Introduction to bash script (1)

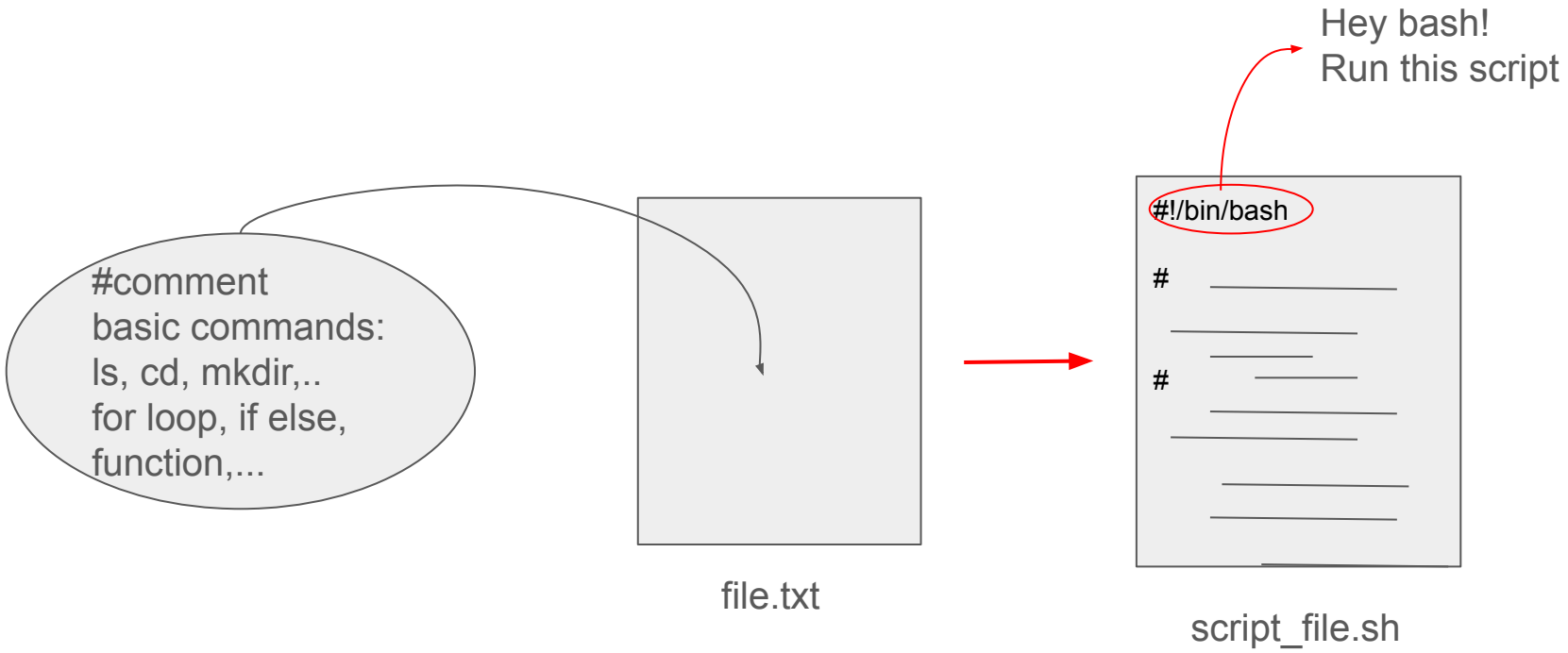
Huyha

23/05/2024

Contents

More: Extension Linux bash script learning

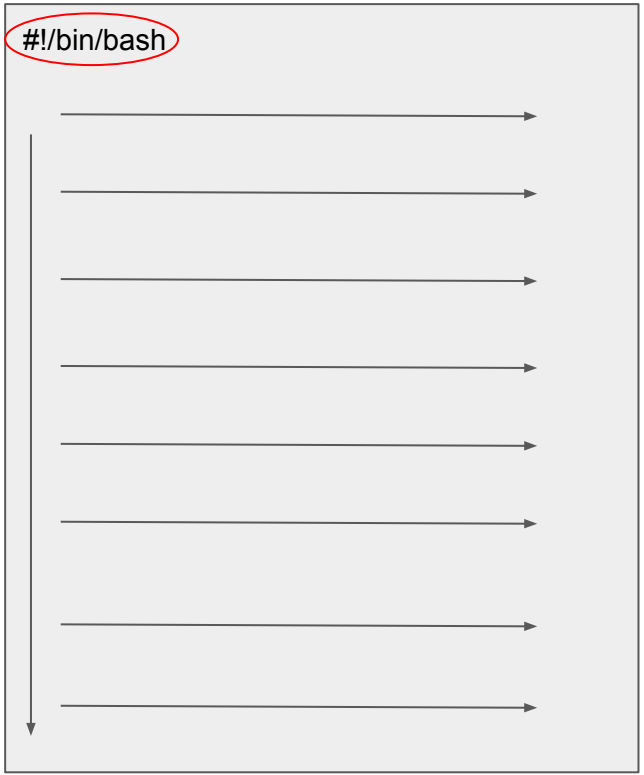
1. What is bash script?



-**Robust** and **reproducible**

Source: Khainguyen

1. What is bash script?



script_file.sh

#!/bin/bash
#! called “shebang"
/bin/bash is the path to the bash program file
->It make bash to do the script command

If the commands in this are in python, it is called python script, R is called R script

Commands read from **left to right** and from **top to bottom**

Source: Khainguyen

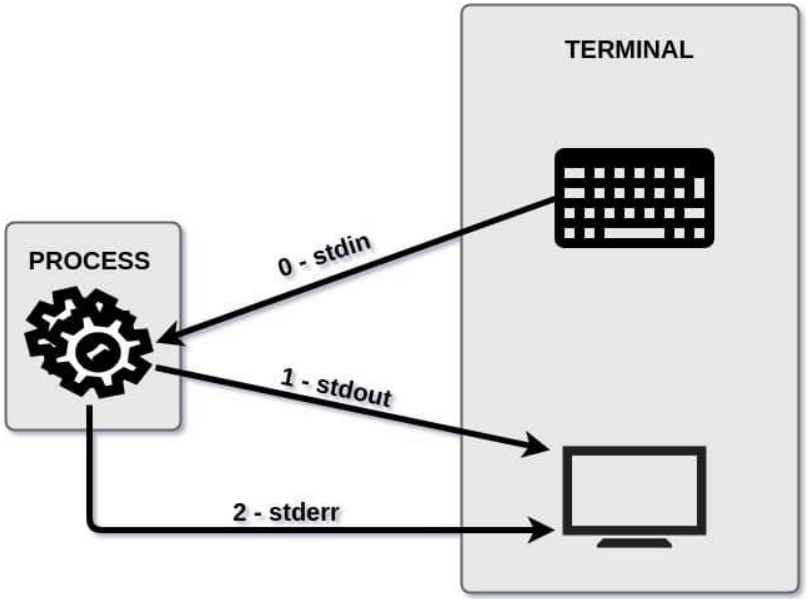
Write your first bash script

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ nano myscript.sh
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ ls -lh
total 4.0K
-rw-rw-r-- 1 huyha huyha 55 May  8 16:11 myscript.sh
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ chmod u+wrx myscript.sh
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ ls -lh
total 4.0K
-rwxrw-r-- 1 huyha huyha 55 May  8 16:11 myscript.sh
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ ./myscript.sh
Hello world
My first script
```

baschscript > $ myscript.sh

```
1   #!/bin/bash
2
3   echo "Hello world"
4   echo "My first script"
5
```

DATA STREAM



Run Linux command

- standard **input (stdin)**
- standard **output (stdout)** – displays the output
- standard **error (stderr)** – displays error output

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ expr 1 + 1
2
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ cd ?
bash: cd: ?: No such file or directory
```

2. Function syntax

1. The most widely used format is:

```
<function name> () {
        <commands>
}
```

Alternatively, the same function can be one line:

```
<function name> () { <commands>; }
```

2. The alternative way to write a bash function is using the reserved word **`function`**:

```
function <function name> {
        <commands>
}
```

Or in one line:

```
function <function name> { <commands>; }
```

2. Function example

baschscript > $ myscript.sh

```
1   #!/bin/bash
2
3   function home ) {
4   cd /home/huyha
5   }
6
7   function home () { cd /home/huyha; }
8
9   haha () {
10  cd /home/huyha
11  }
```

```
(base) huyha@dummycomputer:~$ cd $a
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ home
(base) huyha@dummycomputer:~$ cd $a
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ haha
(base) huyha@dummycomputer:~$
```

3. Variables, argument

Variables

$name = variable

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ echo "good morning huyha"
good morning huyha
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ name=huyha
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ echo "good morning name"
good morning name
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ echo "good morning $huyha"
good morning
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ echo "good morning $name"
good morning huyha
```

3. Arguments

baschscript > $ myscript.sh

```
1   #!/bin/bash
2
3   #$0
4   echo "This script name is $0"
5
6   #
7   echo "The first argument: $1"
8   echo "The second argument: $2"
9   echo "The third argument: $3"
10  #
11  echo "Total argument: $#"
12  #
13  a=$1
14  b=$2
15  c=$3
16  echo "sum $(( $a + $b + $c ))"
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ ./myscript.sh 1 2 3
This script name is ./myscript.sh
The first argument: 1
The second argument: 2
The third argument: 3
Total argument: 3
6
```

4. Basic math

```bash
#!/bin/bash

expr 1 + 1

echo "1 + 1" | bc

echo $((1+1))
```

```bash
#!/bin/bash

# Addition
result=$((5 + 3))
echo "5 + 3 = $result"

# Subtraction
result=$((10 - 4))
echo "10 - 4 = $result"

# Multiplication
result=$((6 * 2))
echo "6 * 2 = $result"

# Division
result=$((20 / 5))
echo "20 / 5 = $result"

# Modulus
result=$((15 % 4))
echo "15 % 4 = $result"
```

Guess question 1

GO gosocrative.com ROOM HA251

Script

```
#!/bin/bash

num1=5
num2=10
num3=$(($num1+$num2))

echo $(($num1 + $num2 - $num3))
echo "$num3"
echo " $1 % $num3" | bc
```

Input 1 `./question1.sh 10`

2 `./question1.sh 15`

Guess question 1

GO gosocrative.com ROOM HA251

Script

```
#!/bin/bash

num1=5
num2=10
num3=$(($num1+$num2))

echo $(($num1 + $num2 - $num3))
echo "$num3"
echo " $1 % $num3" | bc
```

Input 1 `./question1.sh 10`

2 `./question1.sh 15`

TRUE OR **FALSE**

1. 0, 15, 10
2. 0, 15, 0

Result question 1

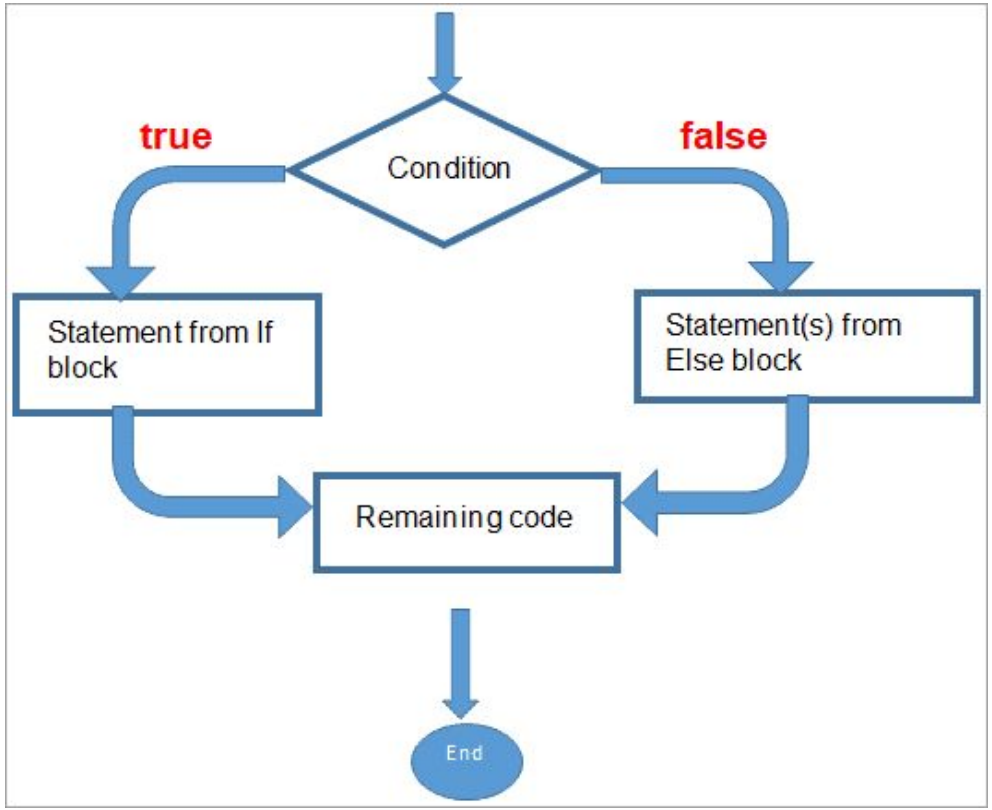
```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ ./question1.sh 10
0
15
10
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ ./question1.sh 15
0
15
0
```

TRUE OR **FALSE**

1. 0, 15, 10
2. 0, 15, 0

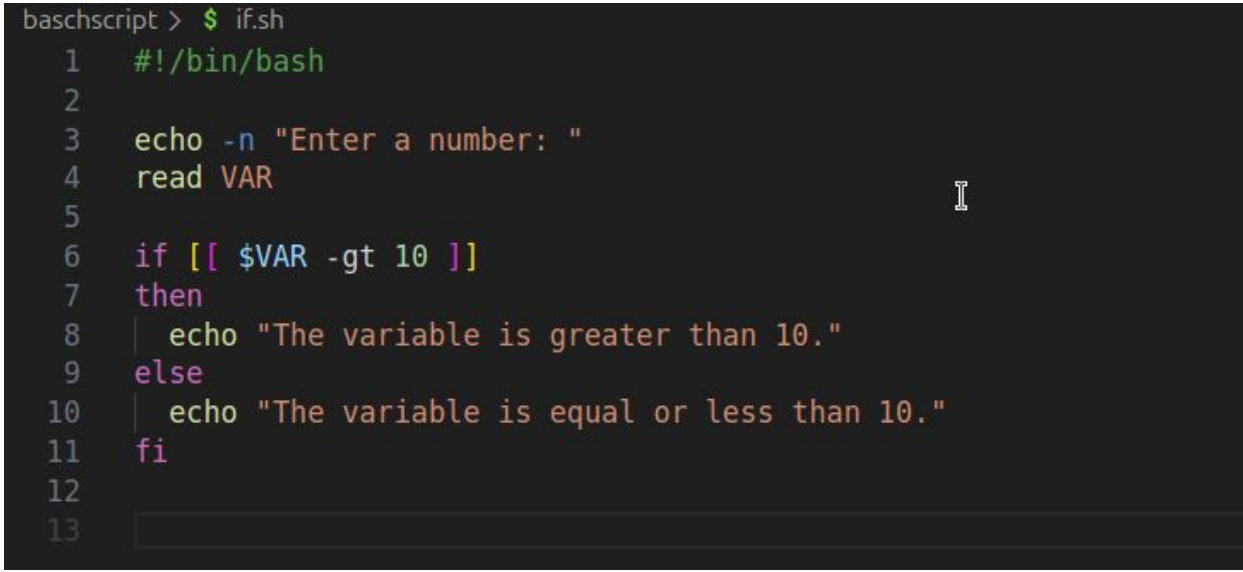
5. If Statement, Case Statement, (Condition)

```
if TEST-COMMAND
then
  STATEMENTS
fi
```



5. If Statement, Case Statement, (Condition)

If syntax



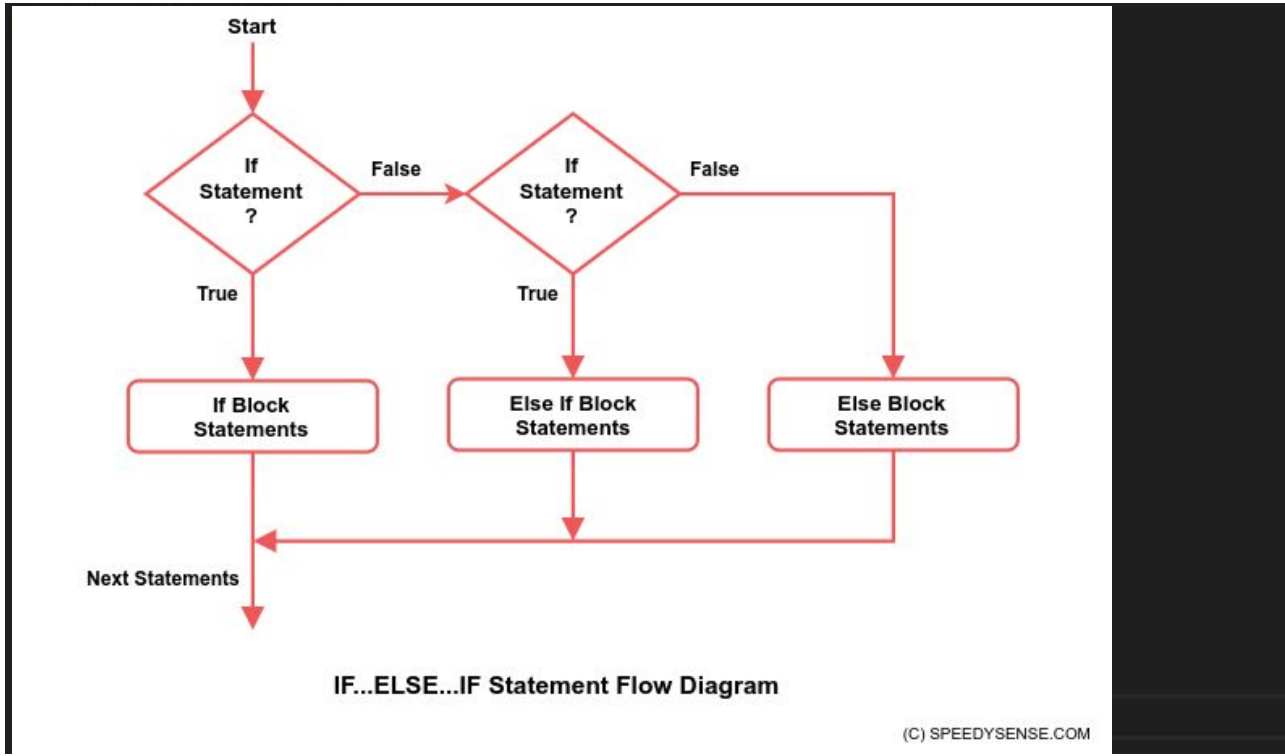
Source: https://linuxize.com/post/bash-if-else-statement/

5. IF statement (continue)

if...elif...else

The Bash `if...elif...else`

```
if TEST-COMMAND1
then
  STATEMENTS1
elif TEST-COMMAND2
then
  STATEMENTS2
else
  STATEMENTS3
fi
```



Source: https://linuxize.com/post/bash-if-else-statement/

5. IF statement (continue)

if...elif...else

The Bash `if...elif...else`

```
if TEST-COMMAND1
then
  STATEMENTS1
elif TEST-COMMAND2
then
  STATEMENTS2
else
  STATEMENTS3
fi
```

baschscript > $ if2.sh

```
1   #!/bin/bash
2
3   echo -n "Enter a number: "
4   read VAR
5
6   if [[ $VAR -gt 10 ]]
7   then
8     echo "The variable is greater than 10."
9   elif [[ $VAR -eq 10 ]]
10  then
11    echo "The variable is equal to 10."
12  else
13    echo "The variable is less than 10."
14  fi
15
```

Source: https://linuxize.com/post/bash-if-else-statement/

5. NESTED IF

Question:

I have item price

How to find which number is the largest?

baschscript > $ if3.sh

```
#!/bin/bash
$1=VAR1
$2=VAR2
$3=VAR3

if [[ $VAR1 -ge $VAR2 ]]
then
  if [[ $VAR1 -ge $VAR3 ]]
  then
    echo "$VAR1 is the largest number."
  else
    echo "$VAR3 is the largest number."
  fi
else
  if [[ $VAR2 -ge $VAR3 ]]
  then
    echo "$VAR2 is the largest number."
  else
    echo "$VAR3 is the largest number."
  fi
fi
```

Source: https://linuxize.com/post/bash-if-else-statement/

5. Case statement

Case syntax

1.Case <variable> in

2.Pattern1 | pattern2) action1

3.;;

4.Pattern 3) action 2

5.;;

6.*) action

7.;;

8.esac

5. Condition

`[[ -z STRING ]]`	Empty string
`[[ -n STRING ]]`	Not empty string
`[[ STRING == STRING ]]`	Equal
`[[ STRING != STRING ]]`	Not Equal
`[[ NUM -eq NUM ]]`	Equal
`[[ NUM -ne NUM ]]`	Not equal
`[[ NUM -lt NUM ]]`	Less than
`[[ NUM -le NUM ]]`	Less than or equal
`[[ NUM -gt NUM ]]`	Greater than
`[[ NUM -ge NUM ]]`	Greater than or equal
`[[ STRING =~ STRING ]]`	Regexp
`(( NUM < NUM ))`	Numeric conditions

File conditions

`[[ -e FILE ]]`	Exists
`[[ -r FILE ]]`	Readable
`[[ -h FILE ]]`	Symlink
`[[ -d FILE ]]`	Directory
`[[ -w FILE ]]`	Writable
`[[ -s FILE ]]`	Size is > 0 bytes
`[[ -f FILE ]]`	File
`[[ -x FILE ]]`	Executable
`[[ FILE1 -nt FILE2 ]]`	1 is more recent than 2
`[[ FILE1 -ot FILE2 ]]`	2 is more recent than 1
`[[ FILE1 -ef FILE2 ]]`	Same files

`[[ ! EXPR ]]`	Not
`[[ X && Y ]]`	And
`[[ X || Y ]]`	Or

PRACTICE 1

Input 3 number

+Calculate **Sum**,**Mean**

+Detect which is the biggest(optional)

+Detect if the biggest (or 3 number input) is **odd** or **even**

Result

6. Exit code

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ true; echo "$?"
0
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ false; echo "$?"
1
```

In a bash script, an exit code is a numerical value returned by a command or script to indicate its success or failure. Exit codes are useful for scripting as they allow you to determine the outcome of a command or script and take appropriate actions based on that outcome.

Output 0 stand for **TRUE value (success command)**

Output 1 stand for **FALSE value (fail command)**

$? Use to check exit code

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ touch /root/test.txt;echo "$?"
touch: cannot touch '/root/test.txt': Permission denied
1
(fail command)
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ touch ~/test.txt;echo "$?"
0
(success command)
```

6. Control Exit code

```bash
#!/bin/bash

touch /root/test.txt

if [[ $? -eq 0 ]];then
    echo "Done"
else
    echo "Error"
fi
echo "$?"
```

```
touch: cannot touch '/root/test.txt': Permission denied
Error
0
```

```bash
#!/bin/bash

touch /root/test.txt

if [[ $? -eq 0 ]];then
    echo "Done"
    exit 0
else
    echo "Error"
    exit 1
fi
echo "$?"
echo "bla bla bla"
```

```
touch: cannot touch '/root/test.txt': Permission denied
Error
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ echo $?
1
```

Guess question 2

```
#1
ls > name.txt
test -s name.txt
echo "1: $?"
#2
a=005
test $a -eq 5
echo "2: $?"
#3
b=1
test 001 == $b
echo "3: $?"
#4
test -d /
echo "4: $?"
#5
false && test 2 -gt 1
echo 5: $?
```

GO gosocrative.com ROOM HA251

1. 1,0 2,0 3,1 4,0 5,1
2. 1,0 2,0 3,0 4,1 5,0
3. 1,1 2,0 3,1 4,0 5,1
4. 1,1 2,1 3,0 4,0 5,1
5. 1,0 2,1 3,0 4,1 5,0

GO gosocrative.com ROOM HA251

Result

7. Cheatsheet

Bash scripting cheatsheet

Example

```
#!/usr/bin/env bash

NAME="John"
echo "Hello $NAME!"
```

Variables

```
NAME="John"
echo $NAME
echo "$NAME"
echo "${NAME}!"
```

String quotes

```
NAME="John"
echo "Hi $NAME"  #=> Hi John
echo 'Hi $NAME'  #=> Hi $NAME
```

Conditional execution

```
git commit && git push
git commit || echo "Commit failed"
```

Functions

```
get_name() {
  echo "John"
}

echo "You are $(get_name)"
```

See: Functions

Shell execution

```
echo "I'm in $(pwd)"
echo "I'm in `pwd`"
# Same
```

See Command substitution

Conditionals

```
if [ -z "$string" ]; then
  echo "String is empty"
elsif [ -n "$string" ]; then
  echo "String is not empty"
fi
```

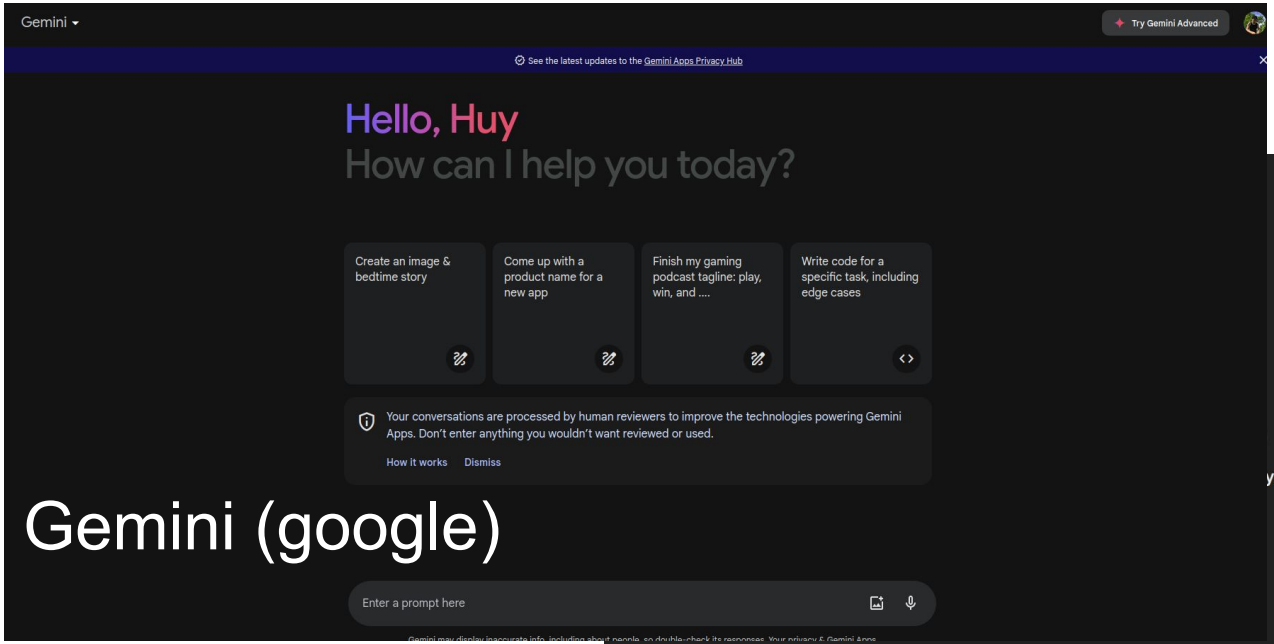
Brace expansion

```
echo {A,B}.js
```

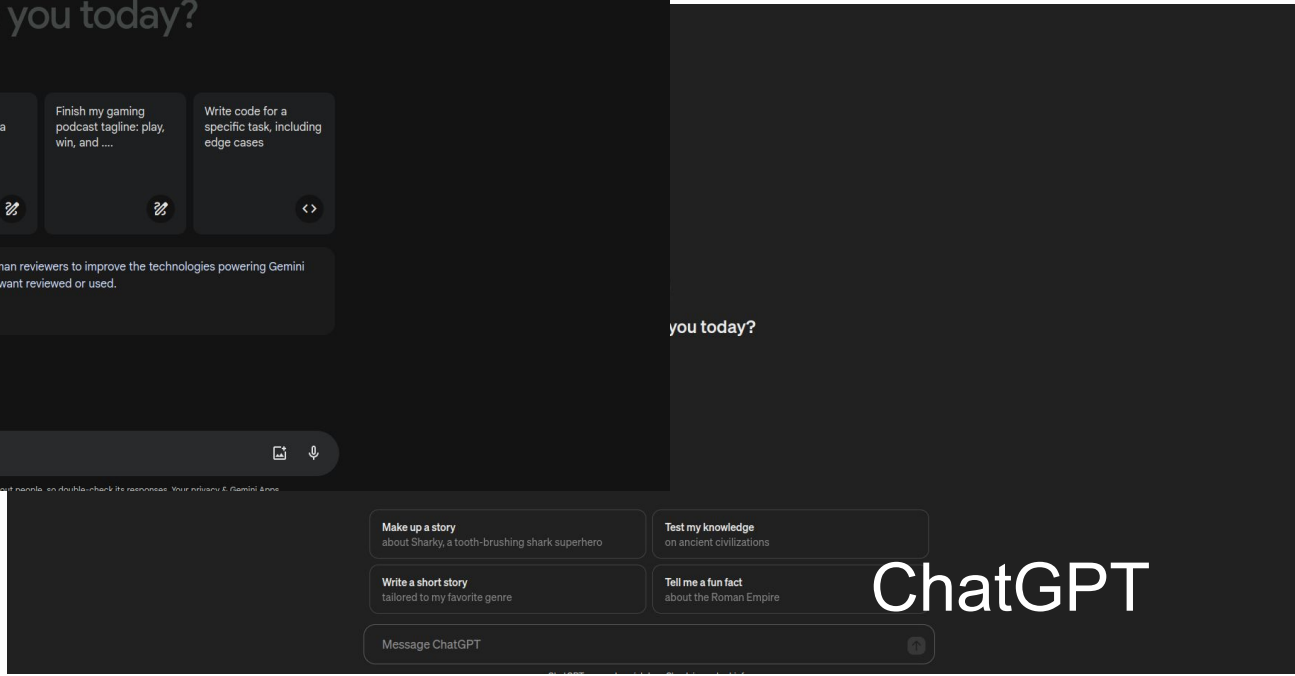
Strict mode

```
set -euo pipefail
IFS=$'\n\t'
```

7. Use AI to learn code



Gemini (google)



ChatGPT

Summary

1. What is bash script?
   +Write your first bash script (Hello world)
   +Datastream
2. Function
3. Variables, argument
4. Basic math
5. IF Statement (Condition) + Case statement Condition
6. Exit Code
7. Cheat Sheet and AI

Extension Linux bash script learning

8. Homework

1/ Use this code to down

```
mkdir ./homework
curl -o ./homework/fa1.txt https://drive.usercontent.google.com/download?id=1RvsIx3_qGI_znCAWHDycHKfhUQxm5U6j&export=download&authuser=0
curl -o ./homework/fa2.txt https://drive.usercontent.google.com/download?id=1znAljOXBLbeq4qt0asce185mzKkPgmxv&export=download&authuser=0
(If you don't have curl please install it.-(sudo apt install curl))
Guilde
```

+ Use “Grep” to extract the organism name of fa1.txt and fa2.txt
+ Use “Grep” and “wc” to find the nucleotide length of file
+ Compare the nucleotide length between fa1.txt and fa2.txt

2/ Output the organism name and length of the bigger nucleotide length file

```
>KT719404.1 Porcine circovirus 2 strain PCV2, complete genome
ACCAGCGCACTTCGGCAGCGGCAGCACCTCGGCAGCACCTCAGCAGCAACATGCCCAGCAAGAAGAATGG
AAGAAGCGGACCCCAACCCCATAAAAGGTGGGTGTTCACTTTGAATAATCCTTCCGAAGACGAGCGCAAG
AAAATACGGGAGCTTCCAATCTCCCTGTTTGATTATTTTATTGTTGGCGAGGAGGGTAATGAGGAAGGAC
GAACACCTCACCTCCAGGGGTTCGCTAATTTTGTGAAAAAGCAAACTTTTAATAAAGTGAAGTGGTATCT
TGGTGCCCGCTGCCACATCGAGAAAGCCAAAGGAACTGATCAGCAGAATAAAGAATATTGCAGTAAAGAA
GGCAACTTACTTATTGAGTGTGGAGCTCCTAGATCTCAAGGACAACGGAGTGACCTGTCTACTGCTGTGA
GTACCTTGTTGGAGAGCGGGAGTCTGGTGACCGTTGCAGAGCAGCACCCTGTAACGTTTGTCAGAAATTT
CCGCGGGCTGGCTGAACTTTTGAAAGTGAGCGGGAAAATGCAGAAGCGTGATTGGAAGACTAATGTACAC
GTCATTGTGGGGCCACCTGGGTGTGGTAAAAGCAAATGGGCTGCTAATTTTGCAGA
```

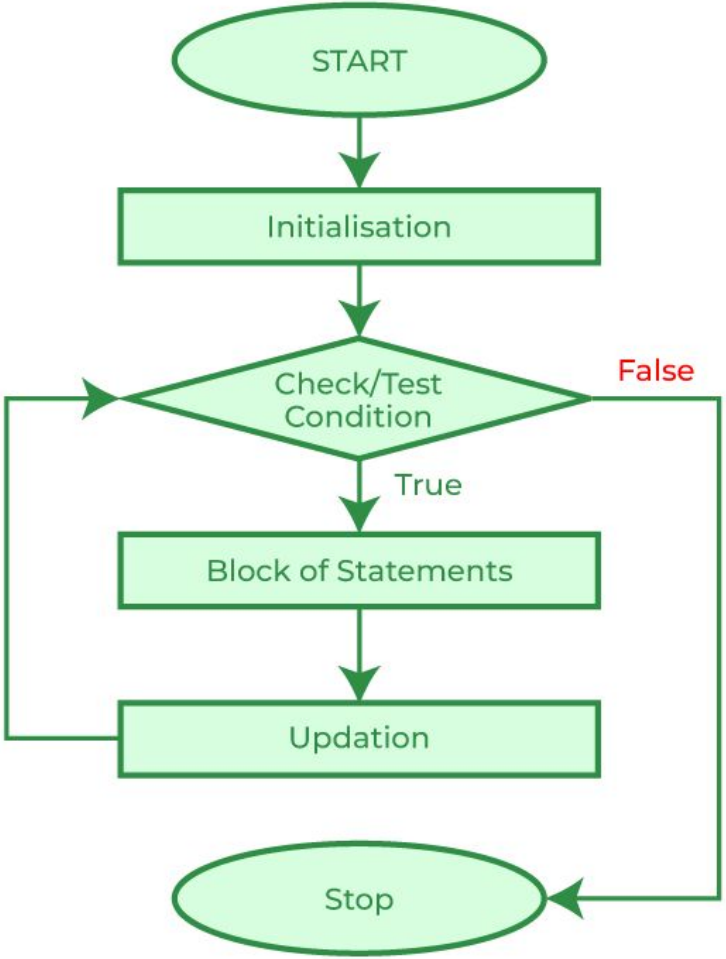
Introduction to bash script (2)

Huyha

9. For loop, While loop

Basic syntax

```bash
for item in list
do
    # commands to execute for each item
done
```



Loop structure

9. For loop, While loop

Loops

Basic for loop

```
for i in /etc/rc.*; do
  echo "$i"
done
```

Ranges

```
for i in {1..5}; do
    echo "Welcome $i"
done
```

With step size

```
for i in {5..50..5}; do
    echo "Welcome $i"
done
```

Reading lines

```
while read -r line; do
  echo "$line"
done <file.txt
```

C-like for loop

```
for ((i = 0 ; i < 100 ; i++)); do
  echo "$i"
done
```

Forever

```
while true; do
  ...
done
```

For loop, while loop

Basic loop

baschscript > $ forloop.sh

```
1  #!/bin/bash
2  for (( i = 0 ; i <= 2; i++ ))
3  do
4  echo "$i"
5  done
6
7  for filename in 7 8 9 huy
8  do
9  echo $filename
10 done
11
12 for filename in question*;
13 do
14 echo $filename
15 done
```

```
(base) huyha@dummycomput
0
1
2
7
8
9
huy
question1.sh
question2.sh
```

For loop, while loop

Example **range**

baschscript > $ forRange.sh

```
1  #!/bin/bash
2
3  for i in {10..20}
4  do
5  echo $i
6  done
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ ./forRange.sh
10
11
12
13
14
15
16
17
18
19
20
```

For loop, while loop

Example **Break, Continue**

baschscript > $ forRange1.sh

```
1   #!/bin/bash
2
3   for i in {10..20}
4   do
5   if [[ $i -eq 12 ]]
6   then
7   echo i touch 12
8   break
9   fi
10  echo $i
11  done
```

```
10
11
i touch 12
```

baschscript > $ forRange1.sh

```
1   #!/bin/bash
2
3   for i in {10..20}
4   do
5   if [[ $i -eq 12 ]]
6   then
7   echo i touch 12
8   continue
9   fi
10  echo $i
11  done
```

```
10
11
i touch 12
13
14
15
16
17
18
19
20
```

For loop, while loop

Reading file

baschscript > $ forRead.sh

```
1  #!/bin/bash
2
3  while read name
4  do
5  echo $name
6  done<name.txt
```

File name.txt

baschscript > name.txt

```
1   Maya Chen 1112
2   Liam Miller 1114
3   Elena Garcia 1112
4   Noah Jones 3333
5   Sophia Lee 1212
6   Ethan Brown 12312312
7   Olivia Williams  12312312
8   Benjamin Davis 123213
9   Ava Wilson 14125421
10  Michael Taylor 121221
11  Evelyn Thomas
12  Matthew Moore
13  Isabella Garcia
14  Daniel Hernandez
15  Charlotte Young
16  William Allen
```

Row

Word

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ ./forRead.sh
Maya Chen 1112
Liam Miller 1114
Elena Garcia 1112
Noah Jones 3333
Sophia Lee 1212
Ethan Brown 12312312
Olivia Williams 12312312
Benjamin Davis 123213
Ava Wilson 14125421
Michael Taylor 121221
Evelyn Thomas
Matthew Moore
Isabella Garcia
Daniel Hernandez
Charlotte Young
William Allen
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ ./forRead1.sh
Maya
Chen
1112
Liam
Miller
1114
Elena
Garcia
1112
Noah
```

baschscript > $ forRead1.sh

```
1  #!/bin/bash
2
3  for i in $(cat ./name.txt)
4  do
5  echo $i
6  done
```

Practice 2

Write a script that prints the numbers from 15 to 115 (distance between 2 number is 15). Additionally:

1. **Highlight reaching 60:** When the script encounters the number 60, print a message indicating it reached 60.
2. **Identify odd/even:** For each number, determine if it's odd or even and print that information alongside the number.(optional)
3. **Divisibility by 10:** Indicate if the number is divisible by 10 (a multiple of 10)(optional)

```
15 is odd and indivisible for 10
30 is even and divisible for 10
45 is odd and indivisible for 10
60 is even and divisible for 10
number is reach 60
75 is odd and indivisible for 10
90 is even and divisible for 10
105 is odd and indivisible for 10
```

Result

11. What awk can do?

Porcine circovirus 2 isolate PCV2 P1 OK/USA, complete genome

GenBank: MW051676.1

FASTA Graphics

Go to:

```
LOCUS       MW051676               1761 bp    DNA     circular VRL 21-JUL-2021
DEFINITION  Porcine circovirus 2 isolate PCV2 P1 OK/USA, complete genome.
ACCESSION   MW051676
VERSION     MW051676.1
KEYWORDS    .
SOURCE      Porcine circovirus 2
  ORGANISM  Porcine circovirus 2
            Viruses; Monodnaviria; Shotokuvirae; Cressdnaviricota;
            Arfiviricetes; Cirlivirales; Circoviridae; Circovirus; Circovirus
            porcine2.
REFERENCE   1  (bases 1 to 1761)
  AUTHORS   Paim,W.P., Maggioli,M.F., Weber,M.N., Rezabek,G., Narayanan,S.,
            Ramachandran,A., Canal,C.W. and Bauermann,F.V.
  TITLE     Virome characterization in serum of healthy show pigs raised in
            Oklahoma demonstrated great diversity of ssDNA viruses
  JOURNAL   Virology 556, 87-95 (2021)
   PUBMED   33550118
REFERENCE   2  (bases 1 to 1761)
  AUTHORS   Paim,W.P., Maggioli,M.F., Weber,M.N., Rezabek,G., Narayanan,S.S.,
            Ramachandran,A., Canal,C.W. and Bauermann,F.V.
  TITLE     Direct Submission
  JOURNAL   Submitted (27-SEP-2020) Veterinary Pathobiology, Oklahoma State
            University, 250 McElroy Hall, Stillwater, OK 74074, USA
COMMENT     ##Assembly-Data-START##
            Assembly Method       :: Geneious Prime v. 2020.2.1
            Assembly Name         :: PCV2 P1 OK/USA
            Coverage              :: 12.5X
            Sequencing Technology :: Illumina
            ##Assembly-Data-END##
FEATURES             Location/Qualifiers
     source          1..1761
                     /organism="Porcine circovirus 2"
                     /mol_type="genomic DNA"
                     /isolate="PCV2 P1 OK/USA"
                     /host="swine"
                     /db_xref="taxon:85708"
                     /country="USA"
                     /collection_date="2018"
```

Customize view

Analyze this sec
Run BLAST
Pick Primers
Highlight Sequenc
Find in this Sequer

Related informa
Protein
PubMed
Taxonomy
RefSeq Genome f
RefSeq Genome S

Recent activity
Porcine circov OK/USA, com
pcv2 (12756)
Porcine circov genome
pcv2 complete

```
                     GWLPWDDLLRLCDRYPLTVETKGGTVPFLARSILITSNQTPLEWYSSTAVPAVEALYR
                     RITSLVFWKNATEQSTEEGGQFVTLSPPCPEFPYEINY"
     CDS             complement(351..665)
                     /codon_start=1
                     /product="ORF3 protein"
                     /protein_id="QXO84985.1"
                     /translation="MVTIPPLVSRWFPVCGFRVCKISSPFAFATPRWPHNDVYIRLPI
                     TLLHFPAHFQKFSQPAEISDKRYRVLLCNGHQTPALQQGTHSSREVTPLSLRSRSSTF
                     YQ"
     CDS             complement(1024..1728)
                     /codon_start=1
                     /product="capsid protein"
                     /protein_id="QXO84986.1"
                     /translation="MTYPRRRFRRRRHRPRSHLGQILRRRPWLVHPRHRYRWRRKNGI
                     FNTRLSRTIGYTVKKTTVRTPSWNVDMMRFNINDFLPPGGGSNPLTVPFEYYRIRKVK
                     VEFWPCSPITQGDRGVGSTAVILDDNFVTKANALTYDPYVNYSSRHTITQPFSYHSRY
                     FTPKPVLDRTIDYFQPNNKRNQLWLRLQTTGNVDHVGLGTAFENSIYDQDYNIRITMY
                     VQFREFNLKDPPLNPN"
ORIGIN
        1 gcacttcggc agcggcagca cctcggcagc acctcagcag caacatgccc agcaagaaga
       61 gtggaagaag cggaccccaa ccacataaaa ggtgggtgtt cacgctgaat aatccttccg
      121 aagacgagcg caagaaaata cgggagctcc caatctccct atttgattat tttattgttg
      181 gcgaggaagg taatgaggag ggccgaacac cccacctaca ggggttcgct aattttgtga
      241 agaagcaaac ttttaataaa gtgaagtggt attttggtgc ccgctgccac atcgagaaag
      301 cgaaaggaac agatcagcag aataaagaat attgcagtaa agaaggcaac ttactgatag
      361 aatgtggagc tcctagatct caaggacaac ggagtgacct ctctactgct gtgagtacct
      421 tgttggagag cgggagtctg gtgaccgttg cagagcagca ccctgtaacg tttgtcagaa
      481 atttccgcgg gctggctgaa cttttgaaag tgagcgggaa aatgcagaag cgtgattgga
      541 agacgaatgt acacgtcatt gtggggccac ctgggtgtgg caaaagcaaa tgggctgcta
      601 attttgcaga cccggaaacc acatactgga aaccacctag aaacaagtgg tgggatggtt
      661 accatggtga agaagtggtt gttattgatg acttttatgg ctggctgccg tgggatgatc
      721 tactgagact ctgtgatcga tatcctttga ctgttgagac taaaggtgga actgtacctt
      781 ttttggcccg cagtattctg attaccagca atcagacccc gttggaatgg tactcctcaa
      841 ctgctgtccc agctgtagaa gctctctatc ggaggattac ttccttggta ttttggaaga
      901 atgctacaga acaatccacg gaggaagggg gccagttcgt caccctttcc cccccatgcc
      961 ctgaatttcc atatgaaata aattactgag tctttttttat cacttcgtaa tggttttat
     1021 tattcagtta gggttaagtg gggggtcttt aagattaaat tctctgaatt gtacatacat
     1081 ggttatacgg atattgtagt cctggtcgta tatactgttt tcgaacgcag tgccgaggcc
     1141 tacatggtct acatttccag tagtttgtag tctcagccag agttgatttc ttttgttatt
     1201 gggttggaag taatcgattg tcctatcaag gacaggtttc ggggtaaagt accgggagtg
     1261 gtaggagaag ggctgggtta tggtatggcg ggaggagtag tttacatagg ggtcataggt
     1321 tagggcattg gcctttgtta caaagttatc atctagaata acagcagtgg agcccactcc
     1381 cctgtcaccc tgggtgattg gggagcaggg ccagaattca accttaacct tccttattct
     1441 gtagtattca aagggcacag tgagggggtt tgagccccct cctgggggaa gaaaatcatt
     1501 aatattaaat ctcatcatgt ccacattcca ggagggcgtt ctgactgtgg ttttcttgac
     1561 agtgtaaccg atggtgcggg agaggcgggt gttgaagatg ccatttttcc ttctccagcg
     1621 gtaacggtgg cgggggtgga cgagccaggg ccggcggcgg aggatctggc caagatggct
     1681 gcgggggcgg tgtcttcgtc tgcggaaacg cctccttgga tacgtcatcg ctgaaaacga
     1741 aagaagtgcg ctgtaagtat t
//
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk '$1=="AC"{print $2}' U31362.1.gb |tr -d ';'
U31362
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk '$1=="DE"{$1="";print $0}' U31362.1.gb
 Human immunodeficiency virus type 1 isolate IND7 envelope glycoprotein
 gp120 (env) mRNA, partial cds.
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk '/SQ/{get
line;while(i=1){print $0;getline; if ($1== "//"){break} } }' U31362.1.gb | tr -d '0-9'
     atgggagtga gggggatact gaggaattat caacaatggt ggatatgggg catcttaggc
     ttttggatgt taatgatttg taatgtggta ggaaacttgt gggtcacagt ctattatggg
     gtacctgtgt gggaagaagc aaaaactact ctattctgtg catcagatgc taaagcatat
     gagacagaag tgcataatgt ctgggctaca catgcctgtg tacccacaga ccccaaccca
     caagaaatat ttttggaaaa tgtaacagaa aattttaaca tgaggaaaaa tgacatggtg
     aatcagatgc atgaggatgt aatcagttta tgggatcaaa gcctaaagcc ctgtgtaaag
     ttgaccccac tctgtgtcac tttagaatgt agacaggtta atgttacca
     caggttaacg ctaccagtaa tggagaagaa ataaaaaata taaaaaatt
     tcaaccacag aaataagaga taggaagcag acagcgtatc gacttttt
     ctagtaccac ttgataacaa gaatgggagc aactctagta agtatatat
     aatacctcag ccataacaca agcctgtcca aaggtcactt ttgatccaa
     tattgtactc cagctggtta tgcgattcta aagtgtaatg ataagacat
     ggaccatgcc ataatgttag cacagtacaa tgtacacatg gaattaag
     actcaactac tgttaaatgg tagcctagca gaagaagaga taataatta
     ctgacagaca atgtcaaaac aataatagta catcttaatc aatctgtag
     acaagaccca acaataatac aagaaaaagt ataaggatag gaccaggac
     gcaacaggag acataatagg ggacataaga caagcacatt gtaacatta
     tggaatgaaa ctttacaaag ggtaagaaaa aaattagcag aacacttcc
     ataaatttta catcatcctc aggaggggac ctagaaatta caacacata
     agaggagaat ttttctattg taatcaatca ggcctgttta atggtacat
     ggtacaaaag gtaattcaag ctcagtcatc acaatcccat gcagaataa
     aacatgtggc agggggtagg acgagcaatg tatgcccctc ccattgaag
     tgtaaatcaa atatcacagg actactattg gtacgtgatg gaggactag
     gacacagaga cagagacatt cagacctgga ggaggagata tgagggacaa ttggagaagt
     gaattatata aatataaagt ggtaaaaatt aagccattgg gaatagcacc cactacagca
     aaaaggagag tggtggagag agaa
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk '/transla
tion/{while(i=1){print $0;getline; if ($1=="XX"){break} } }' U31362.1.gb|awk '{print
$2}'|tr -d '/translation="'
MGVRGILRNYQQWWIWGILGFWMLMICNVVGNLWVTVYYGVPVWE
EAKTTLFCASDAKAYETEVHNVWATHACVPTDPNPQEIFLENVTENFNMRKNDMVNQMH
EDVISLWDQSLKPCVKLTPLCVTLECRQVNVTSNGTQVNATSNGEEIKNIKNCSFNSTT
EIRDRKQTAYRLFYRLDLVPLDNKNGSNSSKYILINCNTSAITQACPKVTFDPIPIHYC
TPAGYAILKCNDKTFNGTGPCHNVSTVQCTHGIKPVVSTQLLLNGSLAEEEIIIRSENL
TDNVKTIIVHLNQSVEIVCTRPNNNTRKSIRIGPGQTFYATGDIIGDIRQAHCNISEAK
WNETLQRVRKKLAEHFPNKTINFTSSSGGDLEITTHSFNCRGEFFYCNQSGLFNGTYMH
NGTKGNSSSVITIPCRIKQIINMWQGVGRAMYAPPIEGNITCKSNITGLLLVRDGGLGP
SNDTETETFRPGGGDMRDNWRSELYKYKVVKIKPLGIAPTTAKRRVVERE
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk '$1=="AC"{print $2}' U31362.1.gb |tr -d ';'
U31362
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk '$1=="DE"{$1="";print $0}' U31362.1.gb
Human immunodeficiency virus type 1 isolate IND7 envelope glycoprotein
gp120 (env) mRNA, partial cds.
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk '/SQ/{get
line;while(i=1){print $0;getline; if ($1== "//"){break} } }' U31362.1.gb | tr -d '0-9'
     atgggagtga gggggatact gaggaattat caacaatggt ggatatgggg catcttaggc
     ttttggatgt taatgatttg taatgtggta ggaaacttgt gggtcacagt ctattatggg
     gtacctgtgt gggaagaagc aaaaactact ctattctgtg catcagatgc taaagcatat
     gagacagaag tgcataatgt ctgggctaca catgcctgtg tacccacaga ccccaaccca
     caagaaatat ttttggaaaa tgtaacagaa aattttaaca tgaggaaaaa tgacatggtg
     aatcagatgc atgaggatgt aatcagttta tgggatcaaa gcctaaagcc ctgtgtaaag
     ttgaccccac tctgtgtcac tttagaatgt agacaggtta atgttacca
     caggttaacg ctaccagtaa tggagaagaa ataaaaaata taaaaaatt
     tcaaccacag aaataagaga taggaagcag acagcgtatc gacttttt
     ctagtaccac ttgataacaa gaatgggagc aactctagta agtatatat
     aatacctcag ccataacaca agcctgtcca aaggtcactt ttgatccaa
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk '/transla
tion/{while(i=1){print $0;getline; if ($1=="XX"){break} } }' U31362.1.gb|awk '{print
$2}'|tr -d '/translation="'
```

```
#Acessionnumber
awk '$1=="AC"{print $2}' U31362.1.gb |tr -d ':'
#Gene_name
awk '$1=="DE"{$1="";print $0}' U31362.1.gb
awk 'BEGIN{ORS=" "} $1=="DE"{$1="";print $0}' U31362.1.gb
awk '$1=="DE"{print substr($0,6)}' U31362.1.gb
#Test1 SQ
awk '/SQ/{getline;while(i=1) {print $1,$2,$3,$4,$5,$6;getline; if ($1=="//"){break}}} END{gsub(" ","1524",$0)}' U31362.1.gb
#SQ
awk '/SQ/{getline;while(i=1){print $0;getline; if ($1=="//"){break} } }' U31362.1.gb | tr -d '0-9'
#Translation
awk '/translation/{while(i=1){print $0;getline; if ($1=="XX"){break} } }' U31362.1.gb|awk '{print $2}'|tr -d '/translation="'
```

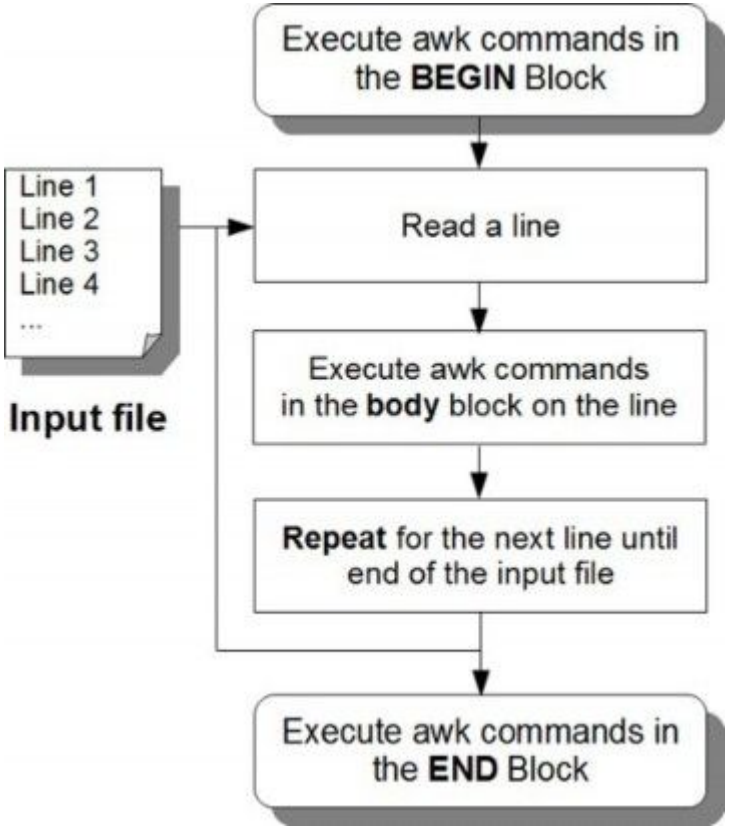
11.1 Introduction about awk

[AWK cheat sheet]

[Basic learning awk]

Command like
grep " ",
wc, tr, loop, echo,
condition, sed…

= **Awk**



https://vds-admin.ru/sed-and-awk-101-hacks/chapter-8-awk-syntax-and-basic-commands

Why we use awk in bioinformatic?

Column number	Title	Definition
1	chrom	Chromosome (e.g. chr3, chrY, chr2_random) or *scaffold* (e.g. scaffold10671) name
2	chromStart	Start coordinate on the chromosome or scaffold for the sequence considered (the first base on the chromosome is numbered 0 i.e. the number is zero-based)
3	chromEnd	End coordinate on the chromosome or scaffold for the sequence considered. This position is non-inclusive, unlike chromStart (the first base on the chromosome is numbered 1 i.e. the number is one-based).
4	name	Name of the line in the BED file
5	score	Score between 0 and 1000
6	strand	DNA strand orientation (positive ["+"] or negative ["-"] or "." if no strand)
7	thickStart	Starting coordinate from which the annotation is displayed in a thicker way on a graphical representation (e.g.: the start codon of a gene)
8	thickEnd	End coordinates from which the annotation is no longer displayed in a thicker way on a graphical representation (e.g.: the stop codon of a gene)
9	itemRgb	RGB value in the form R, G, B (e.g. 255,0,0) determining the display color of the annotation contained in the BED file
10	blockCount	Number of blocks (e.g. exons) on the line of the BED file
11	blockSizes	List of values separated by commas corresponding to the size of the blocks (the number of values must correspond to that of the "blockCount")
12	blockStarts	List of values separated by commas corresponding to the starting coordinates of the blocks, coordinates calculated relative to those present in the chromStart column (the number of values must correspond to that of the "blockCount")

Field

```
browser position      :127471196-127495720
browser hide all
track name="ItemR      mo" description="Item RGB demonstration" visibility=2 itemRgb="On"
chr7    127471196    127472363    Pos1    0    +    127471196    127472363    255,0,0
chr7    127472363    127473530    Pos2    0    +    127472363    127473530    255,0,0
chr7    127473530    127474697    Pos3    0    +    127473530    127474697    255,0,0
chr7    127474697    127475864    Pos4    0    +    127474697    127475864    255,0,0
chr7    127475864    127477031    Neg1    0    -    127475864    127477031    0,0,255
chr7    127477031    127478198    Neg2    0    -    127477031    127478198    0,0,255
chr7    127478198    127479365    Neg3    0    -    127478198    127479365    0,0,255
chr7    127479365    127480532    Pos5    0    +    127479365    127480532    255,0,0
chr7    12748053     127481699    Neg4    0    -    127480532    127481699    0,0,255
```

Why we use awk in bioinformatic?

Column number	Title	Definition
1	chrom	Chromosome (e.g. chr3, chrY, chr2_random) or *scaffold* (e.g. scaffold10671) name
2	chromStart	Start coordinate on the chromosome or scaffold for the sequence considered (the first base on the chromosome is numbered 0 i.e. the number is zero-based)
3	chromEnd	End coordinate on the chromosome or scaffold for the sequence considered. This position is non-inclusive, unlike chromStart (the first base on the chromosome is numbered 1 i.e. the number is one-based).
4	name	Name of the line in the BED file
5	score	Score between 0 and 1000
6	strand	DNA strand orientation (positive ["+"] or negative ["-"] or "." if no strand)
7	thickStart	Starting coordinate from which the annotation is displayed in a thicker way on a graphical representation (e.g.: the start codon of a gene)
8	thickEnd	End coordinates from which the annotation is no longer displayed in a thicker way on a graphical representation (e.g.: the stop codon of a gene)
9	itemRgb	RGB value in the form R, G, B (e.g. 255,0,0) determining the display color of the annotation contained in the BED file
10	blockCount	Number of blocks (e.g. exons) on the line of the BED file
11	blockSizes	List of values separated by commas corresponding to the size of the blocks (the number of values must correspond to that of the "blockCount")
12	blockStarts	List of values separated by commas corresponding to the starting coordinates of the blocks, coordinates calculated relative to those present in the chromStart column (the number of values must correspond to that of the "blockCount")

Record

```
browser position chr7:127471196-127495720
browser hide all
track name="ItemRGBDemo" description="Item RGB demonstration" visibility=2 itemRgb="On"
chr7    127471196    127472363    Pos1    0    +    127471196    127472363    255,0,0
chr7    127472363    127473530    Pos2    0    +    127472363    127473530    255,0,0
chr7    127473530    127474697    Pos3    0    +    127473530    127474697    255,0,0
chr7    127474697    127475864    Pos4    0    +    127474697    127475864    255,0,0
chr7    127475864    127477031    Neg1    0    -    127475864    127477031    0,0,255
chr7    127477031    127478198    Neg2    0    -    127477031    127478198    0,0,255
chr7    127478198    127479365    Neg3    0    -    127478198    127479365    0,0,255
chr7    127479365    127480532    Pos5    0    +    127479365    127480532    255,0,0
```

11.2 Field delimiter

“:” is a field separator

awk -F<value> **#Default is “ “ #Set input field separator**

```
(base) huyha@dummycomputer:~$ echo "Huy:M:150:18:Hochiminh" | awk -F: '{print $1}'
Huy
(base) huyha@dummycomputer:~$ echo "Huy:M:150:18:Hochiminh" | awk -F: '{print $1,$3}'
Huy 150
(base) huyha@dummycomputer:~$ echo "Huy:M:150:18:Hochiminh" | awk -F: '{print $0}'
Huy:M:150:18:Hochiminh
```

awk -f <filename> **#Reading commands to activate in a file**

```
echo '{print $1} END{print "this is the end of process"}' >awk1
awk -f awk1 Test1.txt
```

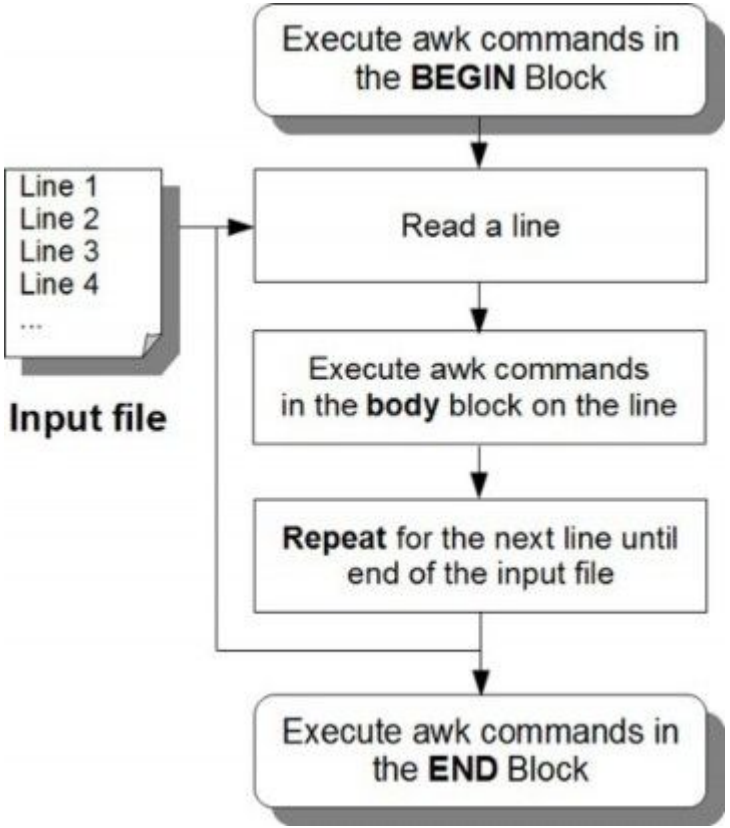
11.1 Awk structure

awk -option

‘**BEGIN**{action beginblock}
/**parten**/ {action block}

END{action endblock}’

inputfile



11.2 AWK Pattern

awk '/female/ {print $0}' Test1.txt

```
Linh female Haiphong 21
Mai female Hochiminh 26
Xuan female Binhdinh 53
Truc female Haiphong 21
```

awk ' $2 == male {print $0}' Test1.txt

```
Nam male Hochiminh 55
Vinh male Hanoi 12
```

awk '/chi/ {print $0}' Test1.txt

```
Nam male Hochiminh 55
Mai female Hochiminh 26
```

11.2 AWK Variable

```
# Access 'my_var'
inside the script

my_var="hello"

awk -v var="$my_var"
 'BEGIN { print var }'

hello
```

Build-in variables

`$0`	Whole line
`$1, $2...$NF`	First, second... last field
`NR`	Number of Records
`NF`	Number of Fields
`OFS`	Output Field Separator (default " ")
`FS`	input Field Separator (default " ")
`ORS`	Output Record Separator (default "\n")
`RS`	input Record Separator (default "\n")
`FILENAME`	Name of the file

Expressions

`$1 == "root"`	First field equals root
`{print $(NF-1)}`	Second last field
`NR!=1{print $0}`	From 2th record
`NR > 3`	From 4th record
`NR == 1`	First record
`END{print NR}`	Total records
`BEGIN{print OFMT}`	Output format
`{print NR, $0}`	Line number
`{print NR " " $0}`	Line number (tab)
`{$1 = NR; print}`	Replace 1th field with line number
`$NF > 4`	Last field > 4
`NR % 2 == 0`	Even records
`NR==10, NR==20`	Records 10 to 20
`BEGIN{print ARGC}`	Total arguments
`ORS=NR%5?",":"\n"`	Concatenate records

AWK Standard Variables (1)

FILENAME: is an AWK **built-in variable** that holds the name of the **current file being processed** by the AWK script.

Input: `awk 'END {print FILENAME}' myfile.txt`

Output: `myfile.txt`

FS: is an AWK **built-in variable** that represents the input **field separator**.

Input:

```
cat myfile.txt
```

```
,V1
```

```
R2,V2
```

```
awk  -v FS=',' '{print $1}' myfile.txt
```

Output:

```
R2
```

AWK Standard Variables (2)

NF: stands for "**Number of Fields**." Field is a part of the input line that is separated by a delimiter, which is **whitespace by default**.

Input: `echo -e "One Two\nOne Two Three\nOne Two Three Four" | awk 'NF > 2'`

Output:

```
One Two Three

One Two Three Four
```

NR: stands for "**Number of Records**." Record is typically **a line of input**.

Input: `echo -e "One Two\nOne Two Three\nOne Two Three Four" | awk 'NR < 3'`

Output:

```
One Two

One Two Three
```

AWK Standard Variables (3)

FNR: is similar to NR, but relative to the current file. When AWK processes multiple files, FNR represents the current line number within the current file being processed.

OFS: stands for "**Output Field Separator**." It determines the character or string used to separate fields when printing the output. The **default value is a whitespace**.

RS: stands for the "**Record Separator**" for input records. It specifies the character or string that separates records (lines) in the input. The **default value is a newline**.

ORS: stands for "**Output Record Separator**." It defines the character or string used to separate output records (lines) when printing the result. The **default value is a newline**.

$0: represents the entire input record (the entire line of input).

$n: represents the nth field in the current input record (line), where the fields are separated by the field separator **FS**.

getline #go down 1 row

11.3 Function

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk 'sub(" ", ":") {print}' Test1.txt
Name:sex city age
Linh:female Haiphong 21
Nam:male Hochiminh 55
Mai:female Hochiminh 26
Vinh:male Hanoi 12
Xuan:female Binhdinh 53
Truc:female Haiphong 21
```

sub(/a/,"b") #change first "a" to b in row

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk 'gsub(" ", ":") { print $0}' Test1.t
Name:sex:city:age
Linh:female:Haiphong:21
Nam:male:Hochiminh:55
Mai:female:Hochiminh:26
Vinh:male:Hanoi:12
Xuan:female:Binhdinh:53:
Truc:female:Haiphong:21
```

gsub(/a/,"b") # change every "a" to b in row

```
awk 'BEGIN{print substr("FirstName",6)}'
Name
```

substring("Hello",3) #return llo

11.4 Operator

Addition and Assignment: +=

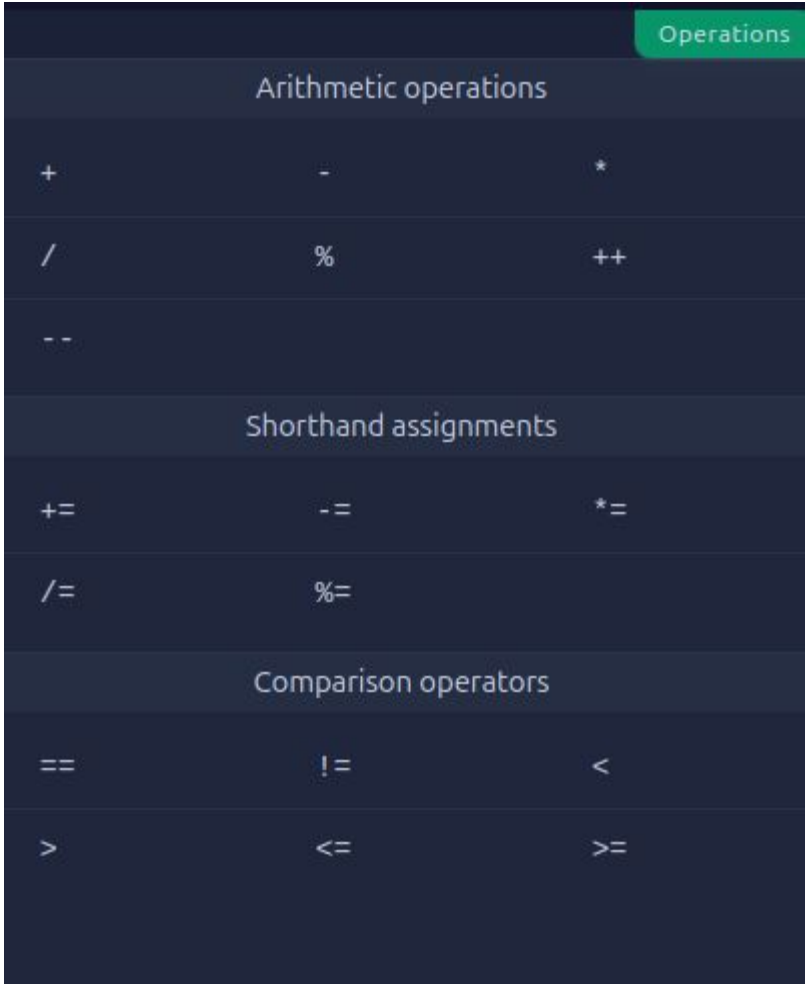
```
num = 5;

num += 3; // Equivalent to 'num = num + 3'

# Now 'num' is 8
```

Subtraction and Assignment: -=

```
num = 10;

num -= 4; // Equivalent to 'num = num - 4'

# Now 'num' is 6
```

Multiplication and Assignment: *=

```
num = 3;

num *= 5; // Equivalent to 'num = num * 5'

# Now 'num' is 15
```



11.4 Increment and decrement operator examples

Pre-increment:

```
num = 3;

result = ++num; // 'num' is incremented
to 3 before its value is assigned to
'result'

print num, result

# Now 'num' is 3, and 'result' is also 3
```

Post-increment:

```
num = 3;

result = num++; // 'num' is used as 3 in
the expression, then it is incremented to
4

print num, result

# Now 'num' is 4, and 'result' is 3
(previous value of 'num')
```

```
awk 'BEGIN {num = 3; preincr = ++num; print num, preincr;
            num = 3; postincr = num++; print num, postincr}'

4 4
4 3
```

11.5 Awk Condition

If statement #if(condition) {command block} else {command block}

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk '{if (NR > 1)  print NR,$0}' Test1.txt
2 Linh female Haiphong 21
3 Nam male Hochiminh 55
4 Mai female Hochiminh 26
5 Vinh male Hanoi 12
6 Xuan female Binhdinh 53
7 Truc female Haiphong 21
```

```
awk 'END{
        if  (NF % 2 == 0)
            {print "even"}
        else
            {print"odd"}
        {print NF}
        }' Test1.txt
```

```
awk 'END{if(NF % 2 == 0) {print "even"} else {print"odd"} {print NF}}' Test1.txt
```

```
even
4
```

11.6 Awk loop

#For (condition; variable; Change) {action block}

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk ' BEGIN{ for ( i=1; i<5; i++ ) { print i }}'
1
2
3
4
```

#While (condition) {action block}

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ awk ' BEGIN{ while ( i<10 ){ i+=2; print i}}'
2
4
6
8
10
```

```
# next statement
{
    if ($1 == "skip") {
        next;  # Skip processing this record and move to the next one
    };
    print "Processing:", $1;
}
```

```
# break statement
BEGIN {
    for (i = 1; i <= 10; i++) {
        if (i == 6) {
            break;  # Exit the loop when 'i' becomes 6
        };
        print "Iteration:", i;
    }
}
```

Homework 1

1. Create file test1.txt and test2 directory in /home/user (~)
2. Write script to find if the files in /home/user is directory or file or other type.

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ ls ~
a.out  Desktop  Documents  Downloads  homework  huyha  miniconda3  Music  Pictures  Public  R  snap  Templates  test2  Videos
```

```
(base) huyha@dummycomputer:~/huyha/homework/Huyha-Learning/baschscript$ ./homework2.sh
a.out is file
Desktop is directory
Documents is directory
Downloads is directory
homework is directory
huyha is directory
miniconda3 is directory
Music is directory
Pictures is directory
Public is directory
R is directory
snap is directory
Templates is directory
test2 is directory
Videos is directory
```

Awk homework 2

```
Down file using below command
curl -o "./U31362.1.gb" https://www.ebi.ac.uk/ena/browser/api/embl/U31362.1?download=rue
cat U31362.1.gb
```

1. Find accession number
2. Find product name
3. Find translation sequence

4. (optional) write script to automatic down file input = list of accession numbers

```
CDS             1..>1524
                /codon_start=1
                /gene="env"
                /product="envelope glycoprotein gp120"
                /db_xref="GOA:Q72858"
                /db_xref="InterPro:IPR000777"
                /db_xref="InterPro:IPR036377"
                /db_xref="UniProtKB/TrEMBL:Q72858"
                /protein_id="AAC55476.1"
                /translation="MGVRGILRNYQQWWIWGILGFWMLMICNVVGNLWVTVYYGVPVWE
                EAKTTLFCASDAKAYETEVHNVWATHACVPTDPNPQEIFLENVTENFNMRKNDMVNQMH
                EDVISLWDQSLKPCVKLTPLCVTLECRQVNVTSNGTQVNATSNGEEIKNIKNCSFNSTT
                EIRDRKQTAYRLFYRLDLVPLDNKNGSNSSKYILINCNTSAITQACPKVTFDPIPIHYC
                TPAGYAILKCNDKTFNGTGPCHNVSTVQCTHGIKPVVSTQLLLNGSLAEEEIIIRSENL
                TDNVKTIIVHLNQSVEIVCTRPNNNTRKSIRIGPGQTFYATGDIIGDIRQAHCNISEAK
                WNETLQRVRKKLAEHFPNKTINFTSSSGGDLEITTHSFNCRGEFFYCNQSGLFNGTYMH
                NGTKGNSSSVITIPCRIKQIINMWQGVGRAMYAPPIEGNITCKSNITGLLLVRDGGLGP
                SNDTETETFRPGGGDMRDNWRSELYKYKVVKIKPLGIAPTTAKRRVVERE"
```

Summary

1. Loop in bash script
2. Class work
3. Introduce about awk - how to manipulate the text
4. Basic awk
5. Home work