Introduction to Unix commands, editors and other commands

20/1/21

Outline

- Why Unix?
- How to connect to a Unix machine?
- Unix commands Overview
- Why and when to write scripts?
- Helpful Unix commands examples and Shell general tips
- Bash programming (if, loops, variables, arrays)

Why Unix?

- Unix derived operating systems (Centos, Fedora, Android, Ubuntu etc) are very popular and widely used
  - Datacenters
  - Web page hosting (web servers)
  - High performance computing
  - Mobile devices (Android)
  - Universities (research and educational purposes)

- Why is so popular
  - Πολλοί χρήστες ταυτόχρονα (multi-user)
  - Πολλές εργασίες ταυτόχρονα (multi- tasking)
  - Open Source
  - Free
  - Unix system commands provide great scripting capabilities

Είσοδος Στο Σύστημα

Remote Access with VPN:

http://its.cs.ucy.ac.cy/images/stories/uploads/guides/vpn.pdf

From a unix shell use ssh command

ssh <hostname>@ <machine>

π.χ. ssh cs05np1@b103ws1.in.cs.ucy.ac.cy

logout ή *exit* ή *CTRL-D*

From windows use an ssh client like putty

- The putty client and the ssh command open a shell that allows interaction with the system through unix commands
- Usually, the default shell is BASH
- Use winscp for file transferring

Το κέλυφος (shell)

- **Κέλυφος (shell)**
  - Διαβάζει τις εντολές του χρήστη,
  - Τις ερμηνεύει και
  - Ξεκινά τα προγράμματα που θα τις εκτελέσουν
- **Παραδείγματα κελυφών:** sh, csh, bash, tcsh
- X Server: UNIX “windows” (τρέχει by default πλέον)
  - Ξεκινά με xinit
  - Ανοίγεις “παράθυρα” με xterm&

Οργάνωση Αρχείων στο UNIX

Δέντρο

- Κατάλογος ρίζα (root directory): "/"
- Κατάλογοι (directories) και υποκατάλογοι (sub-directories)

Βασικές εντολές:
pwd, ls, cd, mkdir, rm, cp, mv, cat, more

Δοκιμάστε τις όλες εκτός **rm * .** π.χ.: **ls –l**
Ειδικοί συμβολισμοί
. Τρέχον κατάλογος (current dir)
.. Κατάλογος που περιέχει τον τρέχον (parent dir)
~ Κατάλογος του χρήστη (user´s dir)
π.χ.: cd ../.. , cd ~/
Για να δείτε τις επιλογές μιας εντολής:
man <command>
ή
google search: man <command>

Δικαιώματα αρχείων

Αλλάξτε τα δικαιώματα των κατάλογων και αρχείων: **chmod**

Δικαιώματα:

r (ανάγνωση), **w** (εγγραφή), **x** (εκτέλεση)

u (χρήστης), **g** (ομάδα), **o** (υπόλοιποι), **a** (όλοι)

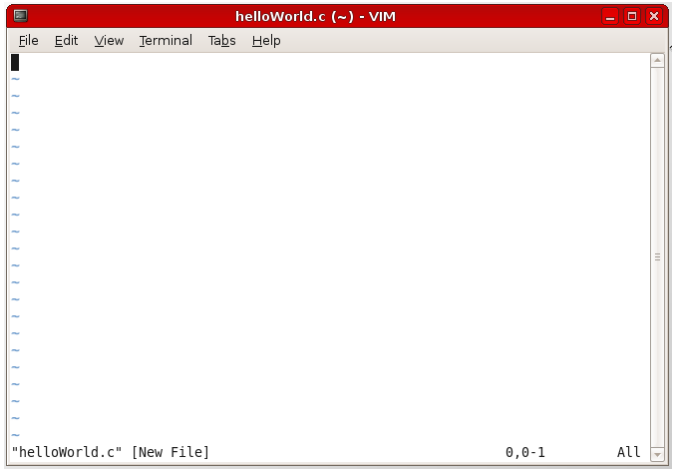
Π.Χ.


```
>ls -l
-rw-r--r--  1 user1  cs 13710 Apr 16:54 x.c
-rw-r--r--  1 user1  cs 68020 Aug 13:45 x.txt
>chmod g+w x.c
>ls -l x.c
-rw-rw-r--  1 user1  cs 13710 Apr 16:54 x.c
```

Επεξεργασία Κειμένου

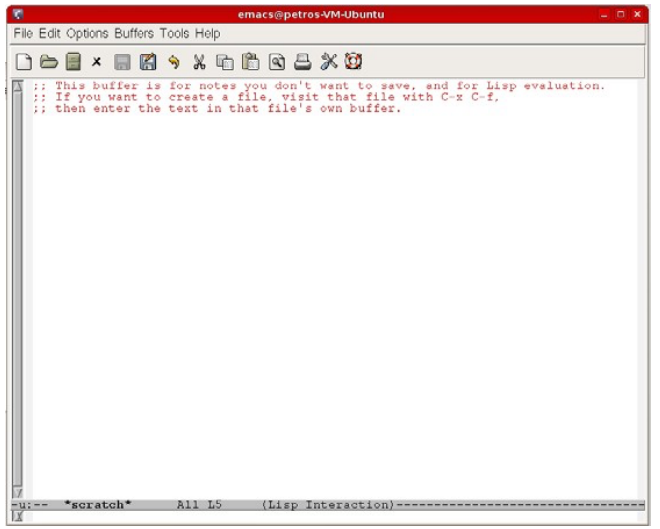
vi

- **i** για να δώσουμε είσοδο
- **ESC** για να δώσουμε εντολές
- **h** μετακίνηση αριστερά
- **l** μετακίνηση δεξιά
- **j** μετακίνηση κάτω
- **k** μετακίνηση πάνω
- **:w** save
- **:q** exit



Some more Commands for VI

http://www.eandem.co.uk/mrw/vim/usr_doc/doc_a4c.pdf



Emacs έχει μενού

Unix commands general syntax

- **commandName switches inputFiles**
- switches are the various command options

E.g. sort -n numbers.txt

1

2

13

- In the above example **commandName=sort** the **–n switch** is used and **inputFile=numbers.txt**
- **sort –n <numbers.txt** is also correct syntax
- **<** implies **stdin (standard input).** A data stream going into a program
- All unix commands expect to read their input either from a file or from standard input stream

Unix Terminal (Bash) Commands

- ssh
  - Secure (ba)sh
  - Connect remotely from a unix terminal to another machine
- scp
  - Secure copy
  - Used for copying files between machines
- vi(m)
  - Terminal based text editor
- chmod
  - Change mode of files (access rights)

Unix Terminal (Bash) Commands

- ls
  - Show contents of current directory
- pwd
  - Print the current directory (complete path)
- cd
  - Change directory
  - New directory defined
    - Relatively to current directory
    - Complete path (from root)
- find
  - Find files on a Unix or Linux system
- grep
  - Search for text inside files
- awk
  - Programming language designed to perform actions on the input stream

Unix Terminal (Bash) Commands

- cp
  - Copy files between directories
- mv
  - Move files from directory to another
- mkdir
  - Create a new directory (folder)
- rm
  - Delete a file (Beware! No Recycle Bin)
- cat
  - Print contents of text files
  - Concatenate multiple files
- wc
  - Calculates words, lines (-l) in text files

Unix Terminal (Bash) Commands

- top
  - Displays a listing with the most CPU-intensive tasks running on the system
- ps
  - Gives a snapshot of the currently running processes (like top)
- kill
  - Immediately terminates a specific process (unless it waits on an I/O operation)
- who
  - Provides a list of users who are currently logged into a machine
- diff
  - Compares and displays line-by-line differences between files
- sdiff
  - Displays side-by-side file comparison
- sudo
  - Execute command as administrator
- Taskset
  - Pin tasks to specific cores
- history
  - Lists the last n commands you executed

Unix Terminal (Bash) Commands

- echo
  - Just print something to the stdout
  - echo $RANDOM 28162 #to print a random number
- bc
  - The unix calculator (especially useful for floating point numbers)
  - echo "2+2" | bc
  - 4

Creating chains of UNIX commands with pipes

- **Unix pipes |**
  - Very powerful tool allows you to perform complex tasks with one liners! (show later examples of cool stuff you can do)
- Pipes redirect stdout of one command to the stdin of another command
- In example **cat numbers.txt | sort -n**

1

2

13

- cat (prints file contents) stdout is redirected to sort stdin
- **cat numbers.txt | sort -n | head -n 1**

1

- Add **head command** to the chain to retrieve the smallest number!

Don’t worry all commands have manual - Man Command

- Man <command>
  - (Almost) Every unix command has a manual page
  - Describes
    - Purpose of command
    - Syntax used for executing command
    - Arguments needed by command
  - Might include
    - Usage examples
    - Similar commands

grep

Εντοπισμός έκφρασης μέσα σε αρχείο και εκτύπωση τις γραμμής που εντοπίστικε.

-n Displays the line number

-v negate the regular expression

--help for more help.

grep –n “Hello World” *.txt

sed – Stream EDitor

s///

`sed s/root/haha/ < /etc/passwd`

Replace the **first** instance of **root** in a line with **haha root is a regular expression.**

`sed s/root/haha/g < /etc/passwd`

Replace every occurrence of root in every line For multiple exression use –e:

`sed –e s/root/haha/ -e s/petrosp/root/ </etc/passwd > /etc/hacked`

Helpful examples - grep example

- Very useful command. Allows to retrieve only lines of interest from a vast amount of data
- **cat example**

  1 Maria Antreou 10
  2 Marios Lazarou 9
  3 Gianna Nikou 5
  4 Eftixios Kiriakou 6
  5 Lazaros Lazarou 9
- **grep ' Lazarou ' example**

  2 Marios **Lazarou** 9
  5 Lazaros **Lazarou** 9
- **Place pattern you are looking for within ''**
- **Use \| to retrieve multiple patterns**
- **grep ‘ Lazarou \| Nikou ' example**

  2 Marios **Lazarou** 9
  3 Gianna **Nikou** 5
  5 Lazaros **Lazarou** 9

Helpful examples - Important grep switches

- -c # return only the number of matching lines
- -o # print only the mached parts of the lines
- -i # ignore case
- -A # print lines after matching lines
- -B # print lines before matching lines
- -R # read all files under directories recursively

Helpful examples – Select columns

- **grep ' Lazarou ' example**

  2 Marios **Lazarou** 9

  5 Lazaros **Lazarou** 9

- Following the previous example – **What can I do if I'm only interested in processing times?**
- **Use cut or awk to isolate the processing time column (in this example the 11th column)**

- cat **example** | **grep ' Lazarou '** | cut -d ' ' -f 3

  9

  9

- cat **example** | **grep ' Lazarou '** | awk '{print $3}'

  9

  9

Helpful examples – Average Min Max

- Now I want to get the max, min and average

```
cat example |  awk ' BEGIN {max=0;min=999999999;} {s=s+$4;c=c+1;
if(max<$4){max=$4};  if($4<min){min=$4}}
END {print max,min,s/c}'
10 5 7.8
```

- **The above is an awk one liner that calculates min max avg!!**

- Other ways for calculating min, max
  - For min

```
cat example | cut -d ' ' -f 4 | sort -k 1 -n | head -n 1
5
```

  - For max

```
cat example | cut -d ' ' -f 4 | sort -k 1 -n | tail -n 1
10
```

Other bash tricks

All unix process have process id (PID)

ps -ef | head

```
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 May10 ?        00:00:02 /sbin/init
root         2     0  0 May10 ?        00:00:00 [kthreadd]
root         3     2  0 May10 ?        00:00:00 [migration/0]
…
```

iplusplus 1000 &

[1] 17050

echo $! #get last launched process pid

17050

echo $? #Get last terminated process exit code

0 # zero code means process exited correctly.. Any other integer usually means abnormal termination

hexdump and objdump

Get the CPU

cat /proc/cpuinfo cat /proc/meminfo

Read Binary Files

hexdump -C hello.out | less

Look into the text Segment of our program.

objdump -d -j .text hello.out

Read the elf file

readelf -a hello.out | less

Μεταγλώττιση Προγράμματος

`gcc prog1.c –o prog1`

time ./**`prog1`**

real 0m5.913s
user 0m5.905s
sys 0m0.002s

gcc pThreads.c -o pThreads –lpthread //compile with Pthreads
time ./pThreads

real 0m0.465s
user 0m0.457s
sys 0m0.004s

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

Shell Scripts

Γιατί scripting?

- Θέλουμε να τρέξουμε την ίδια διεργασία με τις ίδιες ρυθμίσεις αλλά με πολλές διαφορετικές εισόδους
- Θέλουμε να τρέξουμε την ίδια διεργασία με μια είσοδο αλλά πολλές διαφορετικές ρυθμίσεις
- Θέλουμε…

Αν κάνεις το ίδιο πράγμα πολλές φορές τότε γράψε ένα script να το κάνει για σένα

Shell Scripts (2)

- #! /bin/csh
- set APP = hello
- set OUTPUT = hello.output
- echo "Running the program and redirecting output"
    - hello > hello.output
- $APP > $OUTPUT
- echo "The End"

Shell Scripts (3)

```
#! /bin/csh
#
# Variables
#
set PROG        = simulator
set CONFIG      = "-memory 512"
set OUTDIR      = ~/results

                  simulator -memory 512 q3.sql >& ~/results/q3.out
set APPS          simulator -memory 512 q6.sql >& ~/results/q6.out
                  simulator -memory 512 q12.sql >& ~/results/q12.out

@ i = 1
while ($i <= $#APPS)
  $PROG $CONFIG $APPS[$i].sql >&
$OUTDIR/$APPS[$i].out   @ i++
end
```

Shell Scripts (4)

```
#!/bin/tcsh
#
# Variables
#
set SIM = "~/mysims/sim-alpha/sim-alpha"

set CACHESIZES = (16 32 64 128 256)
set CACHELATENCY = (2 2 3 4 6)
set BENCHMARKS = (ammp gcc equake
                  twolf)
foreach bench ($BENCHMARKS)
  foreach cache ($CACHESIZES)
    @ i++;
    $SIM -bench $bench -size $cache -latency
  $CACHELATENCY[$i]  end
  set i = 0;
end
```

https://www.gnu.org/software/bash/manual/bash.pdf
http://www.tldp.org/LDP/abs/abs-guide.pdf
http://www.tldp.org/HOWTO/pdf/Bash-Prog-Intro-HOWTO.pdf

Bash loops

- for var in 1 2 3
  do
  echo “Test message $var”
  done

- var1=1
  var2=2
  if [$var1 –eq $var2]
  then
  echo “Equal”
  else
  echo “Different”
  fi

- var=1
  while [$var –le 10]
  do
  echo “Test message $var”
  var=$(($var + 1))
  done

Bash if operators

-eq equal

-ne not equal

-gt greater than

-ge greater or equal

-lt

-le

-z is null

-n is not null

= for string equality

Bash if examples

```
var1=1
var2=2
if [ $var1 –eq $var2 ]; then
        echo “Equal”
else
        echo “Different”
fi
```

Be careful of white spaces BASH if very sensitive
For instance if[$var –eq $var]; is syntactically wrong

```
varStr1=“str1”
varStr2=“str2”
if [ $varStr1 = $varStr2 ]; then ## for string equality use=
        echo “Equal”
else
        echo “Different”
fi
```