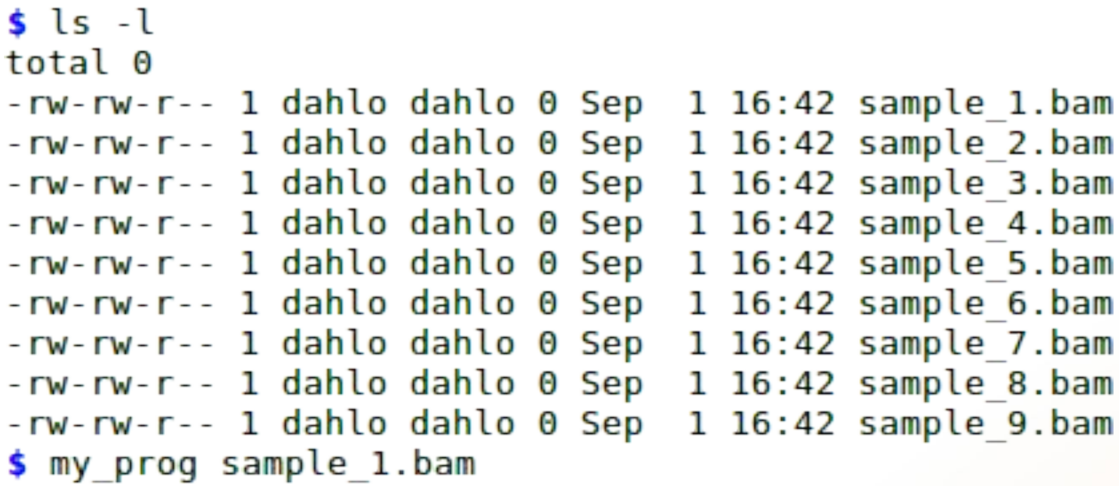
Advanced Linux Usage

2026-03-03

Martin Dahlö
martin.dahlo@nbis.se

Multiple files

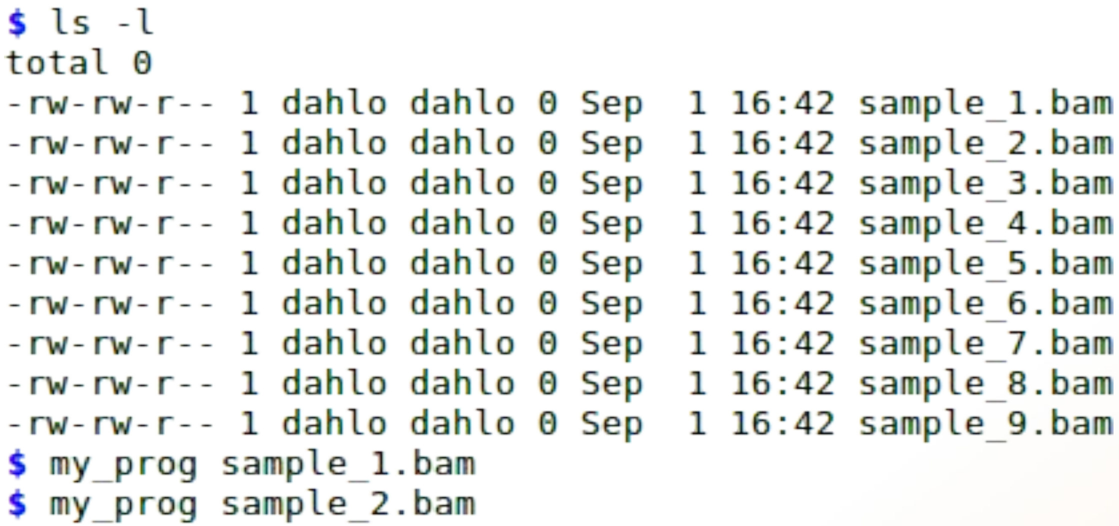
- Same program, many files



```
$ 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
```

Multiple files

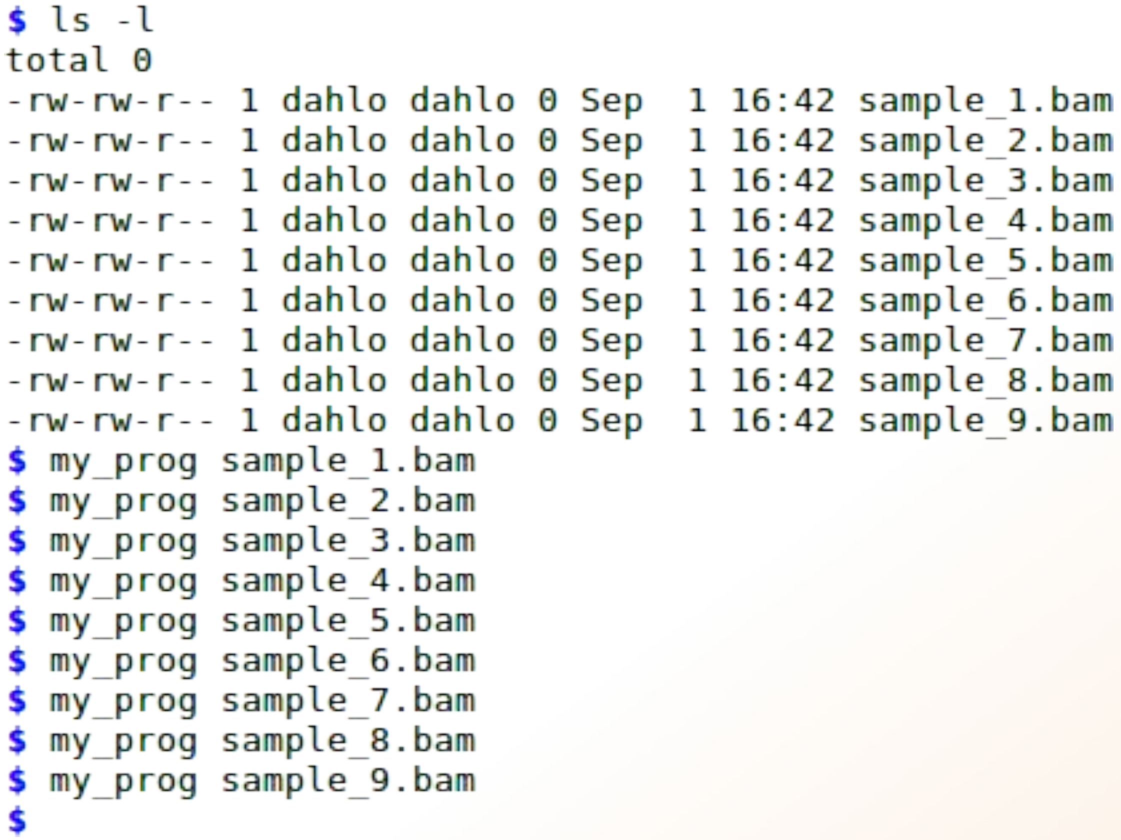
- Same program, many files



```
$ 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
```

Multiple files

- 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

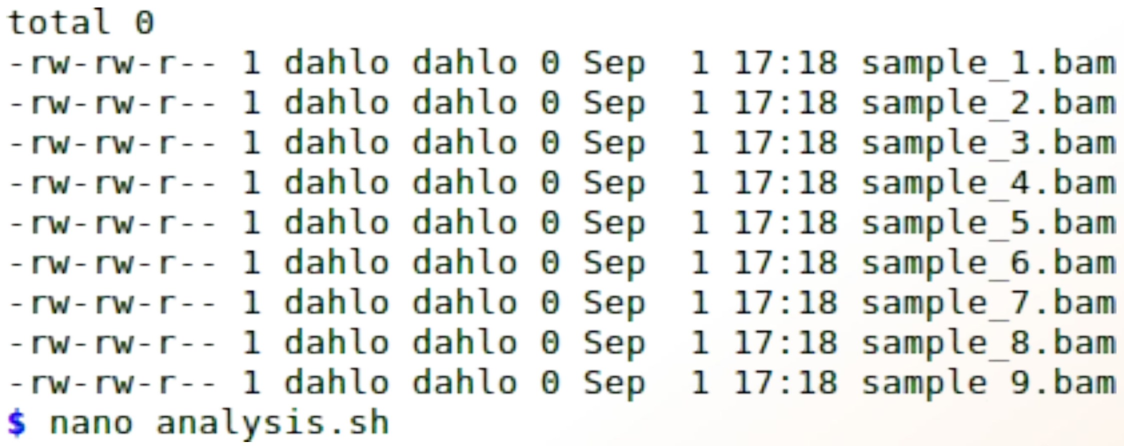
- Same program, many files
  - 10 files? Ok
  - 1000 files? Not ok
- Reproducibility
  - Self and others

Multiple files

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

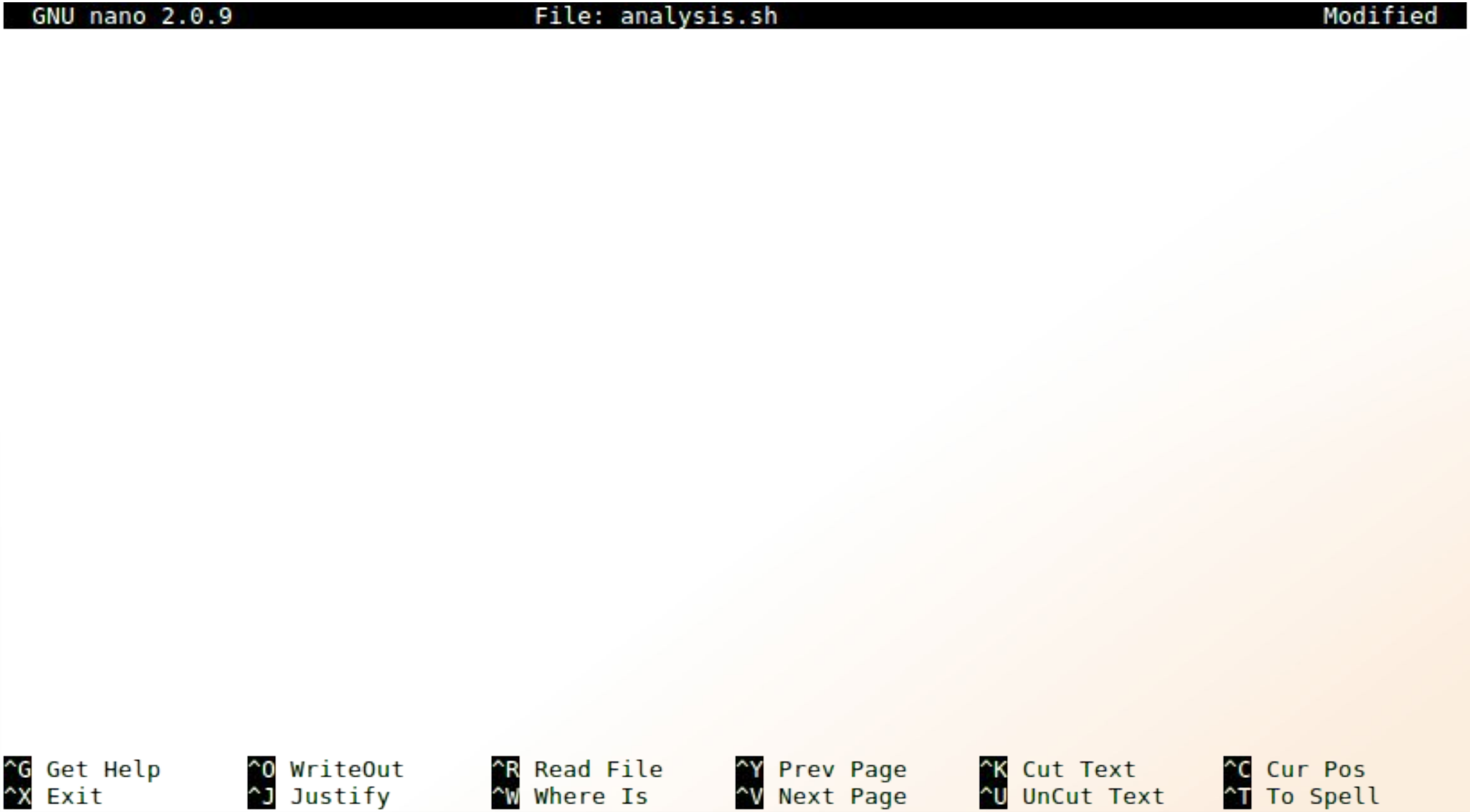
A solution - write a script!

Basic script



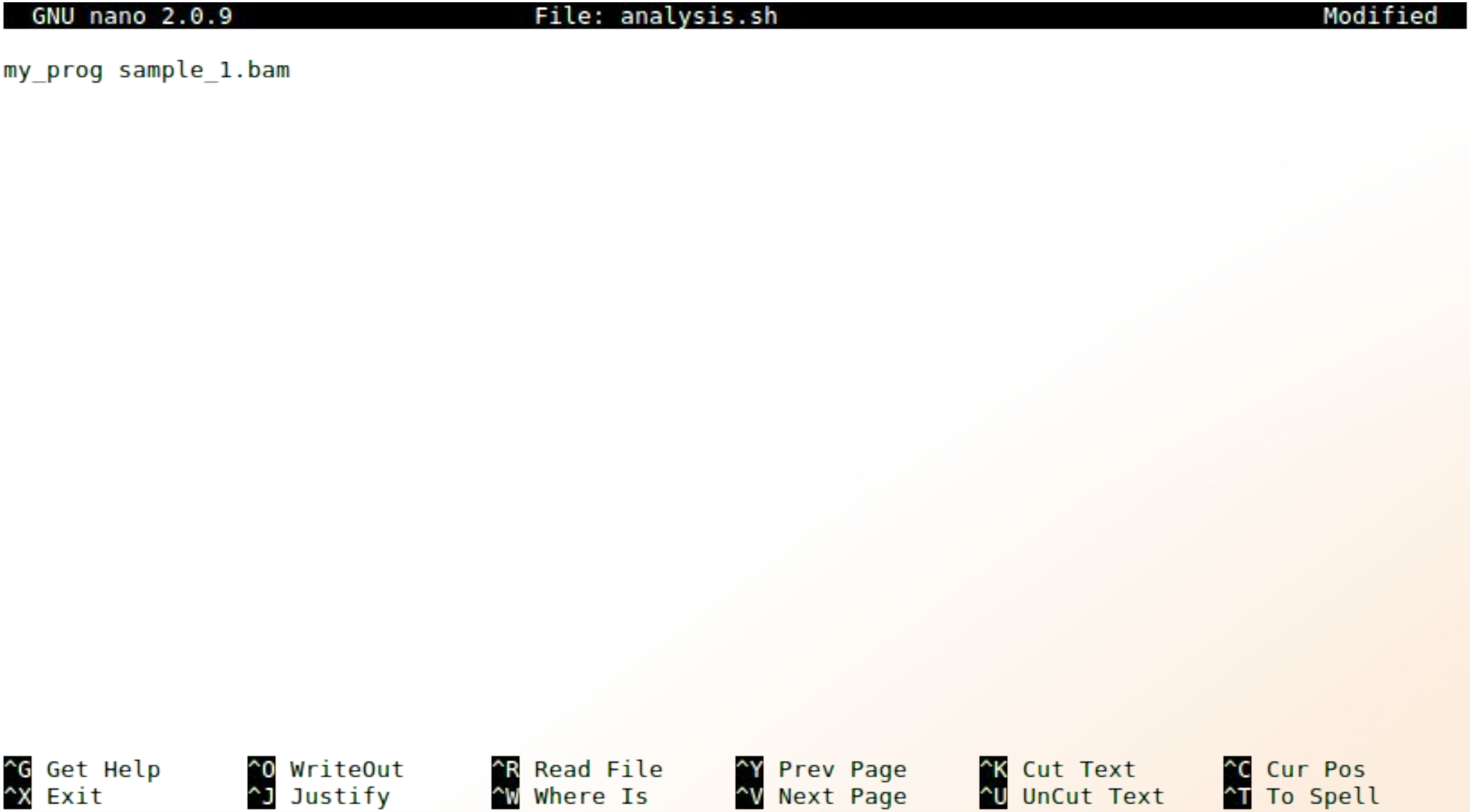
```
total 0
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_4.bam
-rw-rw-r-- 1 dahlo dahlo 0 Sep  1 17:18 sample_5.bam
-rw-rw-r-- 1 dahlo dahlo 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



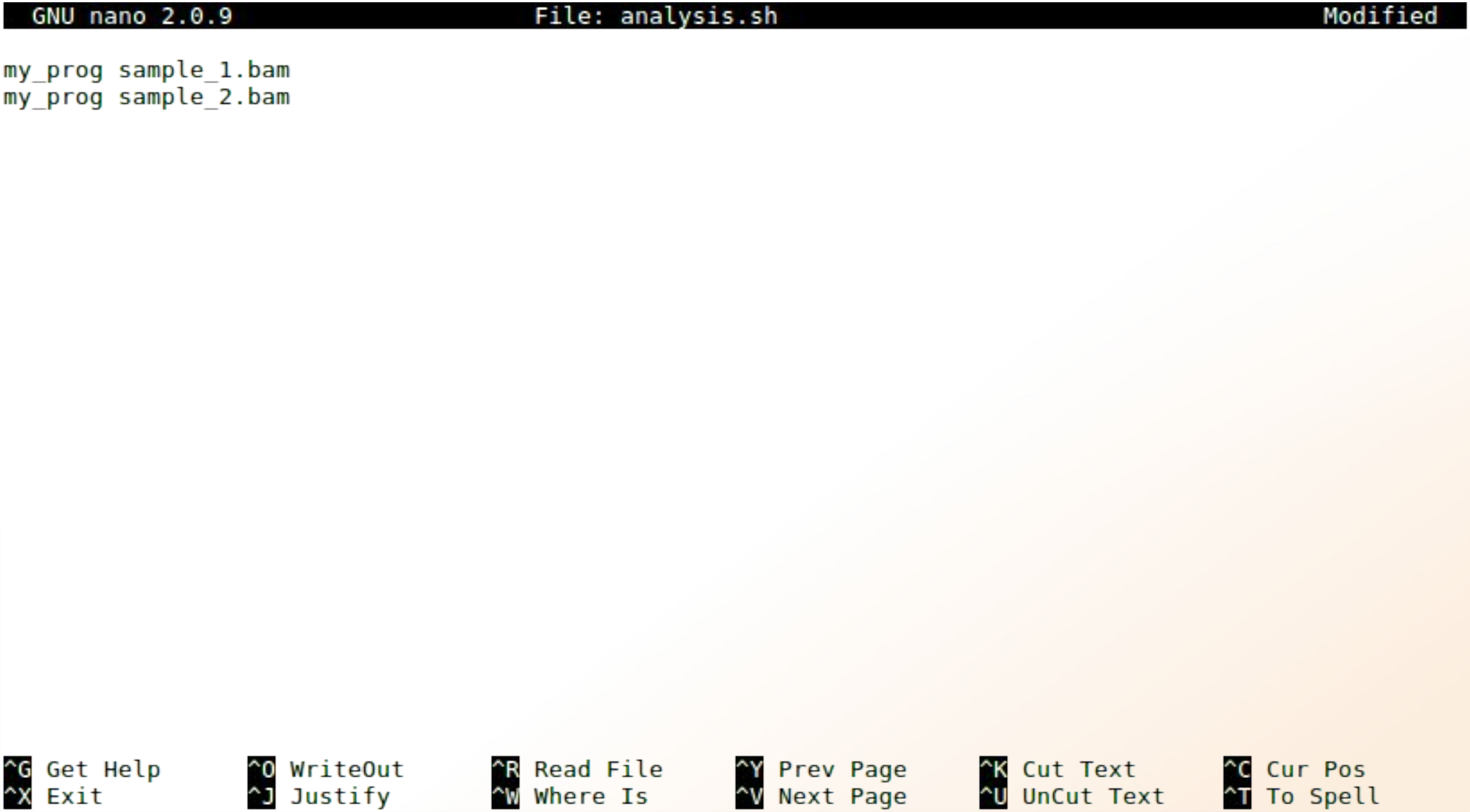
```
  GNU nano 2.0.9                  File: analysis.sh                                       Modified









^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

Basic script



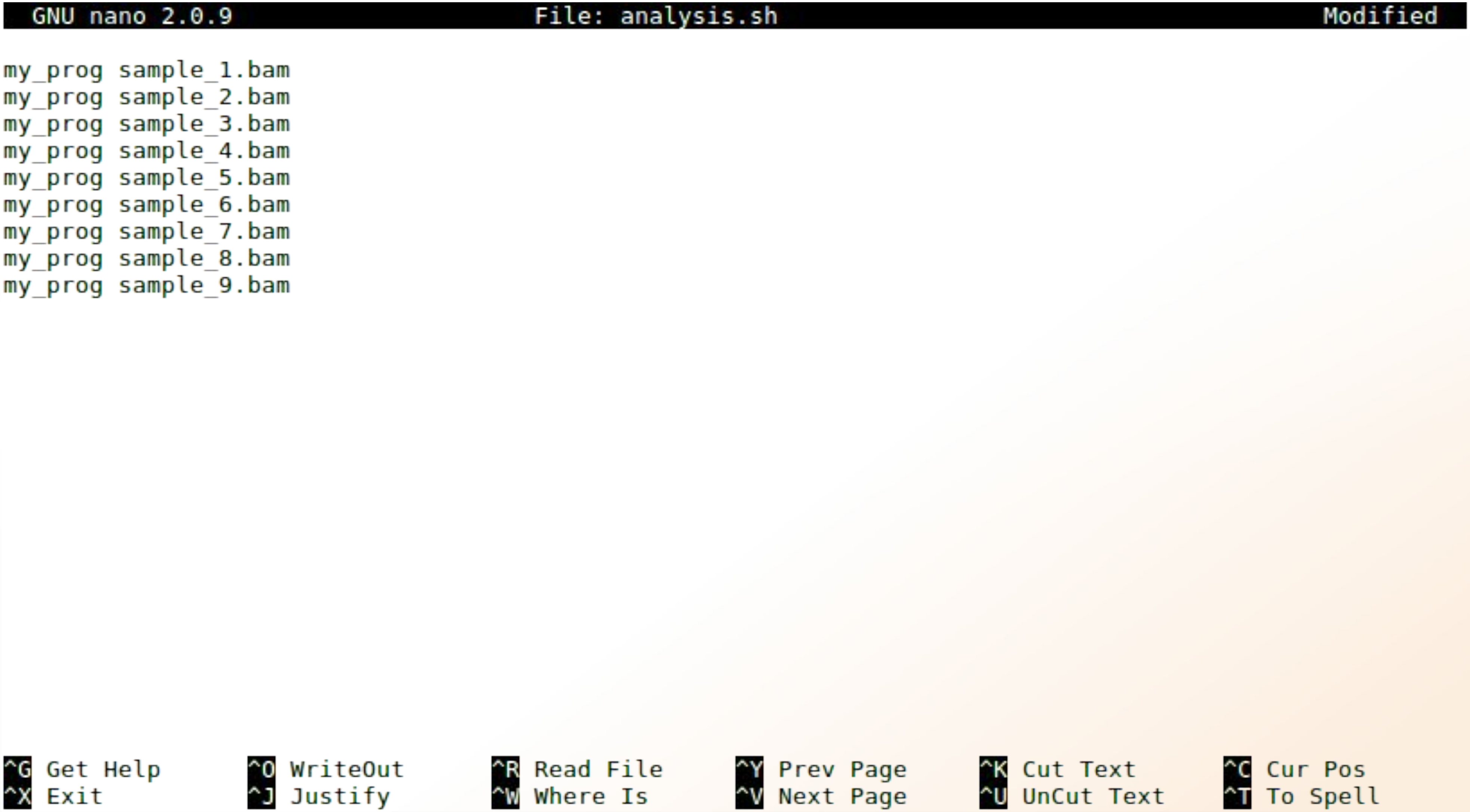
```
  GNU nano 2.0.9                File: analysis.sh                         Modified

my_prog sample_1.bam
















^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

Basic script



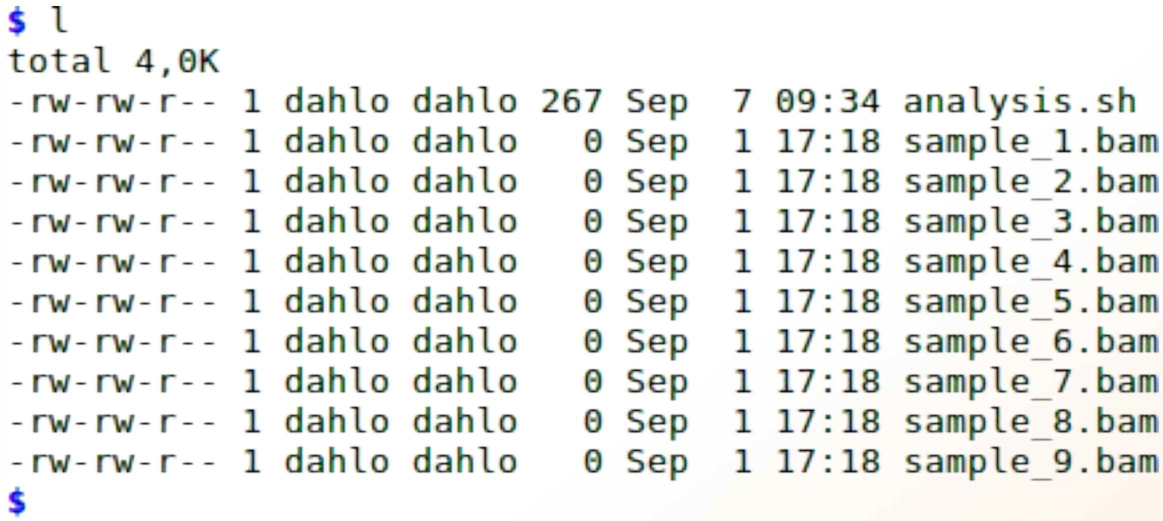
```
my_prog sample_1.bam
my_prog sample_2.bam
```

Basic script



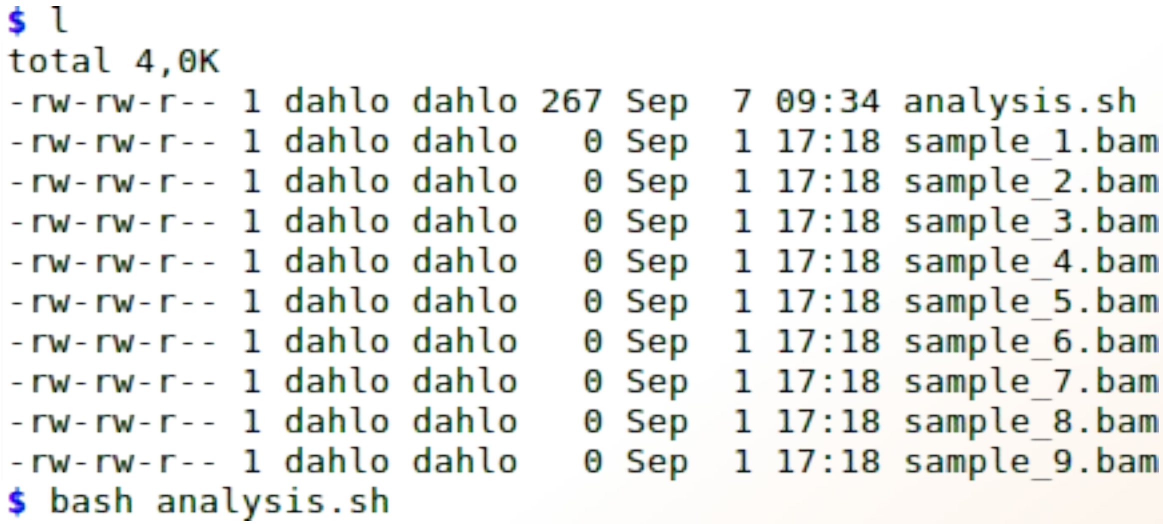
```
  GNU nano 2.0.9                File: analysis.sh                           Modified

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
```

Basic script



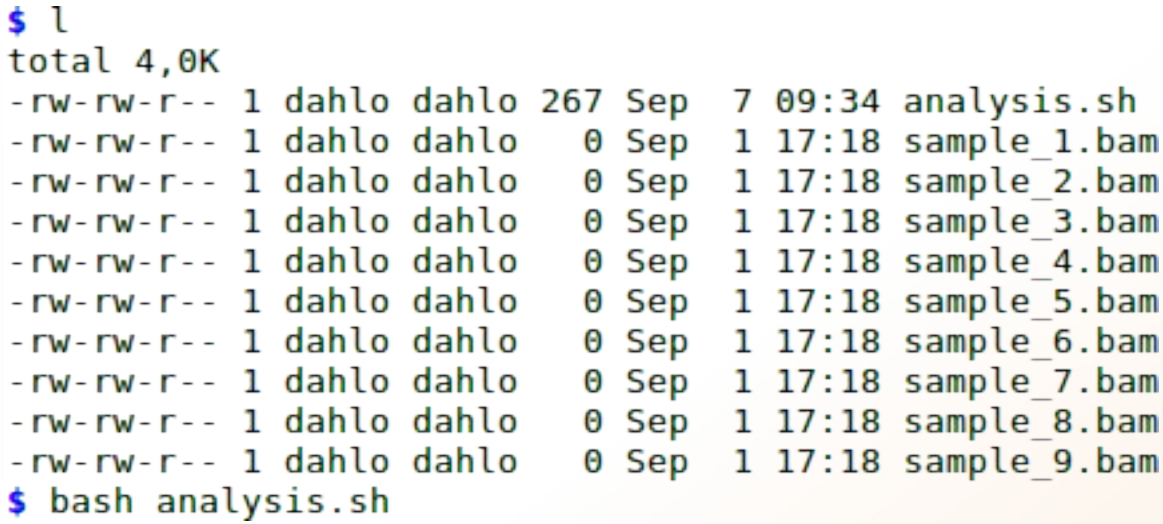
```
$ l
total 4,0K
-rw-rw-r-- 1 dahlo dahlo 267 Sep  7 09:34 analysis.sh
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_1.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_2.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_3.bam
-rw-rw-r-- 1 dahlo dahlo   0 Sep  1 17:18 sample_4.bam
-rw-rw-r--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

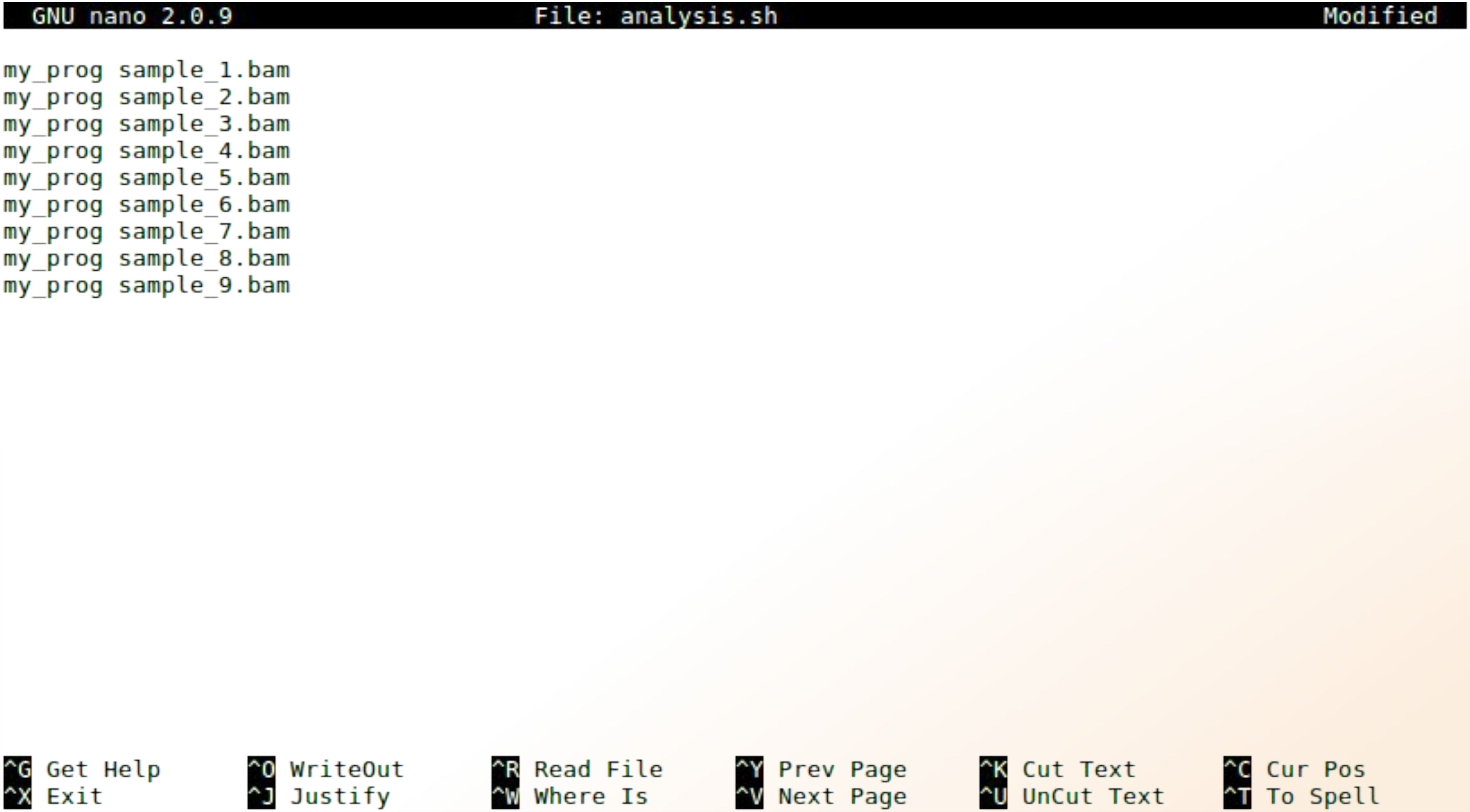
Basic script



```
$ 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
```

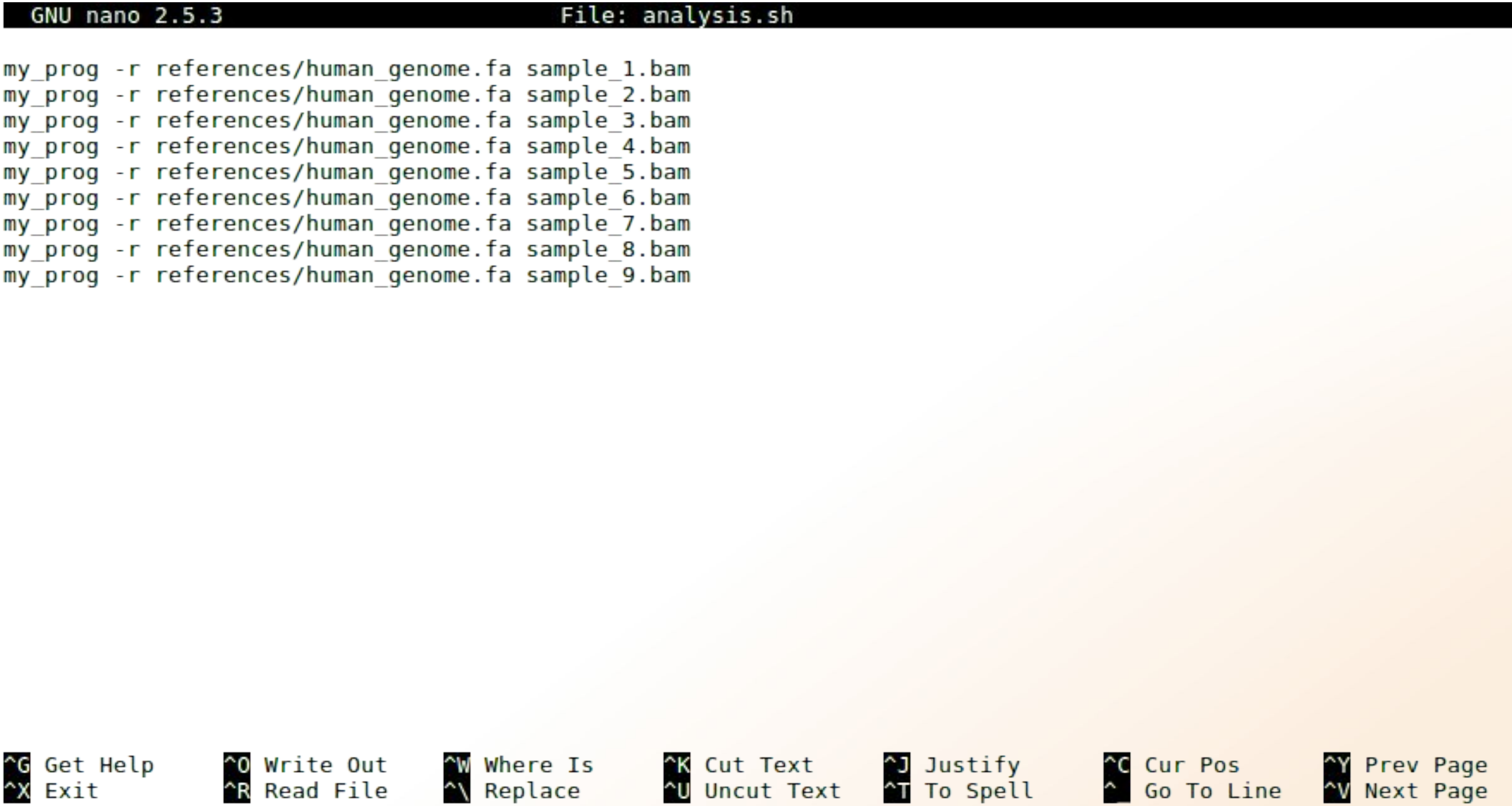
Still not OK for 1000 or more files!

Basic script



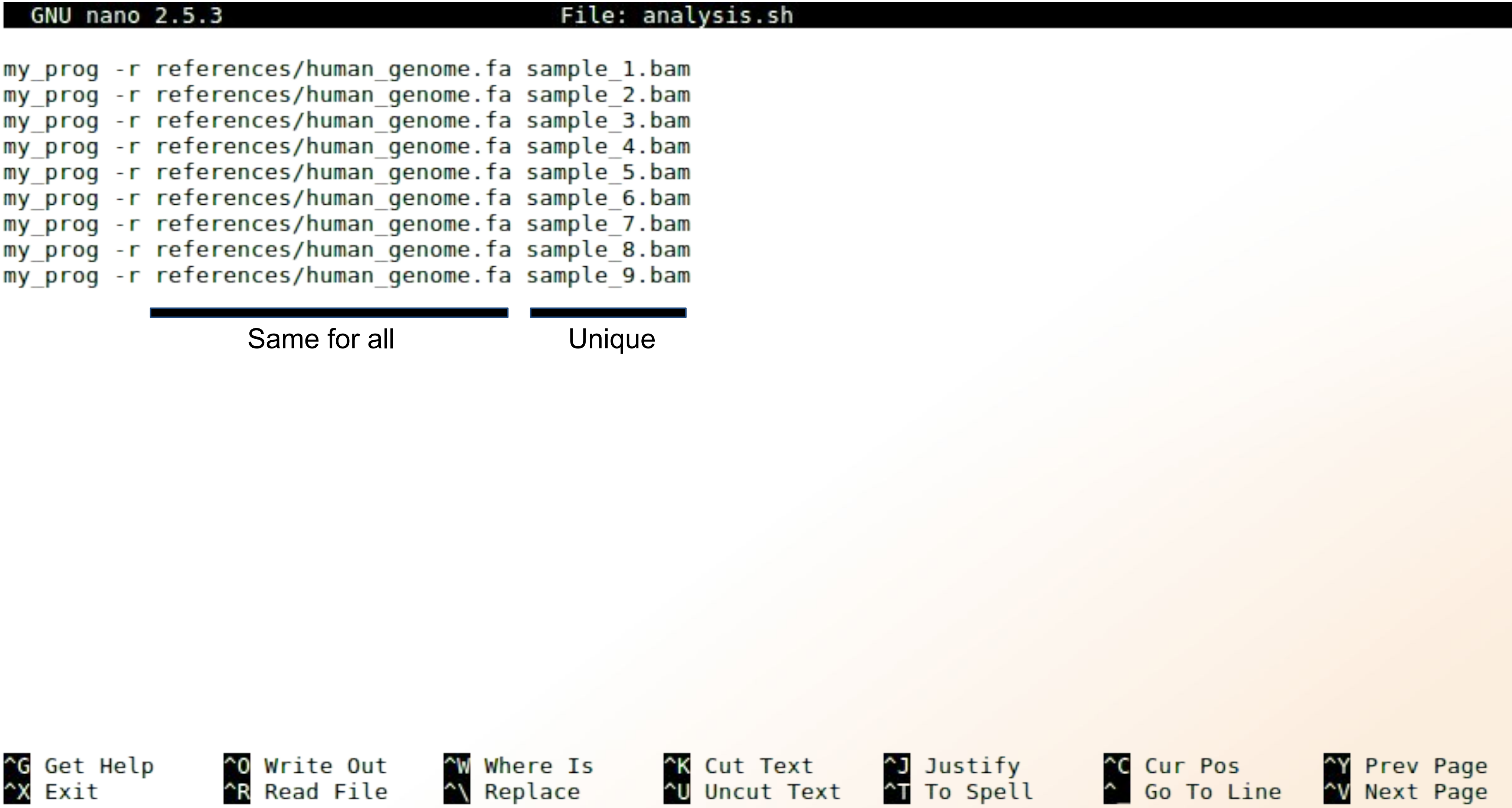
```
  GNU nano 2.0.9              File: analysis.sh                               Modified

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
```

Basic script



```
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
```

Basic script



```
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

Variables

- Assigning

```
my_variable=5
my_variable=”nice text”
```

Variables

- Assigning

```
my_variable=5
my_variable=”nice text”
```

- Using

```
$my_variable
```

Variables

- Assigning

```
my_variable=5
my_variable=”nice text”
```

- Using

```
$my_variable
```

```
$ my_variable=”Martin”
```

Variables

- Assigning

```
my_variable=5
my_variable=”nice text”
```

- Using

```
$my_variable
```

```
$ my_variable=”Martin”
$ echo “Hello $my_variable.”
```

Variables

- Assigning

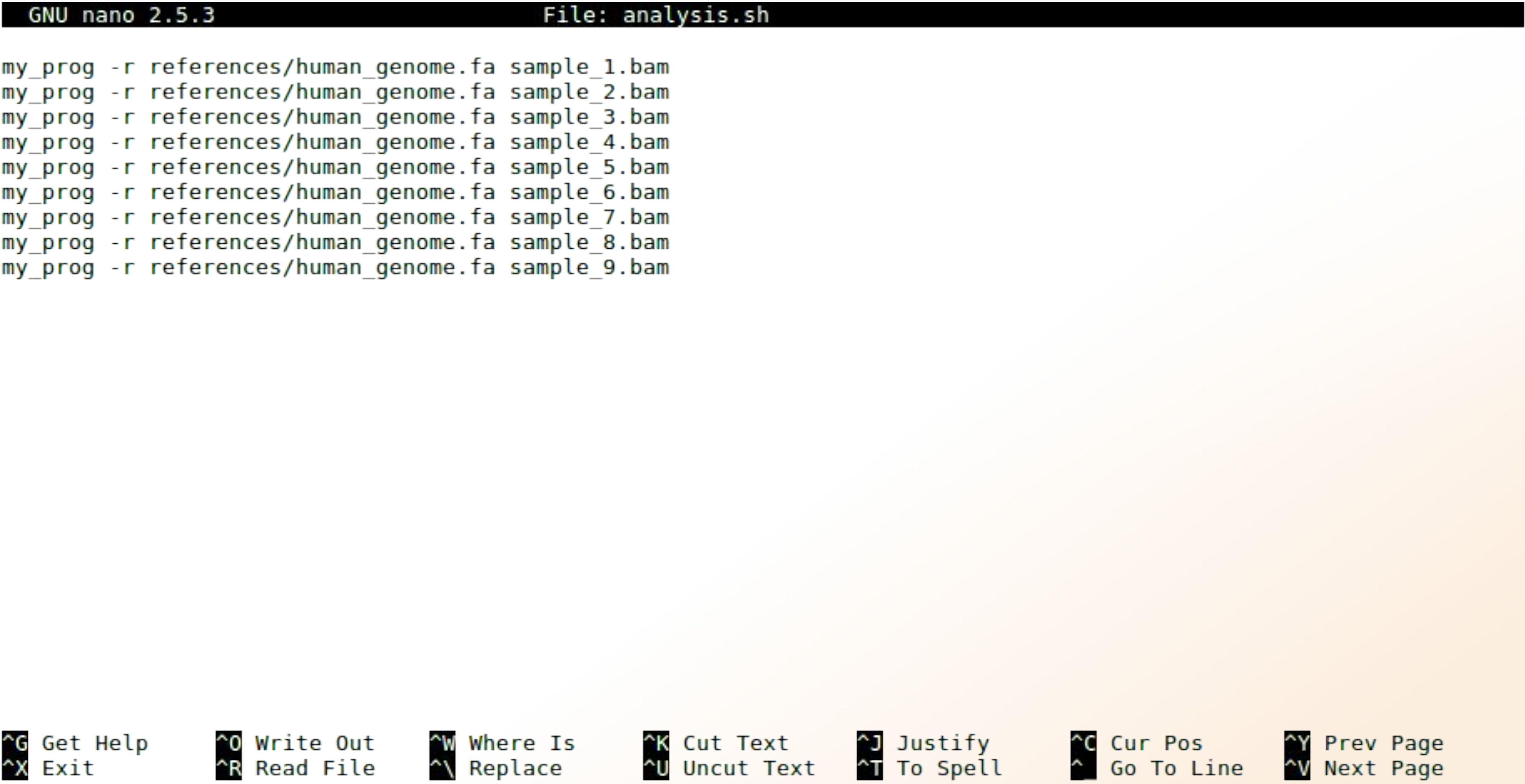
```
my_variable=5
my_variable=”nice text”
```

- Using

```
$my_variable
```

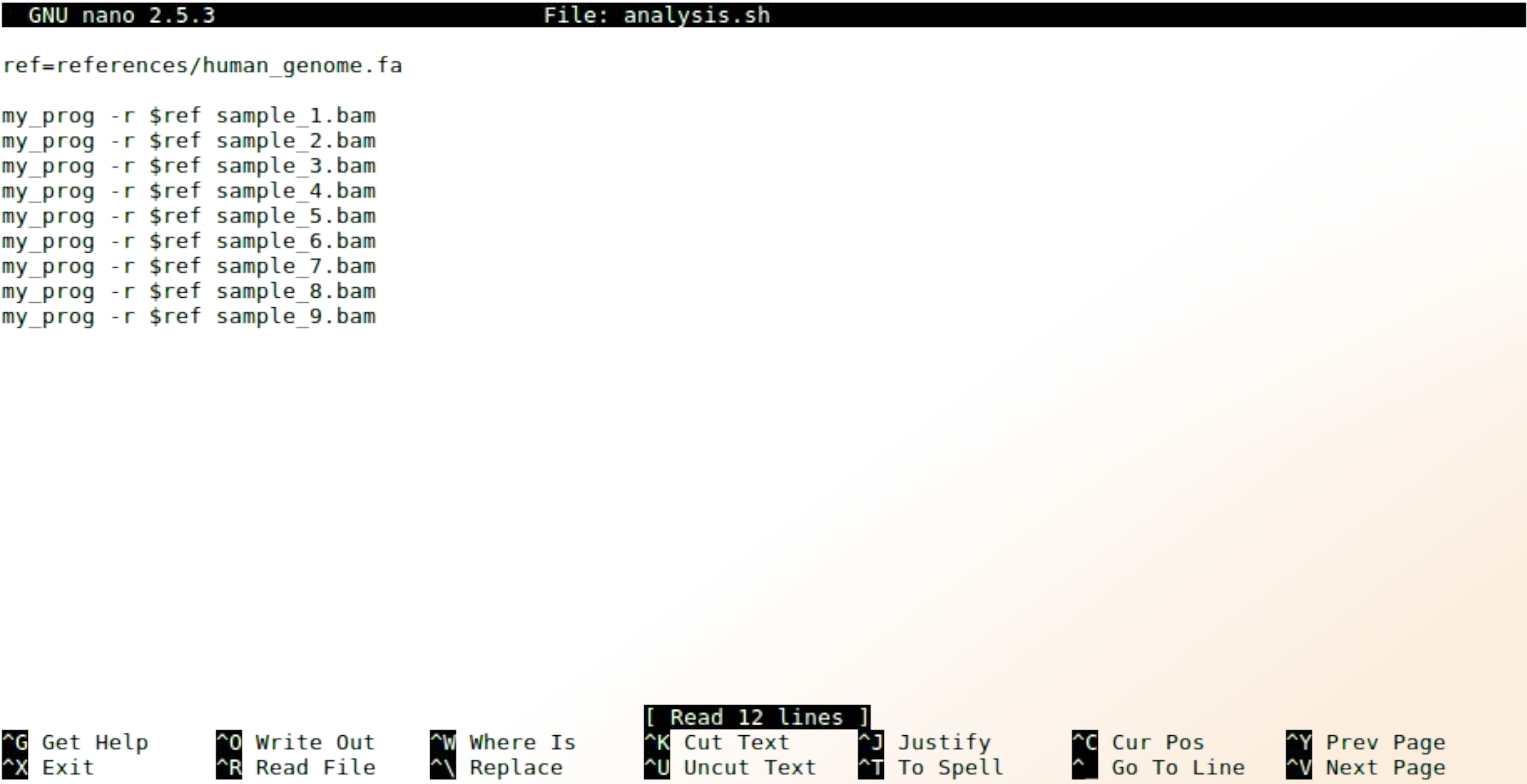
```
$ my_variable=”Martin”
$ echo “Hello $my_variable.”
Hello Martin.
```

Basic script



```
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
```

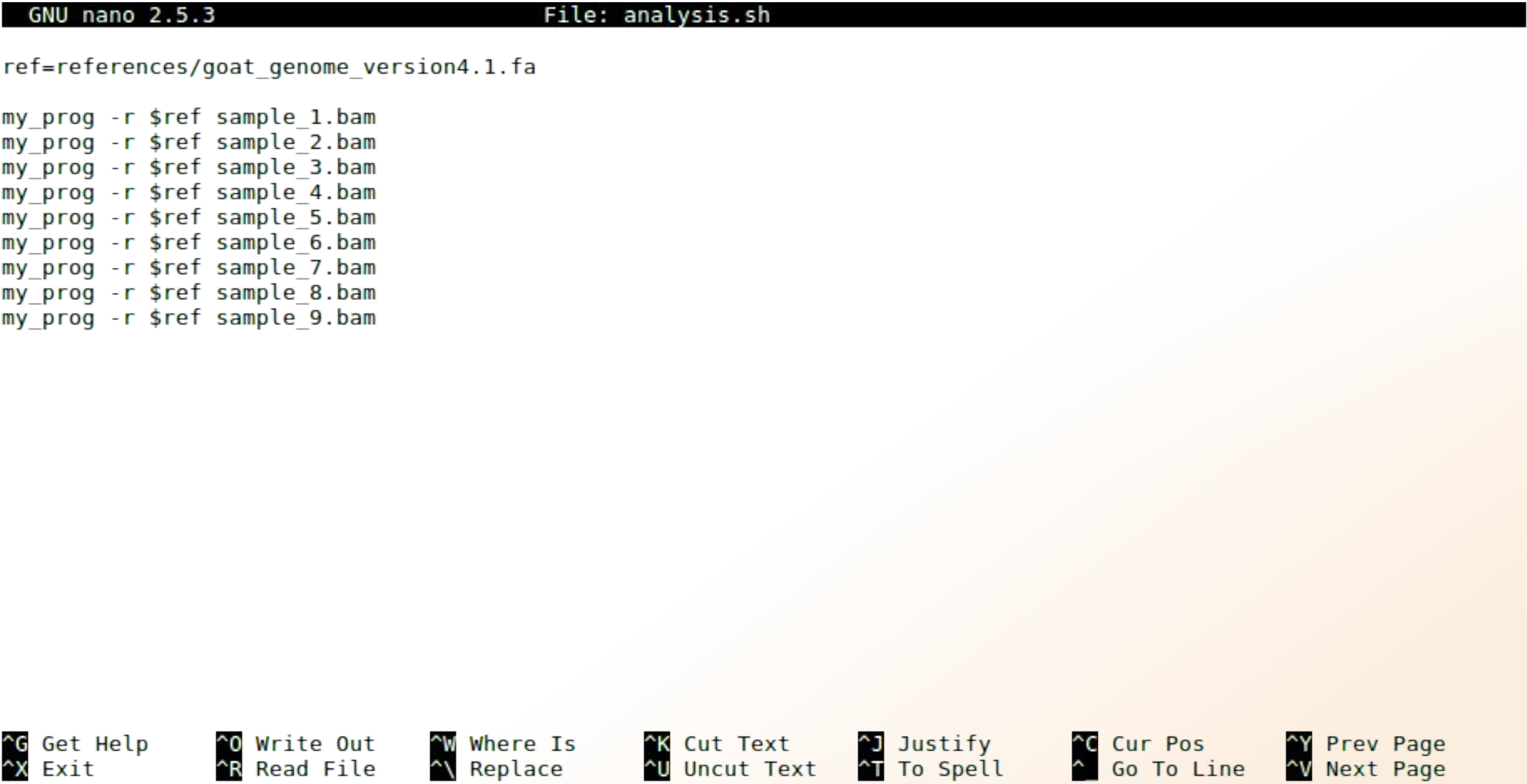
Basic script



```
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
```

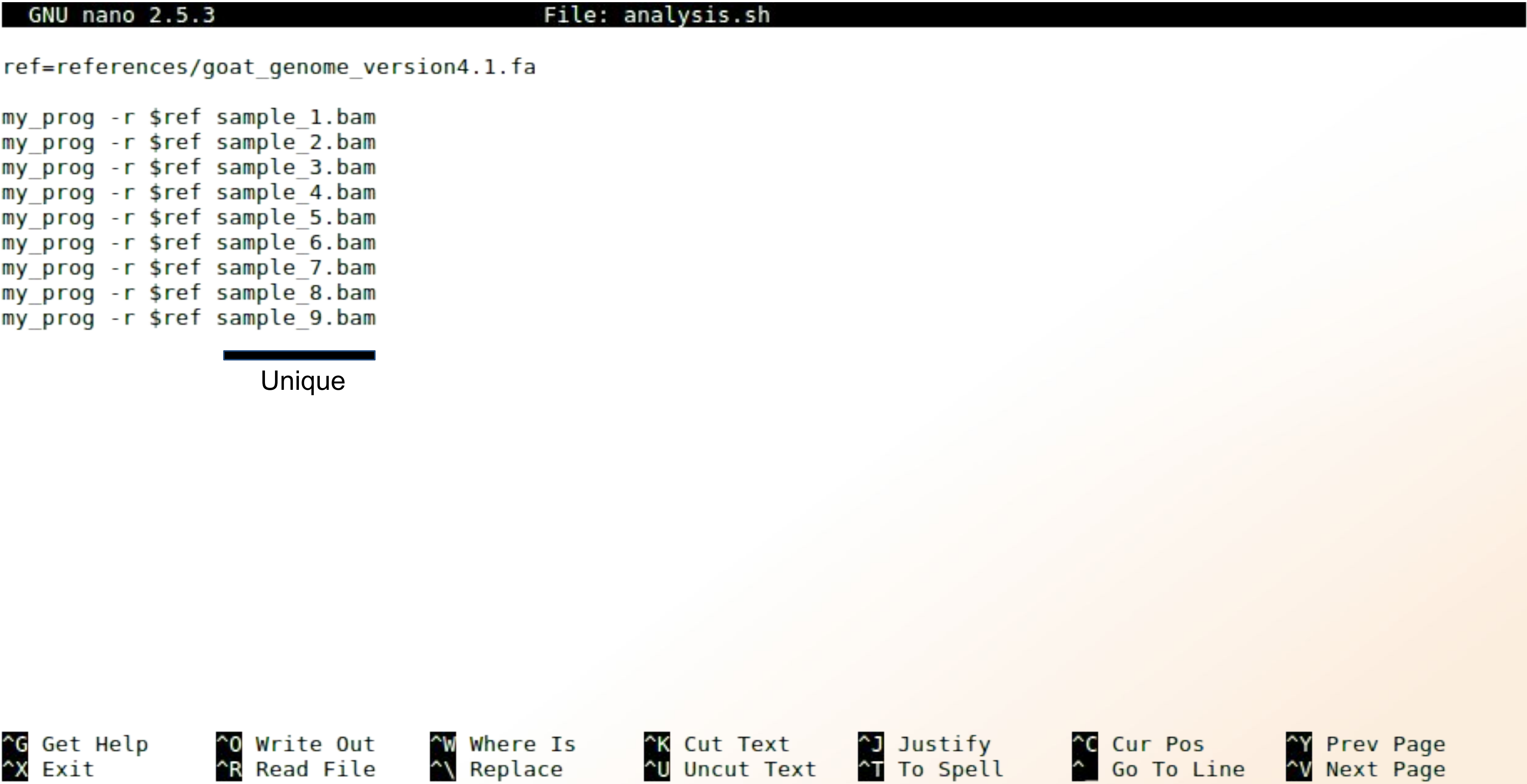
```
[ Read 12 lines ]
```

Basic script



```
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
```

Basic script

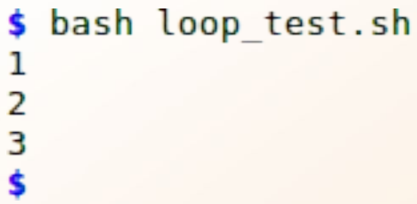


```
 GNU nano 2.5.3                    File: analysis.sh

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

Loops

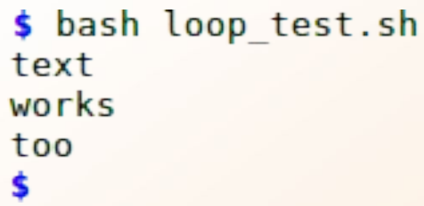
```
for var in 1 2 3;
do
    echo $var
done
```



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

Loops

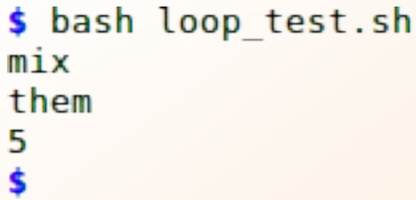
```
for var in text works too;
do
    echo $var
done
```



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

Loops

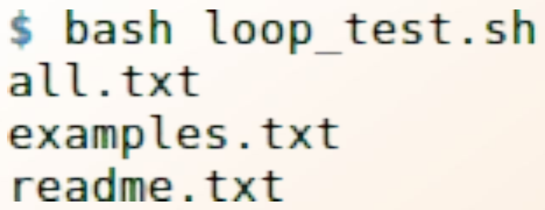
```
for var in mix them 5;
do
    echo $var
done
```



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

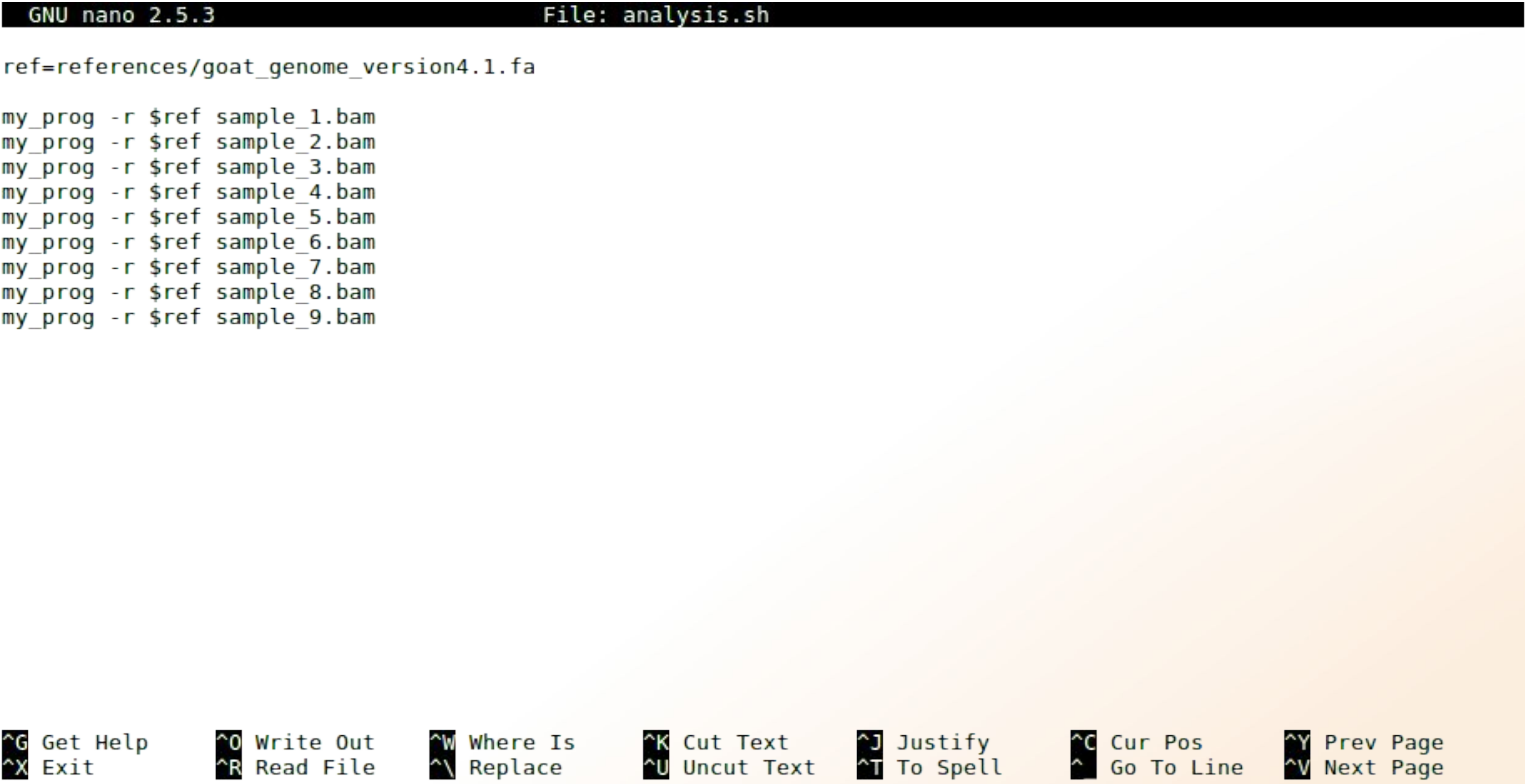
Loops

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



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

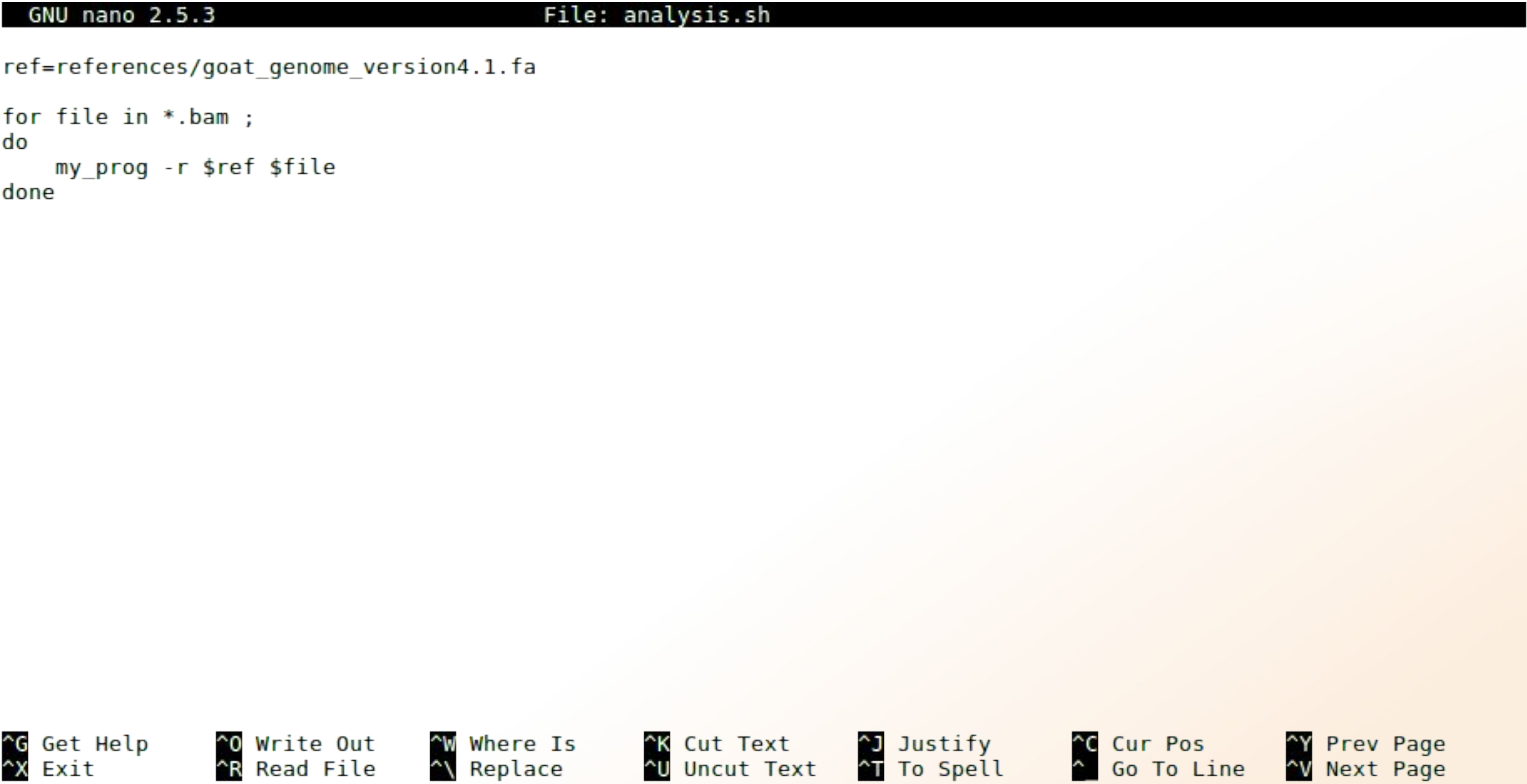
Basic script



```
  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     ^Y Prev Page
^X Exit        ^R Read File   ^\ 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

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     ^Y Prev Page
^X Exit        ^R Read File   ^\ 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

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

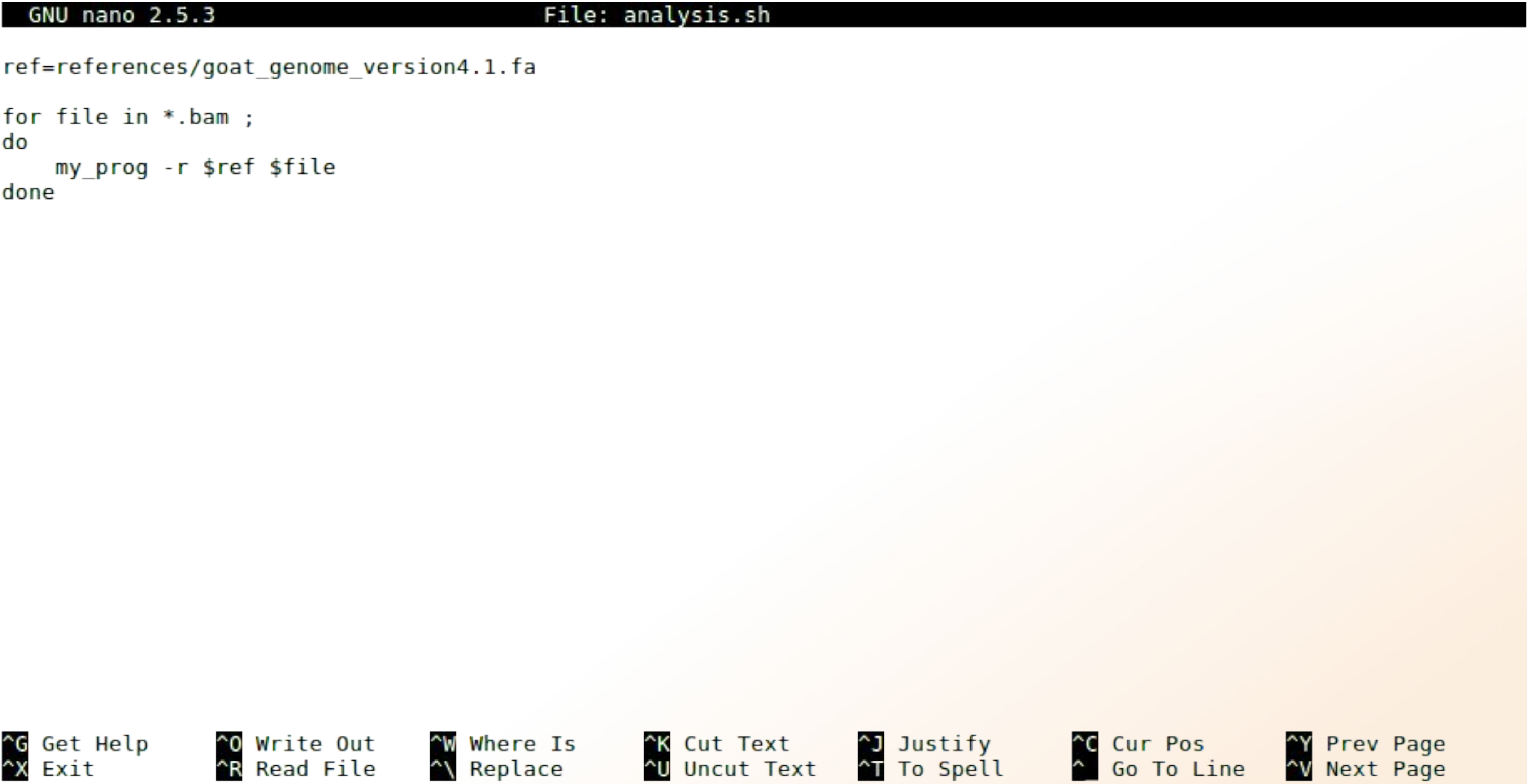
Debugging!

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
```

Basic script

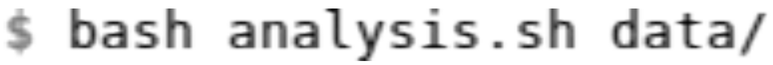


```
ref=references/goat_genome_version4.1.fa

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

Arguments

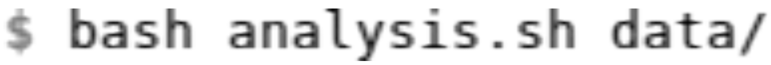
```
$ bash analysis.sh
```

Arguments



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

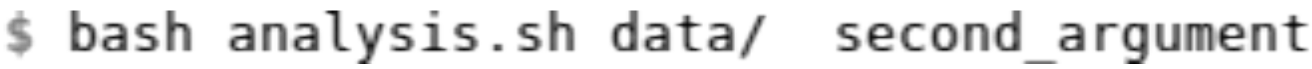
Arguments



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

$1

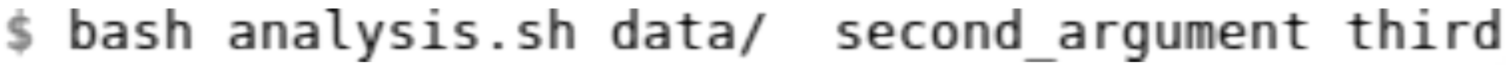
Arguments



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

$1 $2

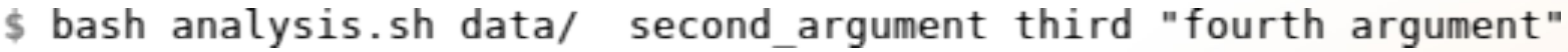
Arguments



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

$1 $2 $3

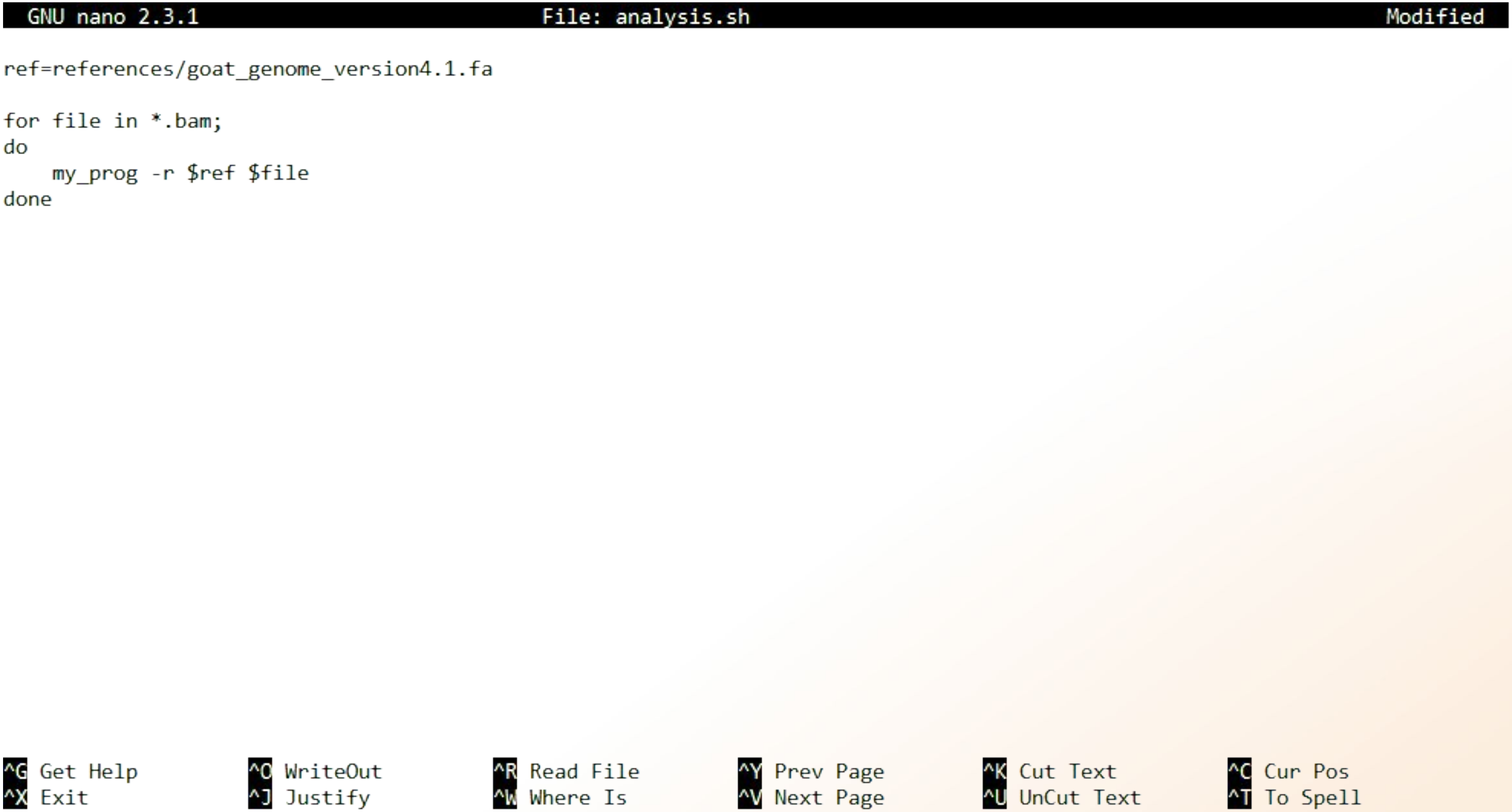
Arguments



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

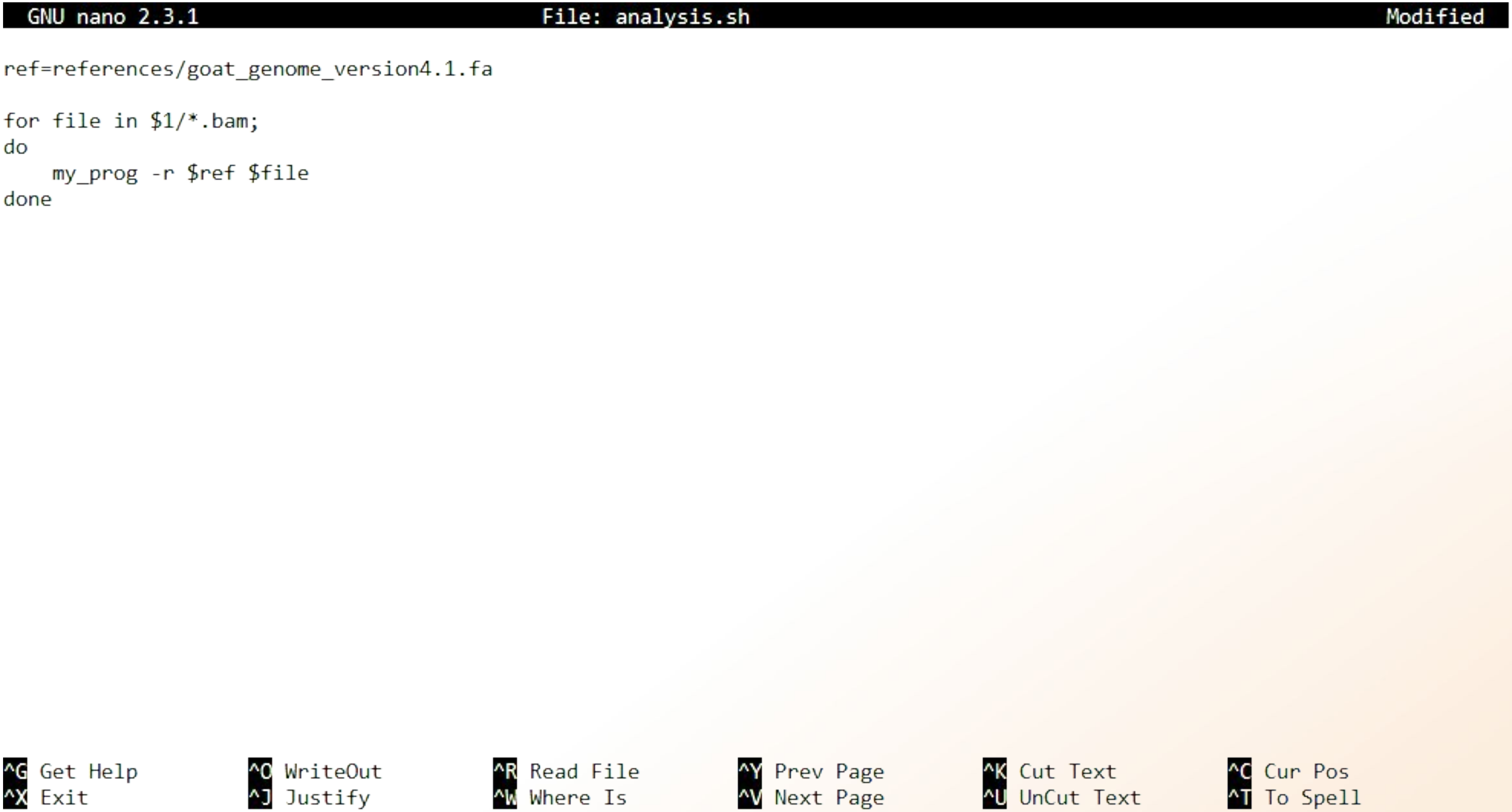
$1 $2 $3 $4

Arguments



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

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

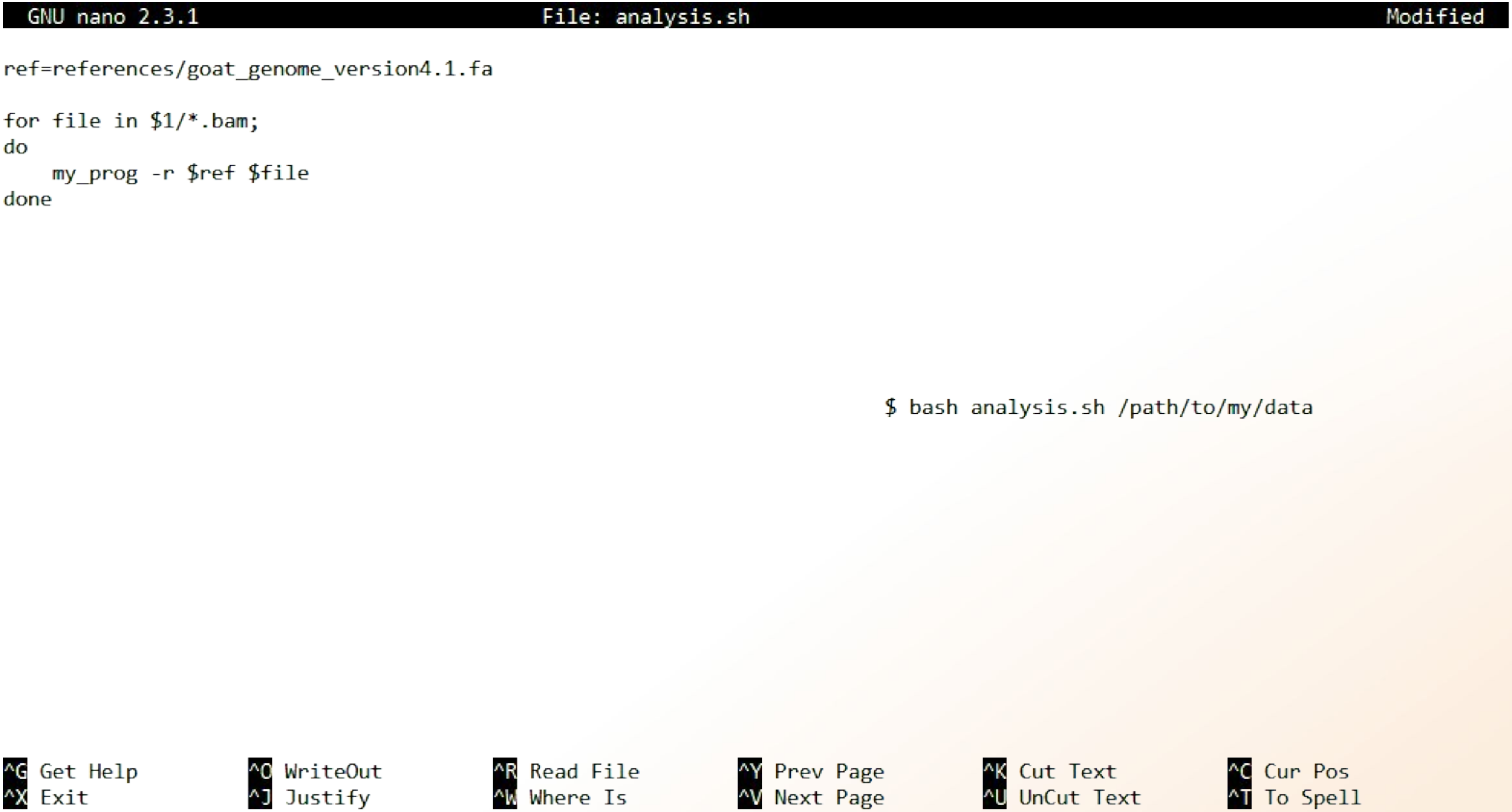
Arguments



```
  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     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

Arguments

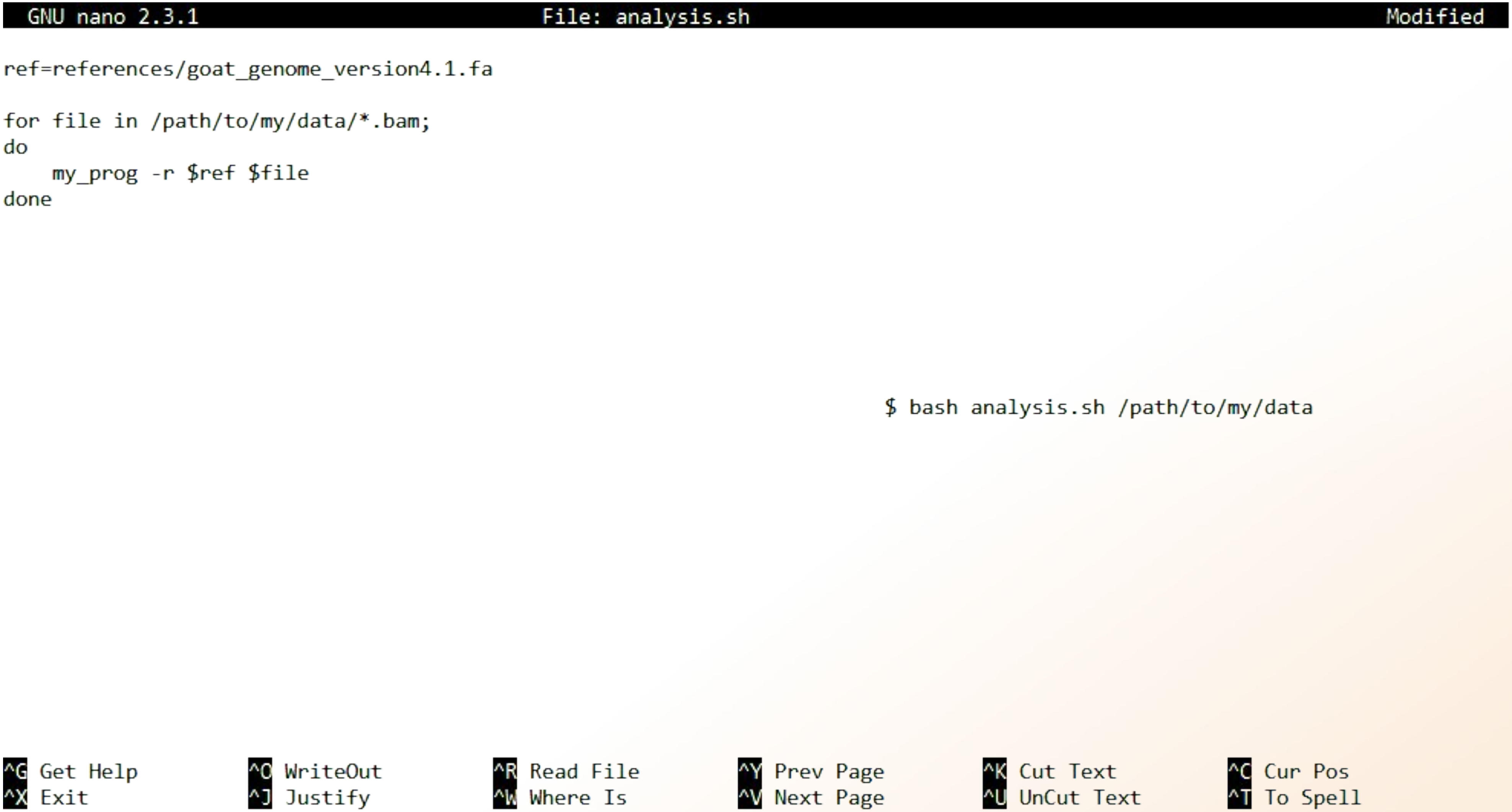


```
  GNU nano 2.3.1                    File: analysis.sh                                          Modified

ref=references/goat_genome_version4.1.fa

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

```
^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

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

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
```

If

- Control statement

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

If

- Control statement

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

result:

```
This is true
```

If

- Control statement

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

result:

If

- Control statement

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

result:

```
This is true
```

If

- Control statement

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

result:

If

- Control statement

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

result:

```
This is true
```

If

- Control statement

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

result:

If

- Control statement

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

result:

```
This is true
```

If

- Control statement

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

result:

If

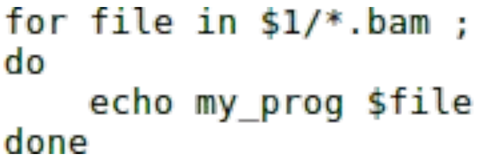
- Control statement (math context)

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

result:

If

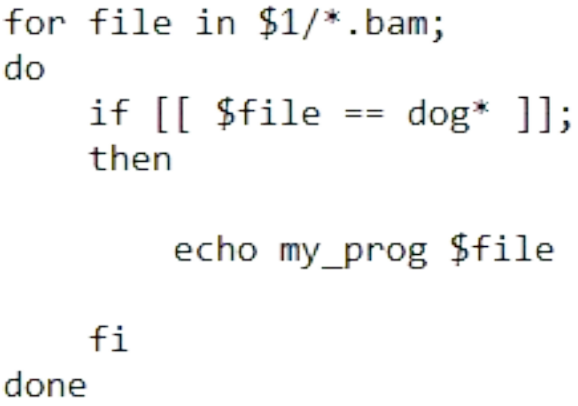
- For all dog samples



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

If

- For all dog samples



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

        echo my_prog $file

    fi
done
```

If

- 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
```

If

- 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
```

If

- 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
```

If

- 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
```

If

- 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
```

If

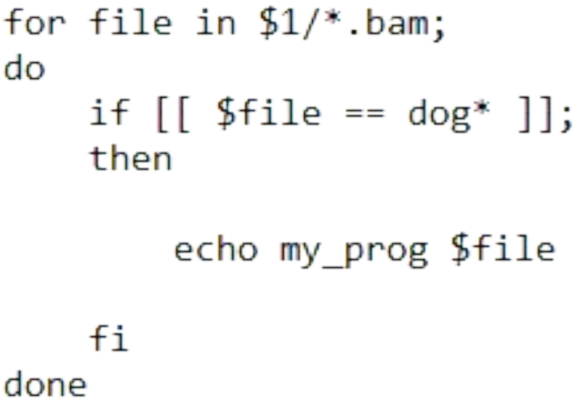
- For all dog samples

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

        echo my_prog $file

    fi
done
```

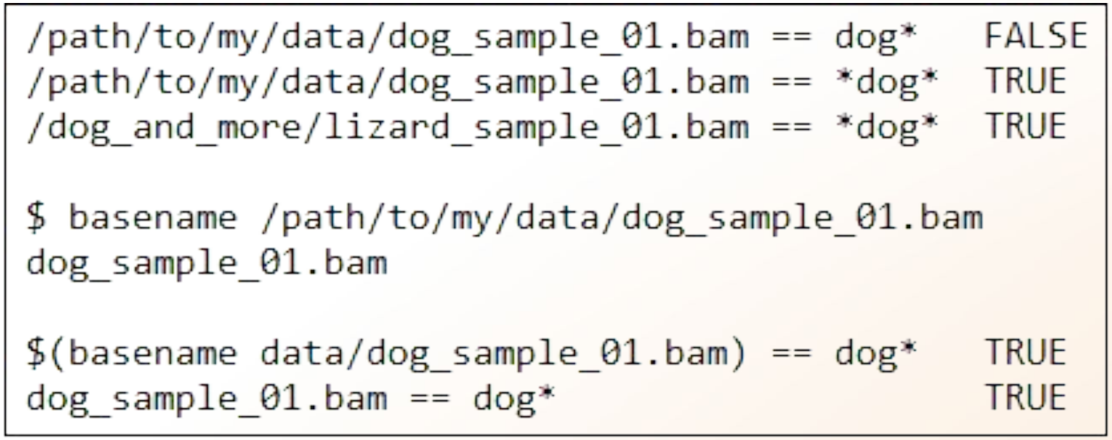
```
/path/to/my/data/dog_sample_01.bam == dog*    FALSE
/path/to/my/data/dog_sample_01.bam == *dog*   TRUE
/dog_and_more/lizard_sample_01.bam == *dog*   TRUE

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

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

If

- For all dog samples



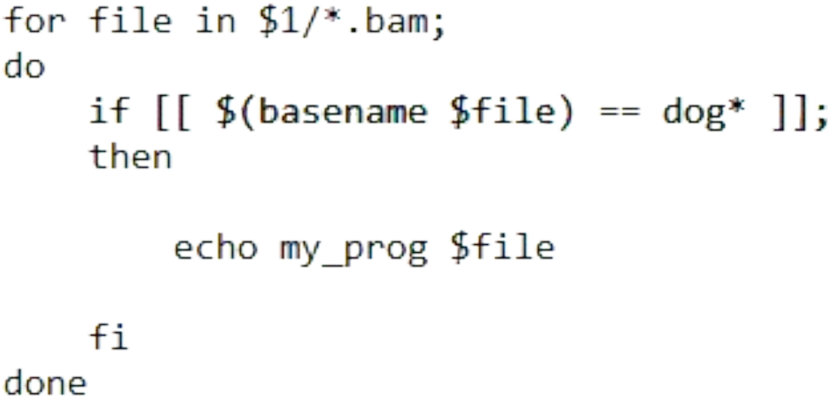
```
for file in $1/*.bam;
do
    if [[ $file == dog* ]];
    then

        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
```

If

- For all dog samples



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

        echo my_prog $file

    fi
done
```

If

- For all dog samples

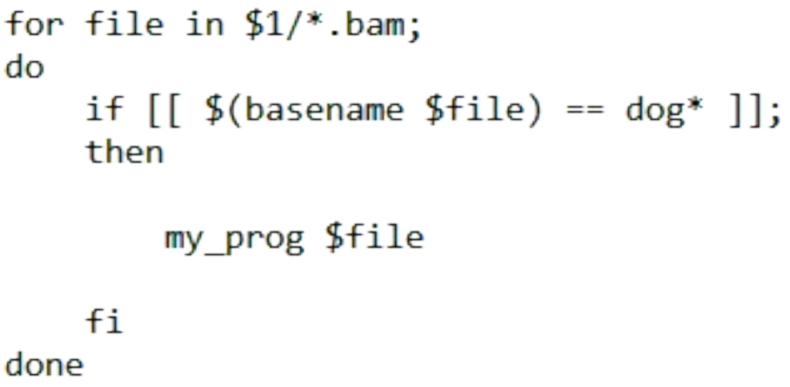
```
for file in $1/*.bam;
do
    if [[ $(basename $file) == dog* ]];
    then

        my_prog $file

    fi
done
```

Different languages

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

Different languages

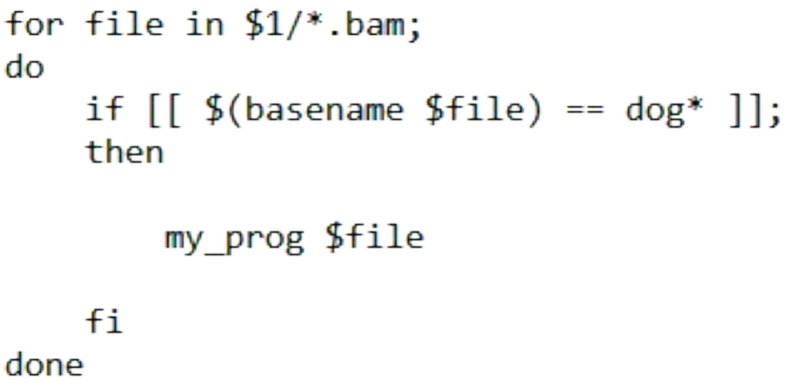
- Programming is programming
    - Perl, Python, **Bash**, and more



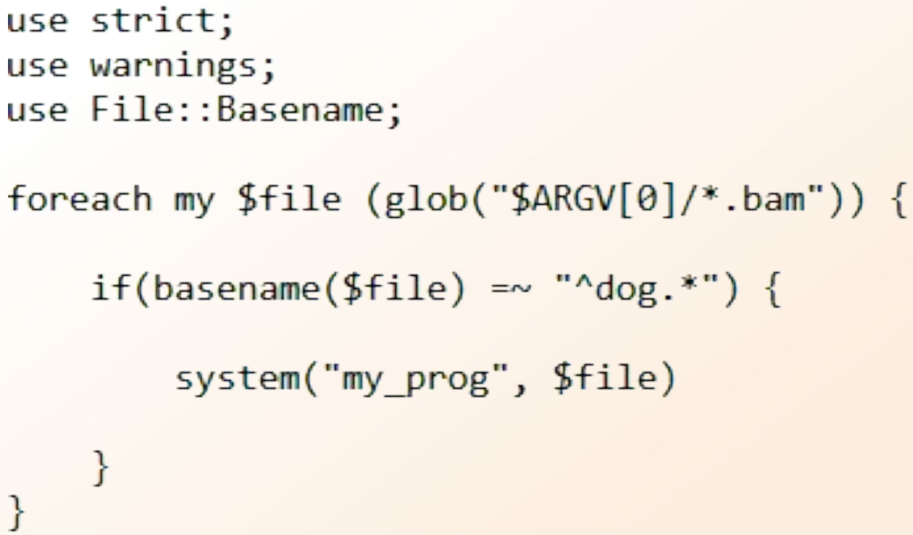
```
for file in $1/*.bam;
do
    if [[ $(basename $file) == dog* ]];
    then

        my_prog $file

    fi
done
```

Different languages

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



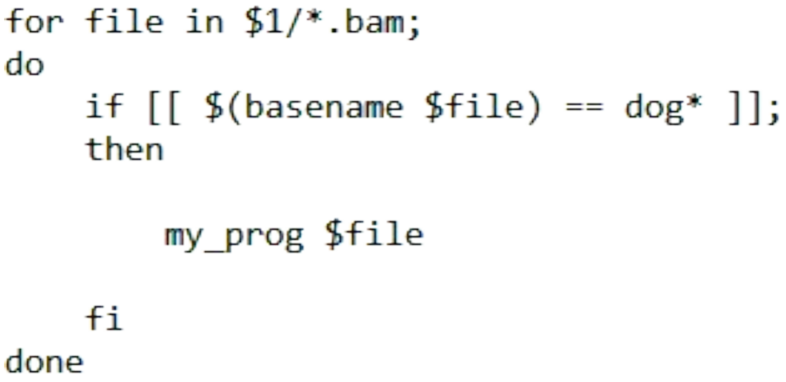
```
for file in $1/*.bam;
do
    if [[ $(basename $file) == dog* ]];
    then

        my_prog $file

    fi
done
```



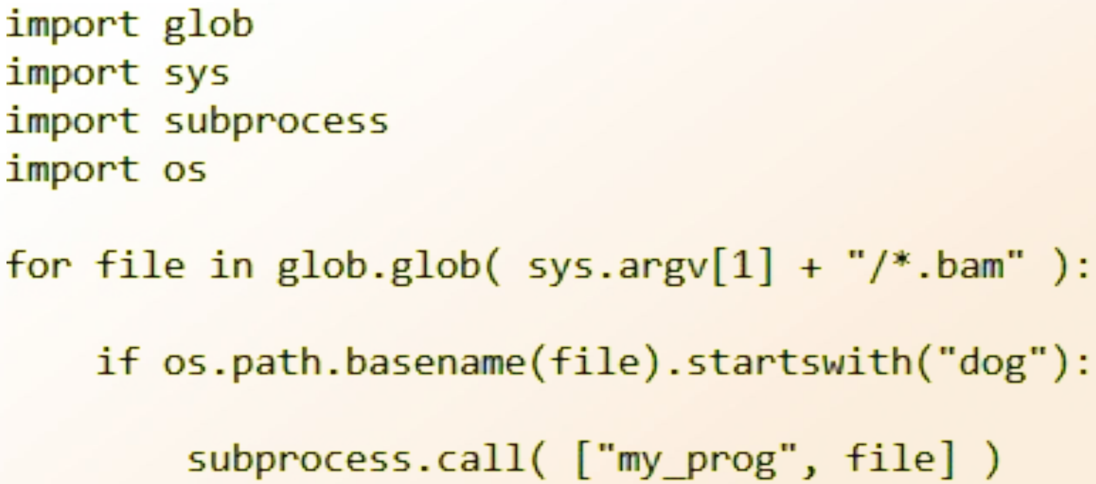
```
use strict;
use warnings;
use File::Basename;

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)

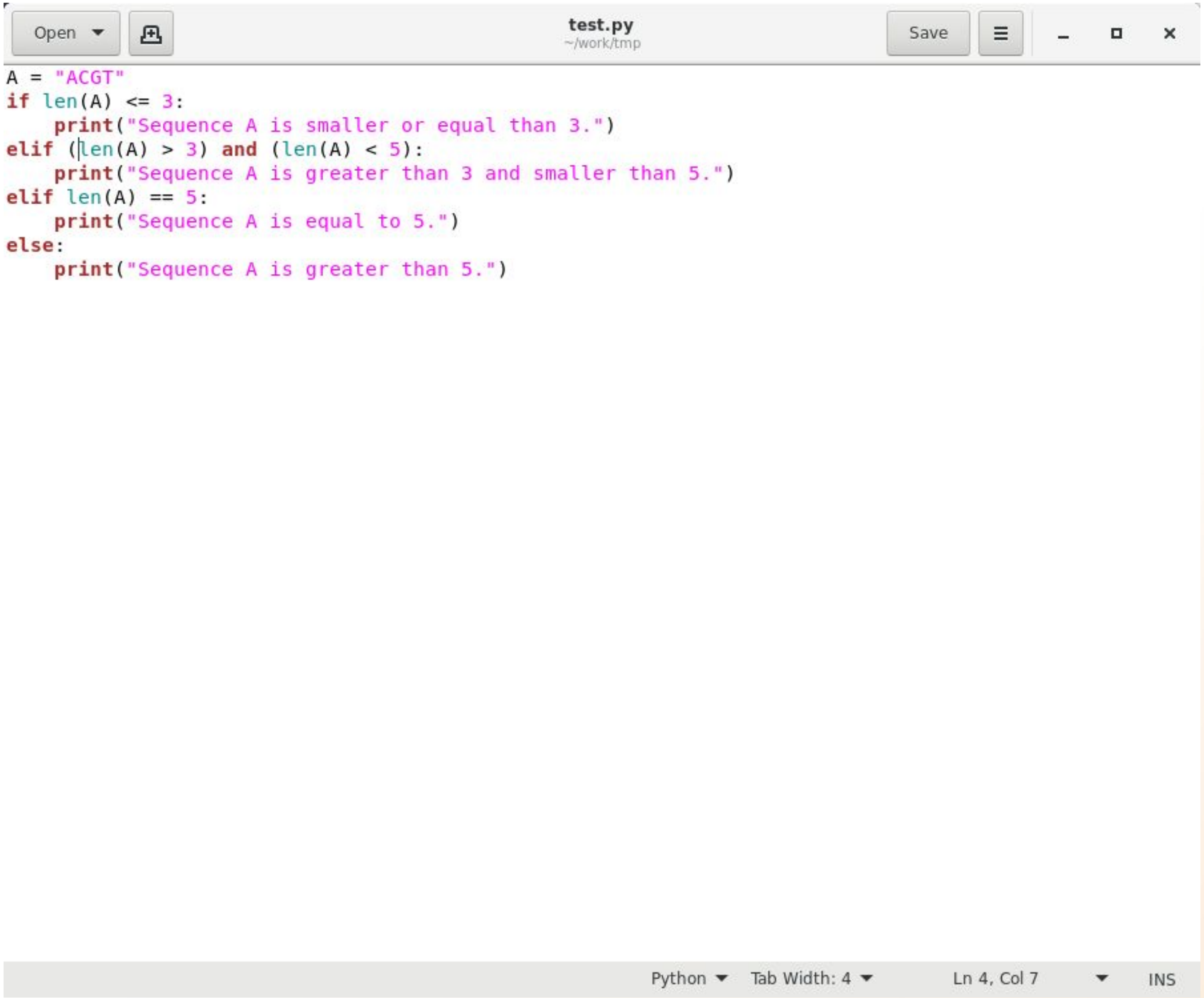
Gedit

- Graphical text editor more similar to what you might be used to

- Launch through command line:

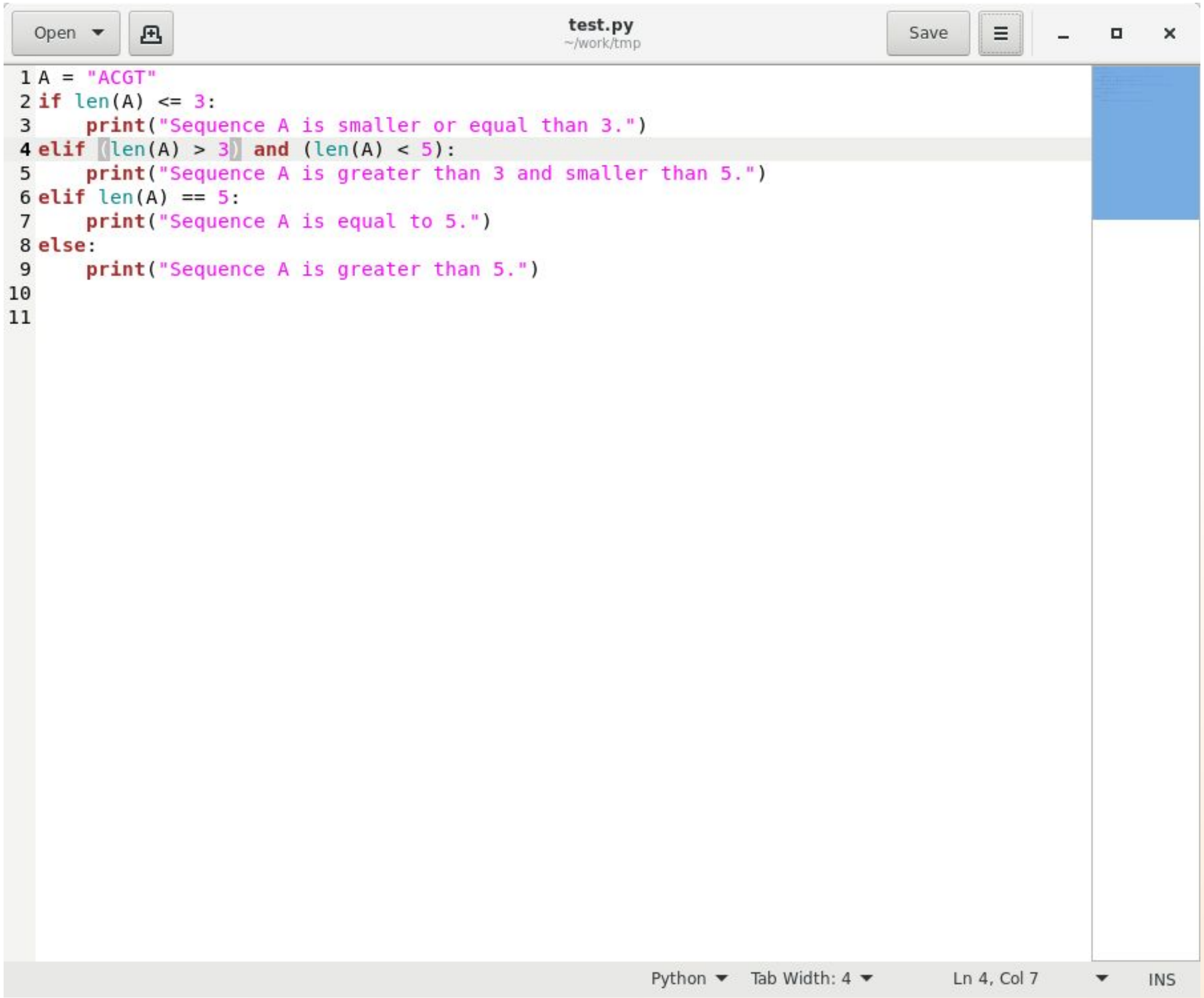
```
$ gedit
```

Gedit



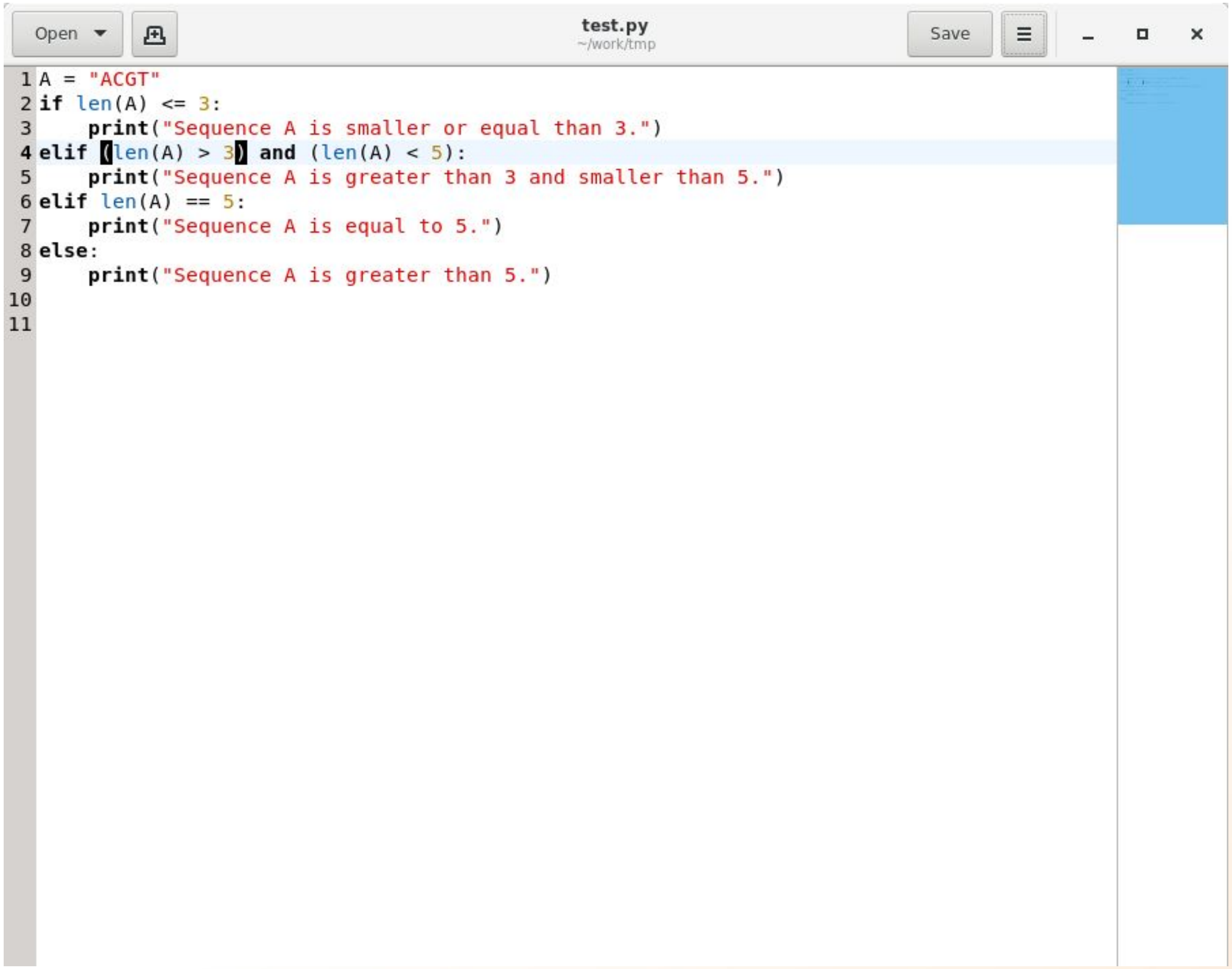
```
A = "ACGT"
if len(A) <= 3:
    print("Sequence A is smaller or equal than 3.")
elif (len(A) > 3) and (len(A) < 5):
    print("Sequence A is greater than 3 and smaller than 5.")
elif len(A) == 5:
    print("Sequence A is equal to 5.")
else:
    print("Sequence A is greater than 5.")
```

Gedit



```python
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.")


```

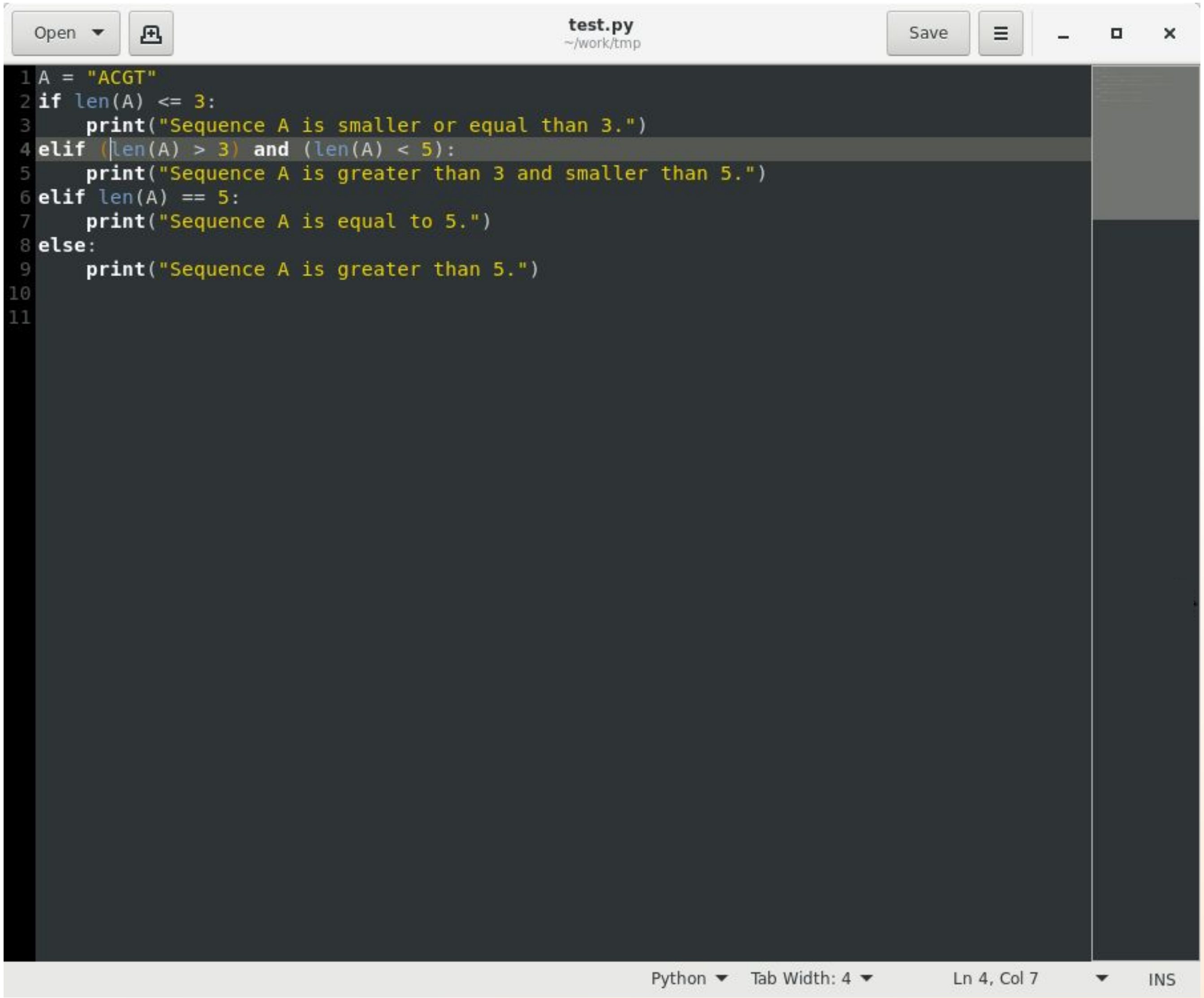
Gedit



```
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.")


```

Theme: Kate

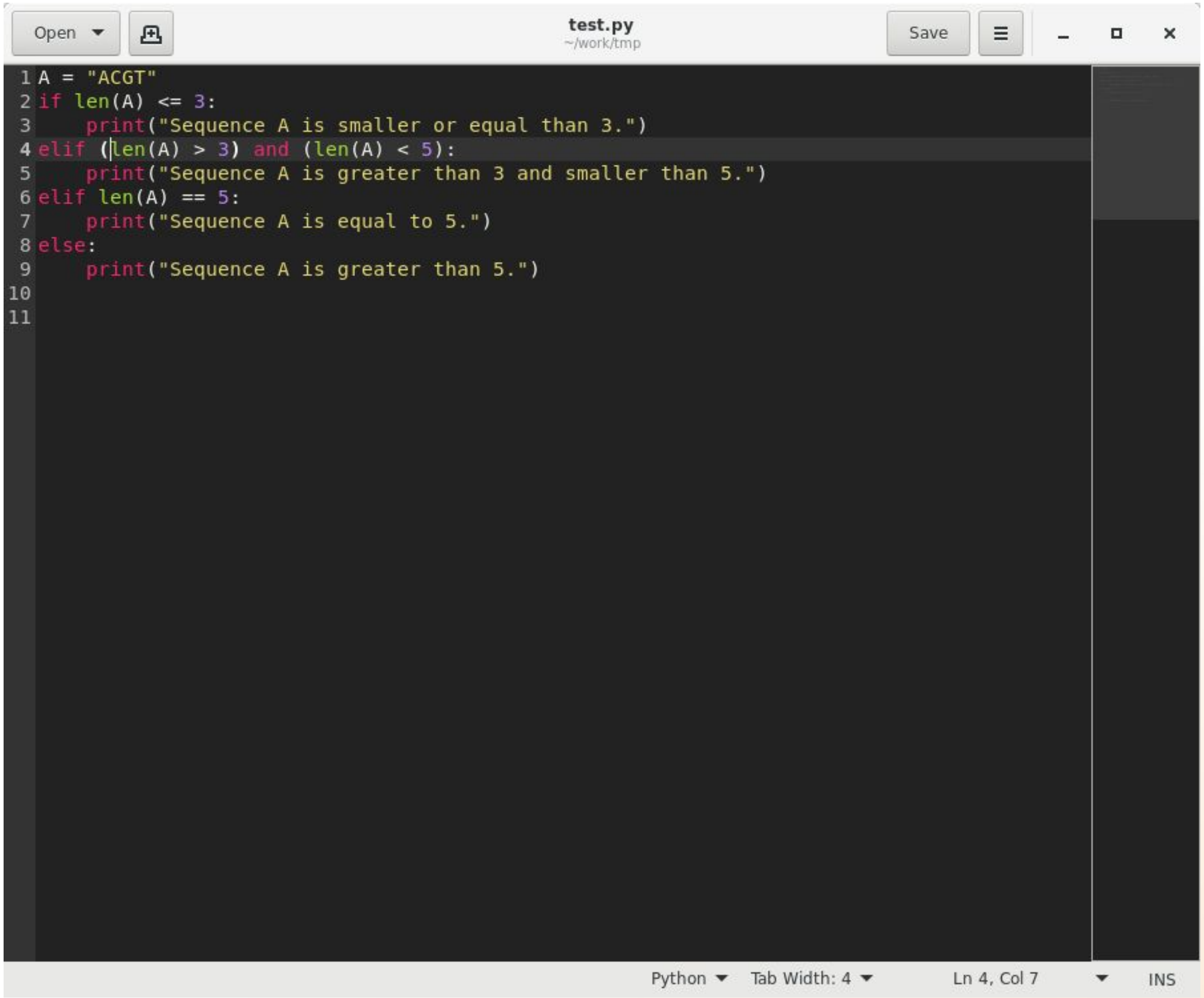
Gedit



```python
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.")
```

Theme: Oblivion

Gedit



```python
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.")


```

Theme: Monokai

Gedit

- 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/2603/topics/linux/lab_linux_advanced.html