Lecture 3

Recap

- truth value = every command execution is a truth value
- variables, how they're created (A=5 B="asdf")
- how to use variables' values = echo $A ${B}

Control mechanisms

FOR

```
for X in a b c 5; do # (equivalent to {)
  echo $X
done # (equivalent to })
```

can be written as well:

```
for X in a b c 5
do # (equivalent to {)
  echo %x
done # (equivalent to {)
```

Logical Expressions

- they are commands being executed behind the scenes

```
! test -f a.txt # gives true if file does not exist,


! test -f a.txt || \ #newline, needed to not get an error
( test -f a.txt && \ #newline
! test-r a.txt )
# gives true if file does not exist or if it exist but it's not readable
# add proper spaces or you get syntax error, space after and before paranthesis
```

IF

```
if true; then
...
elif false; then
...
elif ...; then
else
...
fi
```

WHILE

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

Examples

Cheat sheet

```
#! /bin/bash
# comment that starts any script
chmod # change rwe permissions
```

$@ - all the arguments you give in the execution of the file

break[n], where n is the number of loops you want to exit

continue

```
uniq <<BLABLA
asdf
sadf
asdf
asdf
asdf
qwer
qwer
zxcv
zxcv
zxcv
xzcv
BLABLA # removes duplicates of the lines next to eachother
# asdf
# sadf
# asdf
# qwer
# zxcv
# xzcv
uniq -c <<BLABLA # also outputs the number of appearances
```

```
# to access the vim config files
vim ~./vimrc
# ~ -> home directory
```

```
# Meaning of this regexp
elif echo $X | grep -E -q "^[0-9]+$"; then
echo the string x, grep it look and tell me if you find -E the extended regular expression: from the beginning of the line (^) it cont
```

```
read X Y Z
gea qegaeg gaeg gaegads
echo $X
# gea
echo $Y
# qegaeg
echo $Z
# gaeg gaegads
```

```
if test "$X" == "stop";
# the quotes in "$X"
# if x is empty, then it tests nothing to stop and it stops
# with quotes, you get a blank argument when entering and it work

test == stop
# test: ==: unary operator expected
```

```
test "" == stop
# works
test adsf sdf == stop
# test: too many arguments
test "adsf sdf" == stop
# works
test adsf\ sdf == stop
# works
```

Files

a.sh

```
#!/bin/bash

# file a.sh: Prints every command line argument

for X in  $@; do
  echo $X
done

echo "======"

for X; do
  echo $X
done
```

b.sh

```
#!/bin/bash

for X in $@; do
  if file $X | grep -E -q "ASCII"; then
    echo "$X is a text file"
  elif test -f $X; then
    echo "$X is a file"
  elif test -d $X; then
    echo "$X is a directory"
  elif echo $X | grep -E -q "^[0-9]+$"; then
    echo "$X is a natural number"
  else
    echo "$s is weird"
done
```

c.sh

```
#!/bin/bash

# Read from the command line, and if the input stops, it stops,
# with the command read

while true; do
  read -p "Enter a string: " X
  if test "$X" == "stop";
    break
  fi
  echo "=== $X"
done
```

Command line

may be a comment, or the output of a command

```
# give execution permissions
ls -l
# total 4
# -rw-rw-r--
chmod 700 a.sh

./a.sh asdf eqrt y tyh # first file
#asdf
#eqrt
#y
#tyh
```

```
./a.sh asdf eqrt y tyh # second file
#asdf
#eqrt
#y
#tyh
#======
#asdf
#eqrt
#y
#tyh

file a.as # command that tells us what is written in the file (not cat): file
#a.sh: Bourne-Again shell script, ASCII text executable
file a.sh | grep -E "ASCII"
# a.sh: Bourne-Again shell script, ASCII text executable
# if grep found "ASCII", it gives a 0 as exit code
echo $?
# 0 -> true
file a.sh | grep -E "ASCIIewrwerew"
# 1 -> false
# now, let's tell grep to be quiet and not print everything
file a.sh | grep -q -E "ASCII" # -q does not output anything
# nothing
echo $?
# 0

cp a.sh b.sh
./b.sh a.sh ./bin/ls /etc 1234 12h123h
# a.sh is a text file
# /bin/ls is a file
# /etc is a directory
# 1234 is a natural number
# 12h123h is weird

cp a.sh c.sh
chmod 700 c.sh
./c.sh
Enter a string: asdf
=== asdf
Enter a string: sadf
=== sadf
Enter a string: sadf
=== sadf
Enter a string: stop
```

The most popular name out of all the students

```
# /etc/passwd all students list
# example name:
# ex93:x:2035:2036: andrei.buiciuc - 931 - BUICIUC A. ANDREI: /home/bla bla

awk -F: '{print $5}' /etc/passwd # something something he's really fast
# andrei.buiciuc - 931 - BUICIUC A. ANDREI

awk -F: '{print $5}' /etc/passwd | sed "s/\..*//"
# andrei
# buiciuc

awk -F: '{print $5}' /etc/passwd | sed "s/\..*//"|sort|uniq -c|sort -n -r| head -n 10
# 90 andrei
# 62 alexandru
# 54 andreea
# 36 robert
# ...
# uniq removes duplicates that are next to eachother, that's why we sort
# head -n 10, prints first ten lines
# "s/\..*//"
# s - search for
# / - separator
# \..*// - search for
# / 1st separator
# / 2nd replace with nothing

# remove diacritics, with tr
awk -F: '{print $5}' /etc/passwd | tr 'ȘȚĂÎÂ' 'STAIA' # there is another Ș apparently to be added

# delete everything until the first capital letter
awk -F: '{print $5}' /etc/passwd | tr 'ȘȚĂÎÂ' 'STAIA' | sed "s/[^A-Z]*//"
s - from the beginning of line
[^A-Z] - not capital letter
* - any number of times
// - replace with nothing
```

```
#everything becomes lowercase and sort, initials still there
awk -F: '{print $5}' /etc/passwd | tr 'ȘȚĂÎÂ' 'STAIA' | sed "s/[^A-Z]*//" | tr '[A-Z]' '[a-z]'
|sed "s/[%a-z]/\n/g" | sort | uniq -c |sort -n -r| less


# remove initials
# g - global, every occurance on the line
# sort -n -r
# -n -> numeric
# -r -> reverse
awk -F: '{print $5}' /etc/passwd | tr 'ȘȚĂÎÂ' 'STAIA' | sed  -E"s/[^A-Z]*//" | tr '[A-Z]' '[a-z]'
|sed -E "s/[^a-z]/\n/g" | grep -E "..."| sort | uniq -c |sort -n -r| head -n 10
# 156 andrei
# 128 alexandru
# 77 maria
# 60 mihai
# 59 andreea
```

Processes

- every program in execution is a process

```
ps -ef # print processes
ps -ef | wc -l # print number of processes
```

Script that reads a file with a number, it increments the number and writes the number back 200 times.

d.sh

```
#! /bin/bash

F=$1
N=0

#lt - less than
while test $N -lt 200; do
  K='cat$F' # read line from file
  K='exp $K + 1' # increments
  echo $K > $F
  N='expr $N + 1'
done
```

e.sh

```
#!/bin/bash
echo 0 > x

./d.sh x &
./d.sh x &
./d.sh x &
./d.sh x &
./d.sh x &
# & throws command in background, doesn't wait for it before next command
```

```
chmod 700 d.sh
echo 0 > x
./d.sh x
cat x
# 200
./d.sh x
cat x
# 400
./d.sh x
cat x
# 600
chmod 700 e.sh
./e.sh
ps
cat x
# 7
./e.sh
ps
```

```
cat
# 17
./e.sh
cat x
# 19
```

HW: lookup n++, with the exception to the latest mac os updates (m1 i think)

translates into assembly

```
LOD AX, n
INC AX
STO AX, n
```