

Advanced Linux Usage

2018-09-11

Martin Dahlö
martin.dahlo@scilifelab.uu.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!


```
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
```

^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

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

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

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

```
$ 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
$
```

```
$ 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!

- **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="Rackham"
```

- **Assigning**

```
my_variable=5
```

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

- **Using**

```
$my_variable
```

```
$ my_variable="Rackham"
```

```
$ echo "Hello, $my_variable! "
```

- **Assigning**

```
my_variable=5
```

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

- **Using**

```
$my_variable
```

```
$ my_variable="Rackham"
```

```
$ echo "Hello, $my_variable! "
```

```
Hello, Rackham!
```

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

```
prefix="sample"
```

```
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

```
prefix="sample"
```

```
my_prog ${prefix}_1.bam  
my_prog ${prefix}_2.bam  
my_prog ${prefix}_3.bam  
my_prog ${prefix}_4.bam  
my_prog ${prefix}_5.bam  
my_prog ${prefix}_6.bam  
my_prog ${prefix}_7.bam  
my_prog ${prefix}_8.bam  
my_prog ${prefix}_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


```
prefix="dog"
```

```
my_prog ${prefix}_1.bam  
my_prog ${prefix}_2.bam  
my_prog ${prefix}_3.bam  
my_prog ${prefix}_4.bam  
my_prog ${prefix}_5.bam  
my_prog ${prefix}_6.bam  
my_prog ${prefix}_7.bam  
my_prog ${prefix}_8.bam  
my_prog ${prefix}_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

```
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  
$
```

GNU nano 2.0.9

File: analysis.sh

```
prefix="sample"  
  
for i in 1 2 3 4 5 6 7 8 9;  
do  
    my_prog ${prefix}_${i}.bam  
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

GNU nano 2.0.9 File: analysis.sh

```
prefix="sample"

for i in 1 2 3 4 5 6 7 8 9;
do
    echo my_prog ${prefix}_${i}.bam
done
```

Debugging!

[Wrote 7 lines]

^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

GNU nano 2.0.9 File: a

```
prefix="sample"  
  
for i in 1 2 3 4 5 6 7 8 9;  
do  
    echo my_prog ${prefix}_${i}.bam  
done
```

```
$ bash analysis.sh  
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 Fi
^W Where I

Loop over files

```
$ ls *.bam  
sample_1.bam  sample_3.bam  sample_5.bam  sample_7.bam  sample_9.bam  
sample_2.bam  sample_4.bam  sample_6.bam  sample_8.bam  
$
```

Wildcard *

GNU nano 2.0.9

File: analysis.sh

```
prefix="sample"  
  
for file in $( ls *.bam );  
do  
    echo my_prog $file  
done
```

[Wrote 7 lines]

^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

```
for file in $( ls *.bam );  
do  
    echo my_prog $file  
done
```

[Wrote 7 lines]

^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

Loop over files

```
for file in $( ls *.bam );  
do  
    echo my_prog $file  
done
```

```
$ bash analysis.sh  
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 Fi
^W Where I

Loop over files

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

```
for file in $( ls $1/*.bam );  
do  
    echo my_prog $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

Loop over files

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

```
$ bash analysis.sh /path/to/my/bams
my_prog /path/to/my/bams/sample_1.bam
my_prog /path/to/my/bams/sample_2.bam
my_prog /path/to/my/bams/sample_3.bam
my_prog /path/to/my/bams/sample_4.bam
my_prog /path/to/my/bams/sample_5.bam
my_prog /path/to/my/bams/sample_6.bam
my_prog /path/to/my/bams/sample_7.bam
my_prog /path/to/my/bams/sample_8.bam
my_prog /path/to/my/bams/sample_9.bam
$
```

^G Get Help
^X Exit

^O WriteOut
^J Justify

^R Read F
^W Where

```
for file in $( ls $1/*.bam );  
do  
    my_prog $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

- 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 [[ 5 < 9 ]]; 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 == 9 ]]; 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
```


- For all samples except dog

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

- For all samples except dog

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

- For all samples except dog

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

Ex: `$file` is `/path/to/dog_1.bam`

- For all samples except dog

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

Ex: `$file` is `/path/to/dog_1.bam`

```
basename $file
```

- For all samples except dog

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

Ex: `$file` is `/path/to/dog_1.bam`

```
basename $file
```

```
dog_1.bam
```

- For all samples except dog

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

Ex: `$file` is `/path/to/dog_1.bam`

`basename $file`

`dog_1.bam`

- For all samples except dog

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

Ex: `$file` is `/path/to/dog_1.bam`

`basename $file`

`dog_1.bam`

Different languages

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

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

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

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

```
for file in $( ls $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);
    }
}
```

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

```
for file in $( ls $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 not 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

Laboratory time! (yet again)