ΕΛΠ 605: Προχωρημένη Αρχιτεκτονική Υπολογιστών

Εργαστήριο Αρ. 1

Εισαγωγή στα UNIX και άλλα εργαλεία

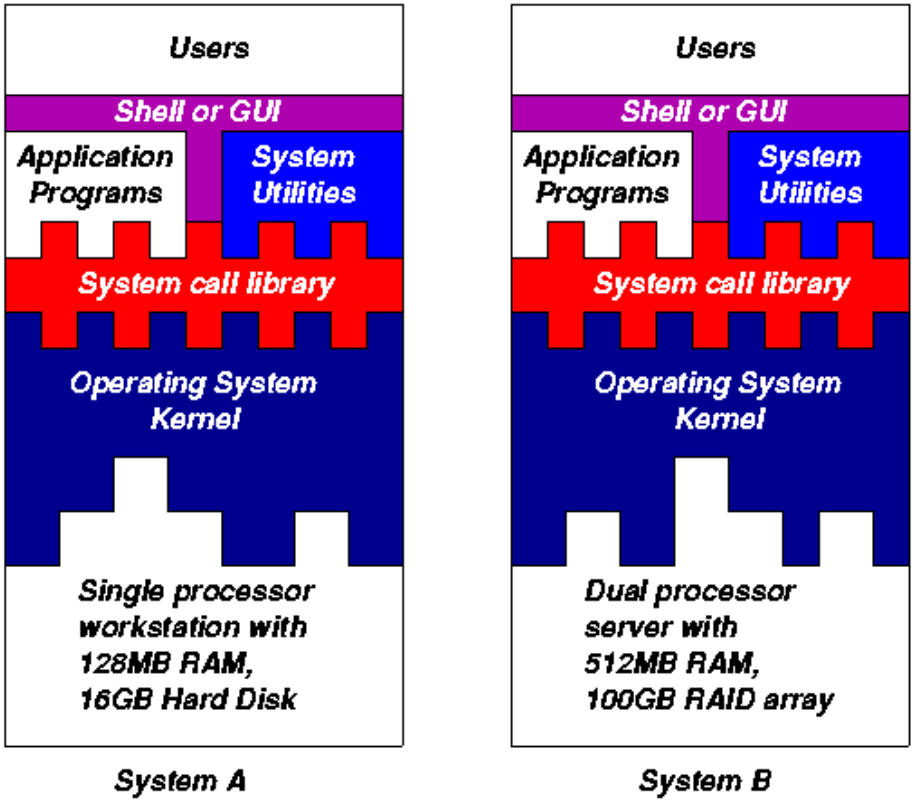
Πέτρος Παναγή, PhD

petrosp@cs.ucy.ac.cy

UNIX

Τι Είναι Λειτουργικό Σύστημα;

Γιατί UNIX;



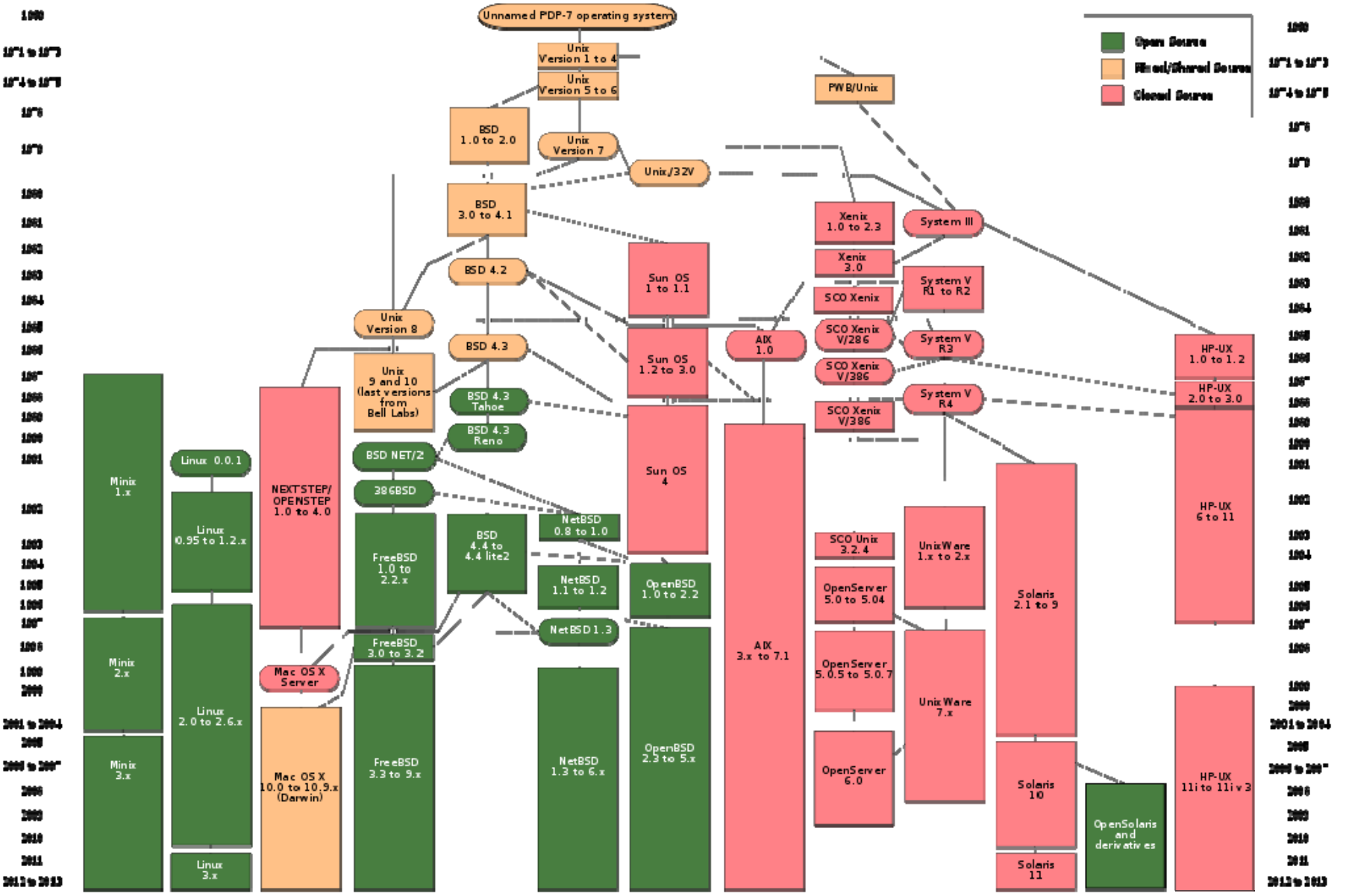
Παραλλαγές Unix:

Linux, AIX, Solaris, Ultrix, Irix, Tru64, FreeBSD

Χαρακτηριστικά:

Πολλοί χρήστες ταυτόχρονα (multi-user)
Πολλές εργασίες ταυτόχρονα (multi-tasking)

UNIX – Η Ιστορία



*

Είσοδος Στο Σύστημα

Text-based (TTY) terminals:

User account:

- Username
- Password
- Το UNIX είναι *case-sensitive* δηλαδή το username `petros` είναι διαφορετικό από `Petros` ή `PETROS`
- Αλλαγή του password με την εντολή *`passwd`*

Remote Login και Logout

- login <hostname>, ssh <hostname>
  - π.χ. ssh cs4058.in.cs.ucy.ac.cy
- *logout* ή *exit* ή *CTRL-D*

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)

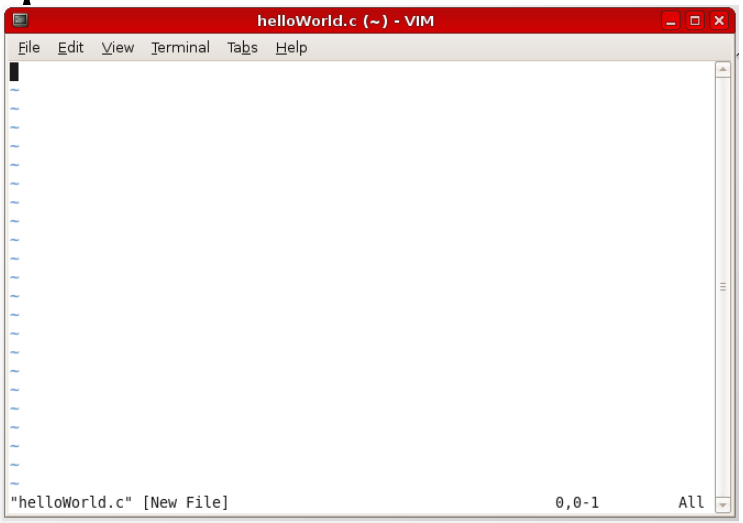
Π.Χ.


```
>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

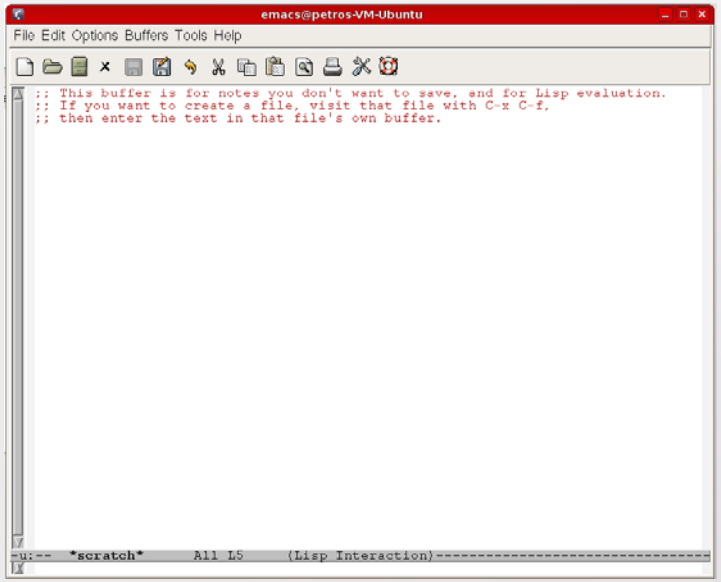
vi

- **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

Μεταγλώττιση Προγράμματος

>gcc hello.c –o hello.out

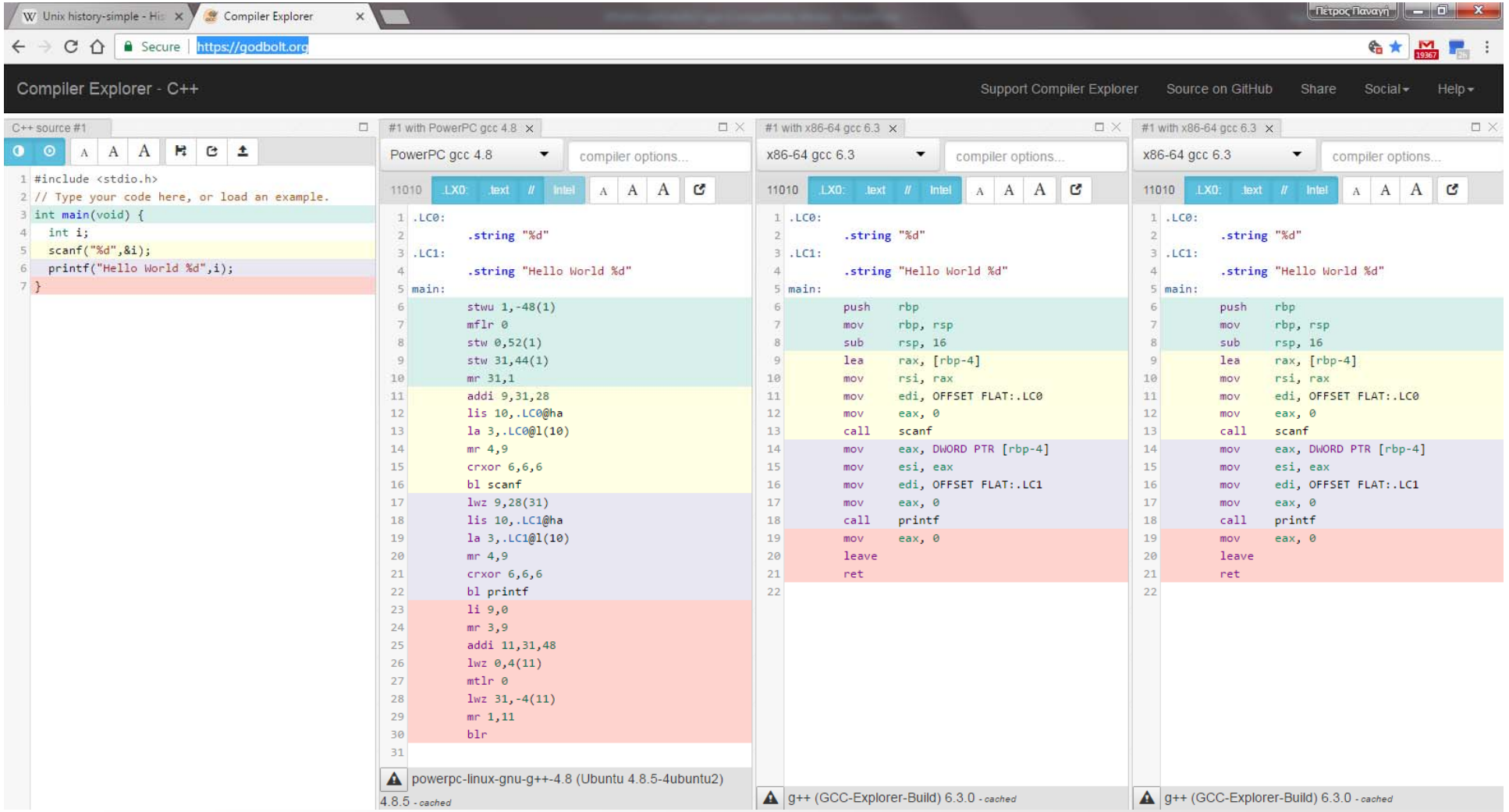
```
petrosp@cs4030:~/EPL370/FALL2009>gcc -O0 prog1.c -o a.out
petrosp@cs4030:~/EPL370/FALL2009>time ./a.out
X = 2255329744
6.289u 0.000s 0:06.28 100.0%    0+0k 0+0io 0pf+0w
petrosp@cs4030:~/EPL370/FALL2009>gcc -O3 prog1.c -o a.out
petrosp@cs4030:~/EPL370/FALL2009>time ./a.out
X = 2255329744
2.594u 0.000s 0:02.59 100.0%    0+0k 0+0io 0pf+0w
```

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

Getting the Assembly of your code.

```
gcc –S hello.c
less main2.s
as main2.s -o a.o
gcc a.o -o a.out
```

Online Tools for gcc assembly generation



https://godbolt.org/

Makefile

https://www.gnu.org/software/make/manual/

Βασική Δομή

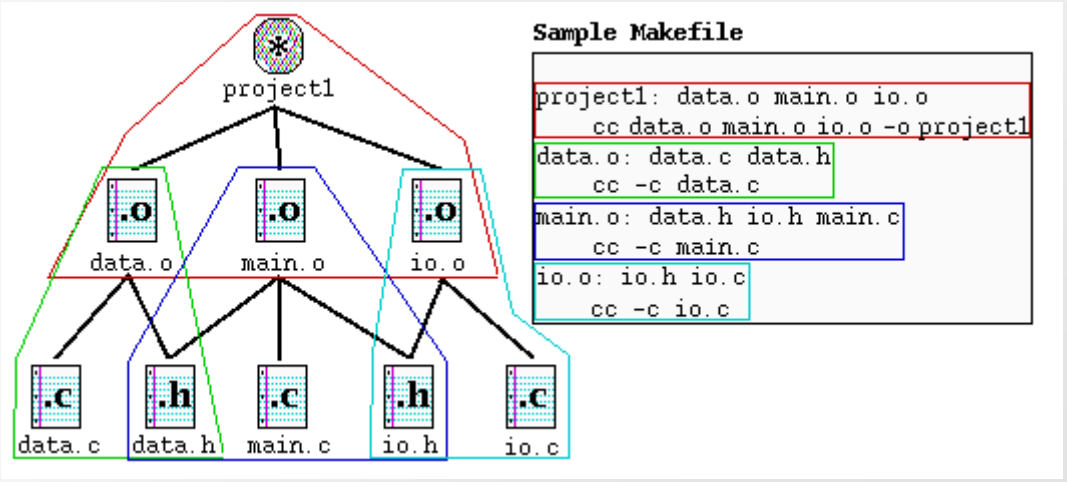
target: dependencies
[tab] system command

Παράδειγμα:

```
all:
    g++ main.cpp hello.cpp factorial.cpp -o hello
```

Και με Dependencies:

```
all: hello

hello: main.o factorial.o hello.o
   g++ main.o factorial.o hello.o -o hello

main.o: main.cpp
   g++ -c main.cpp

factorial.o: factorial.cpp
   g++ -c factorial.cpp

hello.o: hello.cpp
   g++ -c hello.cpp

clean:
   rm -rf *o hello
```



Makefile (2)

Μεταβλητές και Σχόλια

```
# I am a comment, and I want to say that the variable CC will be
# the compiler to use.
CC=g++
# Hey!, I am comment number 2. I want to say that CFLAGS will be the
# options I'll pass to the compiler.
CFLAGS=-c -Wall

all: hello

hello: main.o factorial.o hello.o
    $(CC) main.o factorial.o hello.o -o hello

main.o: main.cpp
    $(CC) $(CFLAGS) main.cpp

factorial.o: factorial.cpp
    $(CC) $(CFLAGS) factorial.cpp

hello.o: hello.cpp
    $(CC) $(CFLAGS) hello.cpp

clean:
    rm -rf *o hello
```

When Debugging TURN OFF Optimizations with -O0

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

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

grep/egrep

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
```

hexdump and objdump

Get the CPU

cat /proc/cpuinfo

cat /proc/meminfo

/sys/devices/system/cpu/cpu0/cache/index0/size

lscpu

lspci

Read Binary Files

hexdump -C hello.out | less (/Hel [enter])

Look into the text Segment of our program.

objdump -d -j .text hello.out

Read the elf file

readelf -a hello.out | less

Regular Expression Quick Reference v1.00

Literal Characters

\f	Form feed
\n	Newline (Use \p in UltraEdit for platform independent line end)
\r	Carriage return
\t	Tab
\v	Vertical tab
\a	Alarm (beep)
\e	Escape
\xxx	The ASCII character specified by the octal number xxx
\xnn	The ASCII character specified by the hexadecimal number nn
\cX	The control character ^X. For example, \cI is equivalent to \t and \cJ is equivalent to \n

Character Classes

[...]	Any one character between the brackets.
[^...]	Any one character not between the brackets.
.	Any character except newline. Equivalent to [^\n]
\w	Any word character. Equivalent to [a-zA-Z0-9_] and [[:alnum:]_]
\W	Any non-word character. Equivalent to [^a-zA-Z0-9_] and [^[:alnum:]_]
\s	Any whitespace character. Equivalent to [\t\n\r\f\v] and [[:space:]]
\S	Any non-whitespace. Equivalent to [^ \t\n\r\f\v] and [^[:space:]] Note: \w != \S
\d	Any digit. Equivalent to [0-9] and [[:digit:]]
\D	Any character other than a digit. Equivalent to [^0-9] and [^[:digit:]]
[\b]	A literal backspace (special case)

[[:class:]]	alnum	alpha	ascii	blank	cntrl	digit	graph
	lower	print	punct	space	upper	xdigit	

Replacement

\	Turn off the special meaning of the following character.
\n	Restore the text matched by the nth pattern previously saved by \(and \). n is a number from 1 to 9, with 1 starting on the left.
&	Reuse the text matched by the search pattern as part of the replacement pattern.
~	Reuse the previous replacement pattern in the current replacement pattern. Must be the only character in the replacement pattern. (ex and vi).
%	Reuse the previous replacement pattern in the current replacement pattern. Must be the only character in the replacement pattern. (ed).
\u	Convert first character of replacement pattern to uppercase.
\U	Convert entire replacement pattern to uppercase.
\l	Convert first character of replacement pattern to lowercase.
\L	Convert entire replacement pattern to lowercase.

Repetition

{n,m}	Match the previous item at least n times but no more than m times.
{n,}	Match the previous item n or more times.
{n}	Match exactly n occurrences of the previous item.
?	Match zero or one occurrences of the previous item. Equivalent to {0,1}
+	Match one or more occurrences of the previous item. Equivalent to {1,}
*	Match zero or more occurrences of the previous item. Equivalent to {0,}
{}?	Non-greedy match - will not include the next match's characters.
??	Non-greedy match.
+?	Non-greedy match.
*?	Non-greedy match. E.g. ^(.*?)\s*$ the grouped expression will not include trailing spaces.

Options

g	Perform a global match. That is, find all matches rather than stopping after the first match.
i	Do case-insensitive pattern matching.
m	Treat string as multiple lines (^ and $ match internal \n).
s	Treat string as single line (^ and $ ignore \n, but . matches \n).
x	Extend your pattern's legibility with whitespace and comments.

Extended Regular Expression

(?#...)	Comment, "..." is ignored.
(?:...)	Matches but doesn't return "..."
(?=...)	Matches if expression would match "..." next
(?!...)	Matches if expression wouldn't match "..." next
(?imsx)	Change matching rules (see options) midway through an expression.

Grouping

(...)	Grouping. Group several items into a single unit that can be used with *, +, ?, |, and so on, and remember the characters that match this group for use with later references.
|	Alternation. Match either the subexpressions to the left or the subexpression to the right.
\n	Match the same characters that were matched when group number n was first matched. Groups are subexpressions within (possibly nested) parentheses.

Anchors

^	Match the beginning of the string, and, in multiline searches, the beginning of a line.
$	Match the end of the string, and, in multiline searches, the end of a line.
\b	Match a word boundary. That is, match the position between a \w character and a \W character. (Note, however, that [\b] matches backspace.)
\B	Match a position that is not a word boundary.