13.03.2024

- echo → print on the screen

  echo "Hello world!"

- V="test"

  ↳ declaring a variable

  echo "$V"

  v1="test2"

  echo "$v1" # test2

  echo "${V}1" # test1

- test.txt

  - remove all lines starting with a number

I) grep -v "^[0-9]" test.txt > test.txt

-v → revert

"^[0-9]" → selection group

II) grep "^[^0-9]" test.txt > test.txt

^ negates the selection group

> → overwrite

>> → append

III sed -i "/^[0-9]/d" test.txt

(/d → delete)

- du file.txt
  ↳ disk usage

- df
  ↳ disk free

- ls [ ] → . → current dir.
  ↳ list   .. → parent dir.
           / → root dir.

- ls -l

rights | Inode (↳ inode? ??) | owner | group | size (bytes) | date | name

_ | _ _ _ (owner) | _ _ _ (group) | _ _ _ (other)

d: dir. | r w x | r w x | r w x
l: link
p: ?

r → read, w → write, x → execute (file) / exchange (dir) (i.e. go into that directory)

read, write: access

- id + tells our user, primary group, ... of secondary groups.

base 8 system (zero)

owner group others

chmod 0[digit] text.txt

change mod. i.e. change permissions

r w x

$7 = 2^2 + 2^1 + 2^0$

ex chmod 451

→ r-- r-x --x

- 1.sh → shell script

./1.sh f1 f2 f3 ...

only for text files, remove all digits

chmode 700 1.sh

sh 1.sh → executes line by line

sh -x 1.sh → also prints each command after execution.

$? -> stores the exit code of the previous executed command

≥0 -> TRUE
<0 -> FALSE
↳ in LINUX

```
#! bin/sh
if test $? -eq 0 ; then
        echo "prev. cmd ok"
else
        echo "prev. cmd failed"
fi # closes the if
```

eq = equal

```
if ___ ;then
___
elif ___ ;then
___
elif ....
.....
else
___
fi
```

---

variable (name does not matter)

```
for i in $@ # all arguments (excludes the 1st arg.)
do
    if file $i | grep -q ":.*text" ; then
        num=`grep -i '^#[0-5]##g' $i`
    fi
done
# file 1.sh : shell ascii text file
```

-q = quiet (do not print on screen)

pipe: output of command1 → input → second command

substitute this with this, to not print on screen

grep -o: only will print only the part which was matched

file text.txt : binary ...

OBS: extensions are only clues for humans!!

Name FirstName grade

2.sh grades.txt

```
#!/bin/sh
if [ ! $# -eq 2 ]; then
    echo "Usage: $0 textfile"
    exit 1
fi
file = $1 # variable
if ! file $file | grep -q ":.*text"; then
    echo "$file is not a text file"
    exit 2
fi
sed -Ei '\n/^([^\n]+)\n+'
```

test ... = [...]

no. of arguments

negate

in place

beginning of word

whitespace

end of word

beginning of word

Name

( ) → to define a group in regex

First Name

(<[^\s]+>)\s+(.*)$/\2\n\3/" file

\2 <- record group from regex

awk -> better at dealing with formatted text files

{...} -> statements to be executed

{...} -> default block | BEGIN{...} | END{...}

$0 in awk is current line { NF -> number of fields (predefined variable)

$1 is the first field

awk '{ printf("%s %s", $2, $1);

for (i=3; i<=NF; i++)

printf(" %s", $i);

printf("\n");

}' $file > $file

1st field

split($1, a, ":")

Name:v1 FirstName:v2 grade:v3

a[1] = Name, a[2] = v

print $0

index ($2, ":")

length ($0)

length (a)

gsub ($4, "a", "b") ; #replace 'a' with 'b' in 4th field

globally substitute

4th field (where)

ARGS →

FILENAME → current filename to be executed.

NR → no. row

NF → no. fields

Questions: 1) What does sed / grep stand for?

2) "-i" in sed command.