

Advanced Linux Usage

2025-11-18

Martin Dahlö
martin.dahlo@nbis.se

Enabler for Life Sciences

- Same program, many files

```
$ ls -l
total 0
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_9.bam
$ my_prog sample_1.bam
```

- Same program, many files

```
$ ls -l
total 0
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_9.bam
$ my_prog sample_1.bam
$ my_prog sample_2.bam
```

- Same program, many files

```
$ ls -l
total 0
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 16:42 sample_9.bam
$ my_prog sample_1.bam
$ my_prog sample_2.bam
$ my_prog sample_3.bam
$ my_prog sample_4.bam
$ my_prog sample_5.bam
$ my_prog sample_6.bam
$ my_prog sample_7.bam
$ my_prog sample_8.bam
$ my_prog sample_9.bam
$
```

Multiple files

- Same program, many files
 - 10 files? Ok
 - 1000 files? Not ok

Multiple files

- Same program, many files
 - 10 files? Ok
 - 1000 files? Not ok
- Reproducibility
 - Self and others

Multiple files

- Same program, many files
 - 10 files? Ok
 - 1000 files? Not ok
- Reproducibility
 - Self and others

A solution - write a script!

Basic script

```
total 0
-rw-rw-r-- 1 dahlo dahlo 0 Sep 1 17:18 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep 1 17:18 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep 1 17:18 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep 1 17:18 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep 1 17:18 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep 1 17:18 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep 1 17:18 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep 1 17:18 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep 1 17:18 sample_9.bam
$ nano analysis.sh
```

Basic script

GNU nano 2.0.9

File: analysis.sh

Modified

^G Get Help
^X Exit

^O WriteOut
^J Justify

^R Read File
^W Where Is

^Y Prev Page
^V Next Page

^K Cut Text
^U UnCut Text

^C Cur Pos
^T To Spell

Basic script

GNU nano 2.0.9

File: analysis.sh

Modified

```
my_prog sample_1.bam
```

^G Get Help
^X Exit

^O WriteOut
^J Justify

^R Read File
^W Where Is

^Y Prev Page
^V Next Page

^K Cut Text
^U UnCut Text

^C Cur Pos
^T To Spell

Basic script

GNU nano 2.0.9

File: analysis.sh

Modified

```
my_prog sample_1.bam
my_prog sample_2.bam
```

^G Get Help
^X Exit

^O WriteOut
^J Justify

^R Read File
^W Where Is

^Y Prev Page
^V Next Page

^K Cut Text
^U UnCut Text

^C Cur Pos
^T To Spell

Basic script

GNU nano 2.0.9

File: analysis.sh

Modified

```
my_prog sample_1.bam
my_prog sample_2.bam
my_prog sample_3.bam
my_prog sample_4.bam
my_prog sample_5.bam
my_prog sample_6.bam
my_prog sample_7.bam
my_prog sample_8.bam
my_prog sample_9.bam
```

^G Get Help
^X Exit

^O WriteOut
^J Justify

^R Read File
^W Where Is

^Y Prev Page
^V Next Page

^K Cut Text
^U UnCut Text

^C Cur Pos
^T To Spell

Basic script

```
$ l
total 4,0K
-rw-rw-r-- 1 dahlo dahlo 267 Sep  7 09:34 analysis.sh
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_9.bam
$
```

Basic script

```
$ l
total 4,0K
-rw-rw-r-- 1 dahlo dahlo 267 Sep  7 09:34 analysis.sh
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_9.bam
$ bash analysis.sh
```

```
$ l
total 4,0K
-rw-rw-r-- 1 dahlo dahlo 267 Sep  7 09:34 analysis.sh
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_6.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_7.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_8.bam
-rw-rw-r-- 1 dahlo dahlo    0 Sep  1 17:18 sample_9.bam
$ bash analysis.sh
```

Still not OK for 1000 or more files!

Basic script

GNU nano 2.0.9

File: analysis.sh

Modified

```
my_prog sample_1.bam
my_prog sample_2.bam
my_prog sample_3.bam
my_prog sample_4.bam
my_prog sample_5.bam
my_prog sample_6.bam
my_prog sample_7.bam
my_prog sample_8.bam
my_prog sample_9.bam
```

^G Get Help
^X Exit

^O WriteOut
^J Justify

^R Read File
^W Where Is

^Y Prev Page
^V Next Page

^K Cut Text
^U UnCut Text

^C Cur Pos
^T To Spell

Basic script

GNU nano 2.5.3

File: analysis.sh

```
my_prog -r references/human_genome.fa sample_1.bam
my_prog -r references/human_genome.fa sample_2.bam
my_prog -r references/human_genome.fa sample_3.bam
my_prog -r references/human_genome.fa sample_4.bam
my_prog -r references/human_genome.fa sample_5.bam
my_prog -r references/human_genome.fa sample_6.bam
my_prog -r references/human_genome.fa sample_7.bam
my_prog -r references/human_genome.fa sample_8.bam
my_prog -r references/human_genome.fa sample_9.bam
```

^G Get Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^ Replace

^K Cut Text
^U Uncut Text

^J Justify
^T To Spell

^C Cur Pos
^ Go To Line

^Y Prev Page
^V Next Page

Basic script

GNU nano 2.5.3

File: analysis.sh

```
my_prog -r references/human_genome.fa sample_1.bam
my_prog -r references/human_genome.fa sample_2.bam
my_prog -r references/human_genome.fa sample_3.bam
my_prog -r references/human_genome.fa sample_4.bam
my_prog -r references/human_genome.fa sample_5.bam
my_prog -r references/human_genome.fa sample_6.bam
my_prog -r references/human_genome.fa sample_7.bam
my_prog -r references/human_genome.fa sample_8.bam
my_prog -r references/human_genome.fa sample_9.bam
```

Same for all

Unique

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos
^X Exit **^R** Read File **^** Replace **^U** Uncut Text **^T** To Spell **^** Go To Line

^Y Prev Page **^V** Next Page

- **Assigning**

```
my_variable=5
```

```
my_variable="nice text"
```

- **Assigning**

```
my_variable=5
```

```
my_variable="nice text"
```

- **Using**

```
$my_variable
```

- **Assigning**

```
my_variable=5
```

```
my_variable="nice text"
```

- **Using**

```
$my_variable
```

```
$ my_variable="Martin"
```

- **Assigning**

```
my_variable=5
```

```
my_variable="nice text"
```

- **Using**

```
$my_variable
```

```
$ my_variable="Martin"
```

```
$ echo "Hello $my_variable."
```

- **Assigning**

```
my_variable=5
```

```
my_variable="nice text"
```

- **Using**

```
$my_variable
```

```
$ my_variable="Martin"
```

```
$ echo "Hello $my_variable."
```

```
Hello Martin.
```

Basic script

GNU nano 2.5.3

File: analysis.sh

```
my_prog -r references/human_genome.fa sample_1.bam
my_prog -r references/human_genome.fa sample_2.bam
my_prog -r references/human_genome.fa sample_3.bam
my_prog -r references/human_genome.fa sample_4.bam
my_prog -r references/human_genome.fa sample_5.bam
my_prog -r references/human_genome.fa sample_6.bam
my_prog -r references/human_genome.fa sample_7.bam
my_prog -r references/human_genome.fa sample_8.bam
my_prog -r references/human_genome.fa sample_9.bam
```

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos **^Y** Prev Page
^X Exit **^R** Read File **^V** Replace **^U** Uncut Text **T** To Spell **^** Go To Line **^V** Next Page

Basic script

GNU nano 2.5.3

File: analysis.sh

`ref=references/human_genome.fa`

```
my_prog -r $ref sample_1.bam  
my_prog -r $ref sample_2.bam  
my_prog -r $ref sample_3.bam  
my_prog -r $ref sample_4.bam  
my_prog -r $ref sample_5.bam  
my_prog -r $ref sample_6.bam  
my_prog -r $ref sample_7.bam  
my_prog -r $ref sample_8.bam  
my_prog -r $ref sample_9.bam
```

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos **^Y** Prev Page
^X Exit **^R** Read File **^L** Replace **^U** Uncut Text **^T** To Spell **^** Go To Line **^V** Next Page

Basic script

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa

my_prog -r $ref sample_1.bam
my_prog -r $ref sample_2.bam
my_prog -r $ref sample_3.bam
my_prog -r $ref sample_4.bam
my_prog -r $ref sample_5.bam
my_prog -r $ref sample_6.bam
my_prog -r $ref sample_7.bam
my_prog -r $ref sample_8.bam
my_prog -r $ref sample_9.bam
```

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos
^X Exit **^R** Read File **^** Replace **^U** Uncut Text **^T** To Spell **^** Go To Line **^Y** Prev Page
 ^V Next Page

Basic script

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa
```

```
my_prog -r $ref sample_1.bam  
my_prog -r $ref sample_2.bam  
my_prog -r $ref sample_3.bam  
my_prog -r $ref sample_4.bam  
my_prog -r $ref sample_5.bam  
my_prog -r $ref sample_6.bam  
my_prog -r $ref sample_7.bam  
my_prog -r $ref sample_8.bam  
my_prog -r $ref sample_9.bam
```

Unique

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos
^X Exit **^R** Read File **^** Replace **^U** Uncut Text **^T** To Spell **^** Go To Line **^Y** Prev Page
 ^V Next Page

```
for var in 1 2 3;  
do  
    echo $var  
done
```

```
$ bash loop_test.sh  
1  
2  
3  
$
```

```
for var in text works too;  
do  
    echo $var  
done
```

```
$ bash loop_test.sh  
text  
works  
too  
$
```

```
for var in mix them 5;  
do  
    echo $var  
done
```

```
$ bash loop_test.sh  
mix  
them  
5  
$
```

```
for var in *.txt;  
do  
    echo $var  
done
```

```
$ bash loop_test.sh  
all.txt  
examples.txt  
readme.txt
```

Basic script

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa

my_prog -r $ref sample_1.bam
my_prog -r $ref sample_2.bam
my_prog -r $ref sample_3.bam
my_prog -r $ref sample_4.bam
my_prog -r $ref sample_5.bam
my_prog -r $ref sample_6.bam
my_prog -r $ref sample_7.bam
my_prog -r $ref sample_8.bam
my_prog -r $ref sample_9.bam
```

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos
^X Exit **^R** Read File **^** Replace **^U** Uncut Text **^T** To Spell **^** Go To Line **^Y** Prev Page
 ^V Next Page

Basic script

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa

for file in *.bam ;
do
    my_prog -r $ref $file
done
```

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos
^X Exit **^R** Read File **^** Replace **^U** Uncut Text **^T** To Spell **^** Go To Line **^Y** Prev Page
 ^V Next Page

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa

for file in *.bam ;
do
    echo my_prog -r $ref $file
done
```

Debugging!

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos
^X Exit **^R** Read File **^** Replace **^U** Uncut Text **^T** To Spell **^** Go To Line **^Y** Prev Page
 ^V Next Page

Basic script

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa

for file in *.bam ;
do
    echo my_prog -r $ref $file
done
```

```
$ bash analysis.sh
my_prog -r references/goat_genome_version4.1.fa sample_1.bam
my_prog -r references/goat_genome_version4.1.fa sample_2.bam
my_prog -r references/goat_genome_version4.1.fa sample_3.bam
my_prog -r references/goat_genome_version4.1.fa sample_4.bam
my_prog -r references/goat_genome_version4.1.fa sample_5.bam
my_prog -r references/goat_genome_version4.1.fa sample_6.bam
my_prog -r references/goat_genome_version4.1.fa sample_7.bam
my_prog -r references/goat_genome_version4.1.fa sample_8.bam
my_prog -r references/goat_genome_version4.1.fa sample_9.bam
```

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text
^X Exit **^R** Read File **^** Replace **^U** Uncut Te

Basic script

GNU nano 2.5.3

File: analysis.sh

```
ref=references/goat_genome_version4.1.fa

for file in *.bam ;
do
    my_prog -r $ref $file
done
```

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos
^X Exit **^R** Read File **^** Replace **^U** Uncut Text **^T** To Spell **^** Go To Line **^Y** Prev Page
 ^V Next Page

Arguments

```
$ bash analysis.sh
```

Arguments

```
$ bash analysis.sh data/
```

Arguments

```
$ bash analysis.sh data/
```

\$1

Arguments

```
$ bash analysis.sh data/ second_argument
```

\$1

\$2

Arguments

```
$ bash analysis.sh data/ second_argument third
```

\$1

\$2

\$3

Arguments

```
$ bash analysis.sh data/ second_argument third "fourth argument"
```

\$1

\$2

\$3

\$4

Arguments

GNU nano 2.3.1

File: analysis.sh

Modified

```
ref=references/goat_genome_version4.1.fa

for file in *.bam;
do
    my_prog -r $ref $file
done
```

^G Get Help
^X Exit

^O WriteOut
^J Justify

^R Read File
^W Where Is

^Y Prev Page
^V Next Page

^K Cut Text
^U UnCut Text

^C Cur Pos
^T To Spell

Arguments

GNU nano 2.3.1

File: analysis.sh

Modified

```
ref=references/goat_genome_version4.1.fa

for file in $1/*.bam;
do
    my_prog -r $ref $file
done
```

^G Get Help
^X Exit

^O WriteOut
^J Justify

^R Read File
^W Where Is

^Y Prev Page
^V Next Page

^K Cut Text
^U UnCut Text

^C Cur Pos
^T To Spell

Arguments

GNU nano 2.3.1

File: analysis.sh

Modified

```
ref=references/goat_genome_version4.1.fa

for file in $1/*.bam;
do
    my_prog -r $ref $file
done
```

```
$ bash analysis.sh /path/to/my/data
```

^G Get Help
^X Exit

^O WriteOut
^J Justify

^R Read File
^W Where Is

^Y Prev Page
^V Next Page

^K Cut Text
^U UnCut Text

^C Cur Pos
^T To Spell

Arguments

GNU nano 2.3.1

File: analysis.sh

Modified

```
ref=references/goat_genome_version4.1.fa

for file in /path/to/my/data/*.bam;
do
    my_prog -r $ref $file
done
```

```
$ bash analysis.sh /path/to/my/data
```

^G Get Help
^X Exit

^O WriteOut
^J Justify

^R Read File
^W Where Is

^Y Prev Page
^V Next Page

^K Cut Text
^U UnCut Text

^C Cur Pos
^T To Spell

- Control statement

```
if condition; then  
    action  
fi
```

- Control statement

```
if true; then  
    echo "This is true"  
fi
```

result:
This is true

- Control statement

```
if false; then  
    echo "This is true"  
fi
```

result:

- Control statement

```
if [ [ "Hello" == "Hello" ] ] ; then  
    echo "This is true"  
fi
```

result:
This is true

- Control statement

```
if [ [ "Hello" == "Hi" ] ]; then  
    echo "This is true"  
fi
```

result:

- Control statement

```
if [ [ "Hello" == "Hel" * ] ] ; then  
    echo "This is true"  
fi
```

result:
This is true

- Control statement

```
if [ [ 5 == 9 ] ]; then  
    echo "This is true"  
fi
```

result:

- Control statement

```
if [[ 5 -lt 9 ]]; then  
    echo "This is true"  
fi
```

result:
This is true

- Control statement

```
if [[ 5 -gt 9 ]]; then  
    echo "This is true"  
fi
```

result:

- Control statement (math context)

```
if (( 5 > 9 )) ; then  
    echo "This is true"  
fi
```

result:

- For all dog samples

```
for file in $1/*.bam ;  
do  
    echo my_prog $file  
done
```

- For all dog samples

```
for file in $1/*.bam;
do
  if [[ $file == dog* ]];
  then
    echo my_prog $file
  fi
done
```

- For all dog samples

```
for file in $1/*.bam;  
do  
  if [[ $file == dog* ]];  
  then  
  
    echo my_prog $file  
  
  fi  
done
```

```
/path/to/my/data/dog_sample_01.bam == dog* FALSE
```

- For all dog samples

```
for file in $1/*.bam;
do
  if [[ $file == dog* ]];
  then

    echo my_prog $file

  fi
done
```

```
/path/to/my/data/dog_sample_01.bam == dog* FALSE
/path/to/my/data/dog_sample_01.bam == *dog* TRUE
```

- For all dog samples

```
for file in $1/*.bam;
do
  if [[ $file == dog* ]];
then
  echo my_prog $file
fi
done
```

```
/path/to/my/data/dog_sample_01.bam == dog* FALSE
/path/to/my/data/dog_sample_01.bam == *dog* TRUE
/dog_and_more/lizard_sample_01.bam == *dog* TRUE
```

- For all dog samples

```
for file in $1/*.bam;
do
  if [[ $file == dog* ]];
then
  echo my_prog $file
fi
done
```

```
/path/to/my/data/dog_sample_01.bam == dog* FALSE
/path/to/my/data/dog_sample_01.bam == *dog* TRUE
/dog_and_more/lizard_sample_01.bam == *dog* TRUE

$ basename /path/to/my/data/dog_sample_01.bam
```

- For all dog samples

```
for file in $1/*.bam;
do
  if [[ $file == dog* ]];
then
  echo my_prog $file
fi
done
```

```
/path/to/my/data/dog_sample_01.bam == dog* FALSE
/path/to/my/data/dog_sample_01.bam == *dog* TRUE
/dog_and_more/lizard_sample_01.bam == *dog* TRUE

$ basename /path/to/my/data/dog_sample_01.bam
dog_sample_01.bam
```

- For all dog samples

```
for file in $1/*.bam;
do
  if [[ $file == dog* ]];
then
  echo my_prog $file
fi
done
```

```
/path/to/my/data/dog_sample_01.bam == dog* FALSE
/path/to/my/data/dog_sample_01.bam == *dog* TRUE
/dog_and_more/lizard_sample_01.bam == *dog* TRUE

$ basename /path/to/my/data/dog_sample_01.bam
dog_sample_01.bam

$(basename data/dog_sample_01.bam) == dog* TRUE
```

- For all dog samples

```
for file in $1/*.bam;
do
  if [[ $file == dog* ]];
then
  echo my_prog $file
fi
done
```

```
/path/to/my/data/dog_sample_01.bam == dog* FALSE
/path/to/my/data/dog_sample_01.bam == *dog* TRUE
/dog_and_more/lizard_sample_01.bam == *dog* TRUE

$ basename /path/to/my/data/dog_sample_01.bam
dog_sample_01.bam

$(basename data/dog_sample_01.bam) == dog* TRUE
dog_sample_01.bam == dog* TRUE
```

- For all dog samples

```
for file in $1/*.bam;
do
  if [[ $(basename $file) == dog* ]];
  then
    echo my_prog $file
  fi
done
```

- For all dog samples

```
for file in $1/*.bam;
do
  if [[ $(basename $file) == dog* ]];
  then
    my_prog $file
  fi
done
```

Different languages

- Programming is programming
 - Perl, Python, Bash, and more

Different languages

- Programming is programming
 - Perl, Python, **Bash**, and more

```
for file in $1/*.bam;
do
  if [[ $(basename $file) == dog* ]];
then
  my_prog $file
fi
done
```

Different languages

- Programming is programming
 - Perl, Python, Bash, and more

```
for file in $1/*.bam;
do
    if [[ $(basename $file) == dog* ]];
then
    my_prog $file
    fi
done
use strict;
use warnings;
use File::Basename;

foreach my $file (glob("$ARGV[0]/*.bam")) {
    if(basename($file) =~ "^dog.*") {
        system("my_prog", $file)
    }
}
```

Different languages

- Programming is programming
 - Perl, Python, Bash, and more

```
for file in $1/*.bam;
do
    if [[ $(basename $file) == dog* ]];
    then

        my_prog $file
    fi
done

import glob
import sys
import subprocess
import os

for file in glob.glob( sys.argv[1] + "/*.bam" ):

    if os.path.basename(file).startswith("dog"):

        subprocess.call( ["my_prog", file] )
```

Different languages

- Programming is programming
 - Perl, Python, Bash, and more
- Start with one, git gud, (learn another)

Different languages

- Programming is programming
 - Perl, Python, Bash, and more
- Start with one, git gud, (learn another)

PYTHON

- Graphical text editor more similar to what you might be used to
- Launch through command line:

```
$ gedit
```

A screenshot of the Gedit text editor interface. The window title is "test.py" located at "~/work/tmp". The file contains the following Python code:

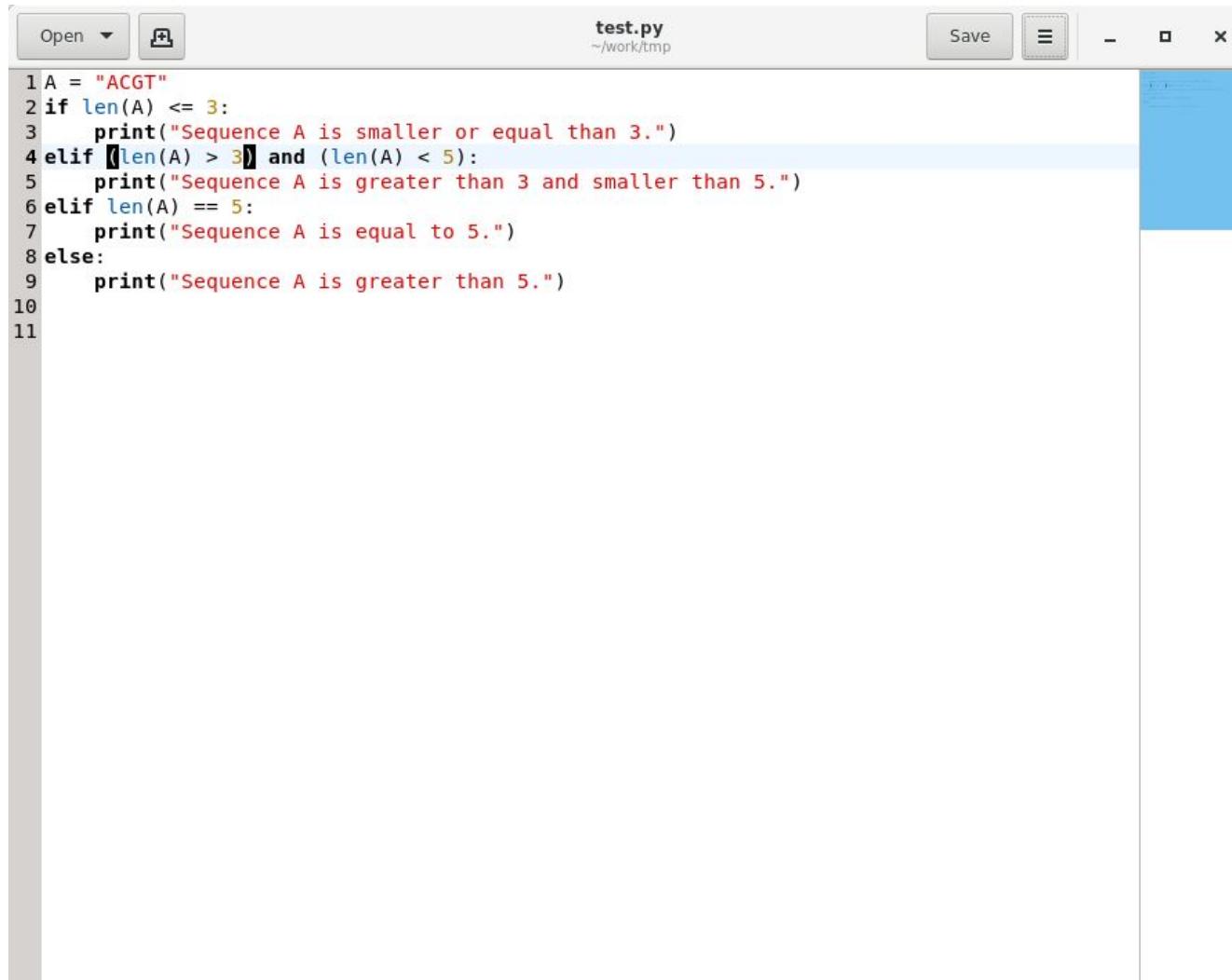
```
A = "ACGT"
if len(A) <= 3:
    print("Sequence A is smaller or equal than 3.")
elif (len(A) > 3) and (len(A) < 5):
    print("Sequence A is greater than 3 and smaller than 5.")
elif len(A) == 5:
    print("Sequence A is equal to 5.")
else:
    print("Sequence A is greater than 5.)
```

The status bar at the bottom shows "Python" as the language, "Tab Width: 4", "Ln 4, Col 7", and "INS" indicating insert mode.

A screenshot of the Gedit text editor interface. The title bar shows "test.py" and the file path "~/work/tmp". The main window displays the following Python code:

```
1 A = "ACGT"
2 if len(A) <= 3:
3     print("Sequence A is smaller or equal than 3.")
4 elif [len(A) > 3] and (len(A) < 5):
5     print("Sequence A is greater than 3 and smaller than 5.")
6 elif len(A) == 5:
7     print("Sequence A is equal to 5.")
8 else:
9     print("Sequence A is greater than 5.")
10
11
```

The code uses color-coded syntax highlighting. The status bar at the bottom indicates the language is "Python", the tab width is "4", the current line is "Ln 4, Col 7", and the mode is "INS".



A screenshot of the Gedit text editor interface. The title bar shows "test.py" and the file path "~/.work/tmp". The menu bar includes "File", "Edit", "View", "Search", "Tools", and "Help". The toolbar contains icons for "Open", "Save", and "Close". The main code area displays the following Python script:

```
1 A = "ACGT"
2 if len(A) <= 3:
3     print("Sequence A is smaller or equal than 3.")
4 elif [len(A) > 3] and (len(A) < 5):
5     print("Sequence A is greater than 3 and smaller than 5.")
6 elif len(A) == 5:
7     print("Sequence A is equal to 5.")
8 else:
9     print("Sequence A is greater than 5.")
10
11
```

The code uses color-coded syntax highlighting: blue for keywords like `if`, `elif`, `else`, and `print`, red for strings and comments, and black for the sequence variable `A` and its value.

Theme: Kate

A screenshot of the Gedit text editor interface. The window title is "test.py" located at "~/work/tmp". The file contains the following Python code:

```
1 A = "ACGT"
2 if len(A) <= 3:
3     print("Sequence A is smaller or equal than 3.")
4 elif (len(A) > 3) and (len(A) < 5):
5     print("Sequence A is greater than 3 and smaller than 5.")
6 elif len(A) == 5:
7     print("Sequence A is equal to 5.")
8 else:
9     print("Sequence A is greater than 5.")
10
11
```

The code uses syntax highlighting where keywords like "if", "elif", and "else" are in blue, and strings are in green. The file is currently in Python mode, as indicated by the status bar at the bottom which shows "Python" and "Tab Width: 4". The status bar also indicates the current line and column as "Ln 4, Col 7".

Theme: Oblivion

A screenshot of the Gedit text editor interface. The window title is "test.py" located at "~/work/tmp". The file contains the following Python code:

```
1 A = "ACGT"
2 if len(A) <= 3:
3     print("Sequence A is smaller or equal than 3.")
4 elif (len(A) > 3) and (len(A) < 5):
5     print("Sequence A is greater than 3 and smaller than 5.")
6 elif len(A) == 5:
7     print("Sequence A is equal to 5.")
8 else:
9     print("Sequence A is greater than 5.")
10
11
```

The code uses Monokai syntax highlighting, where keywords like "if", "elif", and "else" are colored red, and strings are in green. The editor's status bar at the bottom shows "Python" as the language, "Tab Width: 4", "Ln 4, Col 7", and "INS" indicating insert mode.

Theme: Monokai

- Menu - Preferences - View
 - Display line numbers
 - Highlight current line
 - Highlight matching brackets
- Menu - Preferences - Editor
 - Tab width 4
 - Insert spaces instead of tabs
- Menu - Preferences - Fonts & Colors
 - Kate or Oblivion

Laboratory time once again!

https://nbisweden.github.io/workshop-ngsintro/2511/topics/linux/lab_linux_advanced.html