ΤΜΗΜΑ ΥΠΟΛΟΓΙΣΤΩΝ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

Το Σύστημα Unix

Δρ.Γ.Κοροβέσης
1984

ΠΑΤΡΑ

Περί UNIX

Εισαγωγή

Το Λειτουργικό Σύστημα UNIX αρχικά σχεδιάστηκε για τον μίνι η/υ PDP-7 (1969) και αποτελείτο βασικά από ένα text editor.Οι σχεδιαστές του βασίστηκαν στο interactive σύστημα MULTICS.Αρχικά ήταν για ένα χρήστη και σε assembler.

Το 1973 μεταφέρθηκε στα PDP-11,κατασκευάστηκε ξανά από την αρχή στην γλώσσα C, systems programming language.Ο κύριος σκοπός της νέας προσπάθειας ήταν η δυνατότητα μεταφοράς του σ'άλλες μηχανές.Έτσι αναπτύχθηκε και σε multi-user.

Το πρώτο τεστ portability του UNIX έγινε με την μεταφορά του (1977) στη μηχανή Interdata 8/32 με μπόλικες διαφορές από τα PDP-11.

Σήμερα το UNIX τρέχει σε διάφορες μηχανές (μικρές και μεγάλες).

Βασικό Software

Το σύστημα C,text editor, επεξεργαστές κειμένου, SPELL, assemblet,linking loader,symbolic debugger, διάφορες γλώσσες προγραμματισμού,2 compiler-compiler YACC,TGM,macro-processor,software για συντήρηση,ρουτίνες ελέγχου κ.α. Επίσης υπάρχει Δίκτυο επικοινωνιών με άλλα UNIX και άλλα συστήματα.

Αρχές Σχεδιασμού

1. Interactive, ευκολία χρήσης
2. Περιορισμός κύριας μνήμης, υλοποίηση σε μικρές μηχανές
3.Αυτοδύναμο, το σύστημα έχει τα εργαλεία να αυτοσυντηρείται self support.Αυτό υποστήριξε τη βελτίωση του συστήματος καθώς αναπτύσσετο και αργότερα προστέθηκε ο παράγοντας portability.

Τα 1,2,3 θα φανούν στον σχεδιασμό των file systems,I/O,process control, system command interface. Το Σύστημα που φορτώνεται στην μνήμη αποτελείται :

1. Initialisation code
2. Process management
3. System calls
4. Inerrupt handling
5. File management

Υλικό

Θα δούμε αργότερα το UNIX της Ρ.Ε. για τη μηχανή που έχουμε εδώ 3252 Xp (32 bit ruord, 2Mb μνήμη, rew disk 300 Mb, fixed disc 300 Mb, tape unit 800 b.p. μηχανή πιο πολύπλοκη από τα PDP της DEC.

Αρχικές τιμές στο Σύστημα Init. code, χοντρικά.Kernel page registers παίρνουν αρχικές τιμές, ενεργοποιείται το memory management unit, υπολογισμός memory size, δημιουργία μιας καινούργιας διαδικασίας και ενεργοποίηση του scheduler.

Η καινούργια διαδικασία με τη σειρά της έχει την ευθύνη για τα τερματικά δηλαδή δημιουργεί για κάθε τερματικό μια διαδικασία controller, επίσης φτιάχνει μια process που περιοδικά (π.χ. κάθε 30'') γράφει τη κατάσταση του συστήματος αρχείων στο δίσκο και χειρίζεται το login, logout , accounting.

Processes

Ο cpu εκτελεί μια διαδικασία σε 2 καταστάσεις

(α) user mode, που το Σύστημα γνωρίζει ότι ο έλεγχο (control) βρίσκεται σε κάποιο χρήστη user process με συνέπεια ορισμένους περιορισμούς π.χ. τα interrupts δεν εκτελούνται (serviced) σε user mode.

(β) Kernel mode, ο έλεγχος βρίσκεται στο πυρήνα του Συστήματος, μια διαδικασία μ ' αυτό τον τρόπο έχει την ιδιαίτερη μεταχείριση για συγκεκριμένες δουλείες.Μέσα στο UNIX μια διαδικασία αποτελείται από code που εκτελεί κάποιο έργο.Χρήσιμος ορισμός επίσης είναι η εκτέλεση ενός προγράμματος.Όμως μια process ίσως
εκτελείται από πολλά προγράμματα και πολλές διαδικασίες να συνιστούν ένα πρόγραμμα.Τις process μέσα στο σύστημα μπορούμε να τις δούμε σε δυό επίπεδα :

1. Στο πάνω επίπεδο που η έννοια "process" χρησιμοποιείται για να περιγραφτούν οι λειτουργίες του Συστήματος.Αυτές οι διαδικασίες θεωρούμε ότι τρέχουν πάνω σε φανταστικές μηχανές virtual m/c, άρα τρέχουν παράλληλα (σχετικά) στο χρόνο.Δημιουργούνται, παίρνουν και αφήνουν resources,συνεργάζονται, συγκρούονται (conflict) και μπλοκάρουν (process deadlock), ο αριθμός τους μέσα στο σύστημα είναι μεταβλητός στο χρόνο.

2. Στο κάτω επίπεδο (low level), οι "διαδικασίες" είναι παθητικές, πάνω τους εφαρμόζεται ένας νέος processor που γίνεται time shared δίνοντας, υποστηρίζοντας τη θεωρητική του 1.

Εδώ θα τα εξετάσουμε από τη σκοπιά του 2, ειδικότερα με τα εξής:

α. Image structure
β. Λεπτομέρειες εκτέλεσης μιας απεικόνισης
γ. Processor switching

Για κάθε διαδικασία μέσα στο UNIX παρατηρούμε τα εξής:

1. Οι processes βρίσκονται, υπάρχει ένας pointer στο πίνακα system process table.
2. Κάθε διαδικασία έχει χώρο στη μνήμη, που λέγεται per process data area.
3. Processes δημιουργούν ή τερματίζουν άλλες.
4. Οι διαδικασίες demand, obtain, release διάφορα recources του Συστήματος.

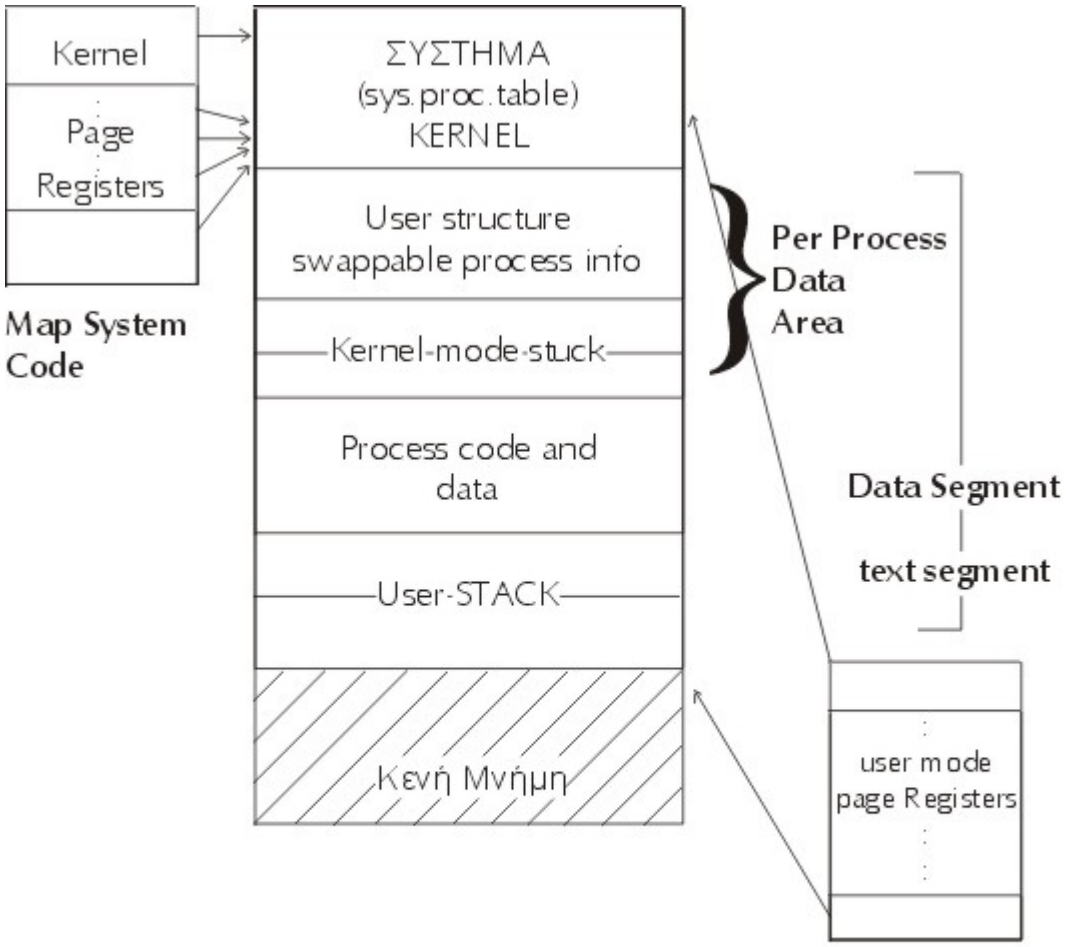
Process Image

Αποτελείται από τους εξής χώρους:

α. Τη δομή "proc".Είναι η αναφορά (pointer) στο πίνακα system process table.Βρίσκεται πάντοτε στη main memory.
β. Ο χώρος Data Segment που αποτελείται από:

1. per process data area
2. user program data
3. program text
4. Stack

γ. Ο χώρος text segment, περιέχει μόνο program text για code sharing.



To Memory map

Η δομή "proc" αποτελείται από 15 στοιχεία απαραίτητα στο Σύστημα για να παρακολουθεί

τις διαδικασίες.Είναι αυτό που είπαμε αναφορά (θέση) στο system process table. Ο πίνακας αυτός βρίσκεται πάντα στο χώρο κύριας μνήμης SYSTEM KERNEY.

1. Η process state : αυτή η μεταβλητή Συστήματος παίρνει τιμές που δηλώνουν τη κατάσταση μιας process (sleeping,running,null)

2. Process flags : περιγράφουν το STATUS της process (in core = εντός κύριας μνήμης, scheduling process, μη αναστελλόμενη = unswappable, Αναστέλεται = swapped, tracing)

3. Η address του χώρου Data Segment : θέση στη μνήμη ή στο δίσκο.

4. Το Μέγεθος του χώρου Data Segment

5. Η προτεραιότητα της διαδικασίας.

6. Όνομα της διαδικασίας.

7. Όνομα Ιδιοκτήτη της διαδικασίας.

8. Όνομα Τερματικού της διαδικασίας.

9. Signal Number που στάλθηκε στη διαδικασία (ίσως).

10. Το synchronisation event πάνω στο οποίο είναι sleeping η διαδικασία.

11. Πληροφορίες για shared code.

12. Πληροφορίες για χρήση του scheduler.

Τα υπόλοιπα στοιχεία βρίσκονται στη δομή "user structure".

<u>Η δομή user</u> βρίσκεται στο χώρο per process data area περιέχει πληροφορίες που παραμένουν εντός Μνήμης, χρήσιμες στο Σύστημα όταν η διαδικασία έχει ανασταλλεί ,δηλαδή Swappable process Information= (process state, user ID, 1/0 parameters, file access control, κ.α.), έχει μεταφερθεί στο Δίσκο.

<u>Image Execution</u>

Για να εκτελεστεί η απεικόνιση μιας process πρέπει να ισχύουν τα εξής :η process να είναι στη μνήμη (in core). Ένας Kernel page Register να είναι pointer στο χώρο per process data area, οι καταχωρητές user page Registers στέλνουν την process πάνω στην real memory από virtual memory.

Αρχικά, το πρόγραμμα που λέγεται bootstrap loader, υλοποιημένο σε hardware, ενεργοποιείται από το πάτημα ενός κουμπιού στην κονσόλα ελέγχου της μηχανής.Σαν στόχο του έχει να φορτώσει στην Μνήμη της μηχανής από το Δίσκο το Λειτουργικό Σύστημα.Ο όρος bootstrap σημαίνει ότι "σηκώνω τον εαυτό μου από τα κορδόνια μου " χαρακτηρίζοντας τη λειτουργία του φορτωτή.

Ο φορτωτής (loader) ενεργοποιεί το πρόγραμμα initialisation code που δίνει τιμές στις παραμέτρους του Συστήματος, φτιάχνει το περιβάλλον unix και μετά με τη σειρά του ενεργοποιεί το πρόγραμμα που κάνει τη δρομολόγηση διαδικασιών δηλαδή του scheduler.

Άρα, η πρώτη διαδικασία από την έναρξη του Συστήματος είναι ο δρομολογητής και "τρέχει" σε Kernel mode. Όλες οι άλλες διαδικασίες τρέχουν σε user mode εκτός των στιγμών που απευθύνονται στα εξαρτήματα του Συστήματος μέσω των εντολών system calls πχ. εντολή για Ε/Ε.

<u>Πως δημιουργούνται οι διαδικασίες στο σύστημα.</u>

Το Unix έχει δύο system call, εντολές Συστήματος που χρησιμοποιούνται για να φτιάχνονται processes διαδικασίες.

(α) Το FORK, η διαδικασία που το χρησιμοποιεί "γεννάει" ένα αντίγραφο της στη Μνήμη, καταλαμβάνει το ΠΑΙΔΙ κάποιο νέο χώρο στη Μνήμη.

(β) Το EXEC f, η διαδικασία που χρησιμοποιεί αυτό το system call επικαλύπτεται στη Μνήμη με το πρόγραμμα του αρχείου " f ", η παλιά διαδικασία παύει να υπάρχει με την έννοια ότι το πρόγραμμα που περιγράφει τα βήματα της "σβήστηκε" από το νέο πρόγραμμα " f ".

Οι παραπάνω εντολές προς το Σύστημα από μόνες τους είναι αδύνατες, σε συνδιασμό όμως συνιστούν ένα ικανό μηχανισμό δημιουργίας διαδικασιών.Στο παρακάτω παράδειγμα το σύμβολο " = " σημαίνει το γνωστό
" := " και το " == " σημαίνει " = ", σύμφωνα με την ορθολογία της γλώσσας C.

<u>Πρόγραμμα Αρχικής Διαδικασίας (Γονιός)</u>

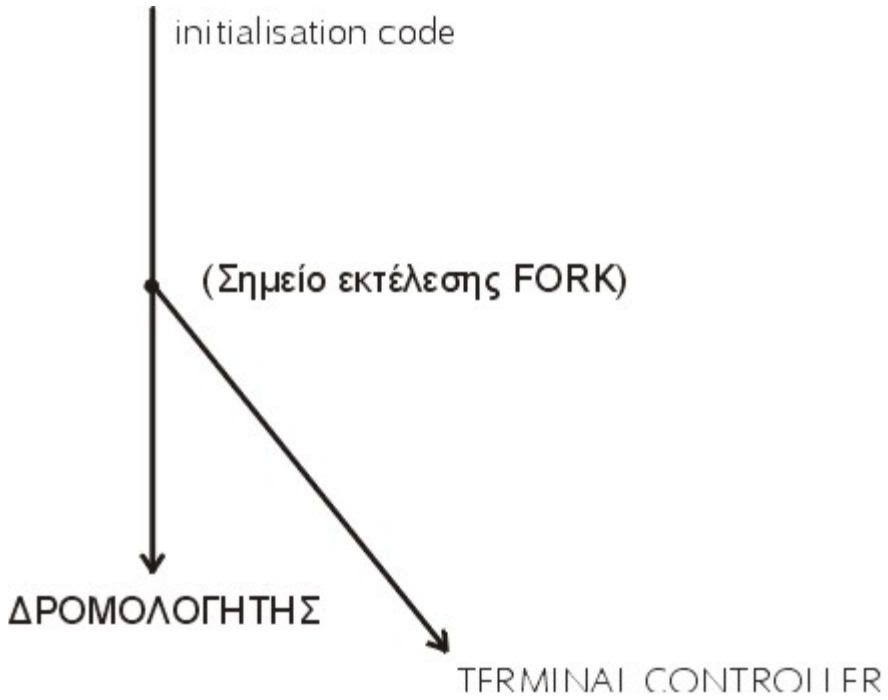
```
int  i  ;

i = forK ( ) ;    Σ' αυτό το σημείο έχουμε 2 διαδικασίες ΓΟΝΙΟΣ και ΠΑΙΔΙ, το παρόν πρόγραμμα
σε ποιά                    αναφέρεται από εδώ και εμπρός ;

if (i = = O)      Η τιμή του i δίνει την λύση

  {
    exec (fileΠΑΙΔΙ, παράμετροι)    [ O για το ΠΑΙΔΙ , = O για το ΓΟΝΙΟ , -1 εννον ]
    exit                [ ΑΥΤΗ Η ΕΝΤΟΛΗ ΔΕΝ ΑΦΗΝΕΙ ΝΑ ΕΚΤΕΛΕΣΤΕΙ ΤΟ ΜΕΡΟΣ ΤΟΥ
ΠΡΟΓΡΑΜΜΑΤΟΣ ]
...}....                    *ΠΟΥ ΑΦΟΡΑ ΤΟΝ ΓΟΝΙΟ
* συνέχεια ΓΟΝΙΟΥ
```

Σε δεύτερο παράδειγμα θα αναφερθούμε πως μετα την ενεργοποίηση του φορτωτή bootstrap loader δημιουργούνται οι πρώτες διαδικασίες στο Σύστημα. Παραστατικά η λειτουργία του FORK δίνεται στο παρακάτω σχήμα



<u>Πρόγραμμα</u>(ορίζει την πρώτη διαδικασία στο Σύστημα)

INITILIALISATION

CODE

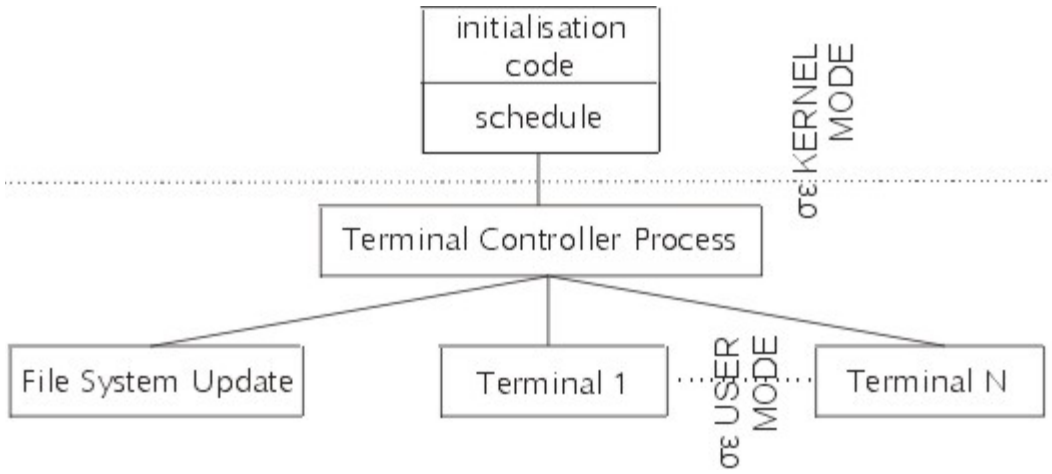
```
if (forK ( )==O)
  {
    exec (terminal_controller,...)
    exit
  }
```

SCHEDULE

Μετά τις Αρχικές τιμές, μέσω FORK και EXEC φτιάχνεται μια νέα διαδικασία που έχει έλεγχο των τερματικών, φτιάχνει με τη σειρά της μια διαδικασία για κάθε τερματικό του Συστήματος και συνεχίζει τη λειτουργία της με το να καταγράφει κάθε 30'' τη κατάσταση του FILE SYSTEM πάνω στο Δίσκο για ασφάλεια στην περίπτωση που "πέσει" το Σύστημα πχ.κόψιμο παροχής ρεύματος.

Η αρχική διαδικασία αφού φτιάξει απογόνους εκτελεί το πρόγραμμα SCHEDULER που δρομολογεί όλες τις διαδικασίες στο Σύστημα.

Στο παρακάτω σχήμα περιγράφεται η διάταξη των πρώτων Διαδικασιών από την έναρξη του Συστήματος.



Οι διαδικασίες Terminal 1,2,...N για κάθε τερματισμό στο Σύστημα είναι η κάθε μια ένας εκτελεστής εντολών δηλαδη command interpreters.Η κάθε μια είναι η εκτέλεση ενός SHELL.

<u>Η δομή του UNIX</u>

Το UNIX αποτελείται από τα εξής software εξαρτήματα :

<u>Το Kernel</u>
είναι ο πυρήνας του Συστήματος που ελέγχει το hardware και εκτελεί μια σειρά από βασικές λειτουργίες "low level".Τα άλλα συστατικά μέρη του UNIX καθώς και προγράμματα των χρηστών απευθύνονται στο Kernel για να εκτελέσει εντολές προς το σύστημα (system calls).

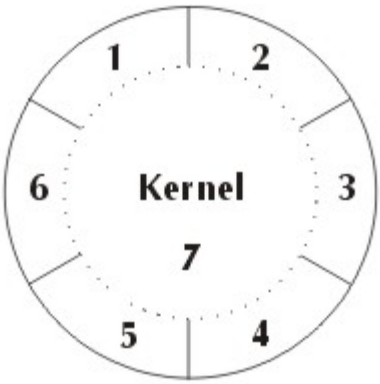
<u>Το SHELL</u> (ειδικό utility)
είναι ο εκτελεστής εντολών προς το UNIX, command interpreter

<u>Τα Utility Programs</u>
είναι commands που υποστηρίζουν μια σειρά δουλειές όπως την αντιγραφή αρχείων,text editing,calculations, αναπτυξη λογισμικού κ.α.

<u>Τα User Programs</u>
είναι τα προγράμματα που αναπτύσουν οι χρήστες από την στιγμή που πάρουν θέση μέσα στο UNIX τότε δεν ξεχωρίζουν από το υπόλοιπο UNIX ή τις εντολές προς αυτό.

<u>Το UNIX Kernel</u>

Εκτελεί βασικές λειτουργίες του Συστήματος, τέτοιες που λέμε ότι είναι "κοντά" στη μηχανή ή αλλιώς λέμε ότι είναι low level.Αυτές οι λειτουργίες δημιουργούν το UNIX environment πάνω σε συγκεκριμένο Hardware.Το πιο "κοντινό" τμήμα του βρίσκεται σε άμεση επαφή με το Hardware και έτσι "κρύβει" από τα άλλα εξαρτήματα του UNIX τη μηχανή καθιστώντας τα ανεξάρτητα της.Τι κάνει το KERNEL περιγράφεται στο επόμενο σχήμα και αποτελείται από τα εξής :



Το KERNEL και οι Λειτουργίες του

1. Management και Ασφάλεια Αρχείων.Το Kernel υλοποιεί το File System, οργανώνει και ελέγχει το mass storage (Δίσκοι,ταινίες) του Συστήματος, επιβάλει κάποια μέτρα Security στην προσπέλαση προς τα Αρχεία.

2. Εκτελεί τις εντολές για I/O, μεταφέροντας data μεταξύ I/O devices.

3. Process Scheduling & Management.Η λειτουργία αυτή του Kernel σχετίζεται με το πρόβλημα της κατανομής του χρόνου CPU μεταξύ διαδικασιών, βρίσκονται σε ανταγωνισμό.Με βάση έναν αλγόριθμο (δίνει σε κάθε process έναν αριθμό "προτεραιότητας") το σύστημα αποφασίζει ποιά διαδικασία θα έχει τον CPU.Ακόμα σχετίζεται με την επικοινωνία μεταξύ των διαδικασιών.

4. Memory management.Το έργο αυτής της Λειτουργίας του Kernel είναι η τοποθέτηση και μεταφορά των διαδικασιών μεταξύ Μνήμης και Δίσκου.Υπάρχουν διάφορες τεχνικές πάνω στις οποίες στηρίζεται αυτή η λειτουργία :
α. process swapping (Αναστολή διαδικασιών)
β. Virtual Memory. Εδώ δεν χρειάζεται ολόκληρη η process να είναι φορτωμένη στη Μνήμη, όπως στη τεχνική "α", αλλά μόνο τα τμήματα της που άμεσα χρειάζονται για την εκτέλεση της.

5. Η τεχνική interrupts και error handling.

6. Λογιστικά του Συστήματος.

7. Στο εσωτερικό ή πιο "χαμηλό" επίπεδο το Kernel "ακουμπάει" το Υλικό της μηχανής.Αυτό το εξάρτημα του Kernel πρέπει να είναι προσαρμοσμένο στο συγκεκριμένο hardware, πχ1.το κάθε Σύστημα (Hware + Sware) έχει τη δικιά του τεχνική για memory management, πχ2. τα εξαρτήματα που έχουν άμεση επαφή με τα I/O devices, αυτά που ονομάζονται Drivers, όλα αυτά έχουν χαρακτηριστικά σε εξάρτηση με το Υλικό και έτσι πρέπει να προσαρμοστούν από το σχετικό μέρος του Kernel.

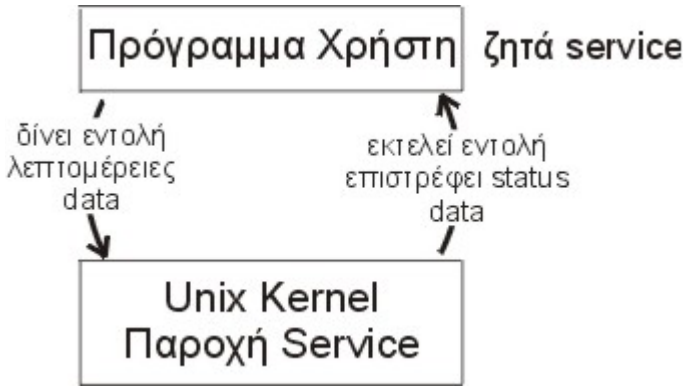
Ο όρος "porting" αφορά την προσαρμογή εξαρτημάτων του Kernel στο Hardware.

<u>]System call interface</u>

Τα διάφορα εργαλεία (utilities) του UNIX και τα προγράμματα εφαρμογών απευθύνονται στο Kernel για εκτέλεση εντολών προς το Σύστημα (system calls), πχ. όταν το πρόγραμμα λογιστικής χρειάζεται στην είσοδο του μια γραμμή data από ένα τερματικό, τότε απευθύνεται στο Kernel (τμήμα 2) έχουμε δραστηριοποίηση του 2 που παίρνει τη γραμμή data πχ. "αβγδεζηθικλμνξοπρστυφχψω" και την "περνάει" στο πρόγραμμα.Ο μηχανισμός των system calls είναι το ενδιάμεσο Kernel <----> προγραμμάτων εφαρμογών.Έτσι οι εντολές προς το Σύστημα αποτελούν το στάνταρ interface του UNIX.Έτσι η μεταφορά (portability, transfer) των προγραμμάτων πάνω στα UNIX γίνεται ανεξάρτητη του Hardware.

<u>Το SHELL</u> (δηλαδή το σύστημα διαλόγου με το UNIX)

Το παρακάτω σχήμα δείχνει τη λειτουργία του SHELL



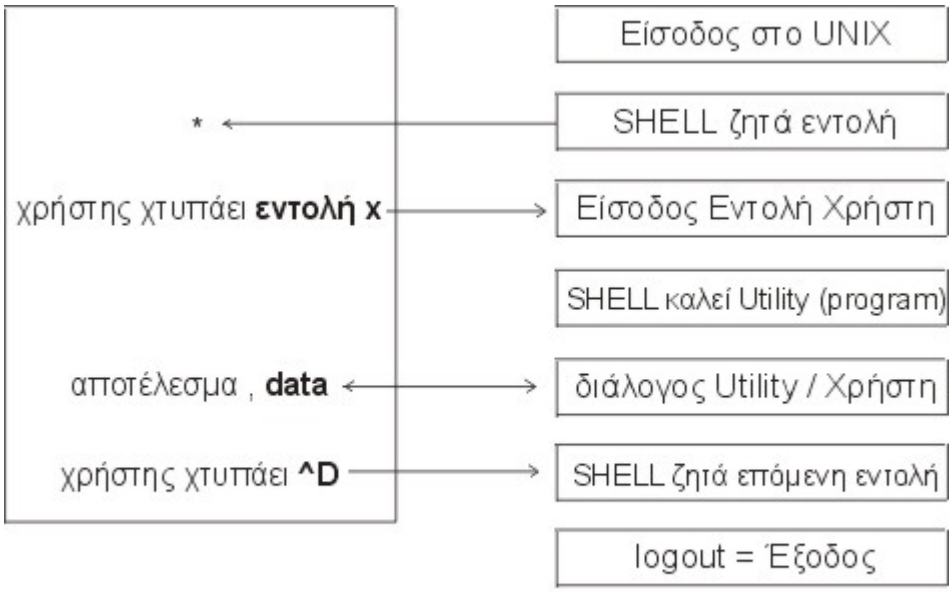
Χρησιμοποιείται πιο συχνά από κάθε άλλο εξάρτημα του UNIX, χειρίζεται το διάλογο μεταξύ Χρήστη και Συστήματος.Αυτό γίνεται με την ερμηνεία των εντολών προς το Σύστημα και την δραστηριοποίηση για την εκτέλεση τους.Είναι δηλαδή ένας command interprefer.Το διάγραμμα ροής που ακολουθεί δείχνει τα βήματα του SHELL.Το συνταχτικό των εντολών είναι :

Όνομα Εντολής	**Options**	**Arguments**
πχ. ls	-l	Αρχείου

δίνει στο τερματικό τον κατάλογο των files, δηλαδή των ονομάτων τους, που περιέχει το file "Αρχείου"

<u>σημείωση :</u> Υπάρχει στο UNIX ειδική κατηγορία αρχείων που λέγονται directory (=κατάλογος) και περιέχουν ονόματα και άλλες λεπτομέρειες αρχείων.

Οθόνη Τερματικού



Τα βήματα του SHELL

Ένα από τα utility του UNIX είναι το ίδιο το SHELL (είναι και αυτό ένα πρόγραμμα) η εντολή **sh**.

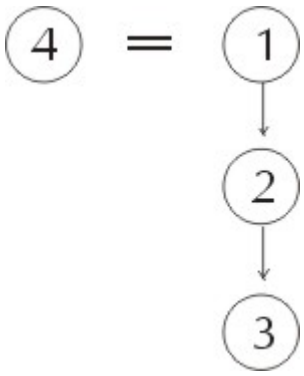
Πχ. Έστω αρχείο ΔΟΚΙΜΗ με περιεχόμενο :

ΔΟΚΙΜΗ

1. os source (κάλεσε τον assembler πάνω στο source)
2. cp a.out testprog (αντέγραψε το αποτέλεσμα του στο αρχείο testprog)
3. testprog (εκτέλεσε το object file testprog)

Η εντολή 4. **sh<ΔΟΚΙΜΗ** κατευθύνει την είσοδο (input) του SHELL πάνω στο αρχείο (command file) ΔΟΚΙΜΗ αντί να είναι πάνω στο τερματικό.

'Ετσι



Σημείωση: Το ρόλο του χαρακτήρα " < " ή " > " σ' αυτή την περίπτωση

Άλλα προγράμματα στο ρόλο του

Εκτός του διαλόγου που επιβάλει το SHELL στο χρήστη το UNIX παρέχει τα μέσα για τροποποίηση του, πχ.ορισμένες κατηγορίες χρηστών να "βλέπουν" μόνο υποσύνολο του Συστήματος, όπως κάποιο παιχνίδι ή τον editor (εντολή **ed**).

Η διαδικασία που χειρίζεται τα login αφού κάνεις έλεγχο του password κ.τ.λ. καλεί το SHELL.Με τροποποίηση του command file που σχετίζεται με αυτή την διαδικασία επιτυγχάνεται ο παραπάνω περιορισμός, δηλαδή αντί του **sh** βάζουμε **ed**.

Αρχειοterminalcontrolerprocess

χειρισμός login / logout / file system update κ.τ.λ.
έλεγχος user name user password
:
:
ed

Κύρια χαρακτηριστικά του SHELL

- Interactive programming (επικοινωνία με το UNIX έχει την μορφή διαλόγου)

- background processing για χρονοβόρα tasks

- input/output redirection (κατεύθυνση εσόδων / εξόδων)

- pipes * (δυνατότητα συνδιασμού προγραμμάτων)

- command files (μια σειρά εντολών πχ. ed,cp,sh,date,ls,cd κ.α. αποθηκεύονται από το χρήστη σ'ένα αρχείο και μπορούν να εκτελεστούν σαν μια εντολή)

- το SHELL σαν γλώσσα προγραμματισμού (με if,do,while,variables)

* **Pipes :** Είναι το μέσο επικοινωνίας διαδικασιών (process communication).Όταν data χρειάζεται να περάσει από process σε process.Στην ουσία είναι ένα αρχείο πάνω στο οποίο γράφουν ή διαβάζουν οι διαδικασίες σε επικοινωνία.

πχ.
sort αριθμός | βρες Λογάριθμότους

Η παραπάνω εντολή δημιουργεί 2 process,το πρώτο βάζει αριθμούς σε διάταξη, η δεύτερη βρίσκει το λογάριθμο τους.

σημείωση: το ρόλο του χαρακτήρα **" 1 "**

Το σύστημα Αρχείων (File System)

Το εξάρτημα αυτό,μια από τις λειτουργίες του Kernel ελέγχει την αποθήκευση πληροφορίας στην Δευτερεύουσα Μνήμη του Συστήματος (mass storage), διευκολύνει την οργάνωση, επιτρέπει την επανεξέταση της (information retrieval).

Κύρια Χαρακτηριστικά του Συστήματος

Hierarchical Structure
Οι χρήστες έχουν τη δυνατότητα να οργανώνουν συσχετιζόμενες πληροφορίες (με την γενικότερη έννοια) σε ομάδα και να επεξεργάζονται μια ομάδα αρχείων σαν μια οντότητα,

File Expansion
Τα αρχελια επεκτείνονται δυναμικά ανάλογα με τις ανάγκες. Ο χρήστης δεν χρειάζεται να αποφασίσει πιο θα είναι το μέγεθος των αρχείων τους.

Structurelless Files
Το σύστημα δεν επιβάλει καμιά εσωτερική δομή στο περιεχόμενο του αρχείου.Η ερμηνεία και η οργάνωση του περιεχομένου είναι στην αποκλειστικότητα του χρήστη.

Ασφάλεια υπάρχει σύστημα ελέγχου προσπέλασης στα αρχεία.

File and Device Independence

Τα αρχεία και τα I/O devices ανήκουν στην ίδια κατηγορία, η επεξεργασία τους είναι η ίδια. Έτσι προγράμματα που έχουν εισόδους/εξόδους πάνω σε αρχεία μπορούν να χρησιμοποιηθούν πάνω σε τερματικά,εκτυπωτές ή να έχουν επικοινωνία (μέσω pipes) με άλλα προγράμματα.

Η έννοια File/Αρχείο

Το μέσο αποθήκευσης κάθε είδους πληροφορίας στο Σύστημα, με δοσμένο όνομα (file name) που αποτελείται από 1-14 χαρακτήρες.

Η έννοια directory / κατάλογος

Αποτελεί την αντιστοιχεία μεταξύ ονομάτων αρχείων και των αρχείων.Η αντιστοιχία αυτή προσδιορίζει κάποιο δομή πάνω στο file system.

Το Σύστημα αρχείων αποτελείται από τρία είδη αρχείων :

(α) Απλά αρχεία (ordinary disk files)
(β) Κατάλογοι (directories)
(γ) Ειδικά αρχεία (special files)
(δ) Pipes

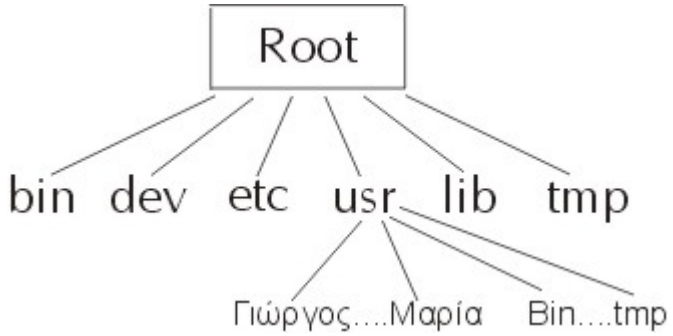
Απλά Αρχεία

Ένα αρχείο περιέχει όποια πληροφορία έχει τοποθετήσει ο χρήστης πχ. object programs,κείμενα κα.Αρχεία με περιεχόμενο κάποιο κείμενο αποτελούνται από μια σειρά χαρακτήρες. Οι γραμμές στο κείμενο ξεχωρίζονται από τον χαρακτήρα " NL ".Προγράμματα σε γλώσσα μηχανής αποτελούνται από μια σειρά words, όπως ακριβώς βρίσκονται στη

Μνήμη. Υπάρχουν βέβαια προγράμματα που οι assembler και ο loader παίρνουν στην είσοδο τους ένα binary αρχείο με συγκεκριμένη δομή .Όμως αυτή η δομή πάνω στα αρχεία είναι υπόθεση των προγραμμάτων που τα χρησιμοποιούν και όχι του Συστήματος Αρχείων.

Κατάλογοι

Κάθε χρήστης έχει το δικό του κατάλογο αρχείων, έχει τη δυνατότητα να φτιάξει υποκαταλόγους (sub - directory). Το σύστημα διατηρεί αρκετούς καταλόγους για τη δική του χρήση. Όπως στο παρακάτω σχήμα :



<u>Ο ρόλος των καταλόγων του Συστήματος:</u>

/ bin	UNIX utilities
/ dev	Μηχανισμοί Ε/Ε, Ειδικά Αρχεία
/ lib	βιβλιοθήκες για επεξεργαστές γλώσσας
/ ete	προγράμματα, πίνακες για system administration
/ tmp	προσωρινά αρχεία
/ usr / bin	βοηθητικός χώρος του /bin
/ usr / games	παιχνίδια
/ usr / lib	επεξεργαστές κειμένου κ.α.
/ usr / spool	αρχεία για εκτύπωση
/ usr / src	προγράμματα σε μορφή C,furtram,parcel κ.τ.λ.

Προσπέλαση στο αρχείο Μαρία εκφράζεται :
/ usr / Μαρία

Κάθε κατάλογος έχει τουλάχιστον δυο στοιχεία στο περιεχόμενο του.Το όνομα " **.**" που σημαίνει τον τωρινό κατάλογο εργασίας χρήστη (current working) και το όνομα "**..**" που αναφέρεται στον πατέρα - κατάλογο, δηλαδή αυτόν που περιέχει τον δοσμένο κατάλογο.Αυτά τα συμβατικά ονόματα αντικαθιστούν τα ονόματα που έχουν οι δοσμένοι κατάλογοι κατάλογοι.Βλέπουμε ότι το Σύστημα Αρχείων, με τους καταλόγους επιβάλει τον περιορισμό να έχει την δομή Δέντρο με αρχική ρίζα τον κατάλογο roat (η /).

Το όνομα Αρχείου είναι απλώς το μέσο προσδιορισμού κάποιου αρχείου στο σύστημα.Ένα αρχείο μπορεί να έχει δύο ή περισσότερα ονόματα.Έτσι λέμε ότι υπάρχουν πολλαπλοί δεσμοί (links) πάνω σε αυτό το αρχείο. Αυτό καθιστά την αντιγραφή αρχείων που είναι κοινά, περιτή.Αυτό ισχύει μόνο για αρχεία που δεν είναι κατάλογοι. Από μια σκοπιά μπορούμε να πούμε ότι το κάθε αρχείο δεν "κατοικεί" σε κάποιο κατάλογο αποκλειστικά.Βέβαια όταν σβήνουν όλοι οι δεσμοί πάνω του τότε αυτό εξαφανίζεται από το Σύστημα.Ο προσδιορισμός κάθε αρχείου μέσα από την έκφραση ολόκληρου του

δρόμου,μέσα από καταλόγους,που οδηγεί σε αυτό, είναι δύσχρηστη διαδικασία.Συνήθως οι χρήστες εργάζονται με μια μικρή ομάδα αρχείων μπλεγμένα σ' ένα κατάλογο.Το σύστημα διατηρεί μεταβλητή που καταγράφει τον τωρινό κατάλογο εργασιών current working directory. Έτσι ο προσδιορισμός ξεκινάει από αυτό και όχι απο το Root.

Ειδικά Αρχεία

Αυτά λειτουργούν σαν τα απλά Αρχεία (ordinary disk files) έχουν την ίδιαιτερότητα ότι σε κάθε ειδικό Αρχείο αντιστοιχεί τουλάχιστον ένας Μηχανισμός Ε/Ε (I/O device).Τα χρησιμοποιούμε για διάβασμα ή γράψιμο όπως και τα απλά αρχεία με την διαφορά όμως ότι εδώ συμβαίνει (το side effect) η μετακίνηση, μεταφορά πληροφορίας στο αντίστοιχο Μηχανισμό ή από αυτόν. Ο κατάλογος του Συστήματος /dev περιέχει όλα αυτά τα ειδικά Αρχεία όμως μπορεί να δημιουργηθεί δεσμός πάνω τους από αλλού πχ. η εκτύπωση πάνω σε χαρτο-ταινία.

/ dev / ppt

Ειδικά αρχεία υπάρχουν για κάθε γραμμή επικοινωνίας, κάθε Δίσκο, κάθε ταινία και Μνήμη.Μ' αυτόν τον τρόπο έχουμε τα εξής οφέλη :

1. Κοινή αντιμετώπιση Μηχανισμού Ε/Ε και Αρχείων
2. Κοινό συνταχτικό για προσδιορισμό τους. Όταν το πρόγραμμα έχει παράμετρο το όνομα αρχείου μπορεί να "πάρει" και το όνομα Μηχανισμού Ε/Ε.
3. Το Σύστημα Ασφάλειας Αρχείων μπορεί να εφαρμοστεί πάνω στους Μηχανισμούς Ε/Ε (I/O devices)

Έχουμε δυο κατηγορίες ειδικών αρχείων :

1. Μπλοκ : Αφορά μονάδα μεταφοράς πληροφορίας (ποσότητα) 512 byte.

2. Χαρακτήρα : Ανά χαρακτήρα πχ. Τηλέτυπο.

<u>Εντολή χρήσης αρχείου</u>

open

close

seek

Ένα αρχείο πρέπει πρώτα να το "ανοίξουμε" με την εντολη

θέση = open (όνομα, γιατί)

Η "θέση" λέγεται προσδιοριστής αρχείου

file descriptor είναι ένας ακέραιος αριθμός

"γιατί" δείχνει τον λόγο προσπέλασης, για διάβασμα, γράψιμο ή και τα δύο.

Για να δημιουργήσουμε ένα αρχείο ή για να επαναγράψουμε πάνω σε παλιό, επικαλύπτοντας το ολοκληρωτικά χρησιμοποιούμε την εντολή **create**.
Εάν το αρχείο "όνομα" είναι ήδη μέσα στο σύστημα τότε η εντολή έχει αποτέλεσμα τη συρίκνωση του σε μέγεθος Ο.
Σε περίπτωση που χρήστες μπαίνουν με ταυτόχρονα μέσα στο ίδιο αρχείο, η λογική σειρά των πράξεων τηρήται αλλά δεν είναι ασφαλής.Αυτός ο κίνδυνος θα ήταν πραγματικό πρόβλημα μόνο σε μεγάλα αρχεία πχ. τραπεζικών πληροφοριών.

Εκτός περιπτώσεων που θα αναφέρουμε παρακάτω το διάβασμα/γράψιμο αρχείων είναι ακολουθιακό δηλαδή ανά byte προς το τέλος του αρχείου ή προς την αρχή του.

Σημείωση: Σειριακή προσπέλαση σημαίνει από την αρχή του αρχείου προς το τέλος του μόνο, τον έναν record μετά τον άλλο (πχ.record = 1 byte).Η προσπέλαση σε συγκεκριμένο byte του αρχείου γίνεται με το να προδιορίζουμε την σχετική του θέση από την αρχή του αρχείου (byte Ο). Αυτός ο τρόπος οργάνωσης των αρχείων έχει το πλεονέκτημα να μη "βλέπει" ο χρήστης τα blocks ή τα records των πραγματικών αρχείων (physical files) δηλαδή των χώρων αποθήκευσης πάνω στα μέσα Δίσκους κ.τ.λ.

Θα δούμε παρακάτω πως τελικά υλοποιείται αυτή η απλή δομή των αρχείων μέσα από index blocks, data blocks κ.τ.λ. Έτσι η πληροφορία πάνω στα αρχεία οργανώνεται με βάση την ιδιαίτερη δομή της (logical structure) και όχι τη δομή του τρόπου υλοποίησης των αρχείων (physical structure).Για κάθε "ανοιχτό" αρχείο, μετά από εκτέλεση της εντολής **open** * το σύστημα φτιάχνει έναν pointer (Δείχτη) που στέκεται στο επόμενο byte του αρχείου που θα διαβαστεί ή θα γραφτεί, πχ.Εάν v bytes διαβάζονται, ο δείχτης προχωράει v bytes από την θέση του.Το διάβασμα / γράψιμο γίνεται με τις εντολές :

αποτέλεσμα **: = read** (Προσδ. Αρχ,θέση. byffer,πόσα)
write

* open (όνομα, i) i = Ο για διάβασμα
i = 1 για γράψιμο
i = 2 για διάβασμα και γράψιμο

Οι παράμετροι των εντολών read/write:

"Προσδ. Αρχ." :
Είναι το αποτέλεσμα των εντολών open,create δηλαδή ένας file descriptor.

"θέση.byffer" :
Είναι η θέση (location) του αριθμού " Πόσα" bytes συνεχόμενα, πάνω στα οποία θα αποθηκευτεί η είσοδος (input) πληροφορίας.Όταν πρόκειται για γράψιμο τότε αυτά τα bytes στέλνονται στην έξοδο (output)

Το "αποτέλεσμα" των παραπάνω εντολών είναι ο τελικός αριθμός των bytes που μεταφέρθηκαν.Σε περίπτωση που συμβαίνει κάποιο λάθος στη λειτουργία της εντολής (I/O error) αυτός ο αριθμός διαφαίρει από τον επιθυμητό αριθμό "Πόσα".Δηλαδή η εντολή δεν εγγυάται ότι "πόσα" bytes θα διαβαστούν πχ1. όταν το αρχείο αναφέρεται σε ένα τηλέτυπο που μέγιστο αριθμό έχει μια γραμμή bytes, πχ2. όταν η address του byffer δεν είναι σωστή κ.α.

Όταν το "αποτέλεσμα" είναι ο αριθμός Ο σημαίνει ότι ο Δείχτης κτύπησε το τέλος του αρχείου, δηλαδή μηδέν bytes απομένουν να διαβαστούν.

Όταν γράφουμε πάνω σε ένα αρχείο μόνο τα bytes που θέλουμε να αλλάξουμε επηρεάζονται.Αυτά προσδιορίζονται από το Δείχτη του συστήματος και τη παράμετρο "πόσα".Εάν το τελευταίο byte που θέλουμε να γράψουμε έχει θέση πέρα από το τωρινό τέλος του αρχείου το αρχείο επιμηκείνεται.

<u>Άμεση (Random) Προσπέλαση στα Αρχεία</u>

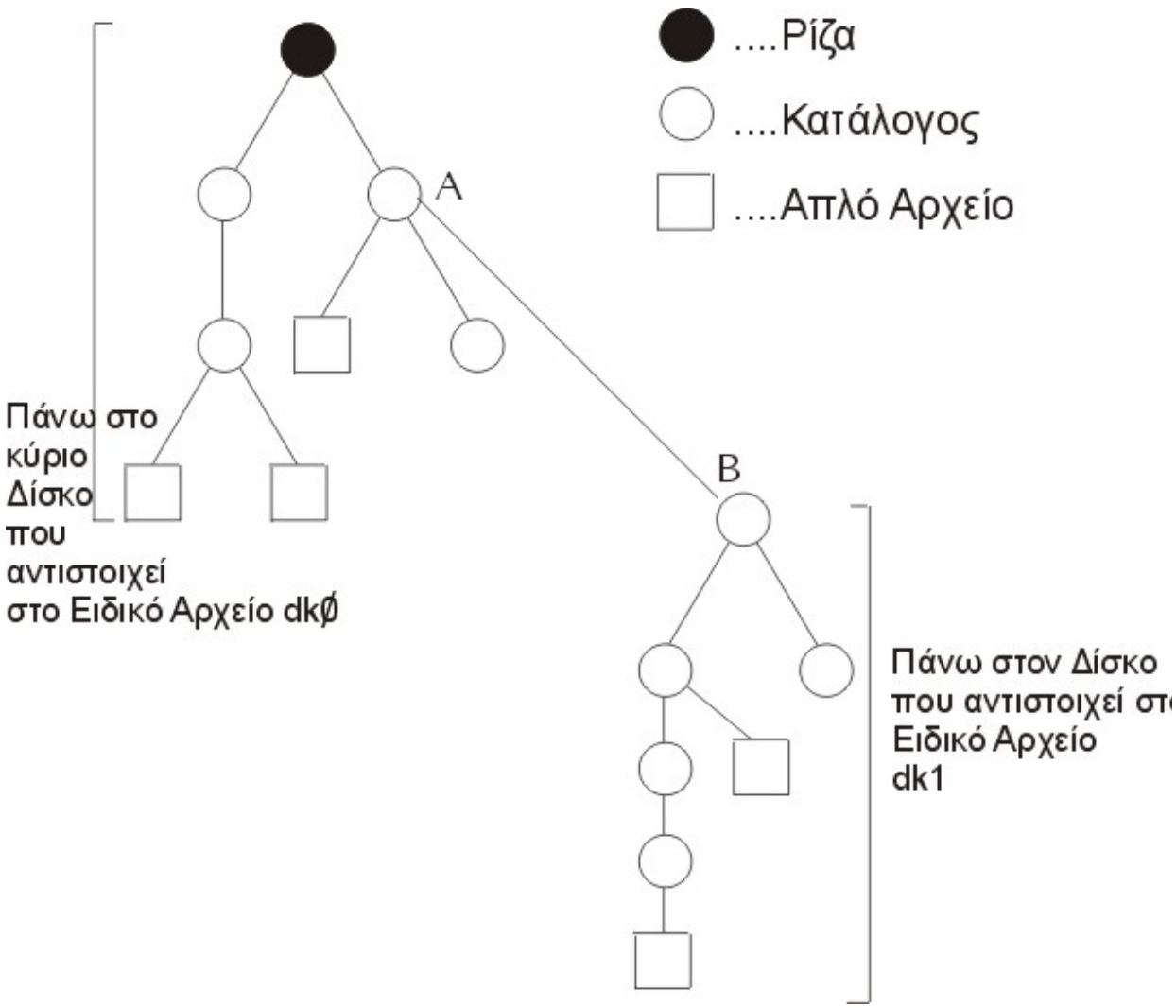
Η άμεση προσπέλαση σε συγκεκριμένο byte, χωρίς να "δούμε" άλλα bytes που μεσολαβούν μεταξύ αρχής του αρχείου (byte Ο) και του συγκεκριμένου, γίνεται απλά με την μετακίνηση του Δείχτη που διατηρεί το σύστημα για κάθε αρχείο στην επιθυμητή θέση.Αυτό είναι δουλειά της εντολής seek :

27 29 -2
θέση **: = seek** (Προς.Αρχ, Αφετηρία,Απόσταση)

"θέση" είναι που θέλουμε να σταθεί ο Δείχτης πάνω σε ποιό byte.Αυτό προσδιορίζεται από τους παραμέτρους που λένε πχ. -2 bytes ξεκινώντας από το τριακοστό byte (δηλαδή byte 29).
Ο Δείχτης τοποθετείται στο byte 27.Τώρα το επόμενο read η write θα αφορά αυτό το byte.Το seek δεν εφαρμόζεται πάνω σε τηλέτυπο,χαρτοταινία . (ΓΙΑΤΙ ;)

<u>Κινητό Σύστημα Αρχείων</u>

Το Σύστημα Αρχείων που εξετάσαμε μέχρι τώρα λειτουργεί κάτω από την υπόθεση ότι όλα τα αρχεία βρίσκονται πάνω σ' ένα μαζικό μέσο αποθήκευσης (single storage volume).Το Σύστημα Αρχείων μπορεί να επεκταθεί πάνω σ' άλλα μέσα αποθήκευσης (σ ' άλλους μηχανισμούς [devices]).Βέβαια η ρίζα (Root) του Συστήματος Αρχείων βρίσκεται πάντοτε στο ίδιο μέσο.Η επέκταση γίνεται με την προσάρτηση του νέου μέσου (volume) πάνω στο Δέντρο.



Η εντολή **mount** A / der / dk1 προσάρτει το (Ανεξάρτητο) κινητό σύστημα αρχείων B σαν υπό-κατάλογο του καταλόγου A.Μετά την προσάρτηση δεν υπάρχει διαφορά μεταξύ των αρχείων πάνω στα δύο μέσα (volumes)
<u>ΠΕΡΙΟΡΙΣΜΟΣ :</u> δεσμοί (links) μεταξύ των αρχείων που βρίσκονται πάνω σε διαφορετικά μέσα αποθήκευσης (direct access storage devices) δεν επιτρέπονται ΓΙΑΤΙ ;

Η εντολή **umount / der / dk1** αποσπά το υπό - Σύστημα αρχείων B, που βρίσκεται πάνω στο κινητό μέσο αποθήκευσης στο δίσκο dk1.Κάθε απόπειρα προσπέλασης πχ. cat A / x αρχείου του A θα δώσει σφάλμα.

<u>Εκτύπωση</u>

Σε κάθε σύστημα με πολλούς ταυτόχρονους χρήστες (multi-user environment) για να αποφύγει την εμπλοκή που προκύπτει αν στέλνουμε στον εκτυπωτή "ταυτόχρονα" εκτυπώσεις το Σύστημα χρησιμοποιεί την τεχνική spooling (τυλίγω πάνω σε κύλινδρο).Αυτή η τεχνική επιτρέπει στον χρήστη να δίνει εντολή για εκτύπωση αρχείων του καθώς προκύπτει η ανάγκη ενώ το Σύστημα οργανώνει την εκτύπωση των στοιχείων Εξόδου με τάξη.Το UNIX περιλαμβάνει διάφορες τέτοιες εντολές (utilities) ανάλογα με τι είδους εκτυπωτές υπάρχουν πάνω στο Σύστημα. Εφαρμόζοντας την τεχνική spooling οι εντολές λειτουργούν με την αντιγραφή των αρχείων για εκτύπωση πάνω σε προσωρινά αρχεία του δίσκου και μετά ένας αριθμός διαδικασιών δευτερεύουσας προτεραιότητας (background

processes) τα εκτυπώνουν ένα προς ένα.Για εκτύπωση έχουμε την εντολή **lp** (line printer) με παραμέτρους τα ονόματα αρχείων (ου), τον αριθμό αντιγράφων, το είδος εκτυπωτή.Η εντολή **cancel** αποσύρει την εντολή για εκτύπωση.

Λογιστικά

Το UNIX διαθέτει ένα αριθμό αρχείων (log files) πάνω τους καταγράφει τις δραστηριότητες του Συστήματος. Σ΄αυτά αντιστοιχούν διαδικασίες (utilities) που εκτυπώνουν τα στοιχεία που συσωρεύονται στα log files.Τα στοιχεία αυτά χρησιμοποιούνται για να προσδιοριστεί το είδος χρήσης του Συστήματος, να εντοπιστεί η ανάγκη για περισσότερη μνήμη, περισσότερο χώρο αποθήκευσης πάνω σε δίσκους,τερματικά, το κόστος χρήσης από διάφορα τμήματα χρηστών.

Το Σύστημα έχει δυο log files ένα για Λογιστικά *Ανά διαδικασία (per process)*. Πάνω σ' αυτό καταγράφεται το όνομα του προγράμματος που εκτελέστηκε, το χρόνο που πέρασε στο Σύστημα, το χρόνο χρήσης της Κεντρικής Μονάδας Επεξεργασιών (CPU), το μέγεθος κύριας μνήμης, τον αριθμό I/O εντολών για Είσοδο / Έξοδο, τη ταυτότητα του χρήστη (id of process owner), το όνομα του τερματικού που ξεκίνησε η process και ένα αρχείο για Λογιστικά *Ανά Είσοδο στο Σύστημα (per login)*. Το Σύστημα καταγράφει πάνω του το όνομα του Χρήστη, το χρόνο ΚΜΕ, το χρόνο σύνδεσης του τερματικού, τον αριθμό processes που εκτελέστηκαν, πληροφορίες σχετικά με την χρήση των Δίσκων.

Ασφάλεια Αρχείων

Η εντολή **chown** Makis Αρχείο Τ αλλάζει τον ιδιοκτήτη του αρχείου. Η εντολή **chgrp.** έτος Δ' Αρχείο αλλάζει την ομάδα - ιδιοκτήτη του αρχείου "ΑρχείοΤ". Η εντολή **chmod** ρ ΑρχείοΤ αλλάζει τους περιορισμούς προσπέλασης πάνω στο αρχείο.Η παράμετρος ρ παίρνει τις παρακάτω τιμές :

O (- - -) καμιά προσπέλαση
1 (- - -) προσπέλαση για εκτέλεση μόνο
2 (w w -) προσπέλαση για γράψιμο μόνο
3 (w w x) προσπέλαση για γράψιμο και εκτέλεση
4 (r - -) προσπέλαση για διάβασμα μόνο
5 (r - x) προσπέλαση για διάβασμα και εκτέλεση
6 (r w x) προσπέλαση για διάβασμα ,εκτέλεση και γράψιμο

Όταν ζητάμε το περιεχόμενο ενός καταλόγου X ls -l X , με την παράμετρο -l ζητάμε μακρά αναφορά (long), το Σύστημα απαντά :

Αρχείο 1	19/7/84	Γιάννης	215	r - x	- - x
Αρχείο 2	8 /8/88	Πέτρος	7	I group	I άλλος

Η εντολή **su** ακολουθούμενη από το password μετατρέπει τον χρήστη σε super -user με κανένα περιορισμό προσπέλασης.

~~~~~~~~~~
~~~~~~~~~~

<u>Υλοποίηση του Συστήματος Αρχείων</u> (ΣΑ)

Κάθε μέσο αποθήκευσης (storage volume) που χρησιμοποιεί το ΣΑ έχει μια χαρακτηριστική δομή (format).Το format ενός δίσκου για το Λειτουργικό Σύστημα 05/32 είναι διαφορετικό από αυτό του UNIX.

Ας υποθέσουμε ότι έχουμε ένα δίσκο με χωρητικότητα 67 Mbyte και η κύρια μνήμη της μηχανής είναι τέτοια που ζητά το χωρισμό του δίσκου σε τρεις φανταστικούς δίσκους (virtual disks), ο καθένας είναι ένα Ειδικό Αρχείο με ονόματα και περιεχόμενο:

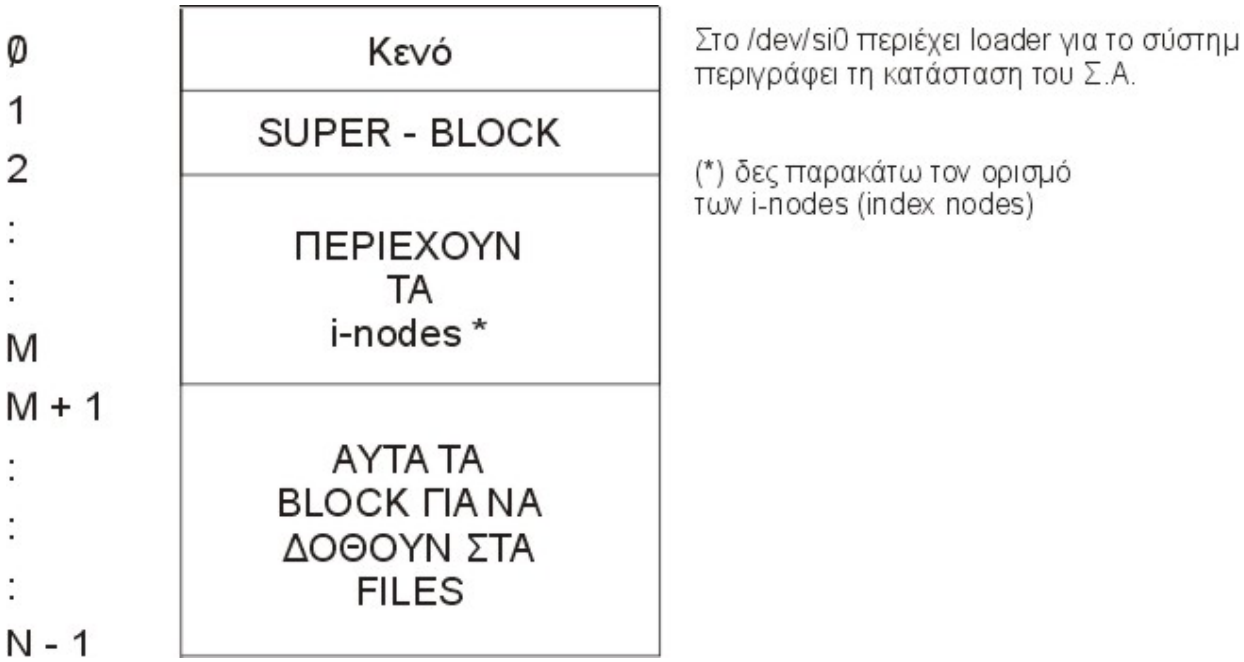
1. / dev / si O περιέχει τον κατάλογο Root, system source,libraries, εντολές.Αποτελείται απο 20.000 blocks
με 1600 block κλεισμένα για swap area από τον scheduler.Είναι μονίμος mounted
(φορτωμένος).

2. / dev / si 1 περιέχει όλους τους χρήστες, εντολές, τα αρχεία χρηστών, είναι φορτωμένο μονίμως, με 64000
blocks μέγεθος.

3. / dev / si 2 Με 47000, χώρος για διάφορες πληροφορίες έμμεσης χρήσης, δεν είναι φορτωμένος μόνιμα.

<u>Σημείωση:</u>
20,000 x 512 + 64,000 x 512 = 67M byte κάθε μπλοκ έχει μέγεθος 512 = 1/2 K bytes

Τα μπλοκ του κάθε φανταστικού δίσκου είναι αριθμημένα από 0 - (N - 1) όπου N είναι το μέγεθος του φανταστικού δίσκου (φ.δ.).Η οργάνωση του κάθε φ.δ. έχει ως εξής:



Τι είναι το SUPER-BLOCK;

Σ' αυτό το χώρο του φ.δ. καταγράφεται η κατάσταση του Σ.Α. Βρίσκεται εντός κυρίας μνήμης όταν το UNIX "τρέχει" και ανανεώνεται πάνω στο Δίσκο κάθε 30''.Αυτό είναι ένα μέτρο προστασίας από βλάβες που "ρίχνουν" το Σύστημα (System Failures).Έχει το εξής περιεχόμενο :

1. Πόσα block έχουν δωθεί για i-nodes
2. Το μέγεθος του Σ.Α. σε block
3. Το χώρο FREE LIST, από block ελεύθερα για χρήση
4. Το χρόνο της ημέρας

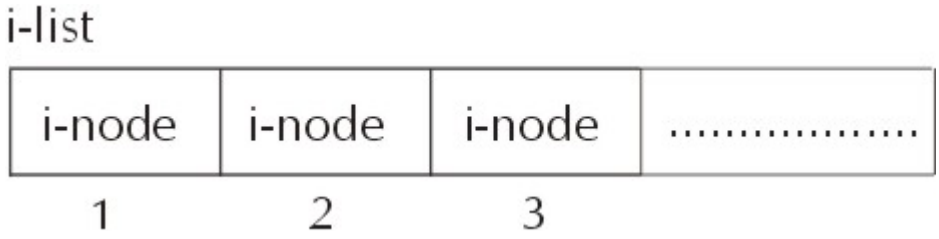
Ο αριθμός των block που χρησιμοποιούνται για i-nodes εξαρτάται από το μέγεθος του Σ.Α. και υπολογίζεται όταν το Σ.Α. δημιουργείται, παίρνοντας υπ' όψη ένα μέσο όρο για το μέγεθος αρχείων. Όταν ένα block παραχωρείται σ' ένα αρχείο ο αριθμός του είναι

1. + 3. < Αριθμ. BLOCK < => γρήγορος έλεγχος Σ.Α.

Όπως αναφέραμε προηγούμενα, ένας κατάλογος (directory) υλοποιεί την αντιστοιχεία μεταξύ ονομάτων αρχείων (file names) και προσδιοριστών αρχείων (file descriptors) που λέγονται i -numbers και χρησιμεύουν στην προσπέλαση αρχείων.

Προσωπικό <--- όνομα καταλόγου
Μισθοί 7
Φόροι 15
Υπάλληλοι 3
Εργάτες 19 <--- index number (συντ. i-number)
(όνομα αρχείου)

Το στοιχείο i-number χρησιμοποιείται σαν δείχτης index πάνω στον πίνακα i-list για προσπέλαση στη πληροφορία που περιγράφει το συσχετιζόμενο αρχείο και λέγεται i-node.



Ένας i-node περιγράφει στο σύστημα το συσχετιζόμενο αρχείο περιέχει την παρακάτω πληροφορία :

1. Το i-node είναι σε χρήση ή όχι
2. Τις θέσεις addresses που κρατάει το αρχείο πάνω στο πραγματικό δίσκο η ταινία.Αποτελείται από 8 words.
3. Το είδος αρχείου (απλό, Κατάλογος, Ειδικό)
4. Μικρό ή Μεγάλο αρχείο
5. Πότε έγινε η τελευταία προσπέλαση/τροποποίηση
6. Το μέγεθος του αρχείου
7. Το σύνολο δεσμών links πάνω στο αρχείο από άλλα αρχεία

8. Ταυτότητα ιδιοχτήτη
9. Άδεια permission για είδος προσπέλασης (read,write,execute) από χρήστες (owner,group, others δηλαδή ιδιοχτήτη,ομάδα,άλλοι)

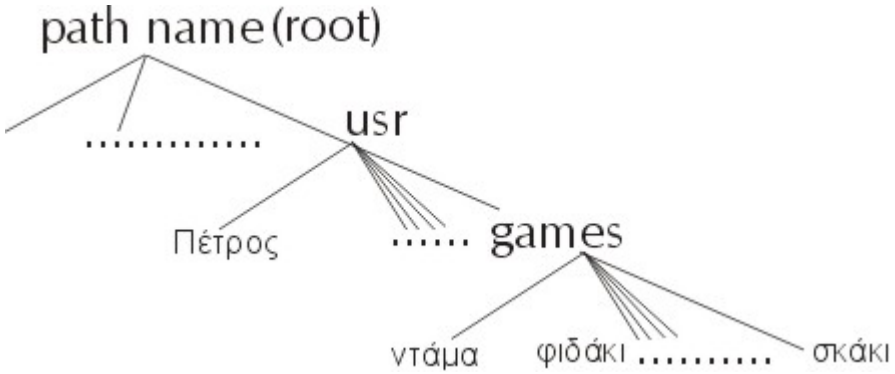
Η εκτέλεση εντολής open ή create έχει σαν αποτέλεσμα τη μετατροπή μιας έκφρασης προσδιορισμού αρχείου
σ' έναν i-number πχ. :

i-number : = create (/usr/games/σκακι, O)
path name *

Από το πίνακα i-list παίρνεται ένας i-node που δεν χρησιμοποιείται και ο αντίστοιχος i-number.Στο κατάλογο που περιέχει το αρχείο που δημιουργείται "σκάκι" μπαίνει το όνομα του αρχείου και ενός i-number.

<u>/ usr / games</u>
ντάμα
:
φιδάκι
:
:
σκάκι 88

* λέμε path name γιατί είναι ο δρόμος που οδηγεί στο αρχείο



Δεσμός (link) πάνω σ' ένα υπαρκτό αρχείο γίνεται ως εξής :

ln Νέο Υπαρκτό

Πρώτα φτιάχνεται ένα νέο όνομα μέσα σ' ένα κατάλογο πχ. στο τωρινό κατάλογο εργασίας. Ο i-number του υπάρχοντος αρχείου αντιγράφεται σ' αυτόν που αντιστοιχεί στο νέο όνομα.Η παράμετρος (7) του i-node που αντιστοιχεί στο Υπαρκτό αρχείο αυξάνεται +1.

Το σβήσιμο αρχείου

rm Νέο

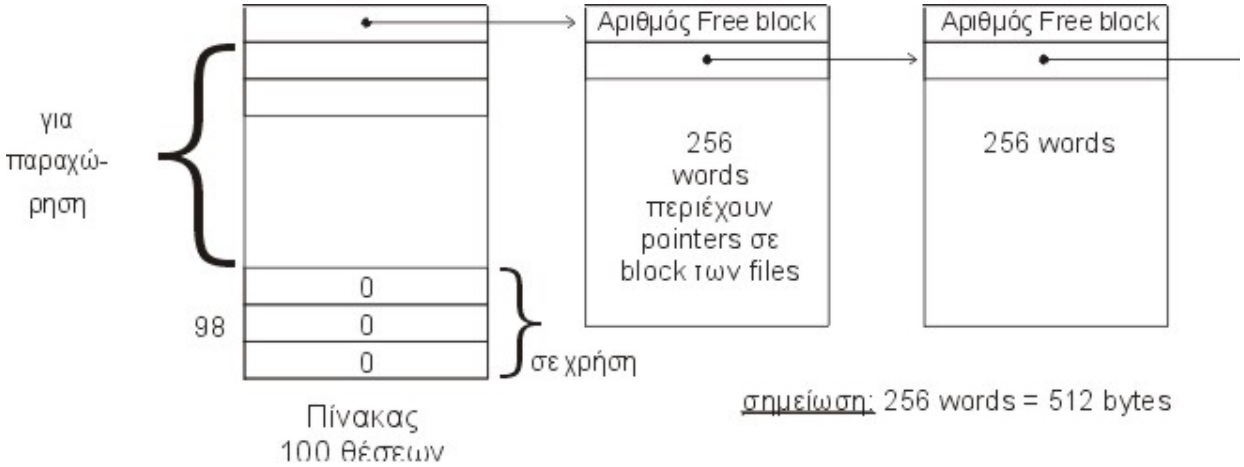
μειώνει αυτή την παράμετρο -1. Όταν η παράμετρος γίνει (7) γίνει μηδέν τότε ο αντίστοιχος i-node μαρκάρεται "κενός" και όλα τα block που δόθηκαν στο αρχείο (δες παρακάτω) επιστρέφονται στο σύστημα.

Πως παραχωρείται ο χώρος Δίσκου σε αρχεία

Ο χώρος Δίσκου (σταθερός ή κινητός ,fixed removable) που φιλοξενεί ένα Σ.Α. διατηρείται σε block μεγέθους 512 bytes = 1/2 Kb με φανταστικές (συμβατικές) διευθύνσεις (adresses) Ο - όριο Μηχανισμού. Όταν το Σ.Α. χρειάζεται χώρο σε Δίσκο ζητά ένα block.

Ο χώρος FREE LIST που βρίσκεται στο χώρο SUPER BLOCK διερευνάται και εάν βρεθεί αχρησιμοποίητο block επιστρέφεται η διεύθυνση του = αριθμός του block.

Ο χώρος FREE LIST έχει την εξής δομή :



Ο πίνακας των 100 θέσεων λειτουργεί σα δέσμη.Η πρώτη θέση δείχνει μια αλυσίδα block το κάθε ένα αποτελείται από 256 words το κάθε word είναι ένα pointer σε block μεγέθους 512 bytes που παραχωρείται σε αρχεία.Οι υπόλοιπες 99 θέσεις περιέχουν τον αριθμό ενός block που είναι ελεύθερο για χρήση η τον αριθμό Ο που σημαίνει πχ. το block 98 είναι σε χρήση.

Δημιουργία / Απόσπαση (σβήσιμο) Αρχείου

Αρχικά όταν ένα αρχείο δημιουργείται του δίνεται ένας i-node από τον πίνακα i-list.Ο πίνακας ερευνάται μέχρι να βρεθεί i-node με παράμετρο (1) να έχει τιμή "όχι σε χρήση". Ο δείχτης του πίνακα i-number που αντιστοιχεί τοποθετείται στον κατάλογο (directory) που περιέχει -, νέο αρχείο μαζί με το όνομα του αρχείου.Οι υπόλοιπες πληροφορίες τοποθετούνται στον i-node.

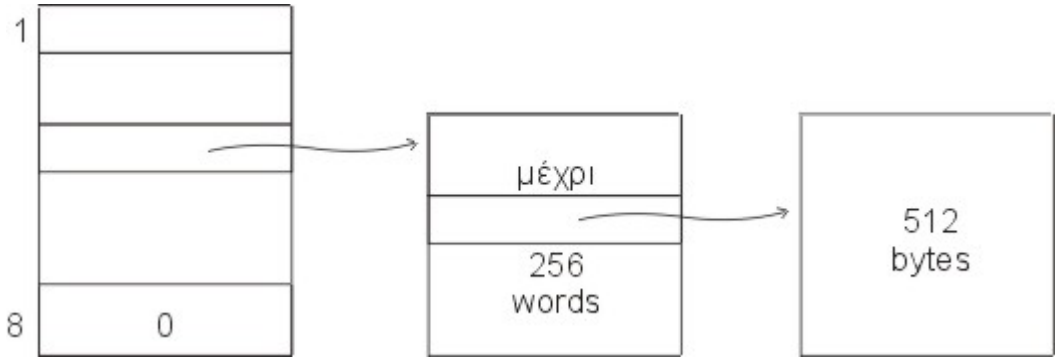
Κατά την απόσταση αρχείου ο i-node επιστρέφεται και τα block των 512bytes που δόθηκαν σ'αυτό επιστρέφοντας στο χώρο FREE LIST.

Προσπέλαση

Εδώ μιλάμε μόνο για απλά αρχεία και Καταλόγους.Η προσπάθεια σε κάποιο block ενός αρχείου καθορίζεται στο σύστημα από το εάν είναι μικρό ή μεγάλο αρχείο.

Μικρά
Η παράμετρος (2) του i-node αποτελείται από 8 words που είναι οι αριθμοί των block.
Άρα όριο = 8 x 512 = 4096 = 4K bytes

Μεγάλα
(α) Το word 8 είναι μηδέν
όριο = 7 x 256 x 512 = 7/8 M bytes



(β) Το word 8 # από το μηδέν άρα
όριο = (7 + 256) 256 x 512 = 34 M bytes

Τα προηγούμενα αφορούσαν απλά αρχεία και καταλόγους.Όταν έχουμε μια εντολή Ε/Ε πάνω σε ένα αρχείο που το i-node δείχνει ότι πρόκειται για ειδικό αρχείο δηλαδή αντιστοιχεί σε κάποιο μηχανισμό (device) τότε η παράμετρος (2) του i-node που αποτελείται από 8 words χρησιμοποιείται ως εξής:

Οι 7 τελευταίες θέσεις αγνοούνται και η πρώτη θέση (το πρώτο word) πέρνεται σαν ένα ζεύγος bytes που συνιστούν το Major device (πρώτο byte) και το minor device (δεύτερο byte).

Το Major device που προσδιορίζει το πρώτο byte ποιός disk drive οδηγός δίσκου που έχει σχέση με συγκεκριμένο controller θα επιλέγει κάποιο interface τηλετύπων.

Το Minor device επιλέγει ποιά ρουτίνα του συστήματος θα αναλάβει να διεκπεραιώσει την εντολή για Ε/Ε είσοδο/έξοδο.

Πως γίνεται το γράψιμο / διάβασμα πάνω σε αρχείο

Στον χρήστη το γράψιμο / διάβασμα πάνω σε αρχείο φαίνεται να γίνεται κατευθείαν δηλαδή μετά από write από χώρο εργασίας του χρήστη στο αρχείο.Ο χώρος εργασίας δεν χρειάζεται πλέον όμως τα πράγματα ,δεν είναι όπως φαίνεται.Το σύστημα χρησιμοποιεί ένα αρκετά πολύπλοκο μηχανισμό από buffers που μεσολαβούν μεταξύ χώρων εργασίας και αρχείου.Στόχος των συστημάτων είναι να περιορισθούν το μέγιστο οι προσπελάσεις σε δίσκους , ώστε οι διάφορες διεργασίες να μην "κολάνε" περιμένοντας για κάποιο disk access.Αυτό βέβαια σε βάρος του χώρου Κύριας Μνήμης που καταλαμβάνουν τα byffers. (σημείωση: Η σχέση χώρου - χρόνου στη μηχανή.)