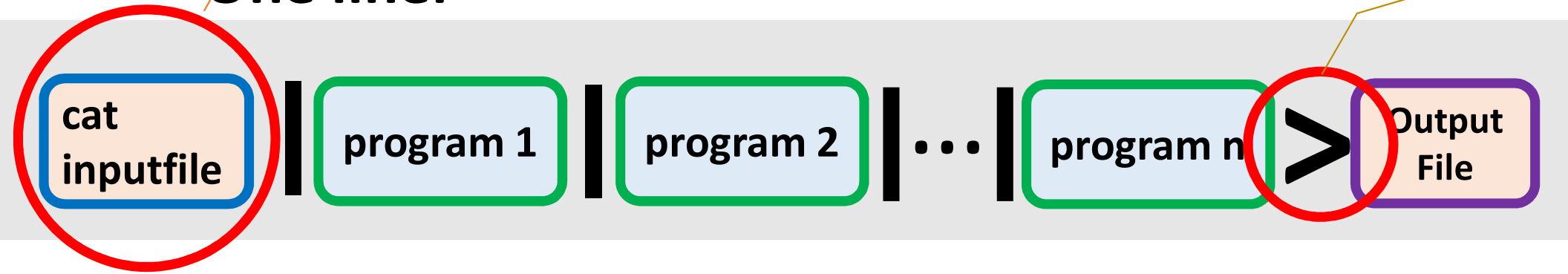


The first program in the chain would read from a file.

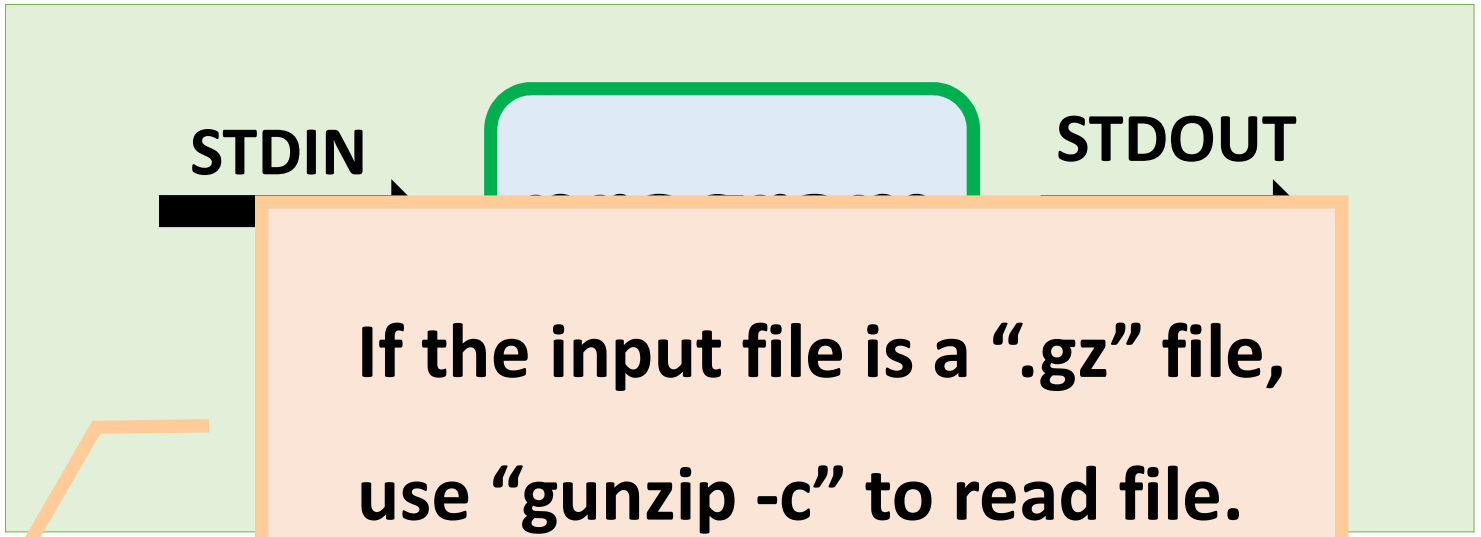
"cat" is often used as the first program.

One liner

Write to file



Three streams for a standard Linux program



If the input file is a “.gz” file,

use “gunzip -c” to read file.

One line:

grep

Search for a pattern and output matched lines

$ cat mydata.txt

```
AAGATCAAAAAAGA
ATTTACGAAAAAAGA
ACCTGTTGGATCCAAAGTT
AAACTTTCGACGATCT
ATTTTTTTAGAAAGG
```

$ cat mydata.txt | grep '[AC]GATC'

```
AAGATCAAAAAAGA
AAACTTTCGACGATCT
```

wc -l Count the number of lines

$ cat mydata.txt

AAGATCAAAAAAGA

ATTTACGAAAAAAGA

ACCTGTTGGATCCAAAGTT

AAACTTTCGACGATCT

ATTTTTTTAGAAAGG

$ cat mydata.txt | grep '[AC]GATC' | wc -l

2

sort

Sort the text in a file

$ sort myChr.txt

```
Chr1
Chr10
Chr2
Chr3
Chr4
Chr5
```

$ sort -V myChr.txt

```
Chr1
Chr2
Chr3
Chr4
Chr5
Chr10
```

$ sort -n myPos.txt

```
1
2
3
4
5
10
```

sort

Sort the text by multiple columns

$ sort -k1,1V -k2,2n myChrPos.txt

```
Chr1    12
Chr1    100
Chr1    200
Chr2    34
Chr2    121
Chr2    300
```

uniq -c

Count the occurrence of unique tags

$ cat mydata.txt

```
ItemB
ItemA
ItemB
ItemC
ItemB
ItemC
ItemB
ItemC
```

$ cat mydata.txt | sort | uniq -c

```
1    ItemA
4    ItemB
3    ItemC
```

Mark sure to run "sort" before "uniq"

Merging files:

cat f1 f2 vs **paste** f1 f2 vs **join** f1 f2

File 1:

Item1

Item2

File2:

Item3

Item4

cat File1 File2 > mergedfile1

Item1
Item2
Item3
Item4

paste File1 File2 > mergedfile2

Item1	Item3
Item2	Item4

* Make sure that that two files has same number of rows and sorted the same way. Otherwise, use “join”

join

Merging two files that share a common field

File 1:

Gene1	DNA-binding
Gene2	kinase
Gene3	membrane

File2:

Gene2	764
Gene3	23
Gene4	34

join -1 1 -2 1 File1 File2 > mergedfile

Gene2	Kinase	764
Gene3	membrane	23

join -1 1 -2 1 -a1 File1 File2 > mergedfile

Gene1	DNA-binding	
Gene2	Kinase	764
Gene3	membrane	23

cut

Output selected columns in a table

$ cat mydata.txt

```
Chr1 1000  2250 Gene1
Chr1  3010 5340 Gene2
Chr1  7500 8460 Gene3
Chr2  8933 9500 Gene4
```

$ cat mydata.txt | cut -f 1,4

```
Chr1  Gene1
Chr1 Gene2
Chr1 Gene3
Chr2  Gene4
```

sed Modify text in a file

$ cat mydata.txt

```
Chr1 1000  2250 Gene1
Chr1  3010 5340 Gene2
Chr1  7500 8460 Gene3
Chr2  8933 9500 Gene4
```

$ cat mydata.txt | sed "s/^Chr//"

```
1     1000 2250 Gene1
1     3010 5340 Gene2
1     7500 8460 Gene3
2     8933 9500 Gene4
```

awk

Probably the most versatile function in Linux

$ cat mydata.txt

```
Chr1 1000  2250 Gene1
Chr1  3010 5340 Gene2
Chr1  7500 8460 Gene3
Chr2  8933 9500 Gene4
```

$ cat mydata.txt |
awk '{if ($1=="Chr1") print $4}'

```
Gene1
Gene2
Gene3
```

A Good Practice:
Create a shell script file for the one liner

```
cat D7RACXX.fastq | \
head –n 40000 | \
grep AGATCGGAAGAGC | \
wc -l
```

Run the shell script

```
sh checkadapter.sh
```

Debug a one-liner

```
gunzip -c human.gff3.gz | \
awk 'BEGIN {OFS = "\t"}; {if ($3=="gene") print $1,$4-1,$5}' | \
bedtools coverage -a win1mb.bed -b stdin -counts | \
LC_ALL=C sort -k1,1V -k2,2n  > gene.cover.bed
```

```
gunzip -c human.gff3.gz | head -n 1000 > tmpfile
```

```
cat tmpfile| \
awk 'BEGIN {OFS = "\t"}; {if ($3=="gene") print $1,$4-1,$5}' | head -n 100
```

Many bioinformatics software support STDIN as input

bwa mem ref.fa reads.fq | **samtools** view -bS **-** > out.bam

Use "-" to specify input from STDIN instead of a file

```
gunzip -c D001.fastq.gz | fastx_clipper -a AGATCG
```

…… | **bedtools** coverage -a **FirstFile** -b **stdin**

BEDtools - An Example

A file: Gene Annotation

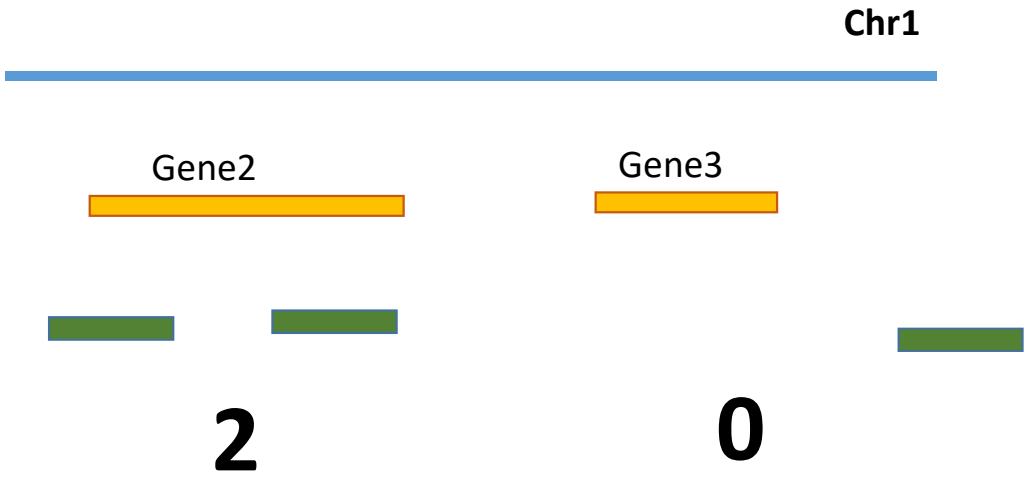
Chr1	1000	2250	Gene1	.	+
Chr1	3010	5340	Gene2	.	-
Chr1	7500	8460	Gene3	.	-
Chr2	8933	9500	Gene4	.	+
Chr2	12000	14000	Gene5	.	+

B file: Recorded Features

Chr1	200	300	Peak1
Chr1	4010	4340	Peak2
Chr1	5020	5300	Peak3
Chr2	8901	9000	Peak4

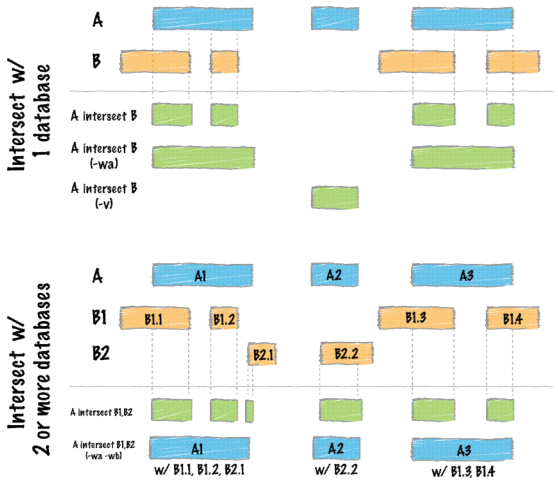
BEDtools coverage

		Number of overlapping peaks
…	Gene1	0
…	Gene2	2
…	Gene3	0
…	Gene4	1

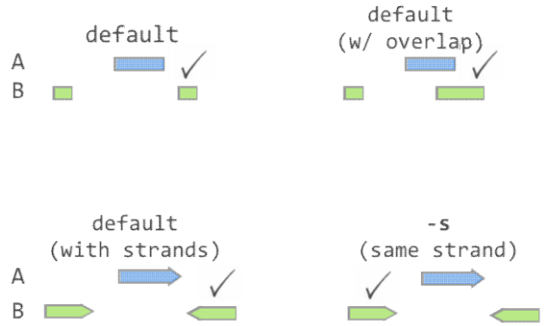


There Are Many Functions in BEDtools

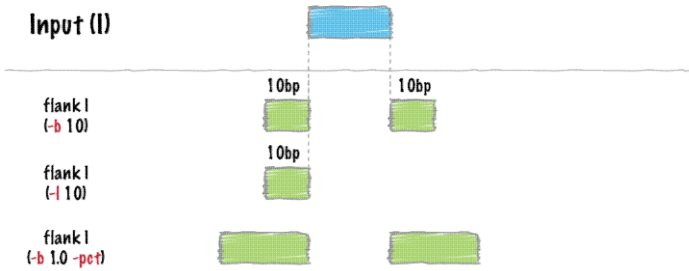
Intersect



Closest



Flanking



Other Tools that Work with Genomic Interval Files

SAM/BAM file	SAMtools, BAMtools, PICARD
VCF file	VCFtools, BCFtools
GFF3/GTF/BED	BEDtools, BEDOPS

Using multi-processor machines

All BioHPC Lab machines feature multiple CPU cores

- general (cbsum1c*b*): 8 CPU cores
- medium-memory (cbsumm*): 24 CPU cores
- marge-memory (cbsulm*): 64+ CPU cores

Using multi-processor machines

Three ways to utilize multiple CPU cores on a machine:

- Using a given program's built-in parallelization, e.g.:

```
blast+ -num_threads 8 [other options]

bowtie -p 8 [other options]
```

Typically, all CPUs work together on a single task. Non-trivial, but taken care of by the programmers.

- Simultaneously executing several programs in the background, e.g.:

```
gzip file1 &
gzip file2 &
gzip file3 &
```

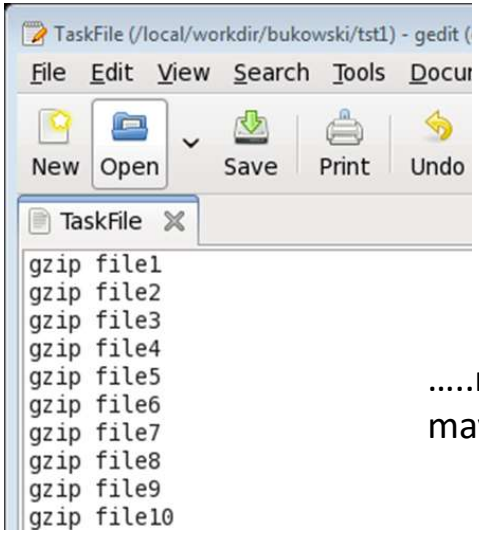
Multiple independent tasks

- If the number of independent tasks is larger than the number of CPU cores - **use a "driver" program:**

```
/programs/bin/perlscripts/perl_fork_univ.pl
```

Using perl_fork_univ.pl

Prepare a file (called, for example, **`TaskFile`**) listing all commands to be executed. For example,



…..number of lines (i.e., tasks)
may be very large

Then run the following command:

```
/programs/bin/perlscripts/perl_fork_univ.pl TaskFile NP >& log &
```

where **`NP`** is the number of processors to use (e.g., 10). The file “log” will contain some useful timing information.

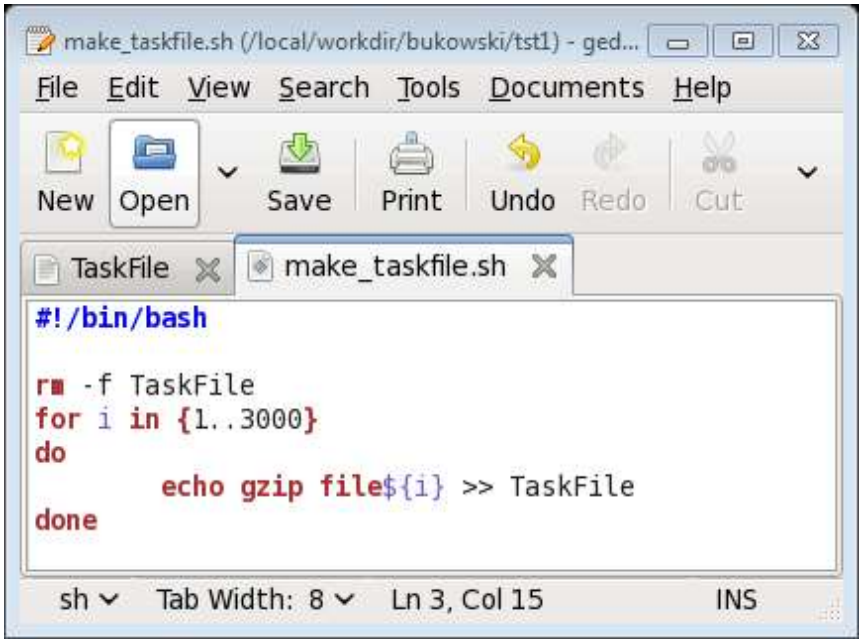
Using perl_fork_univ.pl

What does the script **`perl_fork_univ.pl`** do?

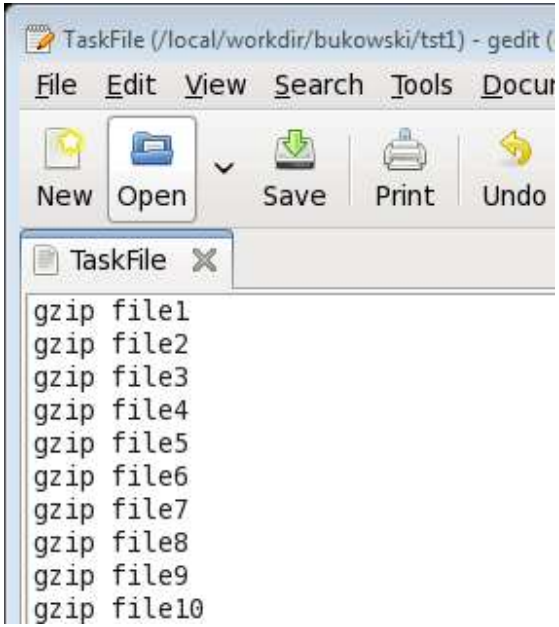
- **`perl_fork_univ.pl`** is an CBSU in-house "driver" script (written in perl)
- It will execute tasks listed in **`TaskFile`** using up to **`NP`** processors
  - The first **`NP`** tasks will be launched simultaneously
  - The **`(NP+1)`**th task will be launched right after one of the initial ones completes and a "slot" becomes available
  - The **`(NP+2)`**nd task will be launched right after another slot becomes available
  - …… etc., until all tasks are distributed
- Only up to **`NP`** tasks are running at a time (less at the end)
- All **`NP`** processors always kept busy (except near the end of task list) – **Load Balancing**

Using perl_fork_univ.pl

Ho to efficiently create a long list of tasks? Can use "**loop**" syntax built into bash:



`TaskFile`



.....

Create an run a script like this, or just type directly from command line, ending each line with RETURN

How to choose number of CPU cores

Typically, determining the right number of CPUs to run on requires some experimentation.

Factors to consider

- total number of CPU cores on a machine: NP <= (number of CPU cores on the machine)
- combined memory required by all tasks running simultaneously should not exceed about 90% of total memory available on a machine; use **`top`** to monitor memory usage
- disk I/O bandwidth: tasks reading/write to the same disk compete for disk bandwidth. Running too many simultaneously will slow things down
- other jobs running on a machine: they also take CPU cores and memory!