ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ UNIX

Περιεχόμενα

ΕΙΣΑΓΩΓΗ 1
Τι είναι το Unix? 1
1.2 Ο πυρήνας (kernel) 1
1.3 Ο φλοιός (shell) 1
2ΕΝΤΟΛΕΣ ΣΤΟ UNIX 2
3 Compiling προγράμματα της C στα UNIX 3
3.1 gcc compiler 3
3.2 Compiling μεγάλα πρόγραμματα που βρίσκονται σε διαφορετικά αρχεία 5
3.3 Η Εντολή Make 5
3.4 Makefile 6

ΕΙΣΑΓΩΓΗ

Τι είναι το Unix?

Τα UNIX είναι ένα λειτουργικό σύστημα που αναπτύχθηκε κυρίως από το 1960 και μετά και συνεχίζει να αναπτύσσεται ακόμα και σήμερα. Περιέχει ένα graphical user interface (GUI) παρόμοιο με αυτό των Microsoft Windows παρέχοντας ένα φιλικό προς τον χρήστη περιβάλλον. Υπάρχουν πολλές διαφορετικές εκδόσεις UNIX ανά τον κόσμο. Οι ποιο δημοφιλείς είναι Sun Solaris , GNU/Linux , και MacOS X.
Τα Unix αποτελούνται απο τρία μέρη τον πυρήνα (kernel), τον φλοιό (shell) και τα προγράμματα.

1.2 Ο πυρήνας (kernel)

Το πρόγραμμα που χρησιμοποιείται για να συνδέει τον χρήστη με την καρδία του UNIX που είναι ο πυρήνας(kernel).Ο πυρήνας φορτώνεται στην μνήμη με το ξεκίνημα του υπολογιστή και διαχειρίζεται το σύστημα μας μέχρι αυτο να σβήσει.Μερικές απο τις λειτουργίες του είναι να δημιουργεί και να ελέγχει διεργασίες, να ελέγχει την μνήμη και τα συστήματα αρχείων καθώς και τους πόρους του συστήματος.Όταν εμείς συνδεόμαστε ο πυρήνας φορτώνει τα προγράμματα στην μνήμη και καθαρίζει το σύστημα όταν αποσυνδεόμαστε.

1.3 Ο φλοιός (shell)

Ο φλοιός είναι υπεύθυνος για να εξασφαλίσει οτι οι εντολές που θα εισαχθούν απο τον χρήστη στο σύστημα μας θα εκτελεστούν κανονικά.Ξεκινάει μετα την διαδικασία εισόδου του χρήστη στο σύστημα και συνεχίζει να τρέχει σαν μια διεργασία μέχρι αυτός να βγεί.Μερικές απο τις δυνατότητες που προσφέρει είναι :

- Δίνει την συνατότητα στον χρήστη να προσαρμόσει το σύστημα σύμφωνα με τις δικές του επιθυμίες
- Κάνει αποθήκευση των ποιο πρόσφατων εντολών για την διευκόλυνση του χρήστη.
- Κάνει διαχείριση εργασιών ωστε να μπορούμε να δημιουργίσουμε νέες, να τις εκτελέσουμε και να τις τερματίσουμε.
- Παρέχει την δυνατότητα σύντμησης μεγάλων ονομάτων και εντολών

- Μας δύνεται η δυνατότητα να γράφουμε προγράμματα τα οποία ονομάζονται scripts και εκτελούνται σαν εντολές.

2ΕΝΤΟΛΕΣ ΣΤΟ UNIX

arch – εμφανίζει την αρχιτεκτονική του συστήματός μας.

kill – τερματίζει μία η περισσότερες διεργασίες

ps – μας δείχνει όλες τις τρέχουσες διεργασίες στο τερματικό.

who – επιστρέφει τον κατάλογο των χρηστών που είναι συνδεδεμένοι στο σύστημα.

date – επιστρέφει την ημερομηνία και ώρα συμφώνα με το ρολόι του συστήματος μας.

chmod – αλλάζει τα δικαιώματα προσπέλασης σε ένα αρχείο.

chmod τριψήφιος οκταδικός αριθμός filename
chmod [user_specifier] mode_change_specifier filename
Στην πρώτη περίπτωση υπάρχουν οι εξής κανόνες: το πρώτο οκταδικό ψηφίο αναφέρεται στον ιδιοκτήτη του αρχείου, το δεύτερο στην ομάδα (group) όπου ανήκει ο ιδιοκτήτης του αρχείου και το τρίτο στους υπόλοιπους χρήστες. Σε κάθε ψηφίο π ροσθέτουμε 4 αν θέλουμε να δώσουμε δικαίωμα ανάγνωσης, 2 αν θέλουμε να δώσουμε δικαίωμα εγγραφής και 1 αν θέλουμε να δώσουμε δικαίωμα εκτέλεσης (το δικαίωμα εκτέλεσης για έναν κατάλογο έχει το νόημα ότι μπορού με να κάνουμε τον κατάλογο αυτό τρέχοντα κατάλογο εργασίας).
Στη δεύτερη περίπτωση το (προαιρετικό) όρισμα user_specifier καθορίζει σε ποιον εφαρμόζεται η αλλαγή των π ρονομίων (**u** = owner – ο ιδιοκτήτης, **g** = group – η ομάδα όπου ανήκει ο ιδιοκτήτης, **o** = other – οι υπόλοιποι χρήσ τες) και το mode_change_specifier δείχνει ποια αλλαγή θα επέλθει (**+r**, **-r**, **+w**, **-w**, **+x**, **-x**)

cat [-options] [filename(s)] – βλέπουμε τα περιεχόμενα ενός αρχείου κειμένου και ανάλογα με τα ορίσματα που θα εισάγουμε έχουμε :
-b τυπώνει αιθμούς γραμμών και παραβλέπει τις κενές γραμμές
-s οπου έχει πολλα κενά τα αντικαθιστά με ένα
-n τυπώνει αριθμούς γραμμών αλλα δεν αγνοεί τις κενές γραμμές
-v δείχνει τους χαρακτήρες που δεν μπορούν να τυπωθούν σαν αριθμούς
Αν δεν εισάγουμε κάποιο όρισμα απλα τυπώνει στην οθόνη μας ότι πληκτρολογήσουμε.

cp file1 file2 – αντιγράφει το αρχείο file1 στο file2.Εαν το αρχείο file2 δεν υπάρχει το δημιουργεί.

diff file1 file2– συγκρίνει τα δυο αρχεία και μας δίνει ως έξοδο τις διαφορές τους.

mv file1 file2 – αλλάζει το όνομα του file1 σε file2.

clear – καθαρίζει την οθόνη του υπολογιστή μας.

gcc [options] [filename(s)] – GNU compiler κατάλληλος για να κάνει compile προγράμματα γραμμένα σε C, C++, Fortran γλώσσες.

ls - Ένας κατάλογος περιέχει πολλά αρχεία και για να δούμε τα ονόματά τους χρησιμοποιού με την εντολή ls.Αν δώσουμε ορίσματα, τότε αν κάποιο από αυτά είναι αρχείο, η ls θα μας παρουσιάσει το όομά του, ενώ αν είναι κατάλογος θα μας δώσει τα ονόματα των αρχείων και καταλόγων που υπάρχουν σε αυτόν.

ln [options] [filename(s)] - Με την εντολή αυτή μπορούμε να αναφερόμαστε σε ένα αρχείο με πολλά ονόματα ή απο διαφορετικούς καταλόγους

3 Compiling προγράμματα της C στα UNIX.

3.1 gcc compiler

Ο μεταγλωττιστής GNU CC ποιό γνωστός ως GCC μπορεί να κάνει μεταγλώττιση σε προγράμματα γραμμένα σε C και C++.Ο Gcc μας δείνει την δυνατότητα ανα πάσα στιγμή να διακόψουμε την διαδικασία της μεταγλώττισης εισάγοντας τις κατάλληλες εντολές.
Ας πάρουμε ένα παράδειγμα:
Υποθέτουμε οτι έχουμε το παρακάτω πρόγραμμα hello.c που θέλουμε να κάνουμε compile απο τα Unix.

```
1   # include <stdio.h>
2   int main (int argc,char*argv[]){
3   printf ("hello world \n");
4   return 0;
5   }
```

Η εντολή που θα πληκτρολογίσουμε στο command line θα είναι :

$ gcc hello.c

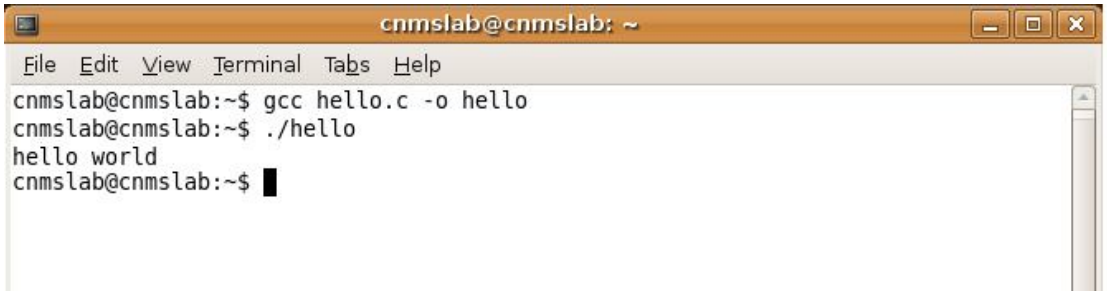
Αφού το πρόγραμμα περάσει με επιτυχία το compile το αποτέλεσμα θα είναι ένα αντικείμενο πρόγραμμα με την ονομασία “a.out”.Η αλλαγή του ονόματος του εκτελέσιμου προγράμματος μας μπορεί να γίνει κατα το compile αντικαθιστώντας την παραπάνω εντολή με :

$gcc hello.c –o hello

Για να τρέξουμε το πρόγραμμα μας δεν έχουμε παρα να πάμε στο directory που βρίσκεται το εκτελέσιμο αρχείο και να πληκτρολογίσουμε :

$ ./ hello

Το πρόγραμμα μας θα εκτελεστεί και στην οθώνη μας θα έχουμε



Κατα την διαδικασία της μεταγλώττισης το πρόγραμμα μας περνάει απο τέσσερα στάδια που είναι:

- Preprocessing
- Compilation
- Assembly
- Linking

Preprocessor Options

- -E περναέι το πρόγραμμα μας απο την διαδικασία του preprocess χωρίς να αφήνει τον compiler να συνεχίσει στα επόμενα στάδια
- -M παρέχει εξαρτήσεις για την εντολή make
- -C κρατάει τα σχόλια του προγράμματος και στο output αρχείο που δημιουργεί

Compiler Options

- -c κάνει μόνο compile το αρχείο και δεν προχωραεί στο επόμενο στάδιο
- -S στέλνει στον assembler ένα αρχείο, το νέο αρχείο που θα δημιουργηθεί θα έχει κατάληξη .s
- -w δεν εμφανίζει warnings
- -W παράγει warnings για κάποιες παρενέργεις που μπορεί να δημιουργεί μια συνάρτηση που καλείται απο το πρόγραμμα(πχ δυσλειτουργία κάποιας συνάρτησης)
- -I ορίζει ένα μονοπάτι για κάποιες επιπλέον βιβλιοθήκες
- -g (-gcoff, -gstabs, -gxcoff, gdwarff) συμπεριλαμβάνει πληροφορίες για debug (μπορεί να προσδιορίζει κάποιον debugger)
- -static κάνει link μια static βιβλιοθήκη με το προγραμμά μας
- -shared σε περίπτωση που υπάρχει δυνατότητα επιλέγει, μια κοινή βιβλιοθήκη απο μια static

Assembler Options

Για να περαστούν οδηγίες στον assembler χρειάζονται οι εντολές να έχουν την εξής μορφη gcc –Wa, -options .

- -ahl δημιουργεί πηγαίο κώδικα σε υψηλό επίπεδο assembly
  πχ. gcc –Wa, -ahl warnings.c
- -as δημιουργεί μια λίστα με σύμβολα
  πχ. gcc –Wa, -as warnings.c

Linker options

- -l default ονομασία βιβλιοθήκης
- -I (dirname) ορίζεται το μονοπάτη για το πού θα ψάξει ο gcc για include αρχεία
  πχ $gcc project.c-I/home/user/include –o project

- -L (dirname)ορίζεται μονοπάτη για το που θα ψάξει ο compiler για βιβλιοθήκη πχ $ gcc project.c –L/home/user/lib -lnew –o project

Ολες οι οδηγίες προς τον gcc compiler που δεν έχουν εκτελεστεί στα ανώτερα στάδια περνάνε στην διαδικασία linking.

3.2 Compiling μεγάλα πρόγραμματα που βρίσκονται σε διαφορετικά αρχεία.

Πολλές φορές κάποιο project που θέλουμε να κάνουμε compile βρίσκεται σε περισσότερα από ένα αρχεία (ας πούμε οτι χωρίζεται σε δύο και έχουν την ονομασία “file1.c” και “file2.c”) με την βοήθεια των παρακάτω εντολών:

$ gcc file1.c file2.c –o program

Το ίδιο αποτέλεσμα μπορεί να επιτευχθεί και με τις ακόλουθες εντολές

$ gcc file1.c
$ gcc file2.c
$ gcc file1.o file2.o –o program

Το πλεονέκτημα της τελευταίας μεθόδου είναι οτι κάνει compile κάθε πρόγραμμα ξεχωριστά και μετά δημιουργείται το αντικείμενο πρόγραμμα.
Εαν αφού δημιουργήσαμε το program.o θέλουμε να αλλάξουμε το file1.c για κάποιο λόγο τότε το αρχείο file2.c δεν χρειάζεται να ξαναγίνει compile.Σε περίπτωση που μπορούσαμε να γλυτώσουμε αυτήν την μεταγλώττιση τότε ο χρόνος που θα χρειάζεται για να γίνει rebuild το “program” θα είναι μικρότερος απο οτι στην αρχή.Σε μεγάλα projects που αποτελούνται απο εναν μεγάλο αριθμό απο προγράμματα μια τέτοια διαχείρηση μπορεί να φανεί πολύ αποδοτική αφού θα μας γλείτωνε απο περιττές μεταγλωτίσεις.Τα Unix για την αυτοματοποίηση αυτής της διαδικασίας έχουν κατάλληλες εντολές τις οποίες θα αναλύσουμε και ποιο κάτω,.

3.3 Η Εντολή Make

Η εντολή make μας αυτοματοποιεί την διαδικασία όταν έχουμε να διαχειριστούμε ένα μεγάλο project με πολλά αρχεία πηγαίου κώδικα. Το make είναι ένα εργαλείο αυτοματοποίησης μιας διαδικασίας πολλών βημάτων, σαν τα shell scripts ή τα batch files στα Windows συστήματα. Η διαφορά του make και το σημείο στο οποίο υπερτερεί από τα προηγούμενα, είναι η χρήση των εξαρτήσεων.
Έστω ότι έχουμε ένα κλασικό project γραμμένο σε C που αποτελείται από τα εξής αρχεία:

$ compiler.c
$ tokens.h
$ lexer.l
$ parser.y
$ common.h
$ seman.c
$ symbol.h
$ symbol.c

Για να δημιουργηθεί το τελικό εκτελέσιμο, πρέπει να εκτελεστούν με την σειρά οι παρακάτω εντολές:

```
$ flex lexer.l -o lexer.c
$ bison parser.y -o parser.c
$ gcc -c lexer.c -o lexer.o
$ gcc -c parser.c -o parser.o
$ gcc -c seman.c -o seman.o
$ gcc -c symbol.c -o symbol.o
$ gcc -c compiler.c -o compiler.o
$ gcc -lfl lexel.o parser.o seman.o symbol.o compiler.o -o compiler
```

Δεν είναι και λίγες για να τις πληκτρολογούμε κάθε φορά που θέλουμε να κάνουμε κάποια αλλαγή σε ένα αρχείο. Μια ιδέα θα ήταν να τις βάλουμε όλες μαζί σε ένα script.Αυτό όμως έχει το μειονέκτημα ότι αν κάνουμε μια αλλαγή σε ένα μόνο αρχείο θα ξαναεκτελεστούν όλες οι εντολές από την αρχή και δεν θα αξιοποιηθεί η πληροφορία ότι μια αλλαγή στο compiler.c θα επηρεάσει μόνο το compiler.o και το τελικό compiler. Αυτό το κενό έρχονται να καλύψουν οι εντολές make και makefile.

3.4 Makefile

Το make χρειάζεται ένα αρχείο ρυθμίσεων που περιέχει τις εντολές. Αυτό λέγεται Makefile και είναι οργανωμένο σε ενότητες που λέγονται rules. Κάθε rule αναφέρεται σε (τουλάχιστον) ένα target. Ένα target είναι ένας στόχος, συνήθως η δημιουργία ενός αρχείου, που πρέπει να πετύχει το make στην διαδικασία του build. Για την επίτευξη ενός στόχου πρέπει να εκτελεστούν κάποιες εντολές και πιθανόν να πρέπει να ικανοποιηθούν κάποιοι άλλοι στόχοι πρώτα, τα λεγόμενα prerequisites:

```
# Structure of the Makefile rule

target: prerequisite_1 prerequisites_2
	command 1
	command 2
	...
	command N
```

Ο κανονας οριζει τρια πραγματα. το ονομα του αρχείου που πρεπει να ανανεωθεί (*στοχος - target*), ποτε πρεπει να ανανεωθει (*εξαρτησεις - dependencies*) και πως θα ανανεωθει (*ενεργειες - actions*).Έτσι στο παράδειγμά μας για να δημιουργηθεί το τελικό εκτελέσιμο compiler φτιάχνουμε ένα rule με target με το ίδιο όνομα και ορίζουμε τα prerequisites και τις εντολές που πρέπει να εκτελεστούν ώστε να ικανοποιηθεί αυτός ο στόχος, δηλαδή η δημιουργία του compiler:

```
compiler: lexer.o parser.o seman.o symbol.o compiler.o
	gcc -lfl lexer.o parser.o seman.o symbol.o compiler.o -o compiler
```

Με το υποχρεωτικό <tab> που υπάρχει στην αρχή κάθε εντολής ξεχωρίζουν οι εντολές που πρέπει να εκτελεστούν για κάθε rule. Με παρόμοιο τρόπο δημιουργούμε τα targets για τα υπόλοιπα αρχεία με τις εντολές που χρειάζεται το καθένα και τα prerequisites του. Το τελικό Makefile έχει ως εξής:

Makefile:

```
lexer.c: lexer.l
	flex lexer.l -o lexer.c

parser.c: parser.y
	bison parser.y -o parser.c

lexer.o: lexer.c tokens.h common.h
	gcc -c lexer.c -o lexer.o -Wall

parser.o: parser.c tokens.h common.h
	gcc -c parser.c -o parser.o -Wall

seman.o: seman.c common.h
	gcc -c seman.c -o seman.o -Wall

symbol.o: symbol.c symbol.h common.h
	gcc -c symbol.c -o symbol.o -Wall

compiler.o: compiler.c
	gcc -c compiler.c -o compiler.o -Wall

compiler: lexer.o parser.o seman.o symbol.o compiler.o
	gcc -lfl lexer.o parser.o seman.o symbol.o compiler.o -o compiler
```

Μπορούμε τώρα να εκτελέσουμε την εντολή make compiler και να δημιουργήσουμε το εκτελέσιμό μας. Βλέπουμε ότι έχουμε προσθέσει την επιπλέον πληροφορία των prerequisites και ότι έτσι μπορεί το make να ξέρει ότι όταν αλλάξει το αρχείο tokens.h δεν χρειάζεται να ξαναδημιουργήσει όλα τα αρχεία από την αρχή αλλά μόνο αυτά που εξαρτώνται, εμμέσως ή αμέσως, από αυτό (στην περίπτωσή μας θα ξαναεκτελεστούν οι κανόνες για τους στόχους lexer.o parser.o και compiler).