# SISTEME DE OPERARE

**Curs 1-3**

**Logare server: cu Putty**

Sau comanda **ssh**

Ctrl+C opreste executia comenzii actuale si inchide procesul

Ctrl+Z stopeaza executia, dar procesul ramane (il gasim cu ps)

Ctrl+S – blocheaza consola de Putty (deblocare cu Ctrl+Q)

**Comenzi: (structura unei comenzi) nume -optiuni argumente**

Echo, whoami

who

**passwd**

ls -l , cat,

mkdir, cp, mv, rm, rmdir, file,

**find / -name \*.c ; find . -type d ; find . -perm 664**

man [sectiune] comanda ; man 3 functii C

man -K keyword

**last, sort, uniq, head, tail, wc**

exit , logout

**Editare fisiere:**

vim, joe, nano, emacs

**vim editor**

* 2 working modes: INSERT / COMMAND
* shortcuts: i, Esc, x, dd, dw, /, V, y, p
* commands: :w, :q, :x, :q!
* autoindent: gg=G
* configure vim by creating **~/.vimrc** file with content:

syntax on  
set tabstop=4  
set expandtab

<https://www.openvim.com/tutorial.html>

<https://vim-adventures.com/>

**Specificarea fisierelor:**

* Cale absoluta cu / (windows \), relativa, generic cu simbolurile wildcard \* ? [oricelitera][!exceptiecaracter]
* Cu aprostroafe sau ghilimere cand avem spatii sau caractere speciale ‘ ‘ “ “

**Fisere standard:**

* Stdin 0
* Stdout 1
* Stderr 2

**Redirectari si pipeing:**

command < fis\_in # redirection of input from fis\_in

command > fis\_out # redirection of output in fis\_out (by rewriting)

command >> fout # appends output to fout

command 2> fis\_out # redirection of error messages in fis\_out (by rewriting)

command 2>> fout # appends error messages to fout

command1 | command2 # the output of the first prima command is piped (as input) to the

second command

separator comenzi: **&&** - and ; **||** - sau ;

(list) list is executed in a subshell environment

{ list; } list is executed in the current shell environment

((aritmExpr)) aritmExpr is arithmetically evaluated

[[ condExpr ]] Returns a status of 0 or 1 depending on the evaluation of the conditional expression

**Moduri de executie**

* Foreground : ls
* Background : sort > B **&**

**Problema: Top utilizatori care petrec timpul pe server:**

last | awk '{print $1, $9}' | sed 's/:/ /' | awk '{print $1, $2\*60+$3}' | awk -f prog.awk | sort -k2 -n -r

(optional si |head -7 vom avea top 7)

[alinacalin@nessie ~]$ cat prog.awk

{ arr[$1]+=$2 }

END {for (u in arr) {

print u, arr[u]

}

}

**Variabile predefinite**

**HOSTNAME** machine name (*linux.scs.ubbcluj.ro*)

**HOME** user's host directory (*/home/scs/an2/gr321/snmr0123*)

**PATH** searchable paths of executable files

**LOGNAME** the name under which the user opened his work session (*snmr0123*)

**SHELL** the type of command interpreter that is used (*/bin/bash*)

**TERM** the type of terminal that is used (*xterm*)

**MAIL** the file containing the user's e-mail (*/var/spool/mail/snmr0123*)

**IFS** shell separators for words

**PS1** the main Unix prompt (*[\u@\h \W]\$*)

**PS2** the Unix secondary prompt (*>*)

6

printenv # listeaza variabilele de mediu

**REGEX – Expresii Regulare Extinse (Extended Regular Expressions)**

**<\ word \><word\>** - defining a word   (end of word \>       beginning of word \<)  
**[A-Z0-9]** - any of the set of characters in []  
      - ex [123] - any of the digits 1, 2, 3  
           [123 ] - 1, 2, 3, or space  
           [a-z] - any lowercase letter  ([[::alpha::]])  
           [aeiou] - any vowel  
**[^...]** - any character except those between []  
       - ex [^0-9] - any character that is not a digit ([[::digit::]]  )  
            [^,.:] - anything except , . or :  
**\*** - previous expression 0 or more times (01)\*   
**+** - previous expression 1 or more times 0+ : 0, 00, 0000

**?** - previous expression 0 or once -?[0-9]+\.[0-9]+ -1.5 0.567

Colou?r -> Color Colour

**.** - any  character, once   
**\** - escape to special characters \?   
**^** - beginning of line ^a   
**$** - end of line ^[0-9 ]\*$

**(** group-expr **)** - group regular expression

**{M,N}** - M to N duplicates of the previous expression 8{3,} : 888 , 88888

01{2, 5} : 011 , 0111, 01111, 011111

(ABC){2} : ABCABC

**|** - or  \<dan\>|\<ana\>

QUIZ REGEX ACTIVITATE BONUS

**Grep – cauta pattern in fisier text**

**grep [optiuni] regex inputfile.txt**

**Optiuni importante:**

-E: use extended regular expressions

-i: ignore the case of your search term

-v: show lines that don’t match, instead of those that do

-c: instead of returning matches, return the number of matches

-x: return only an exact match

-m: stop reading file after n number of matches

-n: print the line number of where matches were found

-q: don’t output anything, but exit with status 0 if any match is found (check that status with echo $?).

-o: print only the matching part of the line

-w: find matches surrounded by space

--color: add color to the matched output

exemple: find all lines that start with a digit

grep -E --color "^[1234567890]" a.txt

Find all lines from a file that contain the word dan

grep -E “\<dan\>” a.txt

Find all lines from a file that start with a word ending in “ing”

grep -E “^[a-zA-Z]\*ing\>” a.txt

Find all lines from a file with odd number of charaters

grep -E "^(..)+$" a.txt

**TUTORIAL:** [**https://www.grymoire.com/Unix/Grep.html**](https://www.grymoire.com/Unix/Grep.html)

**Pentru fisier text de input utilizati fisierul /etc/pseudopasswd . Fisierul standard /etc/passwd contine doar cativa utilizatori.**

**Sed – cauta si inlocuieste pattern in text**

Find and replace: sed **s**/findregex/replace/**gi** file.txt

-E: use extended regular expressions

g-global replacement

i- case insensitive

\1, \2 ... for refering matching groups from the find section with \( group\) in the replace section

& - matching pattern

Transliterate: sed **y**/ab/AB/ file.txt

Delete: sed /regex/**d**file.txt

ex: Replace s with sh

sed -E 's/s/sh/gi' a.txt

Add the prefix 'abc' to each line in a file

sed -E “s/^/abc/” a.txt

Replace any empty line in a file with the word 'empty'

sed -E “s/^$/empty/” a.txt

Rotate with one position every triplet of characters in a file :

ex:  abc => bca

sed -E “s/(.)(.)(.)/\3\1\2/g” a.txt

Double vowels

Sed -E “s/[aeiou]/&&/gi” inputfile.txt

**TUTORIAL**: <https://www.grymoire.com/Unix/Sed.html>

**Awk**  
The essential organization of an AWK program follows the form:  
pattern { action }  
BEGIN          { print "START" }

  /search regex/   { print         }

END            { print "STOP"  }

-Blocks, selectors (BEGIN, END)

**Basic systax**

a) There are only a few commands in AWK. The list and syntax follows:

if ( conditional ) statement [ else statement ]  
while ( conditional ) statement  
for ( expression ; conditional ; expression ) statement  
for ( variable in array ) statement  
break  
continue  
{ [ statement ] ...}  
variable=expression  
print [ expression-list ] [ > expression ]  
printf format [ , expression-list ] [ > expression ]  
next   
exit

b)Built in variables  
print $0 - reffers to entire line  
print $1, $2, $3, $4, $5, $6, $7, $8 - refers to each field

NF -numver of fields variable

NR -number of records (line number)

FILENAME - gives th name of the file being read

c) Change File separator  
  AWK can be used to parse many system administration files. However, many of these files do not have whitespace as a separator. as an example, the password file uses colons. You can easily change the field separator character to be a colon using the "-F" command line option. The following command will print out accounts that don't have passwords:

awk -F: '{if ($2 == "") print $1 ": no password!"}'   /etc/passwd

------------------------------------  
#!/bin/awk -f  
BEGIN {  
 FS=":";  
 }  
{  
 if ( $2 == "" ) {  
 print $1 ": no password!";  
 }  
}

**Useful functions**

* **length(s)**
* **substr(s,p,n)**, **index(s, t)**
* **split(s, a , c)**, **sprintf(format, arg, ...)**

**Running awk file scripts**  
----------------------------------------------------  
A simple shell scrip file name hello.sh  
#!/bin/bash  
echo Hello World  
  
chmod +x hello.sh or chmod 755 hello.sh  
./hello.sh  
------------------------------------------------------

a)As a shell script file name ex1.sh  
#!/bin/sh  
# this is a comment  
awk '  
BEGIN { print "File\tOwner" }  
{ print $8, "\t", $3}  
END { print " - }

chmod +x ex1.sh  
./ex1.sh

b)As an awk script file name ex2.awk  
#!/bin/awk -f  
BEGIN { print "File\tOwner" }  
{ print $8, "\t", $3}  
END { print " - DONE -" }

awk -f ex2.awk input

**Examples:**  
Having a file in which each line contains at least 2 numbers separated by space, calculate

a) sum of first 2 numbers on each line

b) sum of first 2 numbers on odd lines

awk '{if (NR % 2 == 1) print $1+$2}' num.txt

c) sum of first 2 numbers on odd lines with more than 5 numbers on a line

awk '{if (NR % 2 == 1 && NF > 5) print $1+$2}' num.txt

d) sum of numbers in the first column of the file

awk 'BEGIN {n=0} {n=n+$1} END {print n}' num.txt

e) sum of all numbers in the file

awk '{for (i=1; i<=NF; i++) n=n+$i} END {print n}' num.txt

**TUTORIAL**: <https://www.grymoire.com/Unix/Awk.html>

**Alte exercitii (unele nu se aplica la continutul actual al /etc/pseudopasswd deci este posibil sa nu existe linii care se potrivesc – adaptati cerintele)**

1. Display the lines in /etc/passwd that belong to users having three parent initials in their name, even if the initials do not have a dot after them. You will notice that the extended regular expression accepts things that are not really parent initials, but there isn’t much else that we can do ...

grep -E " [A-Z]\.?[A-Z]\.? [A-Z]\.? " /etc/passwd

2. Display the lines in /etc/passwd that belong to users having names of 12 characters or longer (this year there is one with a 13 character name)

grep -E -i "^([^:]\*:){4}[^:]\*[a-z]{12,}" /etc/passwd

3. Convert the content of /etc/passwd using a sort of  Leet/Calculator spelling (eg Bogdan -> B09d4n)

sed "y/elaoszbg/31405289/" /etc/passwd

4. Convert the content of /etc/passwd surrounding with parentheses and sequence of 3 or more vowels

sed -E "s/([aeiou]{3,})/(\1)/gi" /etc/passwd

5. Display the full names (but only the full names) of the students belonging to group 211

awk -F: '$6 ~ /\/gr211\// {print $5}' /etc/passwd

6. Count the numbers of male and female users in /etc/passwd, accepting as true the following incorrect assumptions: All users have their last name as the first name in the user-info filed (5th field) All women have one of their first or middle names ending in the letter “a”

awk –F: -f prog.awk /etc/passwd  
------------prog.awk---

BEGIN {    m=0    w=0}  
# The space at the beginning of the regular# expressions is for not matching the last name

$5 ~ / [a-zA-Z]\*[b-z]\>/ {    m++}  
$5 ~ / [a-zA-Z]\*a\>/ {    w++}  
END {    print "Men:", m    print "Women:", w}

---------------------