**SEMINAR 1**

* Resources at the end of the webpage:
* cs.ubbcluj.ro/~rares/course/os/
* Shell programming – first half of the semester
* C programming – 2nd half
* We’ll focus on using the command line on the server.

Commands have options and arguments.

Teachers recommendation: go through the tutorial (???).

On the website u have tutorials and exercises and so on. TO DO at home!

Commands:

* ls – in Linux (dir in Windows)

ls lists all the files in the current directory

ls has a few options:

-l (L mic)

Ex.:

ls -l

ls -l dir1

* man

man \*command = shows what each command does and all it’s options

for ex: man ls

cat /etc/passwd

* Shows all users (?) on a line, each field is separated by “:”
* ex… is the user, x is the password, same for everyone

cut -d: -f1,5 /etc/passwd

-d: option, -f is the option, 1 and 5 are the fields we want, /etc/passwd is the file we’re extracting information from

3 PR USED TO PROCESS TEXT:

1. grep = print lines matching a pattern

grep Dan /etc/passwd

this selects all the lines that have “Dan” in them

* to make it more specific: we need a language that describes the pattern we are looking for
* for this we have regular expressions:
  + . = matches any single character
  + \ = changes the meaning of the character following it, with this for ex.: \n changes the meaning of “n” to “new line”;  
    this works only for some characters, that are usually not special characters  
    \ is called “escape”
  + [abc] – matches any single character that appears in the list, in this case a,b or c
  + [a-z] matches any single character that belongs to the range, all the lowercase letters in this case
  + [^0-9] matches any character that does NOT belong to the range
  + ^ = beginning of the line
  + & = end of the line
  + \< = beginning of the word
  + \> = end of the word
  + ( ) = group several characters into an expression
  + \* the previous expression zero or more times
  + + = a plus needs at least 1 char to match it, while the star can match 0 characters; the previous expr at least once!
  + ?
  + {m,n} = previous expression at least m times and at most n times;
  + | = this is a logical OR

grep “\<Dan\>” /etc/passwd – shows only the lines that have the word Dan

grep “^.\*$” /etc/passwd

grep “[aeiou]” /etc/passwd – searches for any vowel

grep “[aeiouAEIOU]” /etc/passwd – searches for any vowel lower and uppercase

grep -i “[aeiou]” /etc/passwd – searches for all the vowels case insensitive

-i option ignores the case ( so works for both uppercase and lowercase)

-v – showcase lines that don’t have a certain char

grep -I -v “[a]” /etc/passwd –

-v means it shows all lines that don’t have anything red, so in our case an a or an A

Grep -i “dan” /etc/passwd – all lines that contain “dan” in them, case insensitive

grep -i -o “dan” /etc/passwd – only the red parts ( so every appearance of dan, case insensitive )

grep -i -v -c “dan” /etc/passwd – -i case insentive, -v all that do not have dan, -c counts: so it shows the number of lines that do not have dan in them

CHECK ALL USERNAMES WITHOUT DIGITS IN THEM:

grep -E “^[^0-9]+”

this matches as much as it can

to stop before the first “:”, to take only the username, we do:

grep -E “^[^0-9:]+:” – this goes from the beginning of the line (^) and takes everything that is not in range 0-9 or “:” and stops at the “:”

CHECK ALL USERNAMES WITH AT LEAST 2 VOWELS:

grep -E “^[aeiou]+:” – this matches usernames with only vowels

grep -E “^ [^:]\* [aeiou]+[^:]\*[aeiou]:”

[^:]\* - as many char that are not “:”

Grep -E “^ [^:]\* ([aeiou]+[^:]\*){2,}:”

([aeiou]+[^:]\*){2,} searches for at least 2 vowels

{3} – searches for exactly 3 vowels

{,4} – less than 4 ( this sets the upper limit)

{3,4} – between 3 and 4 vowels etc.

grep -E -v “.” /etc/passwd – all characters that don’t match “.” – at least 1 character ⬄all empty lines

grep -E -v “^(..)\*$” /etc/passwd – takes from the beginning of the line ( ^) and goes until the end of it ($) and repeats 2 characters as many times as needed. -v means that it’s the ones that do not go according to that rule, meaning that we take all the lines with an odd number of characters in them;

nano file.extension / vim file.extension to edit the file ( or create and write into it)

cat file.extension to show the content of the file

**EXTRACT EMAILS FROM A FILE WITH RANDOM TEXT:**

grep – E -i – case insensitive

grep -E -i “\<

email@asda.asd

\< - the beginning of the word

Then we have the characters in the email:

Any character except @\*!?:

[^@\!?\* ]+ - any char that is not in that list

Then we have the char “@”

Then we have the gmai/yahoo/whatever:

[a-z0-9\_-]+ - so at least one character from a-z, digit, \_ or – at least once ( the + shows that)

(\.[a=z0-9\_-]+)

We use \! And \. Because those 2 characters have a special meaning, and we want to work with the actual characters “!” and “.”

Grep -E -i -o “\<[^@\!\*? ] [+@[a-z0-9\_-]+(\.[a-z0-9\_-]+)+\](mailto:+@[a-z0-9_-]+(\.%5ba-z0-9_-%5d+)+\)>”

1. sed

* sed -E “s/dan/#####/gi” /etc/passwd – the i at the end means case insensitive
* this replaces dan with #####, case insensitive because of “i”;
* it replaces all the dans in a line because of “g”
* sed -E “s/dan/####/gi” /etc/passwd | grep “#####” = only shows the lines that contain “#####”, and it showcases the changes with red
* sed -E “s/../2/1/” /file – this replaces any 2 characters the 2nd one with the 1st one ( /2/1 )
* the changes in the file are not SAVED, they are only in the OUTPUT
* to save them:
* sed -E “…” /etc/passwd > new\_file.txt – an example to save them
* sed -E “y///” /file – this codifies a text
* sed -E”y/aeiou/1/2/3/4/5” /file – this codifies the text: all vocals are changed, a – 1, e-2,…
* sed -E “y/t/T/” /file
* sed -E “//d” /file – this deletes the line that have a certain smth in them
* sed -E “/dan/d” /file | grep dan – shows nothing, because all lines that have dan in them are deleted
* it is ⬄to grep -v dan /file | grep dan

1. awk

* each line is a field, and it displays them as tables
* awk looks at things like a table, and u need to specify the delimiter
* example:

BEGIN {s=0 }

Do smth for each line in the file:

…….

END print s

* special VARIABLES:
  + NR – the current line index we are on
  + .
  + .
  + .
* awk -F ( to specify the delimiter)
* awk -F: ‘{print $1}’ /etc/passwd – prints the first field from each line, considering “:” the delimiter
* $1 – means the first field
* $2 – the 2nd field and so on…
* awk -F: ‘/dan/ {print $1}’ /file – this only prints the 1st field of the lines that contain dan (similar to grep)
* awk -F: ‘NR % 2 ==1 {print NR, $1, $5}’ /file – this checks that the current line is on an odd position (1st line, 3rd line, 5th line and so on…)
* if you write a program:
* prog.awk :=
* BEGIN {…}  
  code  
  END{….}
* And run it like awk -F: -f prog.awk /file-that-you-want-to-operate-on