Lecture 1 - 2014/10/01

Few notes Slides

- Course introduction
- UNIX history
- UNIX philosophy
- Introduction to the Shell

Lecture 2 - 2014/10/08

Linux Filesystem

name case sensitive
only 1 directory tree

The directory are organised per meaning of file ... each directory contains a given type of file, obviously you can change it but is better to keep order

DIRECTORY
/sbin, /bin: contains the essential programs (sbin for root)
/usr: contains the most of the users applications
(on /usr/bin, /usr/sbin there is non critical applications like /bin,/sbin)
/etc:
/tmp:
/proc: contains file that represent info about the kernel, it is read-only, not relevant for our course
/home: contains all the home directories (file of the users)
/root: contains the home of the root user
/lib: contains the shared libraries
/var: contains the data written by the applications during the normal activity
(/usr can be read-only, so it is important), usually there are the logs

extensions under unix are only for humman!
A file is an executable if is enabled a flag, the type of the file is detected from its content

Permission on directory:
read: allow to list the files
write: allow to make new files
execute: allow to traverse the directory

EACH FILE HAS 3 MORE BITS
SUID, setuid: allow to execute the file with the permissions of the owner of the file
SGID, setgid: allow to execute the file with the permissions of the group of the owner of the file

HARD LINKS: allow to have "different files" that points to the same file on the disk
(1 file several file name). kernel keep track of the number of the links ===> is like an ALIAS
constraints: no hard link for directory and for file of different partition

if you look the result of 'll' or 'ls -l' you can see a number after the permissions,
this number represent the number of filename associated to the same 'physical' file

SOFT LINKS, symbolic links: is a pointer to a file name (like a link on HTML page)
if you delete the linked file, the link remain there but is DANGLING
if you delete the link you don't delete the linked file

Lecture 3 - 2014/10/22

LINUX TOOLS (2nd part) => Filtering

The most common in "grep" (egrep for the extended version, useful for complex pattern)
there is sort, uniq, ...together allow you to execute "all" set operations
- intersection =>sort ... ... | uniq -d
- union => sort ... ... | uniq
- complement => sort A B B | uniq -u (shows elements of A not in B)
- exclusive or => set ... ... | uniq -u (shows elements not in the intersection)

the instructions work in the assumption of unique lines in the starting files

EXAMPLES

find all broken symbolic links (not with the option)
if I run "file ..." it returns information about the ...
if the ... is a broken simbolic link you can read "broken symbolic link"

```
find . -exec file {} \; | grep "broken"
```

find all the word that is repeated twice (one after the other)

```
cat ... | egrep " ([a-zA-Z]+) \1"
```

Lecture 4 - 2014/10/27

LINUX TOOLSET (part B)

AWK is a programming language. it allows to manipulate strings like sed BUT
=> FOR EACH MATCHED PATTERN IT EXECUTES A PARTICULAR ACTION <=

fields and records are specified by special variables
- $number, in general $(math expression) => field number
- NR => current record
- NF => number of fields in the record (by default the number of words)
- OFS => output field separator
- RS => records separator (by default is the space but you can change)
  - RS="" (RS equal to empty) means that the separator is an empty line
- FS => fields separator

BETTER TO SEE THE SLIDES / WIKIPEDIA / GUIDES

some examples in the slides

variables: automatically initialized to 0 or empty string
- automatic cast to nimber for math operations
- arrays are associative (the index can be also a string) =>
  - WARNING array[50] is the same element of array["50"]
- index in array => says if the index is an array's index => TRUE / FALSE

piping: print ... | cmd calls cmd and puts in its stdin all the printed lines
BETTER TO HAVE PRACTICE...strange behaviour can happen but we have to understand

AWK is very useful for math operations on the shell, text reformation (for example for CVSs)

Examples:
- print the file with one word per line
```
cat ... | tr ' ' '\n'
cat ... | sed 's/ /\n/g'
for word in $(cat ...); do echo $word; done
```
- count how many times is wrote the word XXX in the file FFF
```
cat FFF | grep -oc "XXX"
cat FFF | grep -o "XXX" | wc -l
```
- show the most common word in the document FFF
```
cat FFF | tr 'A-Z' 'a-z' | egrep -o "[a-z']+" | sort | uniq -c | sort -nr | head -n 1
```
- like the previous but get rid of the characters after the apostrophe '
```
cat FFF | tr 'A-Z' 'a-z' | egrep -o "[a-z']+" | egrep -o "^[a-z]*" | sort | uniq -c | sort -nr | head -n 1
cat FFF | tr 'A-Z' 'a-z' | sed '/([a-z]+)'?[a-z]*/ print \1' | sort | uniq -c | sort -nr | head -n 1
```
- From the log file prints all the user and command executed with sudo privileges
```
cat /var/log/auth.log | grep "COMMAND" | sed -r 's/^.*sudo:[ ]+([^ ]*).*COMMAND=([^ ]+)/\1 -- \2/'
```

AWK
- print the first word of each line of FFF with awk
```
cat FFF | awk '{print $1}'
```
- sum all the numbers in the first column of the fike FFF
```
cat FFF | awk '{total+=$1} END {print total}'
```
- show the longest word in the document
```
cat FFF | tr ' ' '\n' | tr 'A-Z' 'a-z' | egrep -o "^[a-z]+" | awk '{print lenght($0) $0}' | sort - r | uniq
```
- sum the number of bytes in the directory grouped per each file extension
```
ls -l | egrep "[ ][^ ]+\. [^ ]+$" | sed -r 's/^.*[a-zA-Z]+ [a-zA-Z]+[ ]+([0-9]+)\. (.*)$/\1 \2/' |
    awk '{total[$2]+=$1} END { for(ext in total) print ext, "total bytes ", total[ext]}'
```

IN THEORY WE CAN PUT ALL INSIDE AWK BUT IS BETTER TO KEEP SIMPLE THE CODE