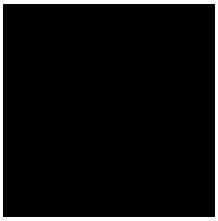


Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Πληροφορικής Και Υπολογιστων

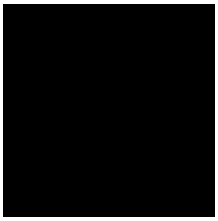
Συστήματα αρχείων στο Linux

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μανώλης Σ. Φανουργάκης

Επιβλέπων : Νεκτάριος Κοζύρης
Επίκουρος καθηγητής ΕΜΠ

Αθήνα, Φεβρουάριος 2004



Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Πληροφορικής Και Υπολογιστών

Συστήματα αρχείων στο Linux

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μανώλης Σ. Φανουργάκης

Επιβλέπων : **Νεκτάριος Κοζύρης**
Επίκουρος Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 20η Φεβρουαρίου 2004

Νεκτάριος Κοζύρης Επίκουρος Καθηγητής ΕΜΠ	Τίμος Σελλής Καθηγητής ΕΜΠ	Παναγιώτης Τσανάκας Καθηγητής ΕΜΠ

Αθήνα, Φεβρουάριος 2004

Μανώλης Σ. Φανουργάκης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Μανώλης Σ. Φανουργάκης 2004
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περιεχόμενα

Εισαγωγή

Σκοπός της παρούσας εργασίας είναι η μελέτη των συστημάτων αρχείων, και το πως αυτά συνεργάζονται με το λειτουργικό σύστημα Linux.

Η **ενότητα 1** ασχολείται με την **περιγραφή του αρχείου και του συστήματος αρχείων**. Δίνεται ο ορισμός τους και αναφέρονται οι ανάγκες που οδήγησαν στη δημιουργία των συστημάτων αρχείων. Εξετάζονται το πως δίνεται στα αρχεία όνομα για να κατηγοριοποιηθεί το περιεχόμενό τους και οι συμβάσεις που ακολουθούνται συχνά στην **ονοματολογία** των αρχείων. Αναφέρονται τα βασικά **χαρακτηριστικά** του κάθε αρχείου τα οποία καταγράφει το σύστημα αρχείων. Αναλύονται οι διάφορες **κατηγορίες** στις οποίες διαφοροποιούνται τα αρχεία λόγω του τρόπου χρήσης τους και το πως γίνεται προσπάθεια με βάση αυτές τις κατηγορίες να γίνεται αποδοτικότερος χειρισμός των αρχείων.

Κατόπιν αναλύονται μερικά βασικά **χαρακτηριστικά του συστήματος αρχείων**. Ένα τέτοιο χαρακτηριστικό είναι η **αξιοπιστία**, το πως δηλαδη το σύστημα αρχείων προσπαθεί να προστατέψει τα πολύτιμα δεδομένα που φυλάσσει ο χρήστης σ΄ αυτό από καταστροφή, και το πως αναπτύσσει τεχνικές για τη διάσωση των αρχείων ακόμα και μετά από πιθανή κατάρρευση του υπολογιστή. Αλλο σημαντικό χαρακτηριστικό είναι η **ασφάλεια**, η εξασφάλιση δηλαδή για το χρήστη ότι τα δεδομένα που έχει αποθηκεύσει στο σύστημα αρχείων δεν κινδυνεύουν από κακόβουλους άλλους χρήστες και από τους κινδύνους που ελλοχεύει το διαδίκτυο.

Τέλος, εξετάζεται τι είναι ο **κατακερματισμός** και πως αυτός μειώνει την απόδοση των συστημάτων αρχείων καθώς και το πως ενεργούν αυτά για να αποτρέψουν την εμφάνισή του. Μια από τις τακτικές που χρησιμοποιούνται γι΄ αυτό είναι η χρήση **extents**. Τέλος, αναφέρεται πως τα συστήματα αρχείων προσπαθούν να αυξήσουν την απόδοσή τους χρησιμοποιώντας την τεχνική του **caching**.

Στην **ενότητα 2** γίνεται μια σύντομη παρουσίαση υλοποιήσεων συστημάτων αρχείων, αναλύεται η δομή και τα χαρακτηριστικά τους και εκθέτονται τα πλεονεκτήματα και τα μειονεκτήματά τους. Το πρώτο σύστημα αρχείων που αναλύεται είναι το **FAT** της Microsoft που είναι ένα από τα πιο απλά και πιο πολύ χρησιμοποιούμενα. Κατόπιν, το **NTFS**, που αναπτύχθηκε για να ξεπεραστούν οι αδυναμίες του FAT και που είναι κατά κάποιο τρόπο ο διάδοχος του. Το NTFS ενσωματώνει πολλά προηγμένα χαρακτηριστικά ώστε να χρησιμοποιείται στο σύγχρονο δικτυακό λειτουργικό των Windows NT.

Ύστερα, περνάμε στο **ext2**, που είναι το κατ΄ εξοχήν σύστημα αρχείων που χρησιμοποιείται στο Linux και σχεδιάστηκε για να προσφέρει μεγάλη απόδοση, σε συνδυασμό με αξιοπιστία και ασφάλεια, καθώς και υποστήριξη πλήθους προηγμένων χαρακτηριστικών. Το **ext3**, από την άλλη είναι μια προέκταση του ext2 που υποστηρίζει την τεχνική journaling, ώστε να εγγυάται ακόμα μεγαλύτερη αξιοπιστία για τα δεδομένα.

Στη συνέχεια εξετάζουμε μερικά πιο προηγμένα συστήματα αρχείων που σχεδιάστηκαν χρησιμοποιώντας προχωρημένες μορφές δομών δεδομένων, όπως τα B-δέντρα. και έχουν σκοπό να προσφέρουν ό,τι αδυνατούν να προσφέρουν τα παραδοσιακά συστήματα αρχείων όπως το ext2: Υποστήριξη και αποδοτικό χειρισμό πολύ μεγάλων μεγεθών αρχείων, συστήματος αρχείων και πλήθους αρχείων. Όλα τα

συστήματα αυτά υποστηρίζουν την τεχνική journaling, αλλά και άλλα πρωτοποριακά χαρακτηριστικά. Το **Reiserfs** σχεδιάστηκε από τον Hans Reiser με τη φιλοσοφία ότι το σύστημα αρχείων πρέπει να προσαρμόζεται στις ανάγκες του χρήστη κι όχι ο χρήστης στα χαρακτηριστικά που του προσφέρει το σύστημα αρχείων. Το Reiserfs παρουσιάζει υψηλή απόδοση στο χειρισμό μεγάλου πλήθους μικρών αρχείων. Το **JFS** αναπτύχθηκε και χρησιμοποιήθηκε από την IBM και στη συνέχεια προσφέρθηκε από την εταιρία στην κοινότητα του ανοιχτό λογισμικού. Το **XFS** πάλι, είναι δημιούργημα της Silicon Graphics.

Κατόπιν, εξετάζεται το δικτυακό πρωτόκολλο του **NFS**, με τη βοήθεια του οποίου μπορούμε να διαμοιράζουμε συστήματα αρχείων σε ένα τοπικό δίκτυο. Το **PVFS**, από την άλλη, είναι ένα εικονικό παράλληλο σύστημα αρχείων που φιλοδοξεί να αυξήσει τις επιδόσεις χρησιμοποιώντας πόρους από πολλούς υπολογιστές ταυτόχρονα, και διαμοιράζοντας το φορτίο ανάμεσά τους.

Τέλος, εξεταζονται δυο συστήματα αρχείων που σχεδιάστηκαν με τελείως διαφορετικό σκοπό και για διαφορετική χρήση από όλα τα προηγούμενα. Το **ISO9660** είναι ένα πρότυπο που χρησιμοποιείται για την τοποθέτηση πληροφοριών μόνο για ανάγνωση στα CDROM. Το **Tmpfs**, από την άλλη, είναι ένα προσωρινό σύστημα αρχείων που συνήθως κάνει χρήση της μνήμης για να προσφέρει ταχύτατη προσωρινή αποθήκευση των δεδομένων.

Στην **ενότητα 3** παρουσιάζεται το **λειτουργικό σύστημα Linux σε σχέση με τα συστήματα αρχείων**. Γίνεται μια σύντομη αναφορά στην ιστορία του και στη συνέχεια αναλύεται η υποστήριξή του για τα συστήματα αρχείων. Η υποστήριξη αυτή γίνεται μέσω του **VFS**, του εικονικού συστήματος αρχείων που χρησιμοποιεί το Linux. Το VFS περιγράφει με γενικό τρόπο τα χαρακτηριστικά όλων των συστημάτων αρχείων. Αναλύεται η δομή του VFS και το πως κωδικοποιεί τις πληροφορίες για κάθε σύστημα αρχείων. Για κάθε σύστημα αρχείων που υποστηρίζει το Linux υπάρχει και μια ξεχωριστή υλοποίηση του VFS. Στη συνέχεια εξετάζεται η διαδικασία με την οποία το VFS καταχωρεί κάθε νέο σύστημα που προσαρτείται στο Linux. Τέλος αναφέρονται μερικά χαρακτηριστικά του VFS που το βοηθούν να αυξήσει την απόδοσή του μέσω της τεχνικής caching, η cache των κόμβων δεικτών, των καταλόγων και η buffer cache.

Στην **ενότητα 4** γίνεται μια μικρή **συγκριτική δοκιμή** δοκιμάζοντας την απόδοση των συστημάτων αρχείων στο Linux σε εργαστηριακό περιβάλλον. Δοκιμάζονται τα ext2, ext3 , Reiserfs, JFS, XFS και FAT. Γίνονται μετρήσεις με τα μετροπρογράμματα Bonnie και postmark και αναλύεται η συμπεριφορά των συστημάτων στις δοκιμές. Στη συνέχεια, εξετάζεται ξεχωριστά το **PVFS**, μετρώντας την απόδοση μια συστοιχίας υπολογιστών για διαφορετικές τοπολογίες. Τέλος, δοκιμάζεται επίσης χωριστά το πρωτόκολλο NFS.

Η **ενότητα 5** παρουσιάζει τη **βιβλιογραφία**.

1. Συστήματα αρχείων

1.1 Ορισμός

Ως απλούστερος ορισμός για την έννοια «Σύστημα αρχείων» μπορεί να δοθεί ο εξής: **Μια δομή δεδομένων που διαχειρίζεται αρχεία**. Πρόκειται λοιπόν για ένα υποσύστημα του λειτουργικού συστήματος, που μας επιτρέπει να χειριζόμαστε τα δεδομένα μας οργανωμένα σε αρχεία.

Το **αρχείο** είναι ένας μηχανισμός αφαίρεσης. Είναι μια αφηρημένη λογική μονάδα αποθήκευσης που προσφέρει έναν απλό, εύκολο και γενικευμένο τρόπο για τη διαχείριση δεδομένων ανεξάρτητα από το φυσικό μέσο αποθήκευσης. Τα αρχεία μεταφράζονται από το λειτουργικό σε περιοχές του αποθηκευτικού μέσου. Έτσι, το λειτουργικό σύστημα είναι σε θέση να χειρίζεται τα δεδομένα σε όλες τις συσκευές εισόδου/εξόδου με τον ίδιο τρόπο. Οι χρήστες και προγραμματιστές του λειτουργικού συστήματος δεν έχουν παρά να χρησιμοποιήσουν τις κλήσεις που εκείνο τους παρέχει για ανάγνωση, εγγραφή, δημιούργια, διαγραφή, αναζήτηση των αρχείων χωρίς να ασχολούνται με τις ιδιαιτερότητες των αποθηκευτικών μέσων.

Για την ευκολία του χρήστη, κάθε αρχείο παίρνει ένα **όνομα**. Έτσι, το σύστημα αρχείων καθιστά πιο εύκολο για τον χρήστη να θυμάται που έχει αποθηκεύσει τα δεδομένα του στο φυσικό μέσο. Το σύστημα αρχείων δομείται ιεραρχικά: Κάθε κατάλογος μπορεί να περιέχει αρχεία και άλλους κατάλογους, εκείνοι με τη σειρά τους νέους υποκατάλογους και ούτω καθ' εξής. Μ΄ αυτό τον τρόπο οι πληροφορίες που περιέχονται στα αρχεία μπορούν να οργανώνονται αποτελεσματικά.

Πρέπει να σημειωθεί ότι συνήθως ανατίθεται στα συστήματα αρχείων η αποθήκευση των δεδομένων για χρονικό διάστημα που υπερβαίνει τη διάρκεια εκτέλεσης των προγραμμάτων ή ακόμα και το χρόνο μεταξύ δυο επανεκκινήσεων του υπολογιστή. Κι αυτό σε αντιδιαστολή με άλλα μέσα αποθήκευσης, όπως τη μνήμη τυχαίας προσπέλασης, τα περιεχόμενα της οποίας χάνονται με τη διακοπή της τροφοδοσίας.

1.2 Χαρακτηριστικά των συστημάτων αρχείων

Η ανάγκη για **δευτερεύοντα μέσα αποθήκευσης** πέρα από τη φυσική μνήμη υπήρξε πάντοτε στους υπολογιστές. Από τις αρχές των δεκαετιών του 1950 και '60 εμφανίζονται τα πρώτα λειτουργικά συστήματα. Τα συστήματα αυτά υποστήριζαν είσοδο και έξοδο μέσω διατρητών καρτών ή μαγνητικών ταινιών. Αργότερα εμφανίστηκαν οι δισκέτες (floppy disks) των 8 και αργότερα των 5,25 και 3,5 ιντσών καθώς και οι σκληροί δίσκοι τύπου Winchester. Σήμερα έχουμε στη διάθεσή μας τους οπτικούς δίσκους CD και DVD που αποθηκεύουν εύκολα, γρήγορα, με μεγάλη αξιοπιστία και με πρακτικά άπειρη διάρκεια ζωής τεράστιες ποσότητες πληροφοριών.

Συχνά, το φυσικό μέσο χωρίζεται σε πολλά διαφορετικά τμήματα, κάθε ένα από τα οποία ονομάζεται **κατάτμηση** (partition). Κάθε κατάτμηση μπορεί να χρησιμοποιηθεί σαν ξεχωριστό μέσο αποθήκευσης και έχουμε τη δυνατότητα να δημιουργήσουμε σε κάθε μια από αυτές ένα ξεχωριστό σύστημα αρχείων. Η τακτική αυτή ακολουθείται συχνά στους σκληρούς δίσκους.

1.2.1 Μπλοκ δεδομένων

Το σύστημα αρχείων χωρίζει το μέσο αποθήκευσης σε τμήματα (**blocks**) που αποτελούν μικρές αυτόνομες μονάδες δεδομένων. Μ' αυτό τον τρόπο μειώνεται σημαντικά η πολυπλοκότητα των διαδικασιών ελέγχου του αποθηκευτικού χώρου εφ' όσον ο αριθμός των μπλοκ είναι πολύ μικρότερος από τον αριθμό των διαθέσιμων χαρακτήρων (bytes). Για παράδειγμα, σε έναν σκληρό δίσκο των 500 MB που είναι χωρισμένος σε μπλοκ των 4 KB, το σύστημα αρχείων έχει την υποχρέωση να διαχειρίζεται τα 125.000 περίπου μπλοκ αντί για τα 500 περίπου εκατομμύρια των συνολικών χαρακτήρων.

Η ελάχιστη δυνατή μονάδα αποθήκευσης σε ένα μέσο ονομάζεται **τομέας** (**sector**). Προφανώς, ένα μπλοκ δεν μπορεί παρά να αποτελείται από έναν ή περισσότερους (ακέραιος αριθμός πάντα) τομείς, συνήθως δε προτιμούνται οι δυνάμεις του 2 (1, 2, 4, 8, κ.ο.κ).

Ένα αρχείο αποτελείται λοιπόν από **ένα ή περισσότερα μπλοκ δεδομένων**.

Πρέπει να τονιστεί ότι η χρήση μπλοκ δεδομένων έχει και τα μειονεκτήματά της. Το τελευταίο μπλοκ σε κάθε αρχείο θα είναι συνήθως μερικώς γεμάτο με δεδομένα. Έτσι, εύκολα βλέπει κανείς ότι για κάθε αρχείο θα υπάρχει σπατάλη αποθηκευτικού χώρου ίσου κατά μέσο όρο με το μισό μέγεθος του μπλοκ. Το πρόβλημα αυτό λέγεται **εσωτερικός κατακερματισμός** (**internal fragmentation**).

Ορισμένα συστήματα αρχείων προσπαθούν να λύσουν το πρόβλημα του εσωτερικού κατακερματισμού χρησιμοποιώντας ένα έξυπνο τέχνασμα: Κατά τη δημιουργία του συστήματος αρχείων ορίζεται μαζί με το μέγεθος του μπλοκ και το μέγεθος του **θραύσματος** (fragment). Το θραύσμα χρησιμοποιείται στη θέση του τελευταίου μπλοκ του αρχείου και ορίζεται με μέγεθος μικρότερο από το μπλοκ. Έτσι, για παράδειγμα, αν έχουμε θραύσμα 8 φορές μικρότερο από το μπλοκ, υποοκταπλασιάζουμε τον εσωτερικό κακακερματισμό. Βέβαια, το χαρακτηριστικό αυτό αυξάνει κατά πολύ την πολυπλοκότητα της διαδικασίας εγγραφής και ανάγνωσης, οπότε είναι και εις βάρος της ταχύτητας.

1.2.2 Ονομασία αρχείων

Το αρχείο προσδιορίζεται από το **όνομά** του, που είναι ένας απλός και εύχρηστος μνημονικός κανόνας για να συγκρατεί ο χρήστης το είδος των πληροφοριών που φυλάσσονται στο αρχείο.

Κάθε αρχείο μπορεί να έχει επίσης μια **επέκταση**. Σε όρισμένα συστήματα αρχείων, η επέκταση αποτελεί ένα προκαθορισμένο τμήμα του αρχείου που διαχωρίζεται δομικά από το όνομα, ενώ σε άλλα όχι.

Όλα τα συστήματα αρχείων έχουν περιορισμούς στην ονομασία των αρχείων, στο μέγεθος και στη χρήση ειδικών χαρακτήρων, άλλα περισσότερους και άλλα λιγότερους. Ακόμα, κάποια συστήματα αρχείων κάνουν διάκριση μεταξύ μικρών και κεφαλαίων γραμματων **(case sensitive),** ενώ άλλα όχι (**case insensitive**).

Αποτελεί κοινή σύμβαση να χωρίζεται η επέκταση με μια τελεία από το όνομα του αρχείου και να δηλώνει τον τύπο του σε αντιδιαστολή με το όνομα, που δηλώνει το περιεχόμενό του. Έτσι, για παράδειγμα, στο σύστημα αρχείων FAT το αρχείο PROG.C συνήθως θα είναι ο κώδικας ενός προγράμματος στη γλώσσα C, ενώ το PROG.EXE είναι η εκτελέσιμη (executable) μορφή του.

Ο διορθωτής κειμένου της γλώσσας C καθώς και ο μεταγλωττιστής της εργάζονται εξ ορισμού με αρχεία με επέκταση .C. Το λειτουργικό σύστημα θεωρεί εξ ορισμού ότι τα αρχεία με κατάληξη .EXE είναι εκτελέσιμα. Έτσι, εφ΄ όσον ακολουθούμε πιστά αυτές τις συμβάσεις, είμαστε με μια ματιά και εμείς και το λογισμικό σε θέση να γνωρίζουμε το περιεχόμενο των αρχείων. Οι επεκτάσεις μπορούν να μας βοηθήσουν ακόμα παραπέρα. Το σύστημα μπορεί να είναι αρκετά έξυπνο ώστε όταν ανοίγουμε ένα συγκεκριμένο αρχείο, να προσπαθήσει να καταλάβει από την επέκτασή του τον τύπο του και να το ανοίξει με το σωστό πρόγραμμα.

Πρέπει, παρ΄ όλα αυτά, να σημειωθεί ότι δεν καθορίζει η επέκταση το περιεχόμενο του αρχείου, απλά συνήθως φροντίζουμε να συμβαίνει το αντίστροφο για διευκόλυνση δική μας και του λογισμικού. Τέλος, δεν εξασφάλιζεται με κάποιο τρόπο από το σύστημα η αντιστοιχία επέκτασης και περιεχομένου. Στο παραπάνω παράδειγμα, κανείς δεν μας εμποδίζει να μετονομάσουμε το αρχείο PROG.C σε PROG.EXE. Αυτό δεν το μετατρέπει με κανέναν τρόπο από πηγαίο κώδικα σε εκτελέσιμο, και αν δοκιμάσουμε να το εκτελέσουμε, δεν θα μπορέσουμε.

1.2.3 Χαρακτηριστικά των αρχείων

Εκτός από το όνομα και την επέκταση, άλλα χαρακτηριστικά των αρχείων που καταγράφονται από τα συστήματα αρχείων είναι:

- **Το μέγεθος** : Δηλώνει από πόσους χαρακτήρες συνολικά αποτελείται το αρχείο. Συνήθως, ο πραγματικό χώρος που καταλαμβάνει ένα αρχείο στο φυσικό μέσο αποθήκευσης είναι λίγο μεγαλύτερος από το μέγεθός του. Αυτό συμβαίνει κυρίως λόγω του εσωτερικού κατακερματισμού, ενώ θα μπορούσαμε με πιο ευρεία θεώρηση να συμπεριλάβουμε στον χώρο που καταλαμβάνει ένα αρχείο και τον χώρο που απαιτείται για τα metadata του, δηλαδή τις πληροφορίες γι΄ αυτό (τα χαρακτηριστικά του για παράδειγμα).
- **Ο ιδιοκτήτης** του αρχείου, όταν μιλάμε για πολυχρηστικά συστήματα, καθώς και η **ομάδα** χρηστών στην οποία ανήκει το αρχείο.
- **Τα δικαιώματα πρόσβασης** στο αρχείο, αν μιλάμε για πολυχρηστικά συστήματα, δηλαδή η

πληροφορία για το ποιοί χρήστες έχουν πρόσβαση στο συγκεκριμένο αρχείο και με ποιο τρόπο.

- **Ο τύπος** του αρχείου, αν πρόκειται δηλαδή για κανονικό αρχείο ή κατάλογο ή κάποιο άλλο είδος που έχει σχέση με το λειτουργικό σύστημα, όπως οι συντομεύσεις (links) ή τα block special files του unix.
- **Η ημερομηνία και ώρα δημιουργίας** ή τελευταίας αλλαγής : Το χαρακτηριστικό αυτό μεταβάλλεται κάθε φορά που κάποιος τροποποιεί το αρχείο και μας δηλώνει πότε έγινε η τελευταία αλλαγή σ΄ αυτό, ή πότε δημιουργήθηκε.
- **Η ημερομηνία και ώρα τελευταίας πρόσβασης** : Το χαρακτηριστικό αυτό μεταβάλλεται κάθε φορά που κάποιος τροποποιεί ή διαβάζει το αρχείο, κάθε φορά δηλαδή που αποκτά πρόσβαση σ΄ αυτό.

Το μέσο μέγεθος των αρχείων είναι ένα χαρακτηριστικό που όσο περνάει ο καιρός αυξάνεται εκθετικά. Χαρακτηριστικό είναι το γεγονός ότι στα τέλη της δεκαετίας του '80, μια εύκαμπτη δισκέτα των 5 και ¼ ιντσών με χωρητικότητα 180 ή 360 KB, και αργότερα οι δισκέτες των 3,5 ιντσών με τα 720 KB ή 1,44 MB τους θεωρούνταν ότι προσέφεραν πολύ αποθηκευτικό χώρο. Το λειτουργικό σύστημα MSDOS, χωρούσε εξ ολοκλήρου σε μία ή δυο τέτοιες δισκέτες! Στα μέσα της δεκαετίας του '90, όταν το CD-ROM άρχισε να διαδίδεται στους προσωπικούς υπολογιστές, τα 650 MB του ήταν ένα τεράστιο νούμερο. Σήμερα (2004), τα 650 MB του εγγράψιμου CD είναι κάτι που θεωρείται σχεδόν τετριμμένο και χρειαζόμαστε συνήθως πάνω από 10 τέτοια δισκάκια για να πάρουμε ένα πλήρες αντίγραφο των αρχείων μας, ενώ τα 4,37 GB του εγγράψιμου DVD θεωρούνται ακόμα σχετικά πολύς χώρος.

Τα δικαιώματα πρόσβασης είναι κάτι που υλοποιείται διαφορετικα από κάθε λειτουργικό σύστημα (ή οικογένεια λειτουργικών συστημάτων) και φυσικά ανάλογα με το αν πρόκειται για πολυχρηστικό σύστημα ή όχι. Είναι ένα πολύ ουσιώδες χαρακτήριστικό, αφ΄ ενός μεν για την προστασία, όσο γίνεται, σημαντικών αρχείων από τυχόντα λάθη του χρήστη (μονοχρηστικά συστήματα) και αφ΄ ετέρου για την ασφάλεια των δεδομένων από μη εξουσιοδοτημένους ή κακόβουλους χρήστες (πολυχρηστικά συστήματα).

Η ημερομηνία τελευταίας αλλαγής είναι ένα εξαιρετικά χρήσιμο χαρακτηριστικό, με τη βοήθεια του οποίου μπορούν να αυτοματοποιηθούν διαδικασίες που έχουν να κάνουν με εξαρτήσεις από πολλά αρχεία. Για παράδειγμα, όταν δημιουργούμε τακτικά εφεδρικά αντίγραφα, αντί να κάνουμε κάθε φορά αντίγραφο για όλα τα αρχεία, είναι πιο έξυπνο να γίνονται αντίγραφα μόνο των αρχείων εκείνων των οποίων η ημερομηνία τελευταίας αλλαγής είναι νεώτερη της τελευταίας λήψης αντιγράφων. Η διαδικασία αυτή λέγεται προσθετικά αντίγραφα (**incremental backup**). Έτσι, χρειαζόμαστε πολύ λιγότερο δευτερεύοντα αποθηκευτικό χώρο, απ΄ ότι αν κάνουμε κάθε φορά πληρη αντίγραφα.

Αντίθετα, η ημερομηνία τελευταίας πρόσβασης δεν έχει την ίδια σπουδαιότητα, καθώς δεν υπάρχουν πολλές χρήσεις της. Επειδή μάλιστα η τροποποίησή της χρειάζεται κάποιο χρόνο να γίνει κάθε φορά που γίνεται πρόσβαση σε κάθε αρχείο, πολλές φορές την απενεργοποιούμε για να επιταχύνουμε το σύστημα αρχείων.

1.2.4 Κατηγορίες αρχείων

Μπορούμε να κατατάξουμε τα αρχεία σε διάφορες κατηγορίες, ανάλογα με τον τρόπο με τον οποίο γίνεται η προσπέλαση σ΄ αυτά.

- **Σειριακά αρχεία** : Είναι τα αρχεία στα οποία η προσπέλαση γίνεται σειριακά, σε μια ανάγνωση για παράδειγμα διαβάζονται όλοι οι χαρακτήρες τους ο ένας μετά τον άλλον, από την αρχή μέχρι το τέλος.
- **Αρχεία τυχαίας προσπέλασης** : Είναι τα αρχεία στα οποία η προσπέλαση γίνεται μη σειριακά: Τοποθετούμαστε (**seek**) πρώτα στο σημείο του αρχείου το οποίο θέλουμε να διαβάσουμε ή να γράψουμε κι ύστερα γράφουμε ή διαβάζουμε κάποιον αριθμό μπλοκ. Κατόπιν, για την επόμενη προσπέλαση, τοποθετούμαστε σε διαφορετική θέση, και ούτω καθ΄ εξής.
- **Σποραδικά αρχεία (sparse files)** : Είναι τα αρχεία αυτά, στα οποία μεγαλο μέρος τους είναι κενό, ενώ δεδομένα υπάρχουν σε μερικά σημεία μόνο. Τα σποραδικά αρχεία παρουσιάζουν γενικά δυσκολίες στο αποτελεσματικό χειρισμό τους από το σύστημα αρχείων.
- **Αρχεία καταγραφής** (logs) : Στα αρχεία αυτά, η πιο συνηθισμένη εργασία που γίνεται είναι η προσθήκη νέων δεδομένων στο τέλος τους (append).

Οι κατηγορίες αυτές προκύπτουν, φυσικά, από τη χρήση και μόνο των αρχείων, και οι χαρακτηρισμοί δεν είναι πάντα απόλυτοι. Εξ΄ άλλου, τα περισσότερα συστήματα αρχείων δεν κάνουν απολύτως καμία διάκριση στα αρχεία. Η μελέτη των παραπάνω γίνεται για να μπορέσουμε να βελτιώσουμε την ταχύτητα χειρισμού κάθε αρχείου ανάλογα με τον τύπο των εργασιών εισόδου/ εξόδου που γίνεται σ΄ αυτό.

Έτσι, στα σειριακά αρχεία είμαστε συνήθως σε θέση να επιτύχουμε την υψηλότερη απόδοση από όλες τις άλλες κατηγορίες. Κι αυτό γιατί γίνεται μια απερίσπαστη συνεχής ανάγνωση μπλοκ από το φυσικό μέσο. Είμαστε σε θέση να χρησιμοποιήσουμε τεχνικές όπως η προανάγνωση (**readahead**) δεδομένων, η ανάγνωση δηλαδή των δεδομένων πριν ακόμα αυτά ζητηθούν και η τοποθέτησή τους στην μνήμη cache για να ανταποκριθεί πολύ πιο γρήγορα το σύστημα όταν αυτά ζητηθούν.

Αντίθετα, στα αρχεία τυχαίας προσπέλασης, η ταχύτητα που επιτυγχάνουμε συνήθως είναι πολύ μικρότερη. Κι αυτό γιατί σε κάθε προσπέλαση διαβάζουμε μικρό αριθμό δεδομένων. Επίσης, η τοποθέτηση (**seek**) σε διαφορετικό σημείο του φυσικού μέσου προσθέτει συνήθως μια μικρή ή μεγάλη καθυστέρηση, ανάλογα με το είδος του φυσικού μέσου. Τα δεδομένα διαβάζονται από το φυσικό μέσο πάντα με τη χρήση χώρων προσωρινής αποθήκευσης (**buffers**), οπότε σε κάθε ανάγνωση μεταφέρεται πάντα ακέραιος αριθμός δεδομένων, πολλαπλάσιο του buffer. Στην περίπτωσή των αρχείων τυχαίας προσπέλασης, συχνά ένα μεγάλο κομμάτι του buffer περιέχει δεδομένα που δε θα χρησιμοποιηθούν.

Τα σποραδικά αρχεία, από την άλλη, είναι λόγω της χρήσης τους ένας πονοκέφαλος για τον κάθε σχεδιαστή συστημάτων αρχείων. Υπάρχει το πρόβλημα του κενού χώρου που δεσμεύεται χωρίς να χρησιμοποιείται. Συχνά το μέγεθος που προκύπτει αν δεσμευτεί χώρος για ολόκληρο το αρχείο είναι πολύ μεγάλο, πράγμα που οδηγεί σε μεγάλη σπατάλη. Επίσης, θα πρέπει να γίνονται συχνά τοποθετήσεις (seeks) σε διαφορετικά σημεία του (πολύ μεγάλου) αρχείου για την προσπέλαση, πράγμα που προσθέτει και μεγάλη

καθυστέρηση. Για την αποφυγή των παραπάνω, υιοθετούνται συχνά πολύπλοκοι αλγόριθμοι που προσπαθούν να δεσμεύουν χώρο μόνο για εκείνα τα τμήματα του αρχείου που περιέχουν δεδομένα.

1.2.5 Ιεραρχικά συστήματα αρχείων

Τα αρχεία ταξινομούνται σε καταλόγους έτσι ώστε κάθε κατάλογος να περιέχει μια λίστα από αρχεία. Με τον τρόπο αυτό είναι εύκολη η ομαδοποίηση των αρχείων που σχετίζονται μεταξύ τους. Τηρουμένων των αναλογιών, θα μπορούσαμε να παρομοιάσουμε κάθε αρχείο με ένα κεφάλαιο του βιβλίου και τον κατάλογο με τον πίνακα περιεχομένων. Ή ακόμα, το καθε αρχείο με ένα έγγραφο που βρίσκεται μέσα σε ένα συρτάρι και τον κατάλογο ως το ευρετήριο των περιεχομένων εγγράφων.

Τα πρώτα συστήματα αρχείων, όπως αυτό του λειτουργικού συστήματος CP/M (δεκαετία 1970), ή του MSDOS στην έκδοση 1.0 (1981) ήταν **μη ιεραρχικά**. Όλα τα αρχεία βρίσκονταν δηλαδή στον ένα και μοναδικό κατάλογο του συστήματος. Αυτό περιόριζε την ευελιξία για τον χρήστη. Όσο αύξανε το πλήθος των αρχείων, τόσο πιο δύσκολη γινόταν η οργάνωσή τους. Επιπλέον, με την ύπαρξη πολλών χρηστών προέκυπταν σοβαρά προβλήματα διαχωρισμού των αρχείων του κάθε χρήστη.

Αντίθετα, στα **ιεραρχικά συστήματα αρχείων**, όπως αυτά που χρησιμοποιεί το λειτουργικό Unix και το MSDOS σε μεταγενέστερες εκδόσεις, έχουμε μια δενδρική δομή καταλόγων. Κάθε κατάλογος μπορεί να έχει και υποκαταλόγους. Ξεκινώντας από το βασικό κατάλογο, τη **ρίζα** (root), επιτρέπονται κατάλογοι και υποκατάλογοι σε οποιοδήποτε βάθος. Έτσι, το σύστημα αρχείων παίρνει μια δεντρική δομή, η οποία ξεκινάει από τη ρίζα, διακλαδίζεται σε πολλούς καταλόγους και υποκαταλόγους και καταλήγει στα αρχεία. Με τη βοήθεια του ιεραρχικού συστήματος αρχείων, ο χρήστης μπορεί να οργανώσει με έναν τρόπο που έχει νόημα όλα τα αρχεία του, καθώς και τα αρχεία του συστήματος.

Έτσι, δημιουργείται και η έννοια του **τρέχοντος κατάλογου**. Ανάλογα με το λειτουργικό σύστημα και την υλοποίησή του, μπορεί να υπάρχει ξεχωριστός τρέχων κατάλογος για κάθε διεργασία, ένας για κάθε σύστημα αρχείων ή μόνος ένας για όλο το λειτουργικό. Αναφορές αρχείων μπορούν να γίνουν και με αναφορά της σχετικής διαδρομής, δηλαδή της θέσης του αρχείου σε σχέση με τον τρέχοντα κατάλογο.

Προφανώς, στα ιεραρχικά συστήματα αρχείων, για να προσδιορίσουμε πλήρως ένα αρχείο χρειαζόμαστε εκτός από το όνομά του και την **απόλυτη διαδρομή** του (absolute path), τη θέση του δηλαδή σε σχέση με τον ριζικό κατάλογο.

Κάθε λειτουργικό σύστημα χρησιμοποιεί κάποιο ειδικό σύμβολο ως διαχωριστή των καταλόγων. Τέτοια σύμβολα είναι τα “/” και “\”.

1.2.6 Αξιοπιστία

Σημαντικό στοιχείο των συστημάτων αρχείων είναι η **αξιοπιστία**. Χρησιμοποιούμε το σύστημα

αρχείων για να αποθηκεύσουμε πολύτιμες πληροφορίες. Σε περίπτωση καταστροφής τους από λάθος του χρήστη, απώλεια τροφοδοσίας, φθορές υλικού ή ακόμα και φυσικές καταστροφές, είναι ιδιαίτερα σημαντικό να μπορούμε να ανακτήσουμε τις πληροφορίες μας, οι οποίες μπορεί να είναι το αποτέλεσμα ανεκτίμητων ωρών εργασίας.

Γι' αυτό το λόγο, τα συστήματα αρχείων σχεδιάζονται με προσοχή ώστε να είναι αφ' ενός μεν **ανθεκτικά στις καταστροφές**, αφετέρου δε τα εργαλεία ανάνηψης να έχουν τη δυνατότητα **να σώζουν όσο πιο πολλές πληροφορίες γίνεται ύστερα από ενδεχόμενη καταστροφή**. Η φύλαξη των πληροφοριών ζωτικής σημασίας για το σύστημα σε πολλαπλά αντίγραφα σε διαφορετικές περιοχές είναι μια από τις τεχνικές που χρησιμοποιούνται.

Πολλές φορές, τμήματα του φυσικού μέσου παθαίνουν βλάβες και δεν είναι πια αξιόπιστα για την αποθήκευση πληροφοριών. Το υπόλοιπο τμήμα όμως του μέσου εξακολουθεί να λειτουργεί σωστά. Πολλά συστήματα αρχείων είναι αρκετά έξυπνα ώστε να μαρκάρουν τα μπλοκ που έχουν πρόβλημα ως κατεστραμμένα μπλοκ (**bad blocks**) και να αποφεύγουν να τα χρησιμοποιούν στο μέλλον. Έτσι δεν είναι άμεσα αναγκαία η αντικατάσταση του φυσικού μέσου.

Βέβαια, παρά την προσεκτική σχεδίαση και τις έξυπνες τεχνικές ανάνηψης των συστημάτων αρχείων, η πείρα δεκαετιών πάνω στους υπολογιστές έχει δείξει ότι μόνο η συχνή τήρηση **εφεδρικών αντιγράφων** σε δευτερεύον μέσο καθιστά το διαχειριστή ενός συστήματος ήσυχο για την προστασία των δεδομένων

1.2.7 Ασφάλεια

Τα συστήματα αρχείων μας βοηθούν επίσης να μοιραστούν δεδομένα και να ανταλλαχθούν πληροφορίες μεταξύ διαφορετικών χρηστών. Σε ένα πολυχρηστικό σύστημα, όι χρήστες μπορούν να επικοινωνούν μεταξύ τους και μέσω του συστήματος αρχείων. Πολύ σημαντική είναι εδώ και η **ασφάλεια** των δεδομένων. Η διασφάλιση δηλαδή ότι κάθε χρήστης θα έχει πρόσβαση μόνο στα δικά του δεδομένα και σ' αυτά των άλλων χρηστών για τα οποία του έχουν επιτρέψει την πρόσβαση.

Η ασφάλεια των δεδομένων έχει σήμερα εξελιχθεί σε ένα πολύ περίπλοκο και ζωτικής σημασίας πρόβλημα. Η εξάπλωση του διαδικτύου και η δυνατότητα σχεδόν κάθε υπολογιστή να συνδεθεί σ' αυτό και να έρθει σε επαφή με οποιονδήποτε άλλο χρήστη του, περιπλέκει ακόμα περισσότερο το πρόβλημα της διασφάλισης των προσωπικών δεδομένων. Η εξάπλωση, τέλος, των διάφορων ηλεκτρονικών **ιών** (viruses) και **σκουληκιών** (worms) που εγκαθίστανται στον υπολογιστή του θύματος και προβαίνούν σε κακόβουλες ενέργειες όπως η παραβίαση του απορρήτου και η μερική ή ακόμα και ολική καταστροφή του υπολογιστή

έχει επιφέρει σε εταιρείας και ιδιώτες τεράστιες χρηματικές και ηθικές ζημιές.

Ο τρόπος που υλοποιείται η ασφάλεια παρουσιάζει βέβαια διαφορές ανάλογα με το σύστημα αρχείων και το λειτουργικό σύστημα. Κοινό σημείο όμως είναι ή προσπάθεια ορισμού δικαιωμάτων πρόσβασης στα αρχεία. Με αυτό τον τρόπο, μπορούν να υπάρξουν περιορισμοί στην πρόσβαση που έχει κάθε χρήστης στα ζωτικά αρχεία του συστήματος ή στα αρχεία των άλλων χρηστών.

Στα πολυχρηστικά λειτουργικά συστήματα των δεκαετιών του 1950 και '60 καθώς και στο μεταγενέστερο Unix (1970 και μετά) δόθηκε μεγάλη σημασία στην ασφάλεια. Εξ' άλλου, η πολυχρηστική φύση τους το απαιτούσε.

Στα λειτουργικά συστήματα που χρησιμοποιήθηκαν στους πρώτους προσωπικούς υπολογιστές, όπως το CP/M και το MSDOS (σύστημα αρχείων FAT) δεν υπήρξε καμία πρόβλεψη ασφαλείας. Κι αυτό γιατί τα συστήματα αυτά προσφέρονταν για οικιακή χρήση και για έναν μόνο χρήστη. Έτσι, το λειτουργικό έδινε στον χρήστη και στα προγράμματα την ελευθερία να έχουν απεριόριστη πρόσβαση στους πόρους του υπολογιστή, ακόμα και στο υλισμικό (hardware) του. Αργότερα, και με την εξάπλωση του διαδικτύου προέκυψαν αρκετά προβλήματα ασφαλείας. Το λειτουργικό σύστημα Windows της Microsoft, στις πολυχρηστικές εκδόσεις του (Windows NT, Windows 2000, Windows XP) ανέπτυξε εκ νέου ένα σύστημα αρχείων (NTFS) το οποίο διαθέτει όλα τα χαρακτηριστικά ασφαλείας που αρμόζουν σε ένα σύγχρονο δικτυακό λειτουργικό.

1.2.8 Κατακερματισμός

Ένα πρόβλημα που συχνά προκύπτει στα συστήματα αρχείων είναι ο εξωτερικός κατακερματισμός (external **fragmentation**) των αρχείων. Κατακερματισμένο αρχείο είναι αυτό του οποίου τα δεδομένα δε βρίσκονται σε συνεχόμενα μπλοκ, αλλά διεσπαρμένα σε διαφορετικές περιοχές του φυσικού μέσου. Για να διαβάσουμε ένα αρχείο θα πρέπει να γίνει προσπέλαση σε όλα τα μπλοκ δεδομένων που αυτό χρησιμοποιεί. Αν τα μπλοκ είναι διασκορπισμένα σε διαφορετικές περιοχές του φυσικού μέσου, θα έχουμε σημαντική καθυστέρηση γιατί πρέπει να προστεθεί ο χρόνος της προσπέλασης αρκετών διαφορετικών περιοχών του φυσικού μέσου. Αντίθετα, αν όλα τα μπλοκ βρίσκονται συγκεντρωμένα στο ίδιο σημείο, η καθυστέρηση αυτή δεν υφίσταται.

Μια καλή τακτική που ακολουθούν πολλά συστήματα αρχείων είναι να προσπαθούν κατα τη δημιουργία του αρχείου να βρουν περιοχές με αρκετά ελεύθερα μπλοκ ώστε να αποθηκευτεί το αρχείο εξ' αρχής χωρίς κατακερματισμό. Κατά τη διάρκεια όμως των επανειλημμένων τροποποιήσεων ενός αρχείου θα χρειαστεί ενδεχομένως κι άλλος ελεύθερος χώρος ο οποίος μπορεί να μην υπάρχει στο τέλος αυτού που αποθηκεύτηκε το αρχείο. Έτσι, μοιραία, καταλήγουμε πάλι στον κατακερματισμό.και μάλιστα τόσο περισσότερο όσο πιο συχνά έχουμε τροποποιήσεις στο αρχείο.

Όσο πιο λίγο ελεύθερο χώρο έχουμε, τόσο πιθανότερο είναι να προκύψει κατακερματισμός των αρχείων. Κι αυτό γιατί τόσο πιθανότερο είναι αφ' ενός να μην μπορεί να αποθηκευτεί ένα αρχείο εξ' αρχής χωρίς κατακερματισμό (να μην υπάρχει δηλαδή συνεχόμενος διαθέσιμος χώρος) και αφ' ετέρου να μην υπάρχει κενός χώρος μετά το τέλος του αρχείου για να αποθηκευτούν τα νέα μπλοκ που θα δεσμευτούν για να κρατήσουν τυχόν τροποποιήσεις του αρχείου. Γι' αυτό το λόγο συνιστάται στους διαχειριστές των λειτουργικών συστημάτων να διατηρούν πάντα καποιο ποσοστό ελεύθερου χώρου (τουλάχιστον 10% του συνόλικού για παράδειγμα) στα συστήματα αρχείων

Για την αντιμετώπιση του κατακερματισμού κυκλοφορούν διάφορα προγράμματα αποκερματισμού **(defragmentation**) τα οποία αναλαμβάνουν να μεταφέρουν τα περιεχόμενα κάθε αρχείου του συστήματος σε συνεχόμενα μπλοκ. Τα προγράμματα αυτά, πολύ συχνά λειτουργούν ακόμα και παράλληλα με τη λειτουργία του υπολογιστή, με το σύστημα αρχείων να χρησιμοποιείται και από άλλα προγράμματα.

1.2.9 Extents

Τα **extents** (εκτάσεις) είναι σύνολα από συνεχόμενα μπλοκ δεδομένων. Χρησιμοποιούνται από πολλά συστήματα αρχείων αλλά και από βάσεις δεδομένων. Ένα extent ορίζεται από τρεις παραμέτρους: το μπλοκ από το οποιο αρχίζει, το μέγεθός του (ακέραιος αριθμός μπλοκ) και τη θέση του πρώτου χαρακτήρα στο αρχείο που καταλαμβάνει το extent.

Τα δεδομένα σε ένα extent θα είναι συνεχόμενα ακριβώς επειδή αυτό αποτελείται από συνεχόμενα μπλοκ. Το γεγονός αυτό οδηγεί σε μεγαλύτερη απόδοση στην ανάγνωση και στην εγγραφή επειδή ελαχιστοποιεί τις αναζητήσεις (seeks) στο φυσικό μέσο. Τέλος, μειώνει το φαινόμενο του κατακερματισμού αφού περισσότερα μπλοκ δεδομένων βρίσκονται κοντά το ένα στο άλλο.

Ένα άλλο ενδιαφέρον σημείο είναι ότι τα extents μας βοηθούν να οργανώσουμε πιο αποτελεσματικά ένα μεγάλο ελεύθερο τμήμα δεδομένων. Εφ΄ όσον ένα extent αποτελείται από πολλά μπλοκ, χρησιμοποιώντας extents θα έχουμε να διαχειριστούμε ακόμα λιγότερες αυτόνομες μονάδες δεδομένων απ΄ ότι αν διαχειριζόμασταν απλά μπλοκ, πράγμα που μειώνει την πολυπλοκότητα του συστήματος αρχείων.

1.2.10 Caching

Η απόδοση ενός συστήματος αρχείων είναι ένα πολύ σημαντικό χαρακτηριστικό του. Ένας τρόπος να αυξηθεί η ταχύτητα ανάγνωσης και εγγραφής στα δευτερεύοντα φυσικά μέσα είναι με ενδιάμεση χρήση της μνήμης. Η διαδικασία αυτή είναι γνωστή ως **caching**.

Η βασική ιδέα είναι να διατηρηθεί στη μνήμη τυχαίας προσπέλασης RAM (ή γενικότερα, *σε ένα πιο γρήγορο φυσικό μέσο*) ένα αντίγραφο από το φυσικό μέσο αποθήκευσης. Όταν ζητηθεί η ανάγνωση από το σύστημα αρχείων ελέγχεται πρώτα αν το ζητούμενο μπλοκ δεδομένων βρίσκεται στην μνήμη cache. Αν ναι, τότε διαβάζεται απ' ευθείας από εκεί. Αν όχι, τότε γίνεται προσπέλαση στο φυσικό μέσο και αφού ικανοποιηθεί η αίτηση το μπλοκ μεταφέρεται και στην μνήμη cache. Όταν η μνήμη cache γεμίσει τότε πρέπει να διαγραφεί ένα μπλοκ από αυτή για να μεταφερθεί ένα νέο.

Για την επιλογή του μπλοκ που θα διαγραφεί για να δημιουργηθεί ελεύθερος χώρος χρησιμοποιούνται διάφοροι αλγόριθμοι. Για παράδειγμα, ο LRU (Last Recently Used) με το σκεπτικό ότι τα μπλοκ που έχουν ζητηθεί πιο πρόσφατα έχουν μεγαλύτερη πιθανότητα να ξαναζητηθούν στο μέλλον επιλέγει το μπλοκ που ο χρόνος της τελευταίας αίτησής του βρίσκεται πιο πίσω από όλα τα άλλα μπλοκ. Άλλος αλγόριθμος βασίζεται στο πόσο συχνά έχει γίνει αίτηση για κάθε μπλοκ και θεωρεί ότι αυτά με τις περισσότερες αιτήσεις έχουν μεγαλύτερη πιθανότητα να ξαναζητηθούν στο μέλλον.

Παρόμοια με την ανάγνωση συμβαίνει και με την **εγγραφή**. Κάθε φορά που γίνεται αίτηση εγγραφής, η εγγραφή γίνεται σε ένα μπλοκ στην μνήμη cache και δε μεταφέρεται στο φυσικό μέσο πριν την διαγραφή του μπλοκ από αυτή. Είναι δυνατόν όμως, για κάποιο λόγο, όπως για παράδειγμα η κατάρρευση του λειτουργικού ή η διακοπή της τροφοδοσίας, να μη μεταφερθεί ποτέ το μπλοκ στο φυσικό μέσο. Στην περίπτωση αυτή μπορεί να καταλήξουμε σε απώλεια των δεδομένων ή και σε καταστροφή του συστήματος αρχείων. Γι' αυτό το λόγο, το χαρακτηριστικό της χρήσης της μνήμης cache για την εγγραφή συχνά δε χρησιμοποιήται, εκτός κι αν υπάρχει αφ΄ ενός μεν απόλυτη εμπιστοσύνη για τη σταθερότητα του λειτουργικού και αφ΄ ετέρου εγγυημένα σταθερή τροφοδοσία μέσω τροφοδοτικού αδιάλειπτης παροχής (UPS).

Δεδομένου ότι η ταχύτητα μεταφοράς δεδομένων από και προς τη μνήμη τυχαίας προσπέλασης είναι από δεκάδες μέχρι και χιλιάδες φορές μεγαλύτερη από την αντίστοιχη ταχύτητα για τα δευτερεύοντα μέσα αποθήκευσης γίνεται φανερό ότι με τη χρήση της τεχνικής caching κερδίζουμε πάρα πολύ σε ταχύτητα.

Σε πολλά μέσα αποθήκευσης, η τεχνική αυτή χρησιμοποιείται κατά κόρον. Για παράδειγμα, οι σύγχρονοι σκληροί δίσκοι έρχονται πάντα εξοπλισμένοι με μνήμη cache ανάγνωσης και εγγραφής, η λειτουργία της οποίας ρυθμίζεται αυτόματα από το ηλεκτρονικό κύκλωμα του σκληρού δίσκου. Η ποσότητα μάλιστα της μνήμης cache που φέρει ένας σκληρός δίσκος αποτελεί ένα από τα σημαντικά χαρακτηριστικά του, και όσο πιο πολλή διαθέτει, τόσο υψηλότερη είναι και η απόδοσή του.

2. Παρουσίαση υλοποιήσεων συστημάτων αρχείων

Στο παρόν τμήμα θα αναφέρουμε μια επιλογή από το πλήθος συστήματων αρχείων που σχεδιάστηκαν για διαφορετικές ανάγκες και με διαφορετικό τρόπο, συχνά δε σε διαφορετικές εποχές.

2.1 Το σύστημα αρχείων FAT (File Allocation Table)

Το σύστημα αρχείων FAT σχεδιάστηκε για το λειτουργικό σύστημα MSDOS και χρησιμοποιήθηκε και αργότερα και στο λειτουργικό των Windows. Το όνομά του το πήρε από τον πίνακα κατανομής αρχείων (File Allocation Table) στον οποίο αποθηκεύει τις καταχωρήσεις των αρχείων και των ελεύθερων blocks.

Πρόκειται για ένα εξαιρετικά απλό σύστημα αρχείων. Δεν διαθέτει χαρακτηριστικά δικαιωμάτων χρήσης, ούτε προστασίας. Το μεγαλύτερο πλεονέκτημά του είναι η συμβατότητα λόγω της εξάπλωσης του MSDOS στους προσωπικούς υπολογιστές. Υποστηρίζεται από πάρα πολλά λειτουργικά συστήματα.

2.1.1 Περιγραφή του συστήματος αρχείων FAT

Υλοποιήθηκε από τη Microsoft ως κύριο σύστημα αρχείων για τα λειτουργικά συστήματα MSDOS και Windows. Χρησιμοποιείται ευρύτατα στις δισκέττες (floppy discs). Υποστηρίζεται σχεδόν από όλα τα υπάρχοντα λειτουργικά συστήματα και γι΄ αυτό χρησιμοποιείται συχνά για το μοίρασμα αρχείων.

Το FAT πρωτοεμφανίστηκε μαζί με την πρώτη έκδοση του MSDOS το 1983. Τότε δεν υποστήριζε κάν υποκαταλόγους, χαρακτηριστικό που προστέθηκε στη δεύτερή του έκδοση.

2.1.2 Δομή του συστήματος αρχείων FAT

Στο FAT η μικρότερη αυτόνομη μονάδα διαχείρισης δεδομένων ονομάζεται **συστάδα** (**cluster**) ή μονάδα αποθήκευσης (**allocation unit**). Κάθε cluster αποτελείται από μία ή περισσότερες φορές την ελάχιστη δυνατή μονάδα αποθήκευσης του φυσικού μέσου.

2.1.2.1 Boot Record

Το boot record, ή boot sector είναι ο πρώτος τομέας του φυσικού μέσου. Η πρόσβαση σ΄αυτό είναι εύκολη, ανεξάρτητα από το αν γνωρίζουμε τον ακριβή τύπο του φυσικού μέσου. Αυτό το κάνει ιδανικό για την αποθήκευση βασικών πληροφοριών για τον τρόπο αποθήκευσης του σύστηματος αρχείων που διαβάζοντάς τις, θα μπορέσουμε να αποκτήσουμε πρόσβαση στο ουσιαστικό κομμάτι του συστήματος αρχείων, τα δεδομένα.

Διαβάζοντας λοιπόν το boot record, μαθαίνουμε τα ακριβή όρια κάθε τμήματος, που, αν και βρίσκονται πάντα με την αυτή σειρά, έχουν διαφορετικά μεγέθη, ανάλογα με το μέγεθος του συστήματος αρχείων και τις παραμέτρους με τις οποίες δημιουργήθηκε.

Τα σπουδαιότερα πεδία του boot record είναι τα εξής:

- **Εντολή άλματος (JMP) για τη ρουτίνα εκκίνησης (boot routine)**: Η εντολή αυτή πραγματοποιεί ένα άλμα στο μέρος που βρίσκεται ο κώδικας εκκίνησης, (κάπου μέσα στο boot record). Η εντολή αυτή είναι απαραίτητη, γιατί η διαδικασία εκκίνησης ενός υπολογιστή δεν είναι τίποτ΄ άλλο από την εκτέλεση των εντολών που βρίσκονται στις πρώτες θέσεις του boot record.
- **Όνομα και αριθμός OEM**: Αυτό το πεδίο χαρακτηρίζει τον κατασκευαστή και του λειτουργικό σύστημα.
- **Χαρακτήρες ανά τομεά (sector)**. Ο τομέας είναι η ελάχιστη μονάδα διαχείρισης δεδομένων του φυσικού μέσου. Για τους σύγχρονους προσωπικούς υπολογιστές, το νούμερο αυτό είναι πάντα 512.
- **Τομείς (sectors) ανά cluster**: Εδώ αποθηκεύεται το πόσοι τομείς περιλαμβάνονται σε ένα cluster, δηλώνεται δηλαδή ουσιαστικά, το μέγεθος του cluster. Δυνατές τιμές είναι οι εξής δυνάμεις του 2: 1,2,4,8,16,32,64 οι οποίες μας δίνουν cluster με μέγεθος 512 bytes, 1, 2, 4, 8, 16 και 32 KB αντίστοιχα.
- **Αριθμός δεσμευμένων τομέων**: Εδώ δηλώνεται ο αριθμός των τομέων στην αρχή του συστήματος αρχείων που είναι δεσμευμένοι. Ο boot sector περιλαμβάνεται στην αρίθμηση και συνήθως είναι ο μόνος, δηλαδή συνήθως το πεδίο αυτό έχει την τιμή 1.
- **Αριθμός πινάκων FAT**: Εδώ δηλώνουμε πόσοι πίνακες FAT υπάρχουν. Επειδή τα περιεχόμενα του πίνακα FAT είναι ζωτικής σημασίας για τα δεδομένα που αποθήκεύονται στο σύστημα αρχείων, φυλάσσεται συνήθως ένα, ή πιο σπάνια, περισσότερα ακριβή αντίγραφα του πίνακα FAT.
- **Αριθμός καταχωρήσεων του ριζικού κατάλογου**: Περιλαμβάνει το πόσες καταχωρήσεις έχει ο ριζικός κατάλογος, που είναι σταθερού μεγέθους. Εξ ου και ο περιορισμός που προκύπτει στον αριθμό των αρχείων και καταλόγων που μπορεί να έχει ο ριζικός κατάλογος.
- **Αριθμός τομέων στο σύστημα αρχείων**: Δίνει τον αριθμό τον τομέων που περιλαμβάνονται στην τρέχουσα κατάτμηση. Το μέγεθος αυτού το πεδίου είναι 32 δυαδικά ψηφία (2 χαρακτήρες) δηλαδή μπορεί να πάρει ως μέγιστη τιμή την 65.535, που για τομέα ίσο με 512 χαρακτήρες, μας δίνει μέγιστο μέγεθος 32 MB. Το όριο αυτό πολύ γρήγορα έγινε ανεπαρκές, γι΄ αυτό και αργότερα προστέθηκε ένα νέο πεδίο στο τέλος του boot record με μεγαλύτερο μέγεθος. Έτσι, όταν το τρέχον πεδίο είναι ίσο με μηδέν, χρησιμοποιείται η τιμή του νέου πεδίου.
- **Περιγραφή μέσου (medium descriptor)**: Το BIOS του υπολογιστή και το λειτουργικό σύστημα χρησιμοποιούν αυτό το πεδίο για να καθορίσουν το είδος (σκληρός δίκος, εύκαμπτη δισκέτα) και τον τύπο (π.χ. γεωμετρία) του φυσικού μέσου.

- **Τομείς ανά πίνακα FAT**: Το πεδίο αυτό δίνει τι μέγεθος του πίνακα FAT σε τομείς.

2.1.2.2. Πίνακας FAT

Ο πίνακας εκχώρησης αρχείων (File Allocation Table, FAT) απ΄ όπου και πήρε το όνομά του το σύστημα αρχείων, είναι μια δομή στην οποία φυλάσσεται από το σύστημα αρχείων η πληροφορία για το που είναι αποθηκευμένο κάθε αρχείο και ειδικότερα ποια clusters περιλαμβάνει.

Ο πίνακας FAT είναι ένας μεγάλος πίνακας από αριθμούς clusters. Σε κάθε καταχώρηση του πίνακα FAT αντιστοιχεί κι ένα cluster. Η αρίθμηση ξεκινάει ουσιαστικά από το 2 και όχι από το 0, γιατί τα πρώτα δυο cluster είναι δεσμευμένα και αποθηκεύεται σ΄ αυτά ένα αντίγραφο του πεδίου περιγραφής μέσου του boot record.

Η τιμή κάθε καταχώρησης του πίνακα FAT μας δείχνει τη χρήση του αντίστοιχου cluster. Υπάρχει συγκεκριμένη τιμή που δηλώνει ότι είναι ελεύθερο, συγκεκριμένες τιμές που δείχνουν ότι είναι δεσμευμένο από το σύστημα, συγκεκριμένη τιμή που δηλώνει ότι το cluster αυτό είναι άχρηστο εξ΄ αιτίας προβλημάτων υλικού (κατεστραμμένο cluster), και συγκεκριμένες τιμές που δηλώνουν ότι το cluster αυτό είναι το τελευταίο ενός αρχείου ή καταλόγου. Αν η τιμή δεν είναι καμιά από όλες τις παραπάνω, τότε το cluster αυτό ανήκει σε ένα αρχείο ή κατάλογο (δεν είναι όμως το τελευταίο του) και η τιμή δείχνει τον αριθμό του **επόμενου cluster** του αρχείου ή καταλόγου.

Κάθε αρχείο ή κατάλογος καταλαμβάνει ένα ή περισσότερα clusters. Ο πίνακας FAT μας βοηθάει να αποκτήσουμε πρόσβαση σ' αυτά.

Γνωρίζοντας το πρώτο cluster κάθε αρχείου ή καταλόγου (την πληροφορία αυτή την παίρνουμε από τον κατάλογο στον οποίο ανήκει), και πηγαίνοντας στην αντίστοιχη εγγραφή στον πίνακα FAT, βρίσκουμε το δεύτερο cluster. Στη συνέχεια, πηγαίνοντας στην εγγραφή που αντιστοιχεί στο δεύτερο βρίσκουμε το επόμενο κ.ο.κ. Σταματάμε όταν συναντήσουμε την ειδική τιμή που δηλώνει ότι φτάσαμε στο τελευταίο cluster. Δηλαδή, τα cluster του αρχείου σχηματίζουν μια συνδεδεμένη λίστα που είναι αποθηκευμένη στον πίνακα FAT.

Ο αριθμός των δυαδικών ψηφίων που χρησιμοποιείται για κάθε εγγραφή του πίνακα αποτελεί το μέγεθος του FAT (FAT size). Συνηθισμένα μέγεθη είναι 12, 16 και 32 δυαδικά ψηφία. Όπως είναι επόμενο, αυτά τα μεγέθη μας δίνουν τη δυνατότητα να περιγράψουμε αντίστοιχα το πολύ μέχρι 2^{12}, 2^{16} και 2^{32} clusters, δηλαδή 4096, 65.536 και περίπου 4 εκατομμύρια αντίστοιχα.

2.1.2.3 Ριζικός κατάλογος

Ο ριζικός κατάλογος, είναι ένας πίνακας , προκαθορισμένου μεγέθους όπως είπαμε, που περιέχει εγγραφές οι οποίες μπορούν να αναπαραστήσουν αρχεία ή καταλόγους κάτω από τον ριζικό κατάλογο.

Κάθε εγγραφή του ριζικού κατάλογου (και κάθε καταλόγου) έχει μήκος 32 χαρακτήρες και περιέχει:

- Το **όνομα** του αρχείου ή καταλόγου που αποτελείται από 8 χαρακτήρες
- Την **επέκταση** που αποτελείται από 3 χαρακτήρες. Σημειώνεται ότι και οι κατάλογοι μπορούν να έχουν επέκταση, που συνήθως όμως δεν χρησιμοποιείται στην πράξη.
- Τα **χαρακτηριστικά** του αρχείου ή καταλόγου που είναι ένας χαρακτήρας, τα 8 δυαδικά ψηφία του οποίου δηλώνουν την έλλειψη συγκεκριμένων χαρακτηριστικών. Τα χαρακτηριστικά αυτά

αναλύονται διεξοδικότερα στη συνέχεια.

- Ένα πεδίο 10 χαρακτήρων που είναι **δεσμευμένο** από το σύστημα.
- Ημερομηνία και **ώρα της δημιουργίας ή τελευταίας τροποποίησης**. Χρησιμοποιούνται 2 χαρακτήρες, δηλαδή 16 δυαδικά ψηφία για την ημερομηνία κι άλλα τόσα για την ώρα. Συγκεκριμένα, από τα 16 πρώτα, αφιερώνονται τα 7 σημαντικότερα στο έτος (+0 ως +127 από το έτος βάσης 1980), τα 4 επόμενα για το μήνα, και τα 5 υπόλοιπα για την ημέρα. Από τα 16 ψηφία που χρησιμοποιούνται για την ώρα, αφιερώνονται τα 5 σημαντικότερα στις ώρες, τα 6 επόμενα στα λεπτά και τα 5 τελευταία στα δευτερόλεπτα. Επειδή τα 5 ψηφία των δευτερολέπτων δίνουν μόνο 2^5=32 συνδυασμούς, καθένας από αυτούς αντιστοιχεί σε χρονική αύξηση δύο δευτερολέπτων, και έχουμε συνολικά 30 τέτοιες αυξήσεις στο λεπτό.
- Το πρώτο cluster του αρχείου ή του κατάλογου.
- Προκειμένου για αρχείο, το μέγεθός του. Το πεδίο αυτό έχει μήκος 4 χαρακτήρες.

Κάθε κατάλογος περιέχει πάντα εξ΄ ορισμού δυο δεσμευμένα αρχεία: Ένα που δείχνει στον εαυτό του και ένα που δείχνει στο γονικό κατάλογο (σ΄αυτόν δηλαδή που περιέχει τον τρέχοντα). Στον ριζικό κατάλογο το δεύτερο αρχείο δεν δείχνει πουθενά. Τα αρχεία αυτά συμβολίζονται με την τελεία και τη διπλή τελεία αντίστοιχα. Μ΄ αυτό τον τρόπο καθίσταται πιο εύκολη η μετακίνηση μέσα στους καταλόγους, χωρίς να χρειάζεται να διατρέχουμε από την αρχή, από τον ριζικό κατάλογο δηλαδή, το σύστημα αρχείων για να εντοπίσουμε τον κατάλογο στον οποίο ανήκει ο τρέχων.

Οι κενές εγγραφές ενός καταλόγου, (αυτές που δεν έχουν χρησιμοποιηθεί ακόμα), έχουν όλα τα πεδία τους μηδενισμένα. Στα αρχεία που σβήστηκαν, απελευθερώνονται όλα τα δεσμευμένα clusters χωρίς να πειράζονται τα δεδομένα τους και τοποθετείται στον κατάλογο, στη θέση του αρχείου, στην αρχή του ονόματος ο ειδικός χαρακτήρας “?”. Πολλά προγράμματα προσπαθούν να αποκαταστήσουν τα από λάθος διαγραμμένα αρχεία ψάχνοντας για αυτές τις εγγραφές και ελπίζοντας ότι εν τω μεταξύ τα clusters τους δεν ξαναχρησιμοποιήθηκαν.

Τα 8 ψηφία των χαρακτηριστικών είναι τα παρακάτω:

- Τα δυο πρώτα ψηφία είναι **δεσμευμένα** από το σύστημα.
- Ψηφίο **αρχειοθέτησης:** Ο λόγος για τον οποίο σχεδιάστηκε αυτό το ψηφίο είναι να βοηθήσει στο να παίρνονται αντίγραφα των αρχείων σε μονιμότερο μέσο. Η λογική με την οποία δημιουργήθηκε είναι η εξής: Κατά τη δημιουργία του αρχείου, καθώς και κατά την δημιουργία αντιγράφου ασφαλείας του το ψηφίο τίθεται ίσο με το μηδέν. Με την τροποποίηση του αρχείου, το ψηφίο τίθεται ίσο με ένα (ή τουλάχιστον υποτίθεται πως το πρόγραμμα που τροποποιεί το φροντίζει αυτό, καθώς το FAT δεν το κάνει). Έτσι, την επόμενη φορά που θα πρέπει να παρθούν αντίγραφα ασφαλείας, θα γνωρίζουμε ότι το αρχείο πρέπει να συμπεριληφθεί σ΄ αυτά. Στην πράξη, ο μηχανισμός αυτός δεν χρησιμοποιείται, γιατί δεν είναι καθ΄ όλα αξιόπιστος. Πέρα από το ότι το πρόγραμμα αρχειοθέτησης και το πρόγραμμα (ή τα προγράμματα) που τροποποιούν τα αρχεία πρέπει να διατηρούν σωστή την κατάσταση των ψηφίων αρχειοθέτησης, οποιοσδήποτε μπορεί να τα αλλάξει χωρίς περιορισμό.
- **Κατάλογος**: Το ψηφίο αυτό καθορίζει αν η τρέχουσα εγγραφή αφορά αρχείο ή κατάλογο.
- **Ετικέτα δίσκου**: Όταν το ψηφίο αυτό είναι ενεργοποιημένο, τότε στην τρέχουσα εγγραφή

αποθηκεύεται η ετικέτα του συστήματος αρχείων. Για το σκοπό αυτό χρησιμοποιούνται οι 11 χαρακτήρες του ονόματος και της επέκτασης. Προφανώς επιτρέπεται μόνο μια ετίκετα για κάθε σύστημα αρχείων, η οποία μπορεί να γραφτεί κατά τη δημιουργία του ή και αργότερα με τη βοήθεια σχετικών εντολών και προγραμμάτων που παρέχονται.

- **Συστήματος**: Το ψηφίο αυτό καθορίζει αν το τρέχον cluster ανήκει σε αρχείο ή κατάλογο του συστήματος. Τα αρχεία ή οι κατάλογοι του συστήματος περιέχουν ζωτικά αρχεία για το λειτουργικό, όπως αρχεία που χρησιμοποιούνται στην εκκίνηση, ή που αποθηκεύουν πληροφορίες για το λειτουργικό. Τα αρχεία αυτά αποκρύπτονται σε πρώτη φάση από τον χρήστη. Αυτό γίνεται για να προστατευθεί το σύστημα από τυχόν λάθη του χρήστη (π.χ. τη διαγραφή αρχείων που μπορεί να οδηγήσει σε αδυναμία σωστής εκκίνησης του συστήματος την επόμενη φορά).
- **Κρυφό**: Τα κρυφά αρχεία, όπως και τα αρχεία συστήματος αποκρύπτονται από το χρήστη. Η διαφορά τους είναι ότι δεν έχουν να κάνουν με το λειτουργικό και την εκκίνηση ή τις ρυθμίσεις του. Ένα παράδειγμα κρυφού αρχείου είναι ένα αρχείο που τοποθετεί το πρόγραμμα Windows Explorer (φυλλομετρητής του συστήματος αρχείων) σε κάθε κατάλογο και αποθηκεύει τις ρυθμίσεις με τις οποίες ο χρήστης θέλει να βλέπει τον κατάλογο, όπως με ποια σειρά θα είναι ταξινομημένα τα αρχεία.
- **Μόνο ανάγνωση**: Τα αρχεία ή κατάλογοι που έχουν ενεργοποιημένο αυτό το ψηφίο δεν μπορούν να τροποποιηθούν άμεσα από το χρήστη. Καθώς όμως στο FAT δεν υπάρχουν δικαιώματα πρόσβασης, ο χρήστης μπορεί πρώτα να μηδενίσει αυτό το ψηφίο και ύστερα να κάνει τροποποίηση. Η χρησιμότητα λοιπόν του ψηφίου είναι απλώς και μόνο για να αποφεύγονται επιπόλαιες και από λάθος τροποποιήσεις. Πολλά προγράμματα αρνούνται να τροποποιήσουν τα αρχεία που είναι σημειωμένα ως μόνο για ανάγνωση, ενώ άλλα τείνουν να ζητούν πρώτα επιβεβαίωση από το χρήστη. Κάποια όμως δεν λαμβάνουν καν υπ΄ όψη τους το χαρακτηριστικό.

2.1.2.4 Δεδομένα

Στην περιοχή δεδομένων αποθηκεύονται τα δεδομένα των αρχείων και οι εγγραφές όλων των καταλόγων εκτός από τον ριζικό. Η περιοχή αυτή καταλαμβάνει το μεγαλύτερο μέρος της κατάτμησης.

Πρόκειται απλά για ένα πίνακα από clusters, σε πλήρη αντιστοιχία με τον πίνακα FAT.

2.1.3 Χαρακτηριστικά του συστήματος αρχείων FAT

2.1.3.1 Ονομασία αρχείων

Το FAT δεν κάνει διάκριση μεταξύ κεφαλαίων και πεζών χαρακτήρων στα ονόματα των αρχείων, είναι δηλαδή **case insensitive**.

Κάθε αρχείο στο FAT έχει ένα όνομα με μέγιστο μήκος 8 χαρακτήρες και μια επέκταση αντίστοιχα με 3. Ο περιορισμός αυτός υπάρχει λόγω του προκαθορισμένου μεγέθους κάθε εγγραφής στον κάθε κατάλογο.

Κάποιοι ειδικοί χαρακτήρες, οι $<>*?."/\[]:;,= δεν επιτρέπεται να εμφανίζονται μέσα στα ονόματα αρχείων. Τέλος, ως ονόματα αρχείων δεν μπορούν να χρησιμοποιηθούν δεσμευμένα ονόματα συσκευών του MSDOS, όπως con, prn και άλλα.

Μέχρι και πριν την έκδοση των Windows 95, το FAT βρισκόταν υπό τον περιορισμό του 8+3 στα ονόματα των αρχείων, πράγμα που καθιστούσε δύσκολη τη ζωή των χρηστών. Οι τελευταίοι ήταν αναγκασμένοι να εφευρίσκουν κρυπτογραφικά ονόματα για τα αρχεία τους, αρκετά μικρά ώστε να ικανοποιούν τον παραπάνω περιορισμό. Όλα αυτά έκαναν φανερή την ανάγκη για υποστήριξη μεγαλύτερων ονομάτων αρχείων. Το μεγάλο πρόβλημα όμως ήταν η συμβατότητα με παλιότερες εκδόσεις του FAT που ήταν εγκατεστημένες στο λειτουργικό MSDOS.

Για να μπορέσει να υποστηρίξει μεγαλύτερα ονόματα αρχείων το FAT, η Microsoft από την πρώτη έκδοση των Windows 95 και μετά, κατέφυγε σε ένα αμφιλεγόμενο σχεδιαστικό τέχνασμα. Με το σύστημα αυτό, κάθε αρχείο μπορεί να έχει όνομα με λιγότερους από τους συνήθεις περιορισμούς στους ειδικούς χαρακτήρες, και με μήκος μέχρι και 255 χαρακτήρες: Κάθε αρχείο το οποίο ο χρήστης έχει ονομάσει με μεγάλο όνομα παίρνει στην καταχώρηση του καταλόγου ένα σύντομο όνομα-ψευδώνυμο που ικανοποιεί τον περιορισμό 8+3. Για παράδειγμα ο κατάλογος “Program files” ονομάζεται “PROGRA~1”, προσέχοντας βέβαια να έχει κάθε αρχείο ένα μοναδικό όνομα-ψευδώνυμο.

Το μεγάλο ονόμα φυλάσσεται μαζί με την αντιστοιχία του με το ψευδώνυμο μέσα σε ειδικές καταχωρήσεις στον καταλόγο στον οποίο ανήκει το αρχείο. Οι καταχωρήσεις μοιάζουν με καταχωρήσεις αρχείων, ο συνδυασμός όμως των χαρακτηριστικών τους είναι αδύνατος για κανονικά αρχεία. Οι καταχωρήσεις αυτές δηλαδή, έχουν και τα 4 τελευταία ψηφία των χαρακτηριστικών (ετικέτα δίσκου, συστήματος, κρυφό, μόνο για ανάγνωση) ίσα με 1. Έτσι, τα αρχεία αυτά αγνοούνται από παλιότερο λογισμικό και δεν αποτελούν ενόχληση. Το λογισμικό όμως που σχεδιάζεται για να υποστηρίξει τα μεγάλα ονόματα διαβάζει και καταλαβαίνει τα αρχεια αυτά και έτσι ανασυνθέτει και παρουσιάζει στον χρήστη τα μεγάλα ονόματα. Καθώς κάθε καταχώρηση έχει μήκος 32 χαρακτήρων, μπορεί να χρειαστούν πολλές τέτοιες για να φυλαχτεί ένα μεγάλο όνομα. Ο τρόπος αυτός έχει το μειονέκτημα να αυξάνει πολύ το μέγεθος των καταλόγων.

Σε γενικές γραμμές, αυτό το τρικ δουλεύει, καθώς τα περισσότερα προγράμματα ξεγελιούνται. Χρειάζεται προσοχή όμως όταν ξανασώζουμε αρχεία με άλλο όνομα, χρησιμοποιώντας παλιά προγράμματα που δεν υποστηρίζουν τα μεγάλα ονόματα, γιατί μπορεί να προκύψουν προβλήματα. Επίσης, αν χρησιμοποιηθεί παλιότερο λογισμικό δημιουργίας εφεδρικών αντιγράφων, μπορεί κατά την επαναφορά να χαθούν τα μεγάλα ονόματα.

Ο ριζικός κατάλογος έχει ένα μικρό όριο στον αριθμό αρχείων και καταλόγων που μπορεί να δεχτεί. Αυτό προκύπτει λόγω του προκαθορισμένου μεγέθους του πίνακά του, όπως αυτός δηλώνεται στο boot record και υλοποιείται σε ξεχωριστό τμήμα από την περιοχή αποθήκευσης των δεδομένων. Συνιστάται να μην χρησιμοποιούνται μεγάλα ονόματα στον ριζικό κατάλογο, ώστε να γίνεται οικονομία στις περιορισμένες του καταχωρήσεις.

2.1.3.2 Μέγιστο μέγεθος συστήματος αρχείων. FAT32

Το FAT επιτρέπει μέγεθος cluster ίσο με 512 bytes, 1, 2, 4, 8, 16 ή 32 KB. Εδώ ισχύει ό,τι γενικά για το μέγεθος του μπλοκ. Μικρό μέγεθος δηλαδή σημαίνει ότι μειώνεται ο εσωτερικός κατακερματισμός, δηλαδή δεν χάνεται χώρος για το τελευταίο μισογεμάτο cluster κάθε αρχείου, ενώ μεγάλο μέγεθος μπλοκ σημαίνει καλύτερη απόδοση.

Λόγω της δομής του όμως, το FAT έχει ένα όριο στο μέγιστο αριθμό clusters που μπορεί να υποστηρίξει. Το όριο αυτό είναι το 2 υψωμένο σε δύναμη ίση με το μέγεθος του πίνακα FAT. Για παράδειγμα, με μέγεθος πίνακα FAT ίσο με 16, δεν μπορούν να υποστηριχτούν πάνω από 2^{16} = 65.536 clusters. Δεδομένου και του περιορισμού στο μέγεθος των clusters, έχουμε ανώτατο όριο μεγέθους του συστήματος αρχείων.

Το FAT, μέχρι και την πρώτη έκδοση των Windows 95, υποστήριζε μέγιστο μεγέθος πίνακα FAT ίσο με 16. Αυτό σήμαινε ότι αν είχε κανείς ένα σχετικά μεγάλο σκληρό δίσκο, έπρεπε ή να τον χώριζε σε πολλές κατατμήσεις ή να χρησιμοπούσε μεγάλο μέγεθος cluster. Για παράδειγμα, για ένα δίσκο 700 MB έπρπε να χρησιμοποιηθεί μέγεθος cluster ίσο με 16 KB, αφού με μέγεθος 8 KB, μπορούσε κανείς να φτάσει μόνο μέχρι τα 512 MB (=65.536 clusters * 8 KB).

Αυτό όμως οδηγούσε σε πολύ μεγάλη σπατάλη χώρου. Και να προστεθεί εδώ ότι η δομή του λειτουργικού συστήματος Windows χρησιμοποιεί σε πολλές περιπτώσεις πολλά και μικρά αρχεία, συντομεύσεις για παράδειγμα. Τα πράγματα έγιναν ακόμα πιο δύσκολα όταν κυκλοφόρησαν σκληροί δίσκοι με μέγεθος πάνω από 2 GB (=65.536 clusters * 32 KB), οπότε η μόνη διέξοδος ήταν ο χωρισμός του δίσκου σε πολλές κατατμήσεις. Αυτό οδήγησε την Microsoft να ενσωματώσει στην επόμενη έκδοση των Windows, την 95 OSR 2, μια καινούρια έκδοση του FAT που υποστήριζε πίνακα FAT με μέγεθος 32 δυαδικών ψηφιών, πράγμα που έλυσε το πρόβλημα. Η έκδοση αυτή έγινε γνωστή ως FAT32 και κυκλοφορεί μέχρι και σήμερα.

2.1.4 Σχόλια

Το FAT, όπως είδαμε, χαρακτηρίζεται από μεγάλη απλότητα.

Η απλότητα του FAT έχει θετικά και αρνητικά. Όσο πιο απλό είναι, τόσο πιο εύκολα υλοποιείται υποστήριξη γι΄ αυτό σε όλα τα λειτουργικά συστήματα. Γι΄ αυτό το λόγο το FAT έχει υποστήριξη σχεδόν παντού και χρησιμοποιείται για ανταλλαγή αρχείων, καθώς και στις εύκαμπτες δισκέττες.

Λόγω της δομής του συστήματος αρχείων, για να γίνει ανάγνωση από αρχεία που καταλαμβάνουν μόνο ένα cluster, δεν απαιτείται πρόσβαση στον πίνακα FAT, πράγμα που σημαίνει ελαφρώς μεγαλύτερη ταχύτητα πρόσβασης σε τέτοια αρχεία.

Στα αρνητικά μπορούμε να αναφέρουμε τους περιορισμούς στο πλήθος περιεχομένων του ριζικού καταλόγου και στην ονομασία των αρχείων, την πλήρη έλλειψη δικαιωμάτων πρόσβασης, τα συνεχή μπαλώματα που έγιναν σ΄αυτό για να το καταστήσουν ικανό να ανταπεξέλθει στις συνεχώς αυξανόμενες ανάγκες των χρηστών, όπως την προσθήκη μεγάλων ονομάτων, την αύξηση του μέγεθους του πίνακα FAT

από 16 σε 32 και άλλα.

Ένα μεγάλο πρόβλημα στην αξιοπιστία του FAT είναι η ζωτικότητα της σημασίας του πίνακα FAT. Αν αυτός καταστραφεί, τότε όλα τα δεδομένα του συστήματος αρχείων είναι πρακτικά χαμένα. Γι΄ αυτό άλλωστε υπάρχουν πολλαπλά αντίγραφά του. Το γεγονός όμως ότι βρίσκονται το ένα σε συνέχεια του άλλου, δηλαδή στην ίδια περιοχή του φυσικού μέσου, καθιστά κάπως αμφίβολη τη χρησιμότητα των αντιγράφων, αφού σε περίπτωση που καταστραφεί η περιοχή που βρίσκεται το πρωτότυπο (σφάλμα υλικού), τίποτα δεν μας εγγυάται ότι η καταστροφή δε θα προχωρήσει και στα επόμενα αντίγραφα που βρίκονται αμέσως μετά. Επιπλέον, όλα τα αντίγραφα ενημερώνονται από το πρωτότυπο σε τακτά χρονικά διαστήματα με αυτόματη λειτουργία του συστήματος, οπότε τίποτα δε μας εγγυάται ότι τυχόν ασυνέπειες ή λάθη δε θα αντικατοπτριστούν και σ΄ όλα τα αντίγραφα.

Επίσης, λόγω της δομής του πίνακα FAT, ακόμα και η αλλοίωση ελάχιστων τμημάτων του είναι δυνατόν να δημιουργήσει μεγάλη ζημιά στα δεδομένα. Για παράδειγμα, ας υποθέσουμε ότι αλλοιώνεται ένας δείκτης επόμενου cluster που βρίσκεται στη μέση ενός αρχείου. Ακόμα και ένα δυαδικό ψηφίο να αλλάξει σ΄ αυτόν το δείκτη, τα περιεχόμενα του αρχείου δε θα είναι προσπελάσιμα από εκείνο το σημείο και μετά. Αντί γι΄ αυτά, θα εμφανίζονται σκουπίδια ή τμήματα άλλων αρχείων στη θέση τους.

Στην τελευταία έκδοση του FAT προστέθηκαν μερικά χαρακτηριστικά που βελτιώνουν κάπως την ασφάλεια που παρέχουν τα αντίγραφα του πίνακα FAT. Δόθηκε η δυνατότητα σε κάθε αντίγραφο να μπορεί να χρησιμοποιηθεί στη θέση του πίνακα FAT, καθώς και η δυνατότητα να απενεργοποιηθεί η αυτόματη διαδικασία που πολύ συχνά μεταφέρει το πρωτότυπο στα αντίγραφα.

Ομοίως, υπάρχει πρόβλημα στην αξιοπιστία από τη ζωτικότητα της σημασίας του boot record. Πολλοί ιοί υπολογιστών στοχεύουν ακριβώς σ΄ αυτό και το καταστρέφουν γράφοντας σκουπίδια πάνω του, καθιστώντας αδύνατη την εκκίνηση του υπολογιστή.

Το FAT χρησιμοποιεί αυτό που ονομάζουμε **linked-list allocation**, δηλαδή δέσμευση συνδεδεμένης λίστας.για την αποθήκευση των δεδομένων στον πίνακα FAT. Δηλαδή, για να διαβάσουμε ένα αρχείο, πρέπει να διατρέξουμε μέσω του πίνακα FAT όλη τη λίστα των clusters που αυτό χρησιμοποιεί. Το ίδιο χρειάζεται να γίνει αν απλώς θέλουμε να προσθέσουμε δεδομένα στο τέλος του, ενώ αν θέλουμε να διαβάσουμε ένα κομμάτι στη μέση του (τυχαία προσπέλαση), πρέπει να διασχίσουμε όλη τη λίστα μέχρι εκείνο το σημείο. Τέλος, για να παρουσιάσουμε τα περιεχόμενα του καταλόγου στον χρήστη θα πρέπει να τα ταξινομήσουμε πρώτα.

Σε περίπτωση που υπάρχει κατακερματισμός, και να σημειωθεί ότι το FAT δεν κάνει τίποτα για να τον ελέγξει και να τον αποτρέψει, τότε είναι πολύ πιθανό να χρειαστούν αλλεπάλληλες προσπελάσεις σε διαφορετικά τμήματα του φυσικού μέσου για να διαβαστεί το αρχείο, πράγμα που μειώνει πάρα πολύ την ταχύτητα ανάγνωσης. Γι΄ αυτό και συνιστάται σε όσους χρησιμοποιούν FAT να τρέχουν τακτικά προγράμματα αποκερματισμού για την επιτάχυνση του συστήματος αρχείων τους. Στο πρόβλημα του κατακερματισμού πρέπει να προστεθουν οι πολύ συχνές προσπελάσεις στον πίνακα FAT, που βρίσκεται στην αρχή του φυσικού μέσου, σε ξεχωριστή περιοχή από τα δεδομένα.

Επίσης, όπως είπαμε, ο κάθε κατάλογος στο FAT είναι μια γραμμική λίστα από εγγραφές σταθερού μήκους 32 χαρακτήρων που καταλαμβάνει όσα clusters χρειάζονται για να χωρέσουν όλα τα περιεχόμενά του. Το γεγονός αυτό καθιστά αρκετά αργό το FAT στη διαχείριση καταλόγων με πολλά περιεχόμενα. Για να προσπελάσουμε ένα αρχείο πρέπει να διαβάσουμε όλο τον κατάλογο μέχρι το σημείο που βρίσκεται το

αρχείο. Η αναζήτηση αρχείων γίνεται γραμμικά στον κατάλογο. Επίσης, για τη δημιουργία ενός νέου αρχείου πρέπει να ελεγχθεί όλος ο κατάλογος για τυχόν ήδη ύπαρξή του. Το πρόβλημα του κατακερματισμού εντείνει τα μειονεκτήματα αυτά.

Για το μέγεθος του κάθε αρχείου στο FAT χρησιμοποείται το αντίστοιχο πεδίο στον κατάλογο που περιέχει το αρχείο. Το πεδίο αυτό είναι μήκους 4 χαρακτήρων ή 32 δυαδικών ψηφιών, επομένως μπορεί να κρατήσει αριθμούς μέχρι 2^{32}=4 GB. Αυτό είναι και το μέγιστο μέγεθος αρχείου στο FAT, το οποίο με τα σημερινά δεδομένα είναι πολλές φορές ανεπαρκές.

2.2 Το σύστημα αρχείων NTFS (NT File System)

Το σύστημα αρχείων NTFS σχεδιάστηκε για το λειτουργικό σύστημα Windows NT (New Technology) της Microsoft (1993) και χρησιμοποιείται σε όλα τα λειτουργικά της εταιρίας που έχουν βάση τα Windows NT, δηλαδή και στα Windows 2000 και XP.

Η Microsoft έχει κυκλοφορήσει πολλές εκδόσεις του NTFS: Τα Windows NT 3.51 και 4 έρχονταν με την έκδοση 1.2. Τα 2000 με την 3.0 και τα XP με την 3.1. Πολλές φορές οι εκδόσεις 3.0 και 3.1 αναφέρονται και ως 5.0 και 5.1 αντίστοιχα, από τον αριθμό της έκδοσης των αντίστοιχων Windows με τα οποία συνοδεύονται. (Τα 2000 θεωρούνται και ως NT 5.0, ενώ τα XP ως 5.1).

Σε μια προσπάθεια δημιουργίας ενός σύγχρονου πολυχρηστικού και δικτυακού λειτουργικού συστήματος και θέλοντας να ξεφύγει από τις αδυναμίες του FAT, η Microsoft σχεδίασε εκ νέου και ενσωμάτωσε πολλά προηγμένα χαρακτηριστικά στο NTFS, όπως ύπαρξη δικαιώματων πρόσβασης και ορίων χρήσης για τους χρήστες για την προστασία των δεδομένων, υποστήριξη και αποτελεσματική διαχείριση μεγάλων δίσκων, δυνατότητα διαφανούς προς τον χρήστη κρυπτογράφησης των δεδομένων, συμπίεσης για την εξοικονόμηση χώρου και άλλα.

Η Microsoft έχει αποκαλύψει ελάχιστες πληροφορίες για την εσωτερική δομή του NTFS κι αυτό καθιστά πολύ δύσκολη την συγγραφή οδηγών για διαφορετικά λειτουργικά συστήματα. Το NTFS είναι σχεδιασμένο περίπου όπως μια βάση δεδομένων. Όταν γίνεται μια αλλαγή στα δεδομένα, πρέπει να γίνουν αλλαγές σε πολλά διαφορετικά σημεία του συστήματος αρχείων, πράγμα που σημαίνει ότι πιθανό λάθος ή παράλειψη ενημέρωσης για όλα αυτά τα σημεία, μπορεί να οδηγήσει σε απώλεια δεδομένων ή και καταστροφή του συστήματος αρχείων.

2.2.1 Δομή του NTFS

Τα περισσότερα από τα μειονεκτήματα του FAT προκύπτουν κατ΄ ευθείαν από την αρκετά απλουστευτική και ξεπερασμένη αρχιτεκτονική που χρησιμοποιεί. Στην εσωτερική δομή του FAT δεν λήφθηκε υπ΄ όψη για χαρακτηριστικα προχωρημένης αξιοπιστίας και ασφάλειας, πράγμα που έκανε πολύ

δύσκολο ή αδύνατο αργότερα την ενσωμάτωση τέτοιων χαρακτηριστικών. Το NTFS από την άλλη, έχει μια προχωρημένη αρχιτεκτονική που όχι μόνο επιτρέπει για τέτοια προχωρημένα χαρακτηριστικά, αλλά καθιστά επίσης δυνατό και εύκολο να προσθέτονται νέα χαρακτηριστικά στο μέλλον με μικρές ή και καθόλου αλλαγές στα υπάρχοντα. Αυτό ακριβώς συνέβη στην έκδοση 5.0 που κυκλοφόρησε με πολλά καινούρια χαρακτηριστικά.

2.2.1.1 Τα πάντα ένα αρχείο

Σχεδόν κάθε στοιχείο στο NTFS είναι ένα αρχείο, συμπεριλαμβανομένων και τον δομών που χρησιμοποιούνται για τη διαχείριση του συστήματος αρχείων και την στατιστική και των έλεγχο των πληροφοριών για το ίδιο το σύστημα αρχείων. Οι πληροφορίες ελέγχου αποθηκεύονται σε ένα σύνολο από ειδικευμένα αρχεία που δημιουργούνται με την δημιουργία του συστήματος αρχείων. Τα αρχεία αυτά ονομάζονται **metadata** και περιέχουν δεδομένα όπως οι λίστες των αρχείων, πληροφορίες για το σύστημα αρχείων, για τη δέσμευση των clusters και άλλα.

Μια εξαίρεση στον κανόνα "Όλα είναι ένα αρχείο" είναι ο boot sector της κατάτμησης που προηγείται των αρχείων metadata και ελέγχει τις πιο βασικές από τις λειτουργίες του NTFS, όπως η εκκίνηση του λειτουργικού συστήματος.

Το ίδιο απλό εννοιολογικό μοντέλο που χρησιμοποιείται για τα αρχεία και τις δομές ελέγχου επεκτείνεται και στο εσωτερικό επίπεδο των αρχείων: Κάθε αρχείο στο NTFS είναι μια συλλογή από χαρακτηριστικά, συμπεριλαμβανομένων και των ίδιων των δεδομένων του αρχείου που απλώς θεωρούνται ένα από όλα τα χαρακτηριστικά. Άλλα χαρακτηριστικά είναι για παράδειγμα το όνομα και το μέγεθος του αρχείου.

Η οργάνωση αυτή, που μοιάζει πολύ με οργάνωση βάσης δεδομένων, δίνει στο σύστημα αρχείων και στο λειτουργικό τη δυνατότητα να δουν κάθε αρχείο σαν ένα αντικείμενο με διάφορα χαρακτηριστικά και να τα διαχειριστούν κατάλληλα. Το γεγονός αυτό καθιστά δυνατό να προστεθούν καινούρια χαρακτηριστικά στο μέλλον.

Εσωτερικά, το NTFS αποθηκεύει όλα τα αρχεία, και τα metadata, σε ένα σύστημα με clusters. Κάθε αρχείο χωρίζεται σε clusters, καθένα από τα οποία περιέχει μια δύναμη του 2 από τομείς των 512 bytes. Υποστηριζόμενα μεγέθη cluster είναι τα 512 bytes, 1, 2, 4, 8, 16, 32 και 64 KB. Επιφανειακά, το σχήμα αυτό θυμίζει εκείνο του FAT, ο τρόπος όμως που διαχειρίζεται τα clusters το NTFS είναι διαφορετικός.

2.2.1.2 Χαρακτηριστικά (attributes)

Στο NTFS, ο πίνακας MFT (**Master File Table**, **Κύριος Πίνακας Αρχείων**) περιέχει καταχωρήσεις για όλα τα αρχεία και τους καταλόγους του συστήματος αρχείων. Οι καταχωρήσεις αυτές δεν είναι τίποτα άλλο από λίστες χαρακτήριστικών.

Τα χαρακτηριστικά είναι απλώς κομμάτια δεδομένων διάφορων ειδών. Το νόημα της πληροφορίας για ένα χαρακτηριστικό εξαρτάται από το πως το λογισμικό εκλαμβάνει και χρησιμοποιεί τα δεδομένα που περιέχονται. Οι κάτάλογοι αποθηκεύονται κατά τον ίδιο γενικό τρόπο όπως και τα αρχεία. Απλά έχουν διαφορετικά χαρακτηριστικά που χρησιμοποιούνται με διαφορετικό τρόπο από το σύστημα αρχείων.

Τα χαρακτηριστικά όλων των αρχείων και των καταλόγων αποθηκεύονται με έναν από τους δύο διαφορετικούς τρόπους, ανάλογα με το είδος των χαρακτηριστικών και κυρίως του μεγέθους τους.

- Τα χαρακτηριστικά που απαιτούν σχετικά μικρό αριθμό δεδομένων αποθηκεύονται κατ΄ ευθείαν στον πίνακα MFT. Αυτά τα χαρακτηριστικά ονομάζονται **resident**. Πολλά από τα απλούστερα και συνηθέστερα χαρακτηριστικά των αρχείων αποθηκεύονται ως resident στον πίνακα MFT. Για παράδειγμα, το όνομα και η ημερομηνία και ώρα της δημιουργίας, τελευταίας τροποποίησης και ανάγνωσης κάθε αρχείου είναι resident για κάθε αρχείο.
- Αν ένα χαρακτηριστικό απαιτεί περισσότερο χώρο απ΄ όσος είναι διαθέσιμος στην εγγραφή του πίνακα MFT, τότε δεν αποθηκεύεται εκεί, αλλά τοποθετείται σε ξεχωριστή περιοχή. Στην εγγραφή τοποθετείται ένας δείκτης που οδηγεί στο μέρος που βρίσκεται το χαρακτηριστικό. Αυτή η διαδικασία ονομάζεται **non-resident** αποθήκευση χαρακτηριστικών.

Στην πράξη, μόνο τα μικρότερα σε μέγεθος χαρακτηριστικά μπορούν να χωρέσουν στις εγγραφές του πίνακα MFT, αφού το μέγεθός τους είναι περιορισμένο. Πολλά άλλα χαρακτηριστικά θα αποθηκευτούν ως non-resident και ειδικά το δεδομένα του αρχείου.

Η non-resident αποθήκευση μπορεί κι αυτή να γίνει με δυο τρόπους. Αν το χαρακτηριστικό δεν χωράει στην εγγραφή του πίνακα MFT, αλλά χωράνε οι δείκτες προς τα δεδομένα, τότε το χαρακτηριστικό τοποθετείται σε μια **ροή δεδομένων** (**data run**) που βρίσκεται σε μια **έκταση** (**extent**), και οι δείκτες προς αυτό στον πίνακα MFT. Για την ακρίβεια, ένα χαρακτηριστικό μπορεί να αποθηκευτεί σε πολλές διαφορετικές ροές, η κάθε μια με ξεχωριστό δείκτη. Αν το αρχείο έχει τόσο πολλά extents που ούτε οι δείκτες προς αυτά δεν χωράνε στην εγγραφή του πίνακα MFT, τότε ολόκληρα τα δεδομένα του χαρακτηριστικού μπορούν να μεταφερθούν σε ένα εξωτερικό χαρακτηριστικό, δηλαδή σε μια ξεχωριστή εγγραφή του πίνακα MFT, ή ακόμα σε πολλαπλά εξωτερικά χαρακτηριστικά.

2.2.1.3 Προκαθορισμένα χαρακτηριστικά

Το NTFS διαθέτει μια σειρά από προκαθορισμένα χαρακτηριστικά.

- **Λίστα χαρακτηριστικών**: Πρόκειται για ένα μετα-χαρακτηριστικό, που περιγράφει δηλαδή άλλα χαρακτηριστικά. Αν είναι απαραίτητο για ένα χαρακτηριστικό να γίνει non-resident, τότε αυτό το χαρακτηριστικό τοποθετείται στην εγγραφή του πίνακα MFT και λειτουργεί ως δείκτης στο non-resident χαρακτηριστικό.
- **Επικεφαλίδα (Header)**: Πρόκειται για ένα σύνολο από χαμηλού επιπέδου δεδομένα που χρησιμοποιούνται εσωτερικά από το NTFS. Περιέχει τον αριθμό σειράς (sequence number) του καταλόγου και δείκτες προς τα χαρακτηριστικά του, καθώς και τον ελεύθερο χώρο στην εγγραφή του πίνακα MFT.
- **Bitmap**: Περιέχει τον δυαδικό χάρτη της δεσμευσης clusters για το αρχείο.
- **Data**: Περιέχει τα δεδομένα του αρχείου. Εξ ορισμού, τα δεδομένα σε ένα αρχείο αποθηκεύονται σε ένα μόνο χαρακτηριστικό, ακόμα κι αν αυτό είναι χωρισμένο σε κομμάτια εξ αιτίας του μεγέθους του.

- **Όνομα αρχείου**: Αποθηκεύει το όνομα (ή τα ονόματα όπως θα δούμε πιο κάτω) που είναι συνδεδεμένο με ένα αρχείο ή κατάλογο.
- **Ευρετήριο ρίζας (Index Root)**: Στο χαρακτηριστικό αυτό βρίσκεται το ευρετήριο των αρχείων που περιέχει ένας κατάλογος, ή μόνο μέρος του αν είναι πολύ μεγάλο σε μέγεθος. Αν ο κατάλογος έχει λίγα περιεχόμενα τότε το χαρακτηριστικό θα χωρέσει όλο στην εγγραφή του πίνακα MFT, αλλιώς κάποιες πληροφορίες πηγαίνουν εκεί και οι υπόλοιπες ως εξωτερικά χαρακτηριστικά.
- **Ευρετήριο δέσμευσης (Index Allocation)**: Αν τα δεδομένα του ευρετήριου του καταλόγου είναι πολλά για να χωρέσουν σε ένα ευρετήρια ρίζας, τότε η εγγραφή του καταλόγου στον πίνακα MFT θα περιέχει ένα χαρακτηριστικό ευρετήριου δέσμευσης στο οποίο θα βρίσκεται ένας δείκτης που θα δείχνει την περιοχή με το υπόλοιπο ευρετήριο.
- **Περιγραφέας ασφαλείας (Security Descriptor)**: Αυτό το χαρακτηριστικό περιέχει πληροφορίες ασφαλείας που ελέγχουν την πρόσβαση στο αρχείο ή στον κατάλογο. **Λίστες ελέγχου πρόσβασης** (**Access Control Lists**, **ACLs**) και άλλα σχετικά δεδομένα αποθηκεύονται σ΄ αυτό το χαρακτηριστικό.
- **Τυπικές πληροφορίες (Standard Information)**: Περιέχει τυπικές πληροφορίες για κάθε αρχείο και κατάλογο, όπως ημερομηνίες και ώρες δημιουργίας, τελευταίας τροποποίησης και ανάγνωσης. Επίσης, τα τυπικά χαρακτηριστικά που χρησιμοποιεί και το FAT, (μόνο για ανάγνωση, κρυφό, συστήματος και άλλα).

Πέρα από αυτά τα προκαθορισμένα χαρακτηριστικά, το NTFS επιτρέπει τη δημιουργία καινούριων χαρακτηριστικών. Οι προγραμματιστές που φτιάχνουν λογισμικό, μπορούν να δημιουργήσουν τα δικά τους χαρακτηριστικά, ανάλογα με τις ανάγκες του λογισμικού τους.

2.2.1.4 Κατάλογοι

Θεωρούμενο εξωτερικά και με βάση τη δομή, το NTFS χρησιμοποιεί τις ίδιες μεθόδους με όλα τα σύγχρονα συστήματα αρχείων για να οργανώνει αρχεία και καταλόγους. Η βάση του συστήματος αυτού είναι ο ριζικός κατάλογος που στην πραγματικότητα είναι ένα από τα σημαντικότερα αρχεία metadata του συστήματος. Στον κατάλογο αυτό αποθηκεύονται αναφορές για αρχεία ή για άλλους καταλόγους. Κάθε τέτοιος κατάλογος μπορεί να περιέχει άλλους υποκατάλογους, κι έτσι το σύστημα αρχείων σχηματίζει μια δεντρική δομή.

Εκείνο που διαφέρει στο NTFS είναι η εσωτερική διαχείριση αυτής της ιεραρχικής δομής. Στο FAT για παράδειγμα, οι κατάλογοι είναι υπεύθυνοι για την αποθήκευση των περισσότερων χαρακτηριστικών των αρχείων. Τα αρχεία αυτά καθ΄ αυτά αποθηκεύουν μόνο τα δεδομένα τους. Στο NTFS, τα αρχεία είναι μια συλλογή από χαρακτηριστικά, οπότε περιέχουν τα ίδια τις πληροφορίες για τον εαυτό τους μαζί με τα δεδομένα τους. Στο NTFS, οι κατάλογοι περιέχουν πληροφορίες μόνο για τον κατάλογο και όχι για τα αρχεία του.

Τα πάντα στο NTFS θεωρούνται αρχεία και το ίδιο ισχύει και για τους καταλόγους. Κάθε κατάλογος διαθέτει μιαν εγγραφή στον πίνακα MFT η οποία είναι η κύρια περιοχή πληροφοριών για τον κατάλογο και περιέχει όσα από τα προκαθορισμένα χαρακτηριστικά αναφέρθηκαν παραπάνω έχουν έννοια για τους καταλόγους.

Αν ο κατάλογος είναι μικρός (λίγα περιεχόμενα), αποθηκεύεται εξ ολοκλήρου στην εγγραφή του πίνακα MFT, όπως ακριβώς τα δεδομένα των μικρών αρχείων. Για τους μεγαλύτερους κατάλογους, οι πληροφορίες μοιράζονται σε πολλαπλές εγγραφές δεδομένων για τις οποίες η κύρια εγγραφή του καταλόγου στον πίνακα MFT διαθέτει δείκτες.

Το NTFS χρησιμοποιεί ένα διαφορετικό τρόπο για την αποθήκευση των καταχωρήσεων ενός καταλόγου. Στο FAT χρησιμοποιείται μια απλή συνδεδεμένη λίστα όπως είδαμε, που είναι πολύ εύκολη στην υλοποίηση αλλά παρουσιάζει μειωμένη απόδοση, ειδικά για καταλόγους με πολλά περιεχόμενα.

Για καλύτερη απόδοση, το NTFS χρησιμοποιεί μια δομή που ονομάζεται B-δέντρο και έχει την προέλευσή της στη σχεδίαση των σχεσιακών βάσεων δεδομένων. Το B-δέντρο είναι με λίγα λόγια μια ισοζυγισμένη δομή αποθήκευσης που παίρνει τη μορφή δέντρου στην οποία τα δεδομένα ισοζυγίζονται ανάμεσα στα κλαδιά του δέντρου.

Εκ κατασκευής, ένα B-δέντρο είναι μια αυτοταξινομούμενη δομή δεδομένων, πράγμα που προσθέτει κάποια καθυστέρηση στην εισαγωγή και εξαγωγή δεδομένων σ΄ αυτό. Για την καθυστέρηση αυτή όμως αποζημιωνόμαστε κατά την χρήση, δηλαδή στην αναζήτηση δεδομένων στο δέντρο. Ο χρόνος που απαιτείται για την εύρεση ενός αρχείου σε έναν κατάλογο του NTFS είναι δραστικά μικρότερος απ΄ ότι στο FAT και για κατάλογους με πολλά περιεχόμενα, οι διαφορές είναι τεράστιες.

2.2.1.5 Αρχεία

Όπως σε όλα τα συστήματα αρχείων, η βασική μονάδα αποθήκευσης στο NTFS από τη σκοπιά του χρήστη είναι τα αρχεία. Το αρχείο είναι μια συλλογή δεδομένων και μπορεί να περιέχει ότιδήποτε: Απλό κείμενο, προγράμματα σε εκτελέσιμη μορφή ή σε πηγαίο κώδικα, αρχεία πολυμέσων, εγγραφές βάσεων δεδομένων και πολλά άλλα. Το σύστημα αρχείων δεν κάνει διάκριση στα αρχεία με βάση το περιεχόμενό τους. Η χρήση του κάθε αρχείου εξαρτάται από το λογισμικό που το διαβάζει και γράφει.

Στο NTFS, τα αρχεία αποθηκεύονται με τη μορφή μιας συλλογής χαρακτηριστικών, στα οποία συμπεριλαμβάνονται και τα ίδια τα δεδομένα Τα βασικα χαρακτηριστικά ενός αρχείου είναι όσα από τα προκαθορισμένα αναφέρθηκαν παραπάνω έχουν έννοια για αρχεία.

Αν ένα αρχείο είναι μικρό σε μέγεθος, τότε μπορεί να βρίσκεται εξ ολοκλήρου μέσα στην εγγραφή στον πίνακα MFT. Το αν θα συμβεί αυτό εξαρτάται κυρίως από το μέγεθός του και το μέγεθος της εγγραφής που χρησιμοποιείται στον πίνακα MFT. Αν κάποιο χαρακτηριστικό του αρχείου είναι μεγάλο σε μέγεθος και δεν χωράει στην εγγραφή, τότε το NTFS αρχίζει μια διαδικασία με την οποία εξάγονται χαρακτηριστικά και τοποθετούνται εκτός της εγγραφής, μετατρέποντας το αρχείο σε non-resident.

Η ακριβής διαδικασία που ακολουθείται είναι η εξής: Πρώτα γίνεται απόπειρα να συμπεριληφθεί όλο το αρχείο στην εγγραφή του πίνακα MFT. Αν δεν χωράει, τότε το χαρακτηριστικό των δεδομένων γίνεται non-resident και στην εγγραφή τοποθετείται μια λίστα από δείκτες προς τη ροή δεδομένων που βρίσκονται τα δεδομένα.

Μπορεί όμως το αρχείο να είναι τόσο μεγάλο που ούτε η αυτή η λίστα να μην χωράει στην εγγραφή. Αν συμβεί αυτό, τότε και αυτή η λίστα γίνεται non-resident και τοποθετείται σε μια ξεχωριστή εγγραφή του πίνακα MFT, ενώ στην εγγραφή του αρχείου τοποθετείται ένας δείκτης προς την ξεχωριστή εγγραφή.

Το NTFS μπορεί να συνεχίσει αυτή τη διαδικασία για πολύ μεγάλα αρχεία, φτιάχνοντας περισσότερες ξεχωριστές εγγραφές στον πίνακα MFT με δείκτες προς τα δεδομένα. Προφανώς, όσο μεγαλύτερο το αρχείο,

τόσο πιο πολύπλοκη η δομή που προκύπτει.

Οι ροές δεδομένων (extents) είναι το μέρος που αποθηκεύονται τα δεδομένα των περισσότερων αρχείων στο NTFS. Αποτελούνται από ομάδες συνεχόμενων clusters στο φυσικό μέσο. Οι δείκτες προς τα δεδομένα περιέχουν μια αναφορά στο σημείο που βρίσκεται η αρχή κάθε ροής και το πόσα clusters αυτή περιλαμβάνει. Έτσι, στο NTFS δεν είναι απαραίτητο να προσπελαστεί κάθε cluster ενός αρχείου για να γνωρίζουμε που βρίσκεται το επόμενο. Η μέθοδος αυτή μειώνει και τα φαινόμενα κατακερματισμού.

2.2.2 Χαρακτηριστικά του NTFS

2.2.2.1 Ονοματολογία αρχείων.

Ένας από τους λόγους για τους οποίους σχεδιάστηκε εκ νέου το NTFS ως σύστημα αρχείων ήταν να ξεφύγει από τους περιορισμούς του FAT στην ονοματολογία των αρχείων.

Τα χαρακτηριστικά που χρησιμοποιούνται για τα ονόματα των αρχείων είναι τα παρακάτω:

- **Μήκος**: Το μέγιστο μήκος για τα ονόματα αρχείων είναι 255 χαρακτηρες.
- **Κεφαλαία/πεζά**: Το NTFS κάνει διάκριση μεταξύ κεφαλαίων και πεζών στα ονόματα των αρχείων. Επιτρέπει κεφαλαία και πεζά μαζί και τα διατηρεί χωρίς να τα μετατρέπει από τη μια μορφή στην άλλη. Η αναφορά όμως σε ένα αρχείο μπορεί να γίνει χωρίς να ληφθούν υπ΄ όψη κεφαλαία/πεζά. Για παράδειγμα το αρχείο Test.txt μπορεί να αναφερθεί και ως test.txt, TEST.TXT και TEST.txt.
- **Χαρακτήρες**: Τα ονόματα μπορούν να περιλαμβάνουν οποιουσδήποτε χαρακτήρες εκτός από τους " / \ < > : | * ? που είναι δεσμευμένοι γιατί χρησιμοποιούνται ως οριοθέτες ή τελεστές στο λειτουργικό σύστημα.
- **Unicode**: Όλα τα ονόματα αρχείων στο NTFS αποθηκεύονται σε μορφή unicode που αφιερώνει 16 δυαδικά ψηφία για κάθε χαρακτήρα και μπορεί να συμπεριλαβει πάρα πολλές χιλιάδες ειδικούς χαρακτήρες, από όλα τα αλφάβητα στον κόσμο, ακόμα και από Ασιατικές γλώσσες που χρησιμοποιούν ιδεογράμματα με χιλιάδες χαρακτήρες.

Το NTFS δημιουργεί αυτόματα για τα αρχεία ονόματα-ψευδώνυμα που ακολουθούν τον περιορισμό των 8+3 όπως στο FAT. Ο λόγος για τον οποίο γίνεται αυτό είναι η συμβατότητα με προηγούμενο λογισμικό και μόνο, αφού το NTFS σχεδιάστηκε εξ αρχής να υποστηρίζει μεγάλα ονόματα. Αν το όνομα του αρχείου εμπίπτει στους περιορισμούς των 8+3 του FAT, δε δημιουργείται ψευδώνυμο.

Το NTFS όμως πηγαίνει ακόμα παραπέρα και υποστηρίζει πολλαπλά ονόματα για κάθε αρχείο. Το ίδιο ακριβώς αρχείο μπορεί να εμφανίζεται σε πολλούς διαφορετικούς καταλόγους με διαφορετικά ονόματα. Υλοποιούνται δηλαδή δεσμοί (hard links) μια ιδέα παρμένη από τα συστήματα αρχείων του unix όπως θα δούμε παρακάτω. Πρόκειται για ένα πολύ ευέλικτο χαρακτηριστικό που διευκολύνει πολύ τη διαχείριση των δεδομένων του συστήματος, αλλά και του χρήστη.

2.2.2.2 Η ασφάλεια στο σύστημα αρχείων NTFS

Το μοντέλο ασφαλείας του συστήματος αρχείων NTFS είναι στην πραγματικότητα ένα υποσύνολο του μοντέλου ασφαλείας των Windows NT.

Η διαχείρηση της ασφάλειας στο σύστημα αρχείων NTFS γίνεται στον πίνακα MFT. Για κάθε αρχείο και κατάλογο υπάρχει ένα χαρακτηριστικό που λέγεται περιγραφέας ασφαλείας. Το πιο σημαντικό από τα περιέχόμενα αυτού του χαρακτηριστικού είναι ένα σύνολο από λίστες που καθορίζουν ποιοι χρήστες μπορούν να αποκτήσουν πρόσβαση σ΄αυτό και με ποιον τρόπο. Οι λίστες αυτές όνομάζονται λίστες ελέγχου πρόσβασης (**Access Control Lists**, **ACLs**).

Οι ACLs περιέχουν καταχωρήσεις που καθορίζουν τι δικαιώματα έχει καθε χρήστης ή ομάδα χρηστών για το κάθε αρχείο.

Κατά τη δημιουργία των Windows NT υπήρχαν έξι βασικά δικαιώματα Το καθένα από αυτά ελέγχει διαφορετικό είδος πρόσβασης:

- **Ανάγνωσης** : Προκειμένου για αρχεία, ελέγχει την ανάγνωση των δεδομένων του. Προκειμένου για καταλόγους, ελέγχει την ανάγνωση των περιεχομένων του καταλόγου.
- **Εγγραφής** : Στα αρχεία ελέγχει την τροποποίηση των δεδομένων. Στους καταλόγους, την τροποποίηση των περιεχομένων του κατάλογου (διαγραφή ή δημιουργία αρχείων για παράδειγμα).
- **Εκτέλεσης** : Ελέγχει το ποιος μπορεί να εκτελέσει ένα αρχείο (πρόγραμμα) ή να μπει στον κατάλογο.
- **Διαγραφής** : Διαγραφή του αρχείου ή του καταλόγου.
- **Αλλαγή δικαιωμάτων** : Ελέγχει το ποιος μπορεί να τροποποίησει τα δικαιώματα ενός αρχείου ή καταλόγου.
- **Οικειοποίηση ιδιοκτησίας** : Ελέγχει το ποιος μπορεί να γίνει ιδιοκτήτης του αρχείου ή καταλόγου.

Αργότερα, με την κυκλοφορία των Windows 2000, τα παραπάνω δικαιώματα αναλύθηκαν σε 13 απλούστερα.

Οι ACLs επιτρέπουν για κάθε αρχείο ή κατάλογο να τεθούν διαφορετικά δικαιώματα για κάθε χρήστη ή ομάδα χρηστών, επιτρέποντας έτσι ένα πολύπλοκο σχήμα ασφάλειας. Υπάρχουν επίσης μηχανισμοί κληρονόμησης δικαιωμάτων· ένα αρχείο κληρονομεί τα δικαιώματα του γονικού καταλόγου για παράδειγμα.

Τέλος, επειδή επιτρέπεται να τεθούν διαφορετικά δικαιώματα για κάθε χρήστη και ομάδα χρηστών και κάθε χρήστης μπορεί να είναι μέλος πολλών διαφορετικών ομάδων, συχνά προκύπτουν συγκρούσεις (**conflicts**) στα δικαιώματα. Για το σκοπό αυτό υπάρχουν μηχανισμοί που εξακριβώνουν ποια δικαιώματα θα επικρατήσουν.

2.2.2.3 Προστασία των δεδομένων στο σύστημα αρχείων NTFS

Κάθε αλλαγή που λαμβάνει χώρα στα δεδομένα ονομάζεται στο NTFS **συναλλαγή** (**transaction**) και αποτελείται στην πραγματικότητα από πολλες μικρότερες αλλαγές σε πολλά διαφορετικά σημεία του συστήματος αρχείων. Είναι σημαντικό για την ασφάλεια των δεδομένων, αυτές οι αλλαγές να εκτελούνται όλες και μάλιστα με τη σωστή σειρά.Το NTFS καταγράφει τις συναλλαγές σε ένα ημερολόγιο εώς ότου

ολοκληρωθούν όλες οι αλλαγές από τις οποίες αποτελούνται και τότε μόνο μεταφέρει τις αλλαγές στο σύστημα αρχείων. Με αυτό τον τρόπο, το NTFS εγγυάται ότι μια συναλλαγή δεν πρόκειται ποτέ να μείνει ημιτελής, πράγμα που μπορεί να οδηγήσει σε απώλει δεδομένων.

Με τη βοήθεια του λειτουργικού, κατατμήσεις με εγκατεστημένο σύστημα αρχείων το NTFS μπορούν να οργανωθούν σε συστοιχίες RAID χωρίς την ανάγκη ύπαρξης ειδικού υλισμικού (ελεγκτής RAID). Η τεχνική RAID χρησιμοποιεί πολλά μέσα αποθήκευσης παράλληλα με στόχο να αυξήσει την ασφάλεια των δεδομένων ή την ταχύτητα του συστήματος αρχείων. Για παράδειγμα, χρησιμοποιώντας RAID, μπορούμε να έχουμε δυο όμοιους σκληρούς δίσκους να λειτουργούν ταυτόχρονα και να αποτελούν ο ένας πιστό αντίγραφο του άλλου. Έτσι, σε περίπτωση που ο ένας από τους δυο παρουσιάσει πρόβλημα λειτουργίας, τα δεδομένα είναι ασφαλή στον άλλο.

Το NTFS, τέλος, υποστηρίζει αυτόματο έλεγχο και καταγραφή των κατεστραμμένων τομέων του φυσικού μέσου ώστε να μη χρησιμοποιούνται για αποθήκευση.

2.2.2.4 Άλλα χαρακτηριστικά του NTFS

Μερικά ακόμα προχωρημένα χαρακτηριστικά που υποστηρίζει το NTFS είναι τα παρακάτω:

- Διαφανής ως προς τον χρήστη **συμπίεση** των αρχείων: Ο χρήστης επιλέγει απλά τα αρχεία ή τους καταλόγους που θελει να συμπιεστούν για εξοικονόμηση χώρου και το σύστημα αρχείων τα συμπιέζει. Όταν κάποιος κάνει αίτηση ανάγνωσης για αρχεία στα οποία έχει εφαρμοστεί συμπίεση, το σύστημα αρχείων αποσυμπιέζει αυτόματα το περιεχόμενό τους πριν την ανάγνωση. Αν ο χρήστης τροποποιήσει τα αρχεία, το σύστημα αρχείων φροντίζει και πάλι να ξανασυμπιέσει τα δεδομένα πριν τα αποθηκεύσει.
- Διαφανής ώς προς τον χρήστη **κρυπτογράφηση** των δεδομένων: Τα κρυπτογραφημένα δεδομένα μπορούν να διαβαστούν μόνο από το λειτουργικό στο συγκεκριμένο υπολογιστή που πραγματοποίησε την κρυπτογράφηση.
- **Quotas**: Στο NTFS είναι δυνατόν να τεθούν όρια χρήσης του συστήματος αρχείων για κάθε χρήστη. Για κάθε ξεχωριστό σύστημα αρχείων NTFS και για κάθε χρήστη, μπορούν να τεθούν όρια σε όγκο δεδομένων, πέρα από τα οποία το σύστημα αρχείων δεν επιτρέπει την περαιτέρω δημιουργία αρχείων. Υποστηρίζεται όριο προειδοποίησης, που όταν ξεπεραστεί, το σύστημα προειδοποιεί ότι ο χρήστης βρίσκεται κοντά στο ανώτατο όριο.
- **Υποστήριξη σποραδικών αρχείων**: Όταν ο χρήστης ή κάποιο πρόγραμμα ενεργοποιήσει σε ένα αρχείο ένα από τα προκαθορισμένα χαρακτηριστικά του NTFS, την ετικέτα σποραδικού αρχείου, το σύστημα αρχείων αναλαμβάνει να διαχειριστεί διαφορετικά το αρχείο αυτό, αποθηκεύοντας μόνο τα πραγματικά δεδομένα του και όχι τον κενό χώρο. Αυτή η διαδικασία γίνεται με διαφάνεια ως προς τον χρήστη, ο οποίος βλέπει το σποραδικό αρχείο σα να ήταν αποθηκευμένο μαζί με τον κενό χώρο.

2.3 Το σύστημα αρχείων Ext2

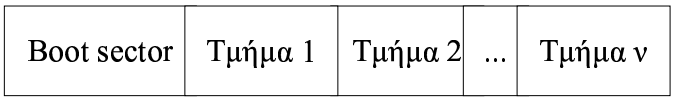
Το σύστημα αρχείων ext2 (**Second Extended Filesystem**) σχεδιάστηκε και υλοποιήθηκε ως ένα αξιόπιστο, γρήγορο και επεκτάσιμο σύστημα αρχείων για το Linux. Είναι, επίσης, μέχρι σήμερα το πιο δημοφιλές στην κοινότητα του Linux και χρησιμοποιείται εξ' ορισμού σ' όλες τις διανομές του λειτουργικού.

Τον πρώτο καιρό της ζωής του, το Linux υποστήριζε ένα και μοναδικό σύστημα αρχείων, βασισμένο σ' εκείνο του λειτουργικού συστήματος minix. Το σύστημα αυτό των αρχείων ήταν εξαιρετικά απλό και είχε πολλούς περιορισμούς, όπως το μέγιστο μηκος ονόματος αρχείου ίσο με 14 χαρακτήρες και το μέγιστο μέγεθος αρχείου ίσο με 64 Mb. Το extfs (Extended File System) σχεδιάστηκε ακριβώς για να υπερπηδηθούν αυτοί οι περιορισμοί και αντικατέστησε τον προκάτοχό του. Αργότερα, προστέθηκαν κι άλλες βελτιώσεις και καινούρια χαρακτηριστικά στο ext και φτάσαμε στο ext2.

2.3.1 Ο κόμβος-δείκτης

Βασική έννοια στο ext2 είναι ο κόμβος-δείκτης (**index-node**, **i-node**). Ο κόμβος-δείκτης αναπαριστά για κάθε αρχείο από πόσα και ποια μπλοκ αποτελείται καθώς και τα χαρακτηριστικά του, όπως το όνομα και τα δικαιώματα πρόσβασης από τους χρήστες. Κάθε αρχείο περιγράφεται από ένα μοναδικό κόμβο-δείκτη. Διατηρείται από το σύστημα αρχείων πίνακας κόμβων δεικτών καθώς και πίνακας μπλοκ δεδομένων. Κάθε κόμβος-δείκτης όπως και κάθε μπλοκ δεδομένων έχει συγκεκριμένη θέση στον αντίστοιχο πίνακα. Οι κατάλογοι είναι απλά ένας ειδικός τύπος αρχείου που περιέχει δεσμούς για τους κόμβους-δείκτες των αρχείων του καταλόγου.

Δομή του κόμβου-δείκτη: Κάθε κόμβος-δείκτης περιέχει τα εξής στοιχεία: Τύπος αρχείου (κανονικό αρχείο, κατάλογος, συμβολικός δεσμός και άλλοι), ποια μπλοκ καταλαμβάνει το αρχείο, δικαιώματα πρόσβασης, χρόνοι τελευταίας ανάγνωσης και τροποποίησης.



(Δομή του ext2)

2.3.2 Δομή του συστήματος αρχείων ext2

Η δομή του ext2 είναι η εξής: Στην αρχή βρίσκεται ο τομέας εκκίνησης (**boot sector**) που συχνά περιέχει πληροφορίες για την εκκίνηση του λειτουργικού συστήματος. Ύστερα έχουμε αρκετά συνεχόμενα όμοια τμήματα.

Αντίγραφο υπερμπλόκ	Αντίγραφο πιν. περιγραφής τμημάτων	Χάρτης πίνακα κόμβων δεικτών	Χάρτης πίνακα μπλοκ δεδομένων	Πίνακας κόμβων δεικτών	Πίνακας μπλοκ δεδομένων

(Δομή τμήματος του ext2)

Το καθένα από αυτά τα τμήματα περιέχει στην αρχή του από ένα αντίγραφο του υπερμπλόκ (**superblock**) καθώς και του πίνακα περιγραφής τμημάτων (**group descriptor table**). Το υπέρμπλοκ περιέχει σημαντικές πληροφορίες για το μέγεθος και τη δομή του συστήματος αρχείων. Ο πίνακας περιγραφής τμημάτων περιέχει την ακριβή δομή και χαρτογράφηση *κάθε* τμήματος. Τα δύο αυτά αντικείμενα βρίσκονται σε πολλά αντίγραφα, ένα για κάθε τμήμα Η ύπαρξη πολλών αντιγράφων καθιστά πιο εύκολη την ανάνηψη του συστήματος αρχείων από καταστροφές.

Στη συνέχεια, σε κάθε τμήμα υπάρχει χάρτης του πίνακα των κόμβων-δεικτών και χάρτης του πίνακα των μπλοκ δεδομένων. Οι χάρτες αυτοί περιέχουν ομάδες από δυαδικά ψηφία στα οποία αποθηκεύεται η πληροφορία αν ο κάθε κόμβος-δείκτης ή το κάθε μπλοκ δεδομένων αντίστοιχα είναι διαθέσιμα ή κατειλημμένα. Ακολουθούν ο πίνακας των κόμβων-δεικτών και ο πίνακας των μπλοκ δεδομένων.

Στο ext2 οι κατάλογοι υλοποιούνται ως λίστες με καταχωρήσεις μεταβλητού μεγέθους. Κάθε καταχώρηση περιγράφει ένα αρχείο. Περιέχει τον αριθμό του κόμβου-δείκτη, το όνομα του αρχείου και το μέγεθός του. Η χρήση μεταβλητού μεγέθους καταχωρήσεων επιτρέπει την υποστήριξη μεγάλων ονομάτων αρχείων χωρίς τη σπατάλη χώρου.

2.3.3 Δεικτοδότηση

Το ext2 χρησιμοποιεί δεικτοδοτημένη δέσμευση (**indexed allocation**) του χώρου στο φυσικό μέσο. Δηλαδή κρατάει σε ένα σημείο δείκτες για το που βρίσκονται τα μπλοκ κάθε αρχείου. Το σημείο αυτό είναι προφανώς ο κόμβος-δείκτης. Κάθε κόμβος δείκτης έχει 12 προκαθορισμένες θέσεις για δείκτες σε μπλοκ δεδομένων. Όταν ένα αρχείο αποτελείται από 12 ή λιγότερα μπλοκ, χρησιμοποιούνται αυτοί οι δείκτες. Για περισσότερα μπλοκ πρέπει να χρησιμοποιηθούν πολλαπλά επίπεδα δεικτών. Έτσι υπάρχουν τρεις ακόμα δείκτες. Ο πρώτος μπορεί να δείξει σε ένα μπλοκ που αποτελείται εξ ολοκλήρου απο δείκτες σε μπλοκ δεδομένων (1ο επίπεδο). Ο δεύτερος σε ένα μπλοκ που αποτελείται από δείκτες σε μπλοκ με δείκτες για τα μπλοκ δεδομένων (2ο επίπεδο) και ο τρίτος για το 3ο επίπεδο δεικτοδότησης. Όταν εξαντλούνται οι δείκτες του πρώτου επιπέδου χρησιμοποιούνται του δεύτερου και κατόπιν του τρίτου. Έτσι το ext2 μπορεί να χειριστεί πολύ μεγάλα αρχεία.

Παράδειγμα: Οι δείκτες στο ext2 έχουν μήκος 32 bit ή 4 χαρακτήρες. Έτσι, όταν έχουμε μπλοκ των 4 KB μπορούν να χωρέσουν 1024 δείκτες σε κάθε μπλοκ. Το πρώτο επίπεδο δεικτοδότησης μπορεί να χειριστεί αρχεία μεγέθους από 12+1 μέχρι 12+1.024 μπλοκ. Το δεύτερο επίπεδο από 12+1.024+1 μέχρι 12+1.024+1.048.576 μπλοκ και το τρίτο από 12+1.024+1.048.576+1 μέχρι 12+1.024+1.048.576+1.073.741.824 δηλαδή πάνω από ένα δισεκατομμύριο μπλοκ που σημαίνει αρχεία με μέγεθος πάνω από 4 TB.(~4.000 GB). Για μέγεθος μπλοκ ίσο με 1 KB μόνο 256 δείκτες μπορούν να χωρέσον

σε ένα μπλοκ και τα παραπάνω νούμερα γίνονται 1 ως 12, 12+1 ως 12+256, 12+256+1 ως 12+256+65.536 και 12+256+65.536+1 ως 12+256+65.536+16.777.216 μπλοκ αντίστοιχα, και το μέγιστο μέγεθος αρχείου είναι κατά τι μεγαλύτερο από 16 GB (=16.777.216 * 1 KB).

Τα πλεονεκτήματα της δεικτοδοτημένης δέσμευσης είναι ότι μπορούμε πολύ γρήγορα να κάνουμε τυχαία προσπέλαση στα αρχεία με τη βοήθεια των δεικτών. Μειονεκτήματα είναι ότι οι δείκτες καταλαμβάνουν κάποιο χώρο στο φυσικό μέσο και ότι για να διαβαστεί το αρχείο πρέπει να διαβαστούν και οι δείκτες μαζί (κάποια καθυστέρηση).

Ο χώρος που καταλαμβάνουν οι δείκτες στο ext2 για αρχεία πέραν των 12 μπλοκ είναι ένα επιπλέον μπλοκ για κάθε (μέγεθος-μπλοκ-σε-byte / 4) μπλοκ. Κι αυτό γιατί οι δείκτες των 4 bytes πρέπει να χωρέσουν σε μπλοκ δεδομένων. Με τα μεγέθη μπλοκ των 1, 2 και 4 KB που υποστηρίζει το ext2 έχουμε 1 μπλοκ δεικτών για κάθε 256, 512 και 1024 μπλοκ δεδομένων αντίστοιχα, δηλαδή από 0,4% μέχρι 0,1% των μπλοκ χρησιμοποιούνται για δείκτες, ποσοστό αρκετά μικρό. Τα ίδια νούμερα ισχύουν και για την καθυστέρηση για την επιπλεον ανάγνωση των δεικτών. Η χρήση της τεχνικής caching εξομαλύνει ακόμα περισσότερο τη διαφορά. Αν δηλαδή τα μπλοκ δεικτών βρίσκονται στη μνήμη, η προσπέλαση δεν καθυστερεί καθόλου.

Για να προσπελαστεί ένα αρχείο γίνεται αναζήτηση στους καταλόγους αρχίζοντας από τον ριζικό για την εύρεση του ονόματος κάθε στοιχείου της διαδρομής του. Από τον κάθε κατάλογο λαμβάνεται ο αριθμός κόμβου-δείκτη του επόμενου στοιχείου μέχρι να φτάσουμε στον κόμβο-δείκτη του τελικού αρχείου. Το ext2 τοποθετεί στην μνήμη cache τους τελευταίους καταλόγους που προσπελάθηκαν προσπαθώντας να μειώσει τον χρόνο αναζήτησης.

Στο ext2, κατά τη δημιουργία του συστήματος αρχείων πρέπει να εκτιμηθεί πόσα αρχεία και κατάλογοι θα χρησιμοποιηθούν και να ληφθεί μέριμνα να δημιουργηθεί και αντίστοιχος αριθμός κόμβων-δεικτών. Ο αριθμός αυτός δεν μπορεί σε καμία περίπτωση να αλλάξει μετά τη δημιουργία του συστήματος αρχείων.

Σε περίπτωση που ο αριθμός των κόμβων-δεικτών υπερτιμηθεί, τότε θα έχουμε σπατάλη χώρου αφού θα έχουμε αφιερώσει χώρο σε κάτι που δε χρειαζόμαστε. Σε περίπτωση που ο αριθμός υποτιμηθεί το πρόβλημα είναι πολύ μεγαλύτερο, αφού όταν εξαντληθούν οι διαθέσιμοι κόμβοι-δείκτες δεν θα είναι πια δυνατή η παραπέρα δημιουργία αρχείων.

2.3.4 Χαρακτηριστικά του συστήματος αρχείων ext2

Το σύστημα αρχείων ext2 υποστηρίζει τους συνήθεις τύπους αρχείων του Unix: κανονικά αρχεία, καταλόγους, αρχεία συσκευών και αρχεία συντόμευσης.

Το ext2 είναι ικανό να χειριστεί χωρίς προβλήματα συστήματα αρχείων με μέγεθος μέχρι και 4TB.

Το ext2 υποστηρίζει **μεγάλα ονόματα αρχείων** με μέγιστο μήκος 255 χαρακτήρες. Οποιοσδήποτε εκτυπώσιμος χαρακτήρας μπορεί να χρησιμοποιηθεί στα ονόματα αρχείων, αν και συνιστάται να αποφεύγονται ειδικοί χαρακτήρες για λόγους ευκολίας χρήσης.

Το ext2 κάνει **διάκριση μεταξύ πεζών και κεφαλαίων** γραμμάτων στα ονόματα των αρχείων **(case sensitive)**.

Το ext2 επιτρέπει στον διαχειριστή να **επιλέξει το μέγεθος του μπλοκ** (block) κατά τη δημιουργία του συστήματος αρχείων. Οι τυπικές διαθέσιμες επιλογές είναι 1, 2 και 4 KB. Η χρήση μεγάλου μεγέθους μπλοκ επιταχύνει το σύστημα καθώς θα πρέπει να γίνει μικρότερος αριθμός προσπελάσεων στο δίσκο για την ανάγνωση ή εγγραφή μεγάλων αρχείων. Από την άλλη, το μεγάλο μέγεθος μπλοκ συνεπάγεται και μεγαλύτερο αριθμό χαμένου χώρου (εσωτερικός κατακερματισμός). Κατά μέσο όρο, το τελευταίο μπλοκ είναι γεμάτο μόνο μέχρι τη μέση με δεδομένα. Αυτό συνεπάγεται χαμένο χώρο ίσο κατά μέσο όρο με μισό μπλοκ για κάθε αρχείο. Έτσι, καθώς αυξάνεται το μέγεθος του μπλοκ αυξάνεται και η σπατάλη χώρου.

Νέα χαρακτηριστικά αρχείων προστέθηκαν τελευταία στο σύστημα αρχείων ext2.

- Αμετάβλητα (**immutable**) αρχεία, τα οποία κανένας χρήστης δεν μπορεί να τα μεταβάλλει παρά μόνο να διαβάσει, τα οποία είναι χρήσιμα για την προστασία ευαίσθητων αρχείων ρυθμίσεων του συστήματος.
- Επεκτάσιμα-μόνο (**append-only**) αρχεία, τα οποία μπορούν να ανοιχτούν για εγγραφή, αλλά τα καινούρια δεδομένα γράφονται στο τέλος τους. Τέτοια αρχεία είναι χρήσιμα για τα αρχεία καταγραφής (log files) του συστήματος.

2.3.5 Δεσμοί

Το ext2 υποστηρίζει **δεσμούς** (**hard links**). Είναι δυνατόν να αντιστοιχιστούν στο ίδιο αρχείο περισσότερα του ενός ονόματα σε διαφορετικά σημεία του δενδρικού συστήματος καταλόγων. Ο δεσμός και το πραγματικό αρχείο μοιράζονται τον ίδιο κόμβο-δείκτη. Η καταχώρηση του δεσμού στον κατάλογο δείχνει στον ίδιο κόμβο-δείκτη του πραγματικού αρχείου. Δεν υπάρχει καμία διάκριση από το σύστημα αρχείων μεταξύ του δεσμού και του πραγματικού αρχείου. Οι περιορισμοί που υπάρχουν για τους δεσμούς είναι ότι δεν μπορεί να γίνει δεσμός με προορισμό κατάλογο και ότι πρέπει να αναφέρονται σε αρχεία στο ίδιο σύστημα αρχείων.

Το ext2 κρατάει για κάθε αρχείο τον αριθμό αναφορών που έχουν γίνει σ' αυτό. Για κάθε αρχείο, ο αριθμός αυτός αρχικά είναι 1. Με κάθε δεσμό που δημιουργείται ο αριθμός αυτός αυξάνεται κατά ένα. Σε κάθε αίτηση διαγραφής του αρχείου ελέγχεται η τιμή του αριθμού αναφορών. Αν είναι μεγαλύτερη του 1, τότε διαγράφεται μόνο η καταχώρηση στον κατάλογο, διαγράφεται δηλαδή ένας δεσμός. Αν ο αριθμός είναι 1 τότε μόνο διαγράφονται και τα περιεχόμενα του αρχείου.

Το ext2 υλοποιεί επίσης **συμβολικούς δεσμούς** (**symbolic links**). Οι συμβολικοί δεσμοί είναι ειδικοί τύποι αρχείων. Χρησιμοποιείται ένας νέος κόμβος-δείκτης μέσα στον οποίο αποθηκεύεται το όνομα και η πλήρης διεύθυνση του προορισμού. Οι συμβολικοί δεσμοί υπερπηδούν τους περιορισμούς των απλών. Να σημειωθεί επίσης ότι δεν γίνεται κανένας έλεγχος αν υπάρχει πραγματικα ο προορισμός στον οποίο δείχνει ο συμβολικός δεσμός. Αν αυτό δε συμβαίνει θα σημειωθεί λάθος κατά την προσπάθεια προσπέλασής του. Τέλος, το ext2 είναι αρκετά έξυπνο ώστε αν υπάρχουν δεσμοί με κυκλική αναφορά να μην δημιουργήται

αδιέξοδος βρόχος (infinite loop) αλλά απλά να επιστρέφεται λάθος στην αίτηση ανάγνωσης ή εγγραφής στο αρχείο.

Ο χρήστης του λειτουργικού βλέπει έναν δεσμό, συμβολικό ή μη, και διαβάζει τα περιεχόμενα του αρχείου ή καταλόγου στο οποίο αναφέρεται με τον ίδιο ακριβώς τρόπο που θα έβλεπε το πραγματικό αρχείο. Υπάρχει δηλαδή απόλυτη διαφάνεια ως προς τον χρήστη.

Οι δεσμοί καθιστούν πιο ευέλικτο το σύστημα αρχείων. Με τη βοήθειά τους μπορούμε να έχουμε τα περιεχόμενα ενός αρχείου ή καταλόγου σε πολλά διαφορετικά σημεία του συστήματος καταλόγων. Αυτό κάνει πιο εύκολη τη χρήση αλλά και τη συντήρηση του λειτουργικού. Ο χρήστης μπορεί, για παράδειγμα να κατασκευάσει συμβολικούς δεσμούς για πιο γρήγορη πρόσβαση σε σημεία που χρησιμοποιεί συχνά. Επίσης, ο διαχειριστής που κάνει συχνές αναβαθμίσεις σε ένα πρόγραμμα ή μια βιβλιοθήκη του συστήματος και για κάποιο λόγο θελει να κρατήσει πολλαπλές εκδόσεις έχει τη δυνατότητα να δημιουργήσει ένα συμβολικό δεσμό που να δείχνει κάθε φορά στην τελευταία έκδοση.

2.3.6 Η ασφάλεια στο ext2

Το ext2 διαθέτει μηχανισμούς για την προστασία και **ασφάλεια** των δεδομένων. Υιοθετείται το κλασικό σχήμα του Unix. Κάθε αρχείο έχει έναν ιδιοκτήτη και μια ομάδα χρηστών στην οποία ανήκει. Υπάρχουν τρεις διαφορετικοί τύποι δικαιωμάτων πρόσβασης για κάθε αρχείο: Για τον ιδιοκτήτη του, για την ομάδα του και τέλος για όλους τους υπόλοιπους. Για κάθε όμάδα από αυτές τις τρεις ορίζονται δικαιώματα ανάγνωσης, γραφής και εκτέλεσης. Έτσι, για να μπορεί ένας χρήστης εκτός του ιδιοκτήτη να διαβάσει το αρχείο πρέπει είτε ο χρήστης να ανήκει στην ομάδα του αρχείου και να είναι ενεργοποιημένο για το αρχείο το δικαίωμα ανάγνωσης για την ομάδα του, είτε να είναι ενεργοποιημένο για το αρχείο το δικαίωμα ανάγνωσης για όλους τους υπόλοιπους χρήστες.

Προκειμένου για κατάλογους, το δικαίωμα εκτέλεσης αντιπροσωπεύει την είσοδο στον κατάλογο, της ανάγνωσης την αναγνωση των περιεχομένων του και της εγγραφής τη δημιουργία νέου αρχείου σ' αυτόν. Τα δικαιώματα μπορεί να τα αλλάξει μόνο ο ιδιοκτήτης του αρχείου καθώς και ο διαχειριστής του συστήματος.

Το απλό αυτό σύστημα μας επιτρέπει αρκετή ευελιξία. Ας πούμε για παράδειγμα ότι έχουμε σε ένα πανεπιστήμιο μια ομάδα φοιτητών που παρακολουθούν μια τάξη. Έχουμε τη δυνατότητα να ορίσουμε στον κάθε φοιτητή να έχει πλήρη πρόσβαση στα δικά του αρχεία, αλλά μόνο πρόσβαση ανάγνωσης στα αρχεία των άλλων. Τέλος, σε όλους τους υπόλοιπους φοιτητές εκτός τάξης μπορούμε να δώσουμε το δικαίωμα της ανάγνωσης και μόνο στα αρχεία της τάξης. Βέβαια, η απλότητα του συστήματος ενέχει και περιορισμούς. Δεν μπορεί να ικανοποιήσει κάποια πιο περίπλοκα σχήματα.

2.3.7 Quotas

Το ext2 υποστηρίζει **quotas**, δηλαδή τον έλεγχο τον ορίων των δυνατοτήτων του χρήστη στο σύστημα

αρχείων. Η βασική ιδέα είναι ότι οι χρήστες υποχρεώνονται να μένουν μέσα σε συγκεκριμένα όρια και όχι να έχουν απεριόριστη ελευθερία στη χρήση κάθε πόρου του λειτουργικού συστήματος, ένα από τα επίπεδα του οποίου είναι και το σύστημα αρχείων. Τα quotas υποστηρίζονται εγγενώς από τον πυρήνα του Linux.

Τα όρια αυτά αφορούν τόσο αριθμό κόμβων-δεικτών όσο και αριθμό μπλοκ δεδομένων. Οι περιορισμοί τίθενται ανά χρήστη ή ομάδα χρηστών και έχουν ισχύ ξεχωριστά για κάθε σύστημα αρχείων .

Οι περιορισμοί που μπορούν να γίνουν είναι οι εξής:

- Χωρίς περίοδο χάριτος : Τίθεται ένα όριο, που αντιπροσωπεύει το μέγιστο όριο χρήσης. Ο χρήστης δεν έχει δικαιοδοσία να φτάσει πέρα από αυτό το όριο. Όταν το προσπαθήσει, το σύστημα απλά δεν τον αφήνει, αλλά παρουσιάζεται σχετικό σφάλμα.
- Με περίοδο χάριτος : Τίθενται δυο διαφορετικά όρια, το soft όριο και το hard όριο. Το hard όριο αντιπροσωπεύει το απόλυτο όριο, το οποίο ο χρήστης δεν μπορεί ποτέ να ξεπεράσει. Όταν ο χρήστης ξεπεράσει το soft όριο, τότε, το σύστημα τον προειδοποιεί ότι έχει στη διάθεσή του όλη τη διάρκεια της περιόδου χάριτος για να σταματήσει να υπερβαίνει το soft όριο. Μετά το πέρας της περιόδου χάριτος, το σύστημα θέτει το soft όριο ως απόλυτο όριο.

2.3.8 Συντήρηση του συστήματος αρχείων ext2

Το ext2 **παρακολουθεί την κατάσταση** του συστήματος αρχείων. Ένα συγκεκριμένο πεδίο στο υπέρμπλοκ χρησιμοποιείται από τον πυρήνα γι' αυτό το σκοπό. Κάθε φορά που γίνεται προσάρτηση του συστήματος αρχείων με επιλογή ανάγνωσης και εγγραφής (read/write) το πεδίο αυτό τίθεται στην τιμή «όχι εντάξει» (**not clean**). Κάθε φορά που γίνεται αποπροσάρτηση του συστήματος αρχείων ή προσάρτηση με επιλογή μόνο για ανάγνωση (read only) το πεδίο αυτό τίθεται στην τιμή «εντάξει» (**clean**). Κατά την εκκίνηση του λειτουργικού ο πυρήνας ελέγχει το πεδίο αυτό. Αν η τιμή του είναι «όχι εντάξει» σημαίνει ότι το σύστημα αρχείων πρέπει να ελεγχθεί για λάθη και ασυνέπειες.

Επίσης, το ext2 διαθέτει τρόπους για τον τακτικό **περιοδικό έλεγχο** του συστήματος αρχείων κατά την εκκίνηση. Στο υπερμπλόκ υπάρχει ένας μετρητής προσαρτήσεων· κάθε φορά που προσαρτείται το σύστημα αρχείων η τιμή του μετρητή αυξάνεται. Όταν ο μετρητής φτάσει σε μια ορισμένη τιμή, (που μπορεί να την θέσει ο διαχειριστής και αποθηκεύεται επίσης στο υπερμπλόκ), γίνεται υποχρεωτικά έλεγχος. Φυλάσσεται επίσης στο υπερμπλόκ η ημερομηνία τελευταίου ελέγχου και μπορεί να οριστεί το μέγιστο χρονικό διάστημα μεταξύ δυο ελέγχων.

Το ext2 δεσμεύει ένα ποσοστό των συνολικών διαθέσιμων μπλοκ, συνήθως το 5%, για χρήση από το διαχειριστή τους συστήματος. Αυτό το γεγονός επιτρέπει στον τελευταίο να μπορεί να διαχειριστεί με ευελιξία τυχόν καταστάσεις όπου κάποια προγράμματα γεμίζουν τελείως το σύστημα αρχείων.

Το ext2 επιτρέπει στους χρήστες να ζητήσουν **ασφαλή διαγραφή** των αρχείων τους. Όταν ένα τέτοιο αρχείο διαγράφεται, τυχαία δεδομένα γράφονται πάνω στα μπλοκ που είχαν εκχωρηθεί γι' αυτό το αρχείο. Αυτό εμποδίζει κακόβουλους χρήστες από την απόκτηση πρόσβασης στα ευαίσθητα δεδομένα των χρηστών.

2.3.9 Θέματα απόδοσης για το σύστημα αρχείων ext2

Όσον αφορά την απόδοση, το ext2 χρησιμοποιεί πολλές τεχνικές για να αυξήσει την αποδοτικότητά του.

- Γίνεται χρήση της τεχνικής **caching**. Τα μπλοκ που έχουν διαβαστεί από το φυσικό μέσο παραμένουν στην πιο γρήγορη φυσική μνήμη μέχρι να αντικατασταθούν από πιο πρόσφατα αναγνωσμένα.
- Επίσης, κατά τη σειριακή ανάγνωση από το φυσικό μέσο, γίνεται από το σύστημα αρχείων ανάγνωση των επόμενων μπλοκ (**readahead**) για να είναι διαθέσιμα στη φυσική μνήμη ώστε να ικανοποιηθούν άμεσα επόμενες αιτήσεις.
- Η χρήση πολλών διαφορετικών τμημάτων στο ext2 συντελεί στο να είναι αρκετά «κοντά» στο φυσικό μέσο η θέση των μπλοκ δεδομένων με τους κόμβους-δείκτες, Αυτό μειώνει τις μετακινήσεις στο φυσικό μέσο που καταναλώνουν άσκοπα χρόνο.
- Τέλος, το ext2 δεσμεύει αυτόματα μέχρι και 8 μπλοκ δεδομένων κατά την εγγραφή. Η τακτική αυτή μειώνει ακόμα περισσότερο τις προσπελάσεις στο φυσικό μέσο.

Περιγράφοντας τη δομή του ext2 αναφέραμε ότι το φυσικό μέσο χωρίζεται σε πολλά παρόμοια τμήματα. Κάθε τέτοιο τμήμα περιέχει ένα τμήμα των συνολικών κόμβων-δεικτών και ένα αντίστοιχο τμήμα από τα συνολικά μπλοκ δεδομένων. Ο λόγος για τον οποίο συμβαίνει αυτό είναι **να βρίσκονται οι κομβοι-δείκτες κοντά** (στο φυσικό μέσο) **στα μπλοκ δεδομένων** στα οποία αναφέρονται. Μ΄ αυτό τον τρόπο μειώνονται οι συνεχείς αναζητήσεις (seeks) στο φυσικό μέσο που επηρεάζουν αρνητικά την απόδοση.

Το ext2, προσπαθεί όσο αυτό είναι δυνατόν, να διατηρεί τους κόμβους-δείκτες και τα μπλοκ δεδομένων για κάθε αρχείο στο ίδιο τμήμα. Επίσης, προσπαθεί να διαμοιράζει τους καταλόγους εξίσου σε όλα τα τμήματα και να τοποθετεί τα αρχεία κάθε καταλόγου στο ίδιο τμήμα με τον κατάλογό τους. Κατά την προσθήκη νέων δεδομένων σε ένα αρχείο, ψάχνοντας για αχρησιμοποίητα μπλοκ δεδομένων, ψάχνει πρώτα στα μπλοκ που βρίσκονται κοντά στο τελευταίο μπλοκ του αρχείου. Το ext2 τα καταφέρνει αρκετά καλά στις παραπάνω προσπάθειες, εφ΄ όσον το σύστημα αρχείων δεν είναι πάνω από περίπου 90% πλήρες, οπότε είναι πλέον αρκετά δύσκολο να τα επιτύχει λόγω της έλλειψης ελεύθερων μπλοκ.

Από τα παραπάνω γίνεται σαφές ότι το ext2 είναι αρκετά έξυπνο ώστε να ελέγχει και να ελαχιστοποιεί τον κατακερματισμό. Για να καταφέρει όμως να αποδώσει το ext2 όσο γίνεται καλύτερα πρέπει να φροντίζουμε η πληρότητά του να μην ξεπερνάει το 90% περίπου.

Επειδή οι κατάλογοι στο ext2 είναι γραμμικές λίστες μεταβλητού μεγέθους, πρέπει να γίνει **γραμμική αναζήτηση** σε κάθε κατάλογο που σημαίνει ότι θα έχουμε καθυστέρηση για καταλόγους με πολλά αρχεία.

Επιπλέον, τα αρχεία κάθε καταλόγου αποθηκεύονται με τη σειρά με την οποία δημιουργήθηκαν, πράγμα που σημαίνει ότι κάθε φορά που ο χρήστης ζητάει μια ταξινομημένη κατ΄ όνομα λίστα με τα αρχεία ενός καταλόγου, η ταξινόμηση πρέπει να γίνεται εκ νέου. Η αδυναμία γρήγορου χειρισμού καταλόγων με πολλά αρχεία είναι ένα από τα μειονεκτήματα του ext2.

Επίσης, υπάρχει ανώτατο όριο στο πλήθος υποκαταλόγων που μπορούν να δημιουργηθούν σε έναν κατάλογο του ext2. Το όριο αυτό είναι 32.768 υποκατάλογοι. Έτσι κι αλλιώς όμως, όταν το πλήθος των αρχείων και υποκαταλόγων σε έναν κατάλογο ξεπερνάει τις 20.000, το ext2 γίνεται τόσο αργό που είναι πρακτικά ασύμφορη η χρήση του.

Τέλος, η έξυπνη τεχνική του ext2 με την άμεση, και απλά, διπλά και τριπλά έμμεση δεικτοδότηση πάσχει από καθυστερήσεις προκειμένου για πολύ μεγάλα αρχεία. Για παράδειγμα, ας σκεφτούμε ότι για να προσπελάσουμε το μπλοκ ενός αρχείου που είναι αρκετά μεγάλο ώστε να χρειάζεται τριπλά έμμεση δεικτοδότηση, θα χρειαστούμε τέσσερεις προσπελάσεις στο φυσικό μέσο, παρά την όποια βελτίωση στην ταχύτητα έχουμε από τη μνήμη cache.

2.4 Το σύστημα αρχείων ext3

2.4.1 Περιγραφή του συστήματος αρχείων ext3

Το σύστημα αρχείων ext3 είναι μια **επέκταση του ext2 που υποστηρίζει journaling**. Διατηρεί όλα τα πλεονεκτήματα και την ευελιξία του ext2 και προσθέτει σ΄ αυτά και τις ωφέλειες από το journaling, δηλαδή τη βελτιωμένη προστασία των δεδομένων και τον πολύ γρήγορο έλεγχο του συστήματος αρχείων ύστερα από καταρρευση του υπολογιστή.

Να σημειωθεί ότι η μετατροπή ένος σύστηματος αρχείων που έχει διαμορφωθεί ως ext2 σε ext3 μπορεί να γίνει εύκολα και γρήγορα με τα εργαλεία που συνοδεύουν το ext2. Επιπλέον, ένα σύστημα αρχείων ext3 μπορεί να προσαρτηθεί και ως ext2, να μη γίνεται χρήση του journaling δηλαδή. Αυτό είναι πολύ εύκολο αφού μοιράζονται την ίδια φυσική δομή.

2.4.2 Δομή του συστήματος αρχείων ext3

Λέμε ότι ένα σύστημα αρχείων είναι **συνεπές** (**consistent**) όταν για κάθε μπλοκ δεδομένων ξέρουμε αν είναι χρησιμοποιημένο ή ελεύθερο, κάθε χρησιμοποιημένο μπλοκ ανήκει σε ένα και μόνο ένα αρχείο ή κατάλογο, και όταν όλα τα αρχεία και οι κατάλογοι του συστήματος μπορούν να προσπελαστούν διασχίζοντας του καταλόγους και τους υποκαταλόγους κάτω από τον ριζικό (όχι χαμένα αρχεία ή κατάλογοι).

Στα συστήματα αρχείων που δεν χρησιμοποιούν journaling, οι αλλαγές στα αρχεία αντικατοπτρίζονται

απευθείας στα μπλοκ δεδομένων. Κάθε αλλαγή συνίσταται σε μια σειρά από βήματα. Για παράδειγμα, όταν παίρνουμε ένα υπάρχον αρχείο και το διορθώνουμε ώστε να προκύψει ένα μεγαλύτερο, έχουμε: εύρεση και δέσμευση κενών μπλοκ δεδομένων, ενημέρωση του κόμβου-δείκτη για τα νέα μπλοκ και το νέο μέγεθος, και τέλος εγγραφή των νέων δεδομένων πάνω στα παλιά μπλοκ καθώς και στα κενά νέα μπλοκ. Για να είναι το σύστημα αρχείων συνεπές, πρέπει ή να εκτελεστούν τα παραπάνω βήματα με επιτυχία ή να μην εκτελεστεί κανένα.

Συνήθως όλα πάνε καλά. Μερικές φορές όμως, συμβαίνει να έχουμε κατάρρευση του υπολογιστή στη μέση μιας εγγραφής. Αυτό μπορεί να συμβεί λόγω απρόβλεπτων παραγόντων όπως διακοπές ρεύματος ή φυσικές καταστροφές, λόγω προβλημάτων υλικού ή και λόγω προβλημάτων στο λογισμικό. Κατά την επανεκκίνηση του υπολογιστή πρέπει να ελεγχθεί το σύστημα αρχείων για το αν είναι συνεπές. Ο έλεγχος γίνεται εξετάζοντας και συγκρίνοντας ένα προς ένα όλα τα δεδομένα (metadata) που είναι αποθηκευμένα στους κόμβους-δείκτες. Όσο μεγαλύτερο σε έκταση είναι το σύστημα αρχείων και όσο πιο πολλά δεδομένα περιέχει, τόσο περισσότερο θα κρατήσει ο έλεγχος αυτός. Η καθυστέρηση αυτή μπορεί να είναι πολύ ενοχλητική για συστήματα που απαιτείται να είναι συνεχώς διαθέσιμα (**high availability systems**), π.χ. διακομιστές.

Η τεχνική του journaling συνίσταται στο να διατηρούμε ένα **ημερολόγιο** (**journal**) στο οποίο καταγράφονται όλες οι συναλλαγές (transactions) που γίνονται στο σύστημα αρχείων, *χωρίς να αντικατοπτρίζονται άμεσα στα δεδομένα*. Σε τακτά χρονικά διαστήματα, όλα τα περιεχόμενα του ημερολογίου μεταφέρονται στα δεδομένα, αδειάζοντάς το.

Σε περίπτωση κατάρρευσης του υπολογιστή, ξέρουμε μετά την επανεκκίνηση ότι το μεγαλύτερο μέρος του συστήματος αρχείων είναι συνεπές. Το μόνο που χρειάζεται να ελέγξουμε είναι τα περιεχόμενα του ημερολογίου και να πραγματοποιηθούν τυχόν εκεί εγγεγραμένες συναλλαγες. Ο έλεγχος και η ανάνηψη είναι ασύγκριτα πιο γρήγορα, τις περισσότερες φορές θέμα ελάχιστων δευτερολέπτων. Το σημαντικότερο όμως είναι ότι η **ανάνηψη είναι ανεξάρτητη του μεγέθους του συστήματος αρχείων**.

Επίσης, το ext3 μας επιτρέπει να διατηρούμε το ημερολόγιο του συστήματος αρχείων σε διαφορετικό φυσικό μέσο. Χρησιμοποιώντας, για παράδειγμα, μια μικρή κατάτμηση σε ένα πιο γρήγορο δίσκο, μπορούμε να αυξήσουμε σημαντικά την απόδοση του συστήματος αρχείων. Ακόμα και μονάδες δυναμικής μνήμης μπορούν να χρησιμοποιηθούν για ακόμα μεγαλύτερη απόδοση.

Άλλα σημαντικά πλεονεκτήματα του journaling που μας προσφέρει το ext3 είναι ακόμα λιγότερος κατακερματισμός και υπό κάποιες συνθήκες υψηλότερη απόδοση στην εγγραφή. Τα παραπάνω εξηγούνται εύκολα αν σκεφτούμε ότι οι εγγραφές στο φυσικό μέσο γίνονται κατά περιόδους, όταν δηλαδή αδειάζουμε τα περιεχόμενα του ημερολογίου, και όχι συνεχόμενα.

Το ext3 υλοποιεί το journaling χρησιμοποιώντας ένα στρώμα διασύνδεσης (interface layer) που λέγεται **Journaling Block Device**, **JBD**. Το JBD έχει σχεδιαστεί ώστε να μπορεί να υλοποιήσει journaling σε οποιοδήποτε είδος φυσικού μέσου. Το ext3 θα ενημερώσει το JBD για όποιες αλλαγές κάνει στα μπλοκ δεδομένων και θα του ζητήσει άδεια πριν το κάνει. Έτσι, ουσιαστικά, το JBD είναι ο διαχειριστής του journaling. Με το JBD γίνεται εύκολη η μετατροπή στο μέλλον και άλλων συστημάτων αρχειων ώστε να υποστηρίζουν journaling.

Ο τρόπος που το JBD πραγματοποιεί το journaling έχει να κάνει με ολόκληρα μπλοκ δεδομένων. Κάθε φορά που γίνεται μια αλλαγή σε δεδομένα σε ένα μπλοκ, το JBD σημειώνει ότι αυτό το μπλοκ έχει αλλάξει. Αυτό, αρχικά φαίνεται λίγο περιττό, αφού στο μπλοκ μπορεί να υπάρχουν και δεδομένα που δεν άλλαξαν. Στην πραγματικότητα όμως η προσέγγιση καθιστά σημαντικά μικρότερη την πολυπλοκότητα της διαχείρισης του journaling και δίνει και τη δυνατότητα για ομαδοποίηση αλλαγών που συμβαίνουν πολλές φορές στο ίδιο μπλοκ. Δηλαδή, πολλές αλλαγές στο ίδιο μπλοκ θα γραφτούν μόνο μια φορά στο φυσικό μέσο. Η προσέγγιση αυτή του journaling ονομάζεται **physical journaling** γιατί ακολουθεί τη δομή των μπλοκ στο φυσικόμέσο.

Αντίθετα, η προσέγγιση κατά την οποία σημειώνουμε για κάθε αλλαγή μόνο τα δεδομένα που άλλαξαν, αρχή, τέλος της αλλαγής και νέα δεδομένα για παράδειγμα ονομάζεται **λογικό journaling** (**logical journaling**) και χρησιμοποιείται από άλλα συστήματα αρχείων που υποστήρίζουν journaling.

2.4.3 Χαρακτηριστικά του συστήματος αρχείων ext3

Με τον όρο **metadata** εννοούμε τις πληροφορίες που αποθηκεύονται σε ένα σύστημα αρχείων και αφορούν τα δεδομένα, δεν εννούμε όμως τα ίδια τα δεδομένα. Στην περίπτωση των ext2 και ext3, τα metadata αποθηκεύονται στους κόμβους-δείκτες (ονόματα αρχείων, έκταση, θέση στο φυσικό μέσο, ιδιότητες και άλλα).

Το σύστημα αρχείων ext3 μας δίνει τρεις δυνατές καταστάσεις λειτουργίας:

- Journal : Στην κατάσταση αυτή, στο ημερολόγιο αποθηκεύονται οι αλλαγές στα metadata καθώς και στα ίδια τα δεδομένα. Γι' αυτό ακριβώς το λόγο είναι η κατάσταση που παρουσιάζει τη μεγαλύτερη καθυστέρηση. Εγγυάται όμως καλύτερα από όλες τις άλλες την ασφάλεια των δεδομένων, αφού ότι και να συμβεί, όλες οι αλλαγές θα βρίσκονται στο ημερολόγιο.
- Ordered : Στο ημερολόγιο αποθηκεύονται μόνο οι αλλαγές στα metadata. Οι αλλαγές στα δεδομένα πρωθούνται στα μπλοκ δεδομένων ακριβώς πριν γραφτούν οι αλλαγές στα αντίστοιχα metadata. Αυτή είναι η εξ΄ ορισμού κατάσταση λειτουργίας.
- Writeback : Στο ημερολόγιο αποθηκεύονται μόνο οι αλλαγές στα metadata. Για τις αλλαγές στα δεδομένα όμως, βασιζόμαστε στην εξ' ορισμού λειτουργία εγγραφής του συστήματος για το πότε θα γραφτούν στα μπλοκ δεδομένων. Αυτή είναι η πιο γρήγορη κατάσταση λειτουργίας.

Η εξ' ορισμού λειτουργία εγγραφής στο Linux γίνεται κάθε 30 δευτερόλεπτα. Δηλαδή, όταν δε χρησιμοποιήται journaling, οι αλλαγές που αιτούνται να γίνουν στο σύστημα αρχείων, μεταφέρονται στο φυσικό μέσο μόνο κάθε 30 ευτερόλεπτα.

Όπως μπορεί κανείς να καταλάβει, αν θέλουμε να κερδίσουμε σε ταχύτητα, πρέπει να κάνουμε κάποιες παραχωρήσεις στην ασφάλεια των δεδομένων και αντίστροφα.

Στην κατάσταση journal, έχουμε τη μέγιστη ασφάλεια: Ό,τι και να συμβεί, το ημερολόγιο είναι εκεί και μας εγγυάται την ακεραιότητα των δεδομένων μας. Το πρόβλημα είναι όμως ότι όλες οι αλλαγές, στα

δεδομένα και στα metadata, γράφονται δυο φορές: μια στο ημερολόγιο, ακριβώς τη στιγμή που έγιναν και μια στα μπλοκ δεδομένων όταν έρθει η ώρα αδειάσματος του ημερολογίου. Αυτό σημαίνει ότι έχουμε επιπτώσεις στην ταχύτητα.

Στην εξ' ορισμού κατάσταση λειτουργίας ordered, μόνο οι αλλαγές στα metadata γράφονται δυο φορές. Οι αλλαγές στα δεδομένα, δεν περνάνε από το ημερολόγιο, αλλά γράφονται ακριβώς πριν γραφτούν οι αλλαγές στα metadata. Εδώ έχουμε σε πολύ μεγάλο βαθμό προστασία των δεδομένων μας. Έχουμε όμως κάπως χαμηλότερη απόδοση σε σχέση με τη μη χρήση journaling, αφού θα έχουμε περισσότερες και συχνότερες προσπελάσεις στο φυσικό μέσο.

Από την άλλη, στην κατάσταση λειτουργίας writeback, αν συμβεί κατάρρευση του συστήματος πριν προλάβει να γίνει λειτουργία εγγραφής, μπορεί να παρουσιαστούν κάποιες ασυνέπειες των δεδομένων ως προς τα metadata. Δηλαδή, για παράδειγμα, μπορεί ένα καινούριο μπλοκ δεδομένων που θα έπρεπε να περιέχει την προσθήκη στο τέλος ενός αρχείου να περιέχει σκουπίδια. Το journaling μας εγγυάται, βέβαια, ότι αυτό ακριβώς είναι το χειρότερο που μπορεί να συμβεί. Δεν υπάρχει κίνδυνος να καταστραφεί το σύστημα αρχείων, καθώς η συνέπειά του είναι κάθε στιγμή εγγυημένη, ακριβώς λόγω του ημερολογίου. Και πάλι όμως, η προοπτική τέτοιων προβλημάτων είναι αρκετά απογοητευτική για πολλούς χρήστες.

2.5 Το σύστημα αρχείων Reiserfs

Το σύστημα αρχείων Reiser πήρε το όνομά του από τον εμπνευστή και αρχικό σχεδιαστή του, Hans Reiser.

Το Reiserfs ανήκει στην κατηγορία των νέων και πολλά υποσχόμενων συστημάτων αρχείων. Μερικά από τα πλεονεκτήματά του είναι η πολύ αυξημένη απόδοση, ειδικά στα μικρά αρχεία, καθώς και σε καταλόγους με μεγάλο πλήθος αρχείων, η εξοικονόμηση του κενού χώρου στο τέλος των αρχείων, η αυτόματη κατανομή κόμβων-δεικτών ανάλογα με τις ανάγκες του συστήματος, η υποστήριξη journaling και άλλα. Από την έκδοση 2.4.1 και μετά, το Reiserfs είναι ενσωματωμένο στον πυρήνα του Linux.

2.5.1 Φιλοσοφία του συστήματος αρχείων Reiserfs

Η βασική σκέψη στη σχεδίαση του Reiserfs είναι να επιτευχθεί ένα περιβάλλον στο οποίο όλοι οι χρήστες και το λογισμικό να μπορούν να αλληλεπιδρούν άμεσα, δυναμικά και αποδοτικά. Γι΄ αυτό πρέπει το σύστημα αρχείων να ικανοποιεί όλες τις ανάγκες των χρηστών του, ώστε να μη χρειάζεται αυτοί να εφευρίσκουν άλλες δομές που θα λειτουργούν πάνω από το συστήμα αρχείων.

Ένα παράδειγμα για τα παραπάνω είναι η απόδοση του συστήματος αρχείων στα μικρά μεγέθη αρχείων, δηλαδή στα αρχεία με μέγεθος μικρότερο του μεγέθους μπλοκ. Στα περισσότερα συστήματα αρχείων η απόδοση για τέτοια αρχεία είναι πολύ μικρότερη σε σχέση με τα μεγάλα αρχεία. Σ' αυτό έρχεται να προστεθεί η σπατάλη χώρου λόγω του εσωτερικού κατακερματισμού. Όλα αυτά ανάγκασαν και αναγκάζουν

τους δημιουργούς λογισμικού (π.χ. βάσεων δεδομένων) να κατασκευάζουν δικές τους δομές που θα διαχειρίζονται τα δεδομένα χρησιμοποιώντας μεγάλα αρχεία για να αποφύγουν όλα τα παραπάνω προβλήματα.

Οι δημιουργοί του Reiserfs πιστεύουν ότι αυτό είναι μια τελείως λάθος προσέγγιση και ότι όσον αφορά αυτόν τον τομέα, τα μέχρι τώρα υπάρχοντα συστήματα αρχείων έχουν αποτύχει, αφού οι χρήστες *αναγκάζονται* να φτιάχνουν μόνοι τους δομές που λειτουργούν πάνω από το σύστημα αρχείων. Επιπλέον, όλα αυτά οδηγούν σε πρακτικές σύμφωνα με τις οποίες ο καθένας προσπαθεί να λύσει τα παραπάνω προβλήματα με το δικό του (συχνά όχι καλύτερο) τρόπο αντί να επιδιώκεται μια συλλογική και πιο οργανωμένη προσέγγιση. Τελικά δηλαδή, έχουμε πολλές μικρές διαφορετικές λύσεις και άλλα τόσα νέα προβλήματα που δημιουργούνται απ΄ αυτές, αντί να έχουμε μια καλά σχεδιασμένη και επικροτούμενη από όλους ενιαία λύση.

Γι΄ αυτούς τους λόγους, το Reiserfs σχεδιάστηκε, μεταξύ άλλων, ώστε να έχει και στα μικρού μεγέθους αρχεία ισοδύναμη απόδοση με τα μεγάλου μεγέθους. Όλα αυτά βέβαια, χωρίς να πρέπει να θυσιαστεί η απόδοση για τις υπόλοιπες κατηγορίες αρχείων. Είναι, επίσης, πάρα πολύ αποτελεσματικό στη διαχείριση καταλόγων με πάρα πολλά αρχεία.

Το Reiserfs είναι ένα πολύ αυστηρά δομημένο σύστημα αρχείων, που αποτελείται από διαφορετικά αντίκείμενα. Κάθε αντικείμενο ελέγχει τις λειτουργίες που συμβαίνουν σ' αυτό, παρέχοντας μια περιορισμένη **διασύνδεση** (**interface**). Επιτρεπτές είναι μόνο οι λειτουργίες που παρέχει η διασύνδεση. Αυτό μπορεί με πρώτη ματιά να φαίνεται ότι περιορίζει τη λειτουργικότητα, είναι όμως μια βασική αρχή του **αντικειμενοστραφούς προγραμματισμού**, η οποία διευκολύνει πάρα πολύ τον προγραμματιστή καθώς η πολυπλοκότητα αυξάνει. Ο έλεγχος για τυχόν παρενέργειες και για σφάλματα γίνεται ασύγκριτα πιο εύκολος. Στην πράξη, πολλά projects ναυάγησαν λόγω της μη τήρησης των παραπάνω κανόνων και της μεγάλης αύξησης της πολυπλοκότητας.

Συχνά ένα πρόγραμμα είναι χωρισμένο σε **στρώματα** (**layers**), οπότε κάθε στρώμα δεν μπορεί να αλληλεπιδράσει παρά μόνο με το αμέσως προηγούμενο και το επόμενό του. Η τακτική αυτή μειώνει επίσης την πολυπλοκότητα του προγράμματος και μεταθέτει τις σχεδιαστικές αποφάσεις κυρίως στη δημιουργία διασυνδέσεων μεταξύ των διάφορων στρωμάτων.

Το Reiserfs σχεδιάστηκε έχοντας στο μυαλό και την παραπάνω φιλοσοφία. Τα αρχεία και οι κατάλογοι στο Reiserfs θεωρούνται ως «αντικείμενα». Επιτρεπτές ενέργειες γι΄ αυτά είναι η ανάγνωση και η εγγραφή χαρακτήρων. Ο χρήστης μπορεί να καθορίσει από ποιον χαρακτήρα θα αρχίσει η ανάγνωση ή η εγγραφή και τον αριθμό των χαρακτήρων που θα διαβαστούν ή γραφτούν. Επιτρέπεται επίσης η αποκοπή χαρακτήρων από το τέλος του αρχείου.

2.5.2 Δομή του συστήματος αρχείων Reiserfs

Το Reiserfs κάνει χρήση των **dancing trees** που είναι μια βελτιωμένη εκδοχή των ισοζυγισμένων B- και B+ δέντρων τα οποία χρησιμοποιούνται και στη διαχείριση βάσεων δεδομένων.

Όλο το σύστημα αρχείων θεωρείται σαν μια μεγάλη βάση δεδομένων. Δημιουργείται ένα δέντρο για

όλο το σύστημα αρχείων, στο οποίο αποθηκεύονται όλα τα χαρακτηριστικά των αρχείων και των καταλόγων. Η αναζήτηση και η προσπέλαση των αρχείων στους καταλόγους γίνεται με βάση τα ταξινομημένα περιεχόμενα του δέντρου. Αυτό κάνει το Reiserfs πολύ αποδοτικό στη διαχείριση καταλόγων, σε σχέση με άλλα συστήματα αρχείων που κάνουν γραμμική αναζήτηση.

Το Reiserfs εισάγει στο δέντρο τα μερικώς γεμάτα τελευταία μπλοκ των αρχείων. Προκειμένου για μικρά αρχεία που καταλαμβάνουν λιγότερο από ένα μπλοκ, τα αρχεία μπορεί να βρίσκονται εξ ολοκλήρου μέσα στο δέντρο, πράγμα που αυξάνει κατά πολύ την απόδοση εφ΄ όσον δεν χρειάζεται επιπλέον προσπέλαση στο χώρο των δεδομένων για την ανάκληση αυτών των μπλοκ. Επιπλέον, με μόνο μια προσπέλαση στο φυσικό μέσο μπορούν να διαβαστούν πάρα πολλά μικρά αρχεία μαζί. Έτσι εξηγείται η ασύγκριτη υπεροχή του Reiserfs στα μικρά αρχεία.

Όσον αφορά την εξοικονόμηση χώρου, υπάρχει και η επιλογή να γίνεται «πακετάρισμα» των τελευταίων δεδομένων των αρχείων (ουρών) ώστε να πιάνουν λιγότερο χώρο και να μην πηγαίνει όλοκληρο μπλοκ χαμένο γι΄ αυτά. Το Reiserfs δηλαδή, τοποθετεί αυτά τα δεδομένα σε ομάδες σε κοινά μπλοκ. Το χαρακτήριστικό αυτό ονομάζεται **tail packing** και από την εμπειρία χρήσης του έχει παρατηρηθεί ότι μπορεί να εξοικονομήσει μέχρι και 5 εώς 6% χώρο στο φυσικό μέσο. Φυσικά αυτό γίνεται με κάποιο κόστος στην ταχύτητα, καθώς υπάρχει επιπλέον καθυστέρηση για να γίνει το «πακετάρισμα» των δεδομένων και για να ξαναγίνεται οποτεδήποτε τα αρχεία αυτά μεταβάλλονται. Γι΄ αυτό το λόγο το χαρακτηριστικό αυτό είναι προαιρετικό, (παράμετρος notail κατά την προσάρτηση του συστήματος αρχείων).

Το Reiserfs δημιουργεί δυναμικά κόμβους-δεικτες στις δομές του δέντρου ανάλογα με τις ανάγκες χρήσης. Αυτό σημαίνει ότι ο διαχειριστής ελευθερώνεται από τη δέσμευση να είναι υποχρεωμένος να ξέρει ή να κάνει εκτίμηση κατά τη δημιουργία του συστήματος αρχείων πόσα αρχεία και καταλόγους θα χρειαστεί.

Το Reiserfs υποστηρίζει journaling στα metadata, πράγμα που προσθέτει στην ασφάλεια των δεδομένων του.

Παρά τα πολλά πλεονεκτήματά του Reiserfs, υπάρχουν και κάποια μειονεκτήματα στα οποία πρέπει να αναφερθούμε. Κάποια εργαλεία δημιουργίας εφεδρικών αντιγράφων ασφαλείας (dump και restore) δεν δουλεύουν με το Reiserfs, λόγω της ύπαρξης κάποιας ασυμβατότητας που πηγάζει από την πρωτοποριακή σχεδίασή του. Για τον ίδιο λόγο, έχουν παρατηρηθεί μικροπροβλήματα στη λειτουργία λίγων μεμονομένων προγράμματων. Επίσης, παρά την ασύγκριτη απόδοση του Reiserfs στα μικρά και στα πολύ μεγάλα αρχεία, η ταχύτητα με την οποία χειρίζεται τα σποραδικά αρχεία (sparse files) είναι πολύ απογοητευτική σε σχέση με άλλα συστήματα αρχείων. Τέλος, κάποιες εκδόσεις του Reiserfs, εμφάνισαν προβλήματα αστάθειας με ορισμένες εκδόσεις του πυρήνα του Linux.

Φυσικά όμως, το Reiserfs είναι ένα ζωντανό project που αναβαθμίζεται και ανανεώνεται συνεχώς και αντιμετωπίζει αργά ή γρήγορα κάθε πρόβλημα που παρουσιάζεται. Δείχνοντας κανείς κάποια προσοχή, περιορίζει τα προβλήματα στο ελάχιστο.

Το πιο εντυπωσιακό για το Reiserfs είναι το πως θα εξελιχθεί στο μέλλον. Η πολιτική του δημιουργού του, Hans Reiser είναι πολύ επιθετική και με καινοτομίες και έχει σχέδια να μετατρέψει το Reiserfs στο μέλλον σε μια πλήρη υψηλής απόδοσης βάση δεδομένων που θα υποστηρίζει συναλλαγές (**transactions**), δυνατότητα ερωτήσεων προς τη βάση (**queries**) και άλλα. Από όλα αυτά βλέπουμε ότι το Reiserfs δεν είναι

απλά ακόμα ένα σύστημα αρχείων υψηλής απόδοσης, αντίθετα δημιουργεί καινούριες πρακτικές και ανοίγει το δρόμο σε πρωτοποριακές προσεγγίσεις επίλυσης παραδοσιακών προβλημάτων αποθήκευσης με το σύστημα αρχείων.

2.6 Το σύστημα αρχείων JFS

Το JFS (Journaling File System) βασίζεται στο ομώνυμο επιτυχημένο σύστημα αρχείων της IBM για το λειτουργικό σύστημα OS2 / Warp. Στις αρχές του 2000, ο κώδικας του JFS δόθηκε στην κοινότητα του ανοιχτού λογισμικού από την IBM. Σύντομα, κυκλοφόρησε και η πρώτη έκδοσή του για το λειτουργικό Linux.

Το JFS χρησιμοποιεί πολλές εξελιγμένες τεχνικές για να επιτύχει υψηλή απόδοση, να υποστηρίξει πολύ μεγάλα αρχεία και εξίσου μεγάλα μεγέθη συστήματος αρχείων, και φυσικά υποστηρίζει journaling.

2.6.1 Δομή του συστήματος αρχείων JFS

Το πρόγραμμα που δημιουργεί το σύστημα αρχείων του JFS φτιάχνει στο φυσικό μέσο ένα σύνολο (**aggregate**). Το aggregate είναι ένας πίνακας από μπλοκ του φυσικού μέσου και περιέχει μια συγκεκριμένη διάταξη που συμπεριλαμβάνει το υπερμπλόκ και έναν πίνακα κατανομής. Το υπερμπλόκ προσδιορίζει το σύστημα αρχείων ως ένα aggregate του JFS, ενώ ο πίνακας κατανομής περιγράφει την κατάσταση όσον αφορά τη χρήση κάθε μπλοκ δεδομένων στο aggregate. Η διάταξη αυτή περιλαμβάνει επίσης το **σύνολο αρχείων** (**fileset**) και τις απαραίτητες πληροφορίες που το περιγράφουν.

Το σύνολο αρχείων περιλαμβάνει αρχεία και καταλόγους. Αρχεία και κατάλογοι περιγράφονται πλήρως από κόμβους-δείκτες. Κάθε κόμβος-δείκτης περιγράφει τα χαρακτηριστικά του αρχείου ή καταλόγου και χρησιμεύει ως το αρχικό σημείο για την εύρεσή του στο φυσικό μέσο. Το JFS χρησιμοποιεί επίσης κόμβους-δείκτες για να περιγράψει και άλλα αντικείμενα του συστήματος αρχείων, όπως το χάρτη που αναπαριστά την κατάσταση όσον αφορά τη δέσμευση και την θέση κάθε κόμβου-δείκτη στο σύνολο αρχείων.

Οι κατάλογοι κωδικοποιούν τα ονόματα των αρχείων που δίνει ο χρήστης σε κόμβους-δείκτες και κατάλογους και σχηματίζουν την παραδοσιακή ιεραρχική δομή ονομάτων του συστήματος αρχείων. Τα αρχεία περιέχουν δεδομένα του χρήστη και δεν υπάρχουν διακρίσεις ως προς τη μορφή των δεδομένων. Δηλαδή, στο JFS, τα αρχεία αντιμετωπίζονται ως ένα **μη ερμηνευόμενο ρεύμα δεδομένων** (data stream). Δομές διευθυνσιοδότησης που βασίζονται σε εκτάσεις (extents) δεδομένων και βρίσκονται στον κόμβο-δείκτη χρησιμοποιούνται για να κωδικοποιήσουν τα δεδομένα στο φυσικό μέσο. Το υπερμπλόκ του aggregate, ο χάρτης δέσμευσης του φυσικού μέσου, οι περιγραφείς των αρχείων (file descriptors), ο χάρτης των κόμβων-δεικτών, οι κόμβοι-δείκτες, οι κατάλογοι και οι δομές διευθυνσιοδότησης αποτελούν τις δομές ελέγχου, ή αλλιώς τα metadata του συστήματος αρχείων JFS.

2.6.2 Journaling

Τα ημερολόγια του JFS διατηρούνται σε κάθε aggregate και χρησιμοποιούνται για να καταγράφουν

πληροφορίες για τις λειτουργίες στα metadata. Το ημερολόγιο έχει μια μορφή που επίσης καθορίζεται από το πρόγραμμα δημιουργίας του συστήματος αρχείων. Το ίδιο ημερολόγιο μπορεί να χρησιμοποιείται ταυτόχρονα από περισσότερα από ένα σύνολα αρχείων μέσα στο aggregate.

Το JFS σχεδιάστηκε ώστε να έχει την τεχνική journaling ενσωματωμένη εξ αρχής, αντί με το σκεπτικό να προστεθεί αργότερα στο υπάρχον σύστημα αρχείων. Ένα σύνολο από χαρακτηριστικά κάνουν το JFS να ξεχωρίζει από τα άλλα συστήματα αρχείων.

Το JFS, αξιοποιώντας την τεχνική του journaling, παρέχει δομική συνέπεια και ικανότητα ανάνηψης, καθώς και πολύ γρηγορότερους χρόνους επανεκκίνησης από ότι τα συστήματα αρχείων που δεν υποστηρίζουν journaling. Το JFS χρησιμοποιεί τεχνικές παρμένες από τα συστήματα διαχειρίσεων βάσεων δεδομένων για την καταγραφή πληροφοριών σχετικά με λειτουργίες που συμβαίνουν στα metadata του συστήματος αρχείων ως ατομικές συναλλαγές (**atomic transactions**). Σε περίπτωση κατάρρευσης του λειτουργικού συστήματος, το σύστημα αρχείων αποκαθίσταται εξετάζοντας το ημερολόγιο και εφαρμόζοντας το περιεχόμενο των αλλαγών που βρίσκονται στα ημερολόγια του συστήματος. Ο χρόνος που απαιτεί η παραπάνω διαδικασία είναι πολύ μικρότερος από το χρόνο που χρειάζεται ένας πλήρης έλεγχος του συστήματος αρχείων.

Το JFS καταγράφει μόνο τις αλλαγές στα metadata και όχι στα δεδομένα αυτά καθ΄ αυτά. Αυτό σημαίνει ότι δεν μπορεί να εγγυηθεί την απόλυτη ασφάλεια των δεδομένων, παρά μόνο του συστήματος αρχείων. Η σημασιολογία της καταγραφής του JFS είναι τέτοια που μια αλλαγή στα metadata του συστήματος αρχείων είναι εγγυημένη να μη χαθεί. Για παράδειγμα, η διαγραφή ενός αρχείου που έχει γίνει επιτυχώς δεν κινδυνεύει ποτέ να χαθεί.

Ο τρόπος της καταγραφής είναι σύγχρονη εγγραφή (τη στιγμή ακριβώς που πραγματοποιείται) στο ημερολόγιο για κάθε λειτουργία κόμβου-δείκτη ή αλλαγής στα metadata. Με όρους απόδοσης, συγκρίνεται αρκετά καλά με πολλά συστήματα που δεν υποστηρίζουν journaling και στηρίζονται σε πολλαπλή και προσεκτική εγγραφή των metadata για να διατηρήσουν τη συνέχεια του συστήματος αρχείων. Είναι όμως κατώτερη από άλλα journaling συστήματα που δεν κάνουν σύγχρονη καταγραφή των αλλαγών στα metadata. Ο τρόπος καταγραφής του JFS έχει βελτιωθεί με τον χρόνο και παρέχει τώρα και ασύγχρονη καταγραφή, πράγμα που αυξάνει την απόδοση του συστήματος αρχείων.

2.6.3 Χαρακτηριστικά του συστήματος αρχείων JFS

Το JFS χρησιμοποιεί δομές που βασίζονται στις εκτάσεις (extents). Χρησιμοποιεί επίσης, ταυτόχρονα επιθετική πολιτική δέσμευσης των μπλοκ δεδομένων. Κάθε έκταση περιγράφεται μοναδικά από τρία χαρακτηριστικά της: Τη θέση από την οποία ξεκινά στο αρχείο, το μήκος της, και τη θέση της στο φυσικό μέσο. Η δομή που χρησιμοποιείται για για τα extents είναι το B+δέντρο, στα φύλλα του οποίου αποθηκεύονται τα παραπάνω τρία χαρακτηριστικά των extents.

Το JFS υποστηρίζει μεγέθη μπλοκ ίσα με 512 bytes, 1, 2, και 4 KB για κάθε σύστημα αρχείων, επιτρέποντας στον χρήστη να βελτιστοποιήσει την χρήση του διαθέσιμου χώρου ανάλογα με το είδος των αρχείων που εξυπηρετούν τις ανάγκες του.

Το JFS δεσμεύει χώρο για τους κόμβους-δείκτες δυναμικά ανάλογα με τη ζήτηση που παρουσιάζεται.

Υπάρχουν δυο διαφορετικές μορφές οργάνωσης καταλόγων στο JFS:

- Ο πρώτος τρόπος αφορά μικρούς καταλόγους και αποθηκεύει τα περιεχόμενα του κατάλογου εξ ολοκλήρου μέσα στον κόμβο-δείκτη του κατάλογου. Έτσι εξουδετερώνεται αφ΄ ενός η ανάγκη για ξεχωριστές λειτουργίες εγγραφής και ανάγνωσης για επιπλέον μπλοκ δεδομένων για τους κατάλογους, αφ΄ ετέρου δε η ανάγκη για δέσμευση και αποδέσμευση ξεχωριστής αποθήκευσης. Μέχρι 8 καταχωρήσεις για κάθε κατάλογο μπορούν να αποθηκευθούν απ΄ ευθείας μέσα στον κόμβο-δείκτη. Σ΄αυτές δεν περιλαμβάνονται οι μόνιμες καταχωρήσεις για τον τρέχοντα και τον γονικό κατάλογο, που φυλάσσονται ξεχωριστά.
- Ο δεύτερος τρόπος χρησιμοποιείται για μεγαλύτερους καταλόγους και αναπαριστά κάθε κατάλογο με ένα B+δέντρο. Πεδίο-κλειδί βάση του οποίου γίνεται η ταξινόμηση του δέντρου είναι το όνομα κάθε αρχείου. Η αναζήτηση, εισαγωγή και διαγραφή γίνονται αρκετά πιο γρήγορα σε σχέση με τους παραδοσιακούς τρόπους οργάνωσης καταλόγων.

Το JFS υποστηρίζει αμφότερα τα πυκνά (dense) και τα σποραδικά (sparse) αρχεία για ολόκληρο το σύστημα αρχείων.

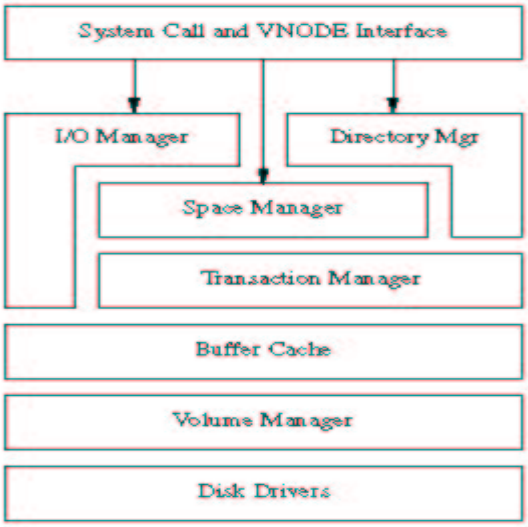
Για τα μεν αραιά αρχεία, επιτρέπεται στα δεδομένα να γράφονται σε τυχαίες θέσεις μέσα στο αρχείο χωρίς να χρειάζεται να αρχικοποιηθούν τα αχρησιμοποίητα ενδιάμεσα μπλοκ. Το αναφερόμενο μήκος του αρχείου είναι αυτό του τελευταίου χαρακτήρα που γράφτηκε, δεν γινεται όμως δέσμευση χώρου για κάθε μπλοκ παρά μόνο όταν γίνει αίτηση εγγραφής γι΄ αυτό. Για παράδειγμα, αν ένα νέο τέτοιο αρχείο δημιουργηθεί σε ένα σύστημα αρχείων με σχετική υποστήριξη, όταν μια εφαρμογή γράψει δεδομένα στο εκατοστό μπλοκ, το JFS θα αναφέρει ότι το μέγεθος του αρχείου είναι 100 μπλοκ παρ΄ όλο που μόνο ένα μπλοκ έχει δεσμευτεί γι΄ αυτό. Αν αργότερα, η εφαρμογή γράψει στο πεντηκοστό μπλοκ, το αρχείο θα καταλαμβάνει στο μέσο έκταση δύο μπλοκ δεδομένων. Και πάλι όμως, το JFS θα αναφέρει μέγεθος 100 μπλοκ. Τα αραιά αρχεία ενδιαφέρουν τις εφαρμογές εκείνες που χρειάζονται ένα πολύ μεγάλο λογικό χώρο, κάνουν όμως χρήση σε ένα μικρό υποσύνολό του.

Το JFS είναι ένα πλήρως 64-μπιτο σύστημα αρχείων. Όλες οι σχετικές δομές και πεδία ελέγχου έχουν μήκος 64 δυαδικά ψηφία. Αυτό επιτρέπει στο JFS να υποστηρίζει πάρα πολύ μεγάλα μεγέθη για τα αρχεία αλλά και για το συστήμα αρχείων. Το μικρότερο υποστηριζόμενο μέγεθος συστήματος αρχείων είναι 16 MB. Το μεγαλύτερο μέγεθος είναι συνάρτηση του μέγεθους μπλοκ και του μέγιστου αριθμού μπλοκ που υποστηρίζουν τα metadata του συστήματος αρχείων. Το JFS υποστηρίζει για μέγεθος μπλοκ 512 bytes και 4 KB μέγιστο μέγεθος συστήματος αρχείων ίσο με 512 TB και 4096 TB αντίστοιχα. Τα ίδια νούμερα ισχύουν και για το μέγιστο μέγεθος αρχείου.

2.7 Το σύστημα αρχείων XFS

Το σύστημα αρχείων XFS αναπτύσσεται από την εταιρία Silicon Graphics International και χρησιμοποιείται στο λειτουργικό της σύστημα IRIX, είναι όμως διαθέσιμο και με τη μορφή ανοιχτού λογισμικού.

2.7.1 Δομή του συστήματος αρχείων XFS



Στην παραπάνω εικόνα φαίνεται η δομή του XFS: Είναι παρόμοια με τη δομή ενός συμβατικού συστήματος αρχείων με την προσθήκη όμως ενός διαχειριστή συναλλαγών (Transaction manager) και ενός διαχειριστή τόμων (Volume manager).

Το XFS είναι χωρισμένο σε διάφορα τμήματα, καθένα από τα οποία είναι υπεύθυνο για ένα ξεχωριστό κομμάτι της λειτουργικότητας του συστήματος αρχείων.

- Το κεντρικό και πιο σημαντικό κομμάτι είναι ο διαχειριστής χώρου (Space Manager). Το τμήμα αυτό διαχειρίζεται τον ελεύθερο χώρο του συστήματος αρχείων, τη δέσμευση των κόμβων-δεικτών, καθώς και την κατανομή του χώρου στα αρχεία.
- Ο διαχειριστής εισόδου/εξόδου (I/O Manager) είναι υπεύθυνος για την ικανοποίηση των αιτήσεων εισόδου/εξόδου και εξαρτάται από τον διαχειριστή χώρου για τη δέσμευση και την καταγραφή του χώρου των αρχείων.
- Ο διαχειριστής καταλόγων (Directory Manager) υλοποιεί το χώρο ονομάτων (namespace) του συστήματος αρχείων XFS.
- Η **buffer cache** χρησιμοποιείται από όλα τα παραπάνω τμήματα για να τοποθετήσει στη μνήμη cache τα περιεχόμενά εκείνων των μπλοκ του φυσικού μέσου που προσπελαύνονται πιο συχνά.
- Ο **διαχειριστής συναλλαγών** (Transaction Manager) χρησιμοποιείται από τα άλλα τμήματα για να ενημερώσει στα metadata τις τροποποιήσεις από τις συναλλαγές του συστήματος αρχείων. Αυτό οδηγεί σε γρήγορη ανάνηψη ύστερα από κατάρρευση του λειτουργικού.

2.7.2 Ομάδες κατανομής (Allocation Groups)

Το σύστημα αρχείων XFS είναι χωρισμένο σε περιοχές που καλούνται ομάδες κατανομής. Οι ομάδες

κατανομής κρατούν το μέγεθος των δομών του XFS μικρό ώστε να μπορούν να λειτουργούν αποδοτικά. Οι ομάδες κατανομής έχουν τυπικά μεγέθη από 500 MB μέχρι 4 GB. Κάθε ομάδα έχει τις δικές τις ξεχωριστές ομάδες δομών για να διαχειρίζεται τον ελεύθερο χώρο και τους κόμβους-δείκτες της. Με το χωρισμό του συστήματος αρχείων σε ομάδες μειώνεται το μέγεθος των δομών ελέγχου. Επιπλέον, γίνεται δυνατή η χρήση δεικτών με σχετική αναφορα΄στις δομές, πράγμα που περιορίζει το μέγεθός τους.

Το XFS χρησιμοποιεί τις ομάδες κατανομής για να πετυχαίνει την τοπικότητα των καταλόγων και των αρχείων τους. Κάθε φορά που δημιουργείται ένας νέος κατάλογος, τοποθετείται σε διαφορετική ομάδα κατανομής. Από τη στιγμή που ένας κατάλογος έχει δημιουργηθεί, γίνεται προσπάθεια να κρατηθούν οι κόμβοι-δείκτες και τα μπλοκ των αρχείων του στην ίδια ομάδα κατανομής. Για μεγάλα αρχεία, καταβάλλεται προσπάθεια να δεσμευονται εκτάσεις (extents) δεδομένων αρχικά κοντά στον κόμβο-δείκτη και αργότερα κοντά στο μπλοκ που είναι πιο κοντά στο προς δέσμευση μπλοκ. Τα αρχεία και οι κατάλογοι, βέβαια, δεν είναι περιορισμένα να βρίσκονται εξ ολοκλήρου σε μόνο μια ομάδα κατανομής.

Ο άλλος λόγος για τον οποίο υπάρχουν οι ομάδες κατανομής είναι η παραλληλοποίηση της διαχείρισης της δέσμευσης των μπλοκ και κόμβων-δεικτών. Σε μεγάλα συστήματα με πολλά αρχεία, και καταλόγους, το να τρέχει μόνο μια διεργασία που τα χειρίζεται, οδηγεί συχνά σε συνωστισμό (bottleneck). Με το να γίνονται οι δομές κάθε ομάδας ανεξάρτητες, το XFS επιτρέπει το να τρέχουν παράλληλα οι διεργασίες για όλες τις ομάδες κατανομής.

2.7.2 Χαρακτηριστικά του συστήματος αρχείων XFS

Η διαχείριση του ελεύθερου χώρου είναι πολύ σημαντικό χαρακτηριστικό για την απόδοση κάθε συστήματος αρχείων. Η αποδοτική κατανομή και απελευθέρωση των μπλοκ δεδομένων και η αποτροπή του συστήματος αρχείων από τον κατακερματισμό είναι ουσιώδη για την καλή απόδοσή του. Το XFS αντικαθιστά τη διαχείριση με βάση τα μπλοκ δεδομένων με μια δομή που στηρίζεται στις εκτάσεις (extents) και υλοποιείται με ένα ζεύγος B+δέντρων για κάθε ομάδα κατανομής. Κάθε καταχώρηση του δέντρου είναι περιγραφή για ελεύθερες εκτάσεις στην ομάδα κατανομής. Κάθε τέτοια περιγραφή περιλαμβάνει το αρχικό μπλοκ της έκτασης και το μήκος της σε μπλοκ. Το ένα από τα δύο δέντρα είναι ταξινομημένο ως προς το αρχικό μπλοκ ενώ το άλλο ως προς το μήκος. Τα δύο αυτά δέντρα επιτρέπουν γρήγορη και πολύ αποτελεσματική αναζήτηση για ελεύθερες εκτάσεις.

Το XFS υποστηρίζει σποραδικά (sparse) αρχεία. Η σχετική υποστήριξη σημαίνει ότι κάθε αρχείο μπορεί να έχει σημεία για τα οποία δεν έχει δεσμευτεί κανένα μπλοκ δεδομένων. Αυτό σημαίνει ότι πρέπει να διατηρηθεί ένα ευρετήριο με τα σημεία του αρχείου για τα οποία έχει όντως δεσμευτεί χώρος. Η υποστήριξη για τα 64 bit που διαθέτει το XFS σημαίνει ότι μπορεί και υποστηρίζει πολύ μεγάλο αριθμό μπλοκ σε κάθε αρχείο. Για να διατηρήσει μικρότερο τον αριθμό των στοιχείων για τα οποία πρέπει να δημιουργηθεί ευρετήριο για κάθε αραιό αρχείο, το XFS χρησιμοποιεί χάρτη εκτάσεων και όχι μπλοκ για να καταγράψει τα περιεχόμενά του. Ο χάρτης αυτός συμπιέζεται για ακόμα μεγαλύτερη οικονομία χώρου.

Εκτός από πολύ μεγάλα αρχεία, το XFS υποστηρίζει και πολύ μεγάλο αριθμό αρχείων. Ο αριθμός αυτός περιορίζεται μόνο από το μέγεθος του συστήματος αρχείων. Αντί να δεσμεύει αρχικά σταθερό αριθμό από κόμβους-δείκτες, το XFS τους δημιουργεί δυναμικά ανάλογα με τις ανάγκες που θα προκύψουν. Σε κάθε ομάδα κατανομής, οι κόμβοι-δείκτες που έχουν δημιουργηθεί φυλάσσονται σε ένα B+δέντρο.

Το XFS υποστηρίζει επίσης πολύ αποδοτικά απεριόριστο αριθμό αρχείων σε κάθε κατάλογο. Οι

καταχωρήσεις για τα αρχεία κάθε καταλόγου φυλάσσονται κι αυτές σε ένα B+δέντρο ταξινομημένα με βάση το όνομά τους.

Συστήματα αρχείων του μεγέθους και της πολυπλοκότητας του XFS θα αργούσαν πάρα πολύ να ανανήψουν μετά από κατάρρευση του συστήματος. Για να αποφύγει τέτοια προβλήματα, το XFS χρησιμοποιεί μια τεχνική παρόμοια με του journaling, με την οποία καταγράφει τις αλλαγές των metadata σε ένα ημερολόγιο

Τεχνικές που χρησιμοποιεί το XFS για να αυξήσει την απόδοσή του είναι:

- Το XFS προσπαθεί να δεσμεύσει συνεχόμενα μπλοκ δεδομένων για κάθε αρχείο, ώστε να αποφεύγεται ο κατακερματισμός και οι περιττές αναζητήσεις στο φυσικό μέσο.
- Επίσης, κατά την εγγραφή, το XFS καθυστερεί να υπολογίσει το μέγεθος κάθε έκτασης που δεσμεύει, και δεν το κάνει παρά μόνο όταν η εγγραφή έχει τελειώσει. Με αυτή την τεχνική, συνήθως όλόκληρο το αρχείο αποθηκεύεται σε μία και μόνο έκταση.
- Για τους ίδιους λόγους, το XFS υποστηρίζει πολύ μεγάλα μεγέθη έκτασης, ώστε να μπορεί να συμπεριλάβει κάθε αρχείο σε μια μόνο έκταση ανεξαρτήτως του μεγέθους του.
- Το XFS υποστηρίζει πολλά διαφορετικά μεγέθη μπλοκ δεδομένων, αρχίζοντας από 512 bytes και φτάνοντας μέχρι και 64 KB, ώστε να μπορεί να ικανοποιήσει τις ανάγκες κάθε χρήσης.

2.8 Το πρωτόκολλο NFS (Network File System)

2.8.1 Εισαγωγή

Όταν, κάποτε, η δικτύωση άρχισε να γίνεται δημοφιλής σε παλιότερα συστήματα που έτρεχαν λειτουργικά του τύπου unix, όλοι οι χρήστες που ήθελαν να διαμοιράζονται αρχεία μεταξύ τους έπρεπε να κάνουν login μέσω του δικτύου σε έναν κεντρικό υπολογιστή στον οποίο βρίσκονταν τα διαμοιραζόμενα αρχεία.

Στους κεντρικούς αυτούς υπολογιστές πολύ γρήγορα συνέβη το φορτίο να είναι πολύ μεγαλύτερο από τον υπολογιστή του κάθε χρήστη και παρατηρούνταν το φαινόμενο του συνωστισμού (bottleneck) κατά το οποίο πάρα πολλοί χρήστες αιτούνταν εξυπηρέτησης από έναν υπολογιστή, χωρίς να ήταν σε θέση αυτός να τους εξυπηρετήσει. Έτσι η ανάγκη για έναν εύκολο και γρήγορο τρόπο να διαμοιράζονται αρχεία μεταξύ πολλών υπολογιστών έγινε γρήγορα φανερή.

Ο πιο εύκολος στη σύλληψη τρόπος για να γίνει αυτό, είναι αυτός κατά τον οποίο ένας διακομιστής εξαγάγει (export) τα συστήματα αρχείων του σε έναν ή περισσότερους υπολογιστές-πελάτες. Οι τελευταίοι εισάγουν (import) τα συστήματα αρχείων και τα παρουσιάζουν στον χρήστη σαν να βρίσκονταν τοπικά στον ίδιο υπολογιστή.

2.8.2 Σχεδίαση του πρωτοκόλλου NFS

Το περισσότερο εμπορικά επιτυχημένο και ευρέως διαδεδομένο πρωτόκολλο απομακρυσμένου συστήματος αρχείων είναι το NFS, που σχεδιάστηκε και υλοποίηθηκε από την εταιρεία Sun Microsystems. Δύο είναι τα στοιχεία που συνέβαλαν στην επιτυχία του. Το ότι η Sun δημοσίευσε τις προδιαγραφές (specifications) του NFS, και το ότι διέθετε την υλοποίησή του σε όποιον τη χρειαζόταν σε τιμή κατώτερη από αυτή που θα κόστιζε στον πελάτη να υλοποιήσει μόνος του ένα δικό του πρωτόκολλο. Έτσι, πολλοί πελάτες στράφηκαν προς το πρωτόκολλο της Sun.

Το NFS σχεδιάστηκε ως μια εφαρμογή που υπακούει στο μοντέλο της σχέσης πελάτη-διακομιστή. Η υλοποίησή του χωρίζεται στο κομμάτι του πελάτη που εισάγει συστήματα αρχείων από το διακομιστή και στο κομμάτι του διακομιστή, που εξάγει τοπικά συστήματα αρχείων σε άλλους υπολογιστές.

Η σχεδίαση του NFS στοχεύει σε πολλά πράγματα:

- Το πρωτόκολλο του NFS σχεδιάστηκε να είναι **χωρίς κατάσταση** (**stateless**). Επειδή ακριβώς δεν υπάρχει κατάσταση που να πρέπει να διατηρηθεί ή να αποκατασταθεί, το πρωτόκολλο είναι ικανό να συνεχίζει να λειτουργεί ακόμα και κατά τη διάρκεια αστοχίας του πελάτη ή του διακομιστή. Είναι δηλαδή πολύ πιο ευέλικτο από ένα σύστημα που λειτουργεί με καταστάσεις.

- Το NFS σχεδιάστηκε ώστε να υποστηρίζει τη σημασιολογία των συστημάτων αρχείων του UNIX. Παρ΄ όλα αυτά, η σχεδίασή του του επιτρέπει να υποστηρίζει τη λιγότερο πλούσια σημασιολογία άλλων τύπων συστημάτων αρχείων, όπως του MSDOS.

- Η προστασία και ο έλεγχος της πρόσβασης ακολουθούν τη σημασιολογία του UNIX. Δηλαδή κάθε χρήστης έχει έναν κωδικό και ανήκει σε ένα σύνολο από ομάδες. Αυτά τα στοιχεία συγκρίνονται με τον ιδιοκτήτη του αρχείου, την ομάδα στην οποία ανήκει, και τα δικαιώματα που δίνει στον ιδιοκτήτη του, στην ομάδα του και σ΄ όλους τους υπόλοιπους. Ο έλεγχος ασφαλείας γίνεται με βάση την υλοποίηση του κάθε σύστηματος, η οποία μπορεί να είναι ικανή να κάνει περισσότερους ή λιγότερους ελέγχους, ανάλογα με τις δυνατότητες του συστήματος αρχείων που υποστηρίζει. Για παράδειγμα, το σύστημα αρχείων του MSDOS δεν μπορεί να υλοποιήσει τον πλήρη έλεγχο ασφαλείας του UNIX και παίρνει τις αποφάσεις του σχετικά με την παραχώρηση πρόσβασης βασιζόμενο στον κωδικό του ιδιοκτήτη και μόνο.

- Το πρωτόκολλο σχεδίασης είναι ανεξάρτητο από τον τρόπο μεταφοράς. Παρ΄ όλο που σχεδιάστηκε για να λειτουργεί κάνοντας χρήση του πρωτοκόλλου UDP, εύκολα μεταφέρθηκε και στο πρωτόκολλο TCP, αλλά και σε άλλα πρωτόκολλα.

Κάποιες από τις αποφάσεις σχεδίασης περιορίζουν το σύνολο των εφαρμογών για τις οποίες το NFS είναι κατάλληλο για χρήση:

- Η σχεδίαση έγινε με τη σκέψη ότι οι διακομιστές και οι πελάτες είναι συνδεδεμένοι σε ένα τοπικό γρήγορο δίκτυο. Το πρωτόκολλο NFS δεν λειτουργεί καλά πάνω από αργές συνδέσεις ή όταν μεταξύ πελατών και διακομιστών μεσολαβούν ενδιάμεσα δίκτυα. Επίσης, υπολειτουργεί για την ασύρματη επικοινωνία η οποία έχει μεγάλες περιόδους λειτουργίας εκτός σύνδεσης.

- Το μοντέλο που χρησιμοποιείται για caching υποθέτει ότι τα περισσότερα από τα αρχεία δεν θα διαμοιράζονται. Αν, αντίθετα, συμβαίνει να μοιράζονται πολλά αρχεία, η απόδοση είναι πολύ χαμηλή, ακριβώς γιατί θα γίνονται συνεχείς τυχαίες προσπελάσεις στο μέσο αποθήκευσης στο διακομιστή.

- Το πρωτόκολλο χωρίς κατάσταση απαιτεί την κατάργηση κάποιας παραδοσιακής σημασιολογίας του UNIX. Το κλείδωμα των αρχείων πρέπει να υλοποιηθεί από μια ξεχωριστή εφαρμογή που να τρέχει στο παρασκήνιο και να υποστηρίζει καταστάσεις.

Παρά τους παραπάνω περιορισμούς, το πρωτόκολλο NFS είναι πολύ διαδεδομένο γιατί τηρεί μια ενδιάμεση στάση μεταξύ της αυστηρής προσήλωσης στη σημασιολογία και της υψηλής απόδοσης. Το χαμηλό κόστος υιοθέτησής του το έχει κάνει πανταχού παρόν.

2.8.2 Δομή και λειτουργία του πρωτοκόλλου NFS

Το πρωτοκόλλο NFS λειτουργεί σαν ένα τυπικό μοντέλο πελάτη-διακομιστή. Ο διακομιστής λαμβάνει μια κλήση απομακρυσμένης διαδικασίας (**Remote Procedure Call**, **RPC**), Μια κλήση απομακρυσμένης διαδικασίας λειτουργεί σχεδόν όπως και μια κλήση τοπικής διαδικασίας. Ο πελάτης κάνει την κλήση και μετά περιμένει για την απάντηση όσο η διαδικασία εκτελείται απομακρυσμένα Για μια απομακρυσμένη κλήση πρέπει για τις παράμετρους, μεταξύ άλλων να αντικατασταθούν τυχόν δείκτες με τα δεδομένα στα οποία δείχνουν, και τυχόν δυαδικά δεδομένα να μετατραπούν σε μια κανονική σειρά ψηφίων ως προς το πιο σημαντικό. Το μήνυμα το λαμβάνει ο διακομιστής, ο οπόιος αφού ανασυνθέσει τις παραμέτρους ξαναμετατρέπει στην τοπική μορφή τα δεδομένα και επεξεργάζεται την αίτηση. Κατόπιν, στέλνει το αποτέλεσμα ξανά κωδικοποιημένο. Ο πελάτης λαμβάνει την απάντηση και χειρίζεται το αποτέλεσμα ακριβώς όπως θα έκανε αν επρόκειτο για μια τοπική κλήση διαδικασίας.

Όταν ο διακομιστής λαμβάνει μια εντολή, απαντάει είτε με το αποτέλεσμα της επεξεργασίας της ή με έναν κωδικό λάθους που εξηγεί γιατί η εκτέλεση δεν είναι δυνατή. Πολλες από τις εντολές του NFS είναι **idempotent**, δηλαδή μπορούν να επαναληφτούν πολλές φορές χωρίς να αλλάξει το τελικό αποτέλεσμα. Για παράδειγμα, η ανάγνωση ή η εγγραφή πολλές φορές του ίδιου μπλοκ ενός αρχείου είναι idempotent εντολή, ενώ η διαγραφή αρχείου δεν είναι, οι επόμενες εντολές θα επιστρέψουν λάθος, μη βρίσκοντας το ήδη διαγραμμένο αρχείο.

Κάθε αρχείο στο διακομιστή περιγράφεται μοναδικά από τον **χειριστή αρχείου** (**file handle**) του, με τον οποίο ο διακομιστής αναφέρεται στο αρχείο. Ο χειριστής αυτός είναι κοινός σε όλο το δίκτυο και χρησιμοποιείται στις λειτουργίες εισόδου/εξόδου για να προσδιορίσει το αρχείο στο οποίο θα γίνει η λειτουργία. Για τη δημιουργία μοναδικών χειριστών για κάθε αρχείο χρησιμοποιούνται τα ονόματα και οι

διαδρομές των αρχείων, αλλά και άλλες πληροφορίες, όπως για παράδειγμα ο αριθμός του κόμβου-δείκτη τους.

Όπως ήδη αναφέρθηκε, το πρωτόκολλο NFS είναι χωρίς κατάσταση (stateless). Αυτό σημαίνει ότι ο διακομιστής δεν χρειάζεται να έχει καμιά πληροφορία για το ποιον πελάτη εξυπηρετεί ή για το ποια αρχεία έχει ήδη ανοίξει ο πελάτης. Κάθε κλήση απομακρυσμένης διαδικασίας περιγράφει πλήρως την ενέργεια της οποίας αιτείται και ο διακομιστής δεν χρειάζεται καμιά άλλη πληροφορία για να την εκτελέσει. Το μεγαλύτερο πλεονέκτημα του πρωτοκόλλου χωρίς κατάσταση είναι ότι σε περίπτωση αστοχίας του διακομιστή, μόλις αυτός επανέλθει είναι αμέσως σε θέση να επεξεργαστεί αιτήσεις.

Ο πελάτης-υπολογιστής του NFS στον οποίο τρέχουν εφαρμογές οι οποιες εργάζονται με αρχεία που βρίσκονται στο διακομιστή πρέπει να ξέρει πως θα ενεργήσει σε περίπτωση που ο διακομιστής πάψει να αποκρίνεται. Υπάρχουν τρεις διαφορετικές προσεγγίσεις γι΄ αυτό:

- Μια προσέγγιση είναι να συνεχίσει ο πελάτης επ΄ άπειρον να προσπαθεί να επικοινωνήσει με το διακομιστή. Η προσέγγιση αυτή χρησιμοποιείται για εφαρμογές στις οποίες τα δεδομένα μεγάλης σημασίας για το σύστημα ή απλά δεν μπορούμε οποιοδήποτε λάθος στα δεδομένα.

- Στο ακριβώς αντίθετο άκρο είναι η προσέγγιση κατά την οποία γίνονται μερικές προσπάθειες επικοινωνίας με το διακομιστή και μετά επιστρέφεται λάθος στην εφαρμογή που έκανε την αίτηση. Το πρόβλημα που προκύπτει εδώ είναι ότι πολλές εφαρμογές αντιλαμβάνονται εσφαλμένα το λάθος αυτό και τερματίζουν αντικανονικά νομίζοντας ότι υπάρχει μόνιμο λάθος στο σύστημα. Ένα πρόσθετο πρόβλημα που προκύπτει είναι ότι είναι δύσκολος ο καθορισμός του ορίου πέρα από το οποίο θα σταματήσουν οι απόπειρες επικοινωνίας με το διακομιστή. Μικρή τιμή σημαίνει ότι πολλές εφαρμογές θα τερματίζουν όταν συμβαίνει κάποια καθυστέρηση στο δίκτυο. Μεγάλη τιμή σημαίνει ότι οι εφαρμογές θα καθυστερούν πολλή ώρα να τερματίσουν σε περίπτωση που ο διακομιστής δεν είναι διαθέσιμος.

- Η ενδιάμεση λύση είναι να προσπαθούν οι εφαρμογές να επικοινωνήσουν με το διακομιστή απ΄ άπειρον, να έλέγχουν όμως ταυτόχρονα για τη λήψη σημάτων τερματισμού και αν αυτά τους αποσταλούν, να τερματίζουν.

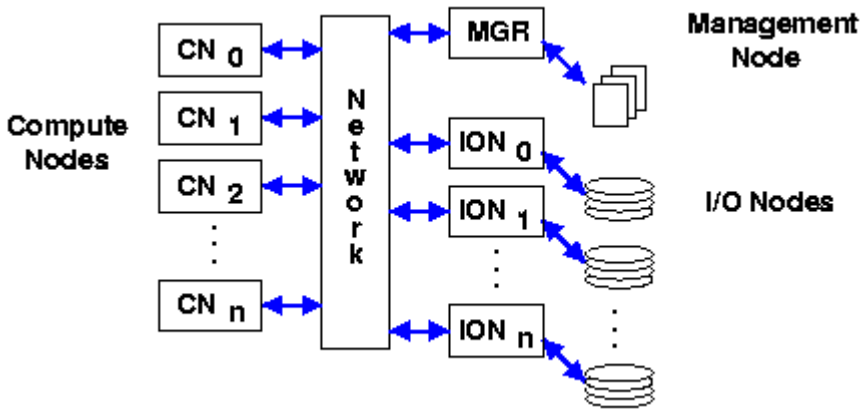
Το NFS δεν είναι ένα ασφαλές πρωτόκολλο, αφού δε σχεδιάστηκε με γνώμονα την ασφάλεια. Βασικός άξονας των όποιων ελέγχων ασφαλείας κανει το NFS είναι ο έλεγχος των εξαγωγών. Για κάθε σύστημα αρχείων που εξάγεται, μπορεί να ρυθμιστούν ως πιθανοί πελάτες είτε συγκεκριμένοι υπολογιστές, είτε σύνολα διευθύνσεων, είτε ολόκληρα δίκτυα. Μπορούν επίσης να ρυθμιστούν διευθύνσεις ή σύνολα διευθύνσεων για τις οποίες θα αρνείται η εξαγωγή.

2.9 Το σύστημα αρχείων PVFS (Parallel Virtual File System)

Τα τελευταία χρόνια, η αύξηση της υπολογιστικής ισχύος είναι πολύ πιο γρήγορη από αυτή των επιδόσεων των συσκευών εισόδου/εξόδου. Αυτό έχει οδηγήσει στο να περιορίζεται σημαντικά η ταχύτητα πολλών εφαρμογών, ειδικά αυτών που διαχειρίζονται μεγάλο αριθμό δεδομένων. Ένας δημοφιλής τρόπος να παρακαμφτεί αυτή η δυσκολία είναι με τη χρήση παράλληλων συστημάτων αρχείων.

Στόχος του PVFS είναι να εξερευνήσει τη σχεδίαση, υλοποίηση και χρήση των παράλληλων λειτουργιών εισόδου/εξόδου και να προσφέρει ένα υψηλής απόδοσης παράλληλο σύστημα αρχείων. Εκτός από κατάλληλη πλατφόρμα για έρευνα πάνω σ΄ αυτό τον τομέα, το PVFS είναι και ένα χρήσιμο εργαλείο για χρήση σε συστοιχίες υπολογιστών.

2.6.1 Δομή του PVFS



Η παραπάνω εικόνα δείχνει το σχήμα λειτουργίας του PVFS. Οι κόμβοι του συστήματος χωρίζονται σε:

- Υπολογιστικούς κόμβους (Compute Nodes) στους οποίους τρέχουν οι εφαρμογές που χρησιμοποιούν τα αρχεία που διαχειρίζεται το PVFS.
- Έναν κόμβο διαχείρισης (Managemente Node), ο οποίος αναλαμβάνει να χειρίζεται τα metadata των αρχείων του PVFS, δηλαδή τις πληροφορίες που αφορούν τα δεδομένα, όχι όμως τα δεδομένα αυτά καθ΄ εαυτά.
- Κόμβους εισόδου/εξόδου (Input / Output Nodes), στους οποίους διαμοιράζονται τα ίδια τα δεδομένα του PVFS.

Ο κόμβος διαχείρισης και οι κόμβοι εισόδου/εξόδου μπορούν να χρησιμοποιηθούν και ως κόμβοι υπολογισμού, ανάλογα με την τοπολογία που επιθυμεί ο διαχειριστής του συστήματος. Για μικρά δίκτυα, πολλές φορές είναι συμφέρον να επικαλύπτονται ώστε να αξιοποιούνται πλήρως όλοι οι πόροι του συστήματος. Αντίθετα σε μεγάλες συστοιχίες, για τους κόμβους εισόδου/εξόδου και διαχείρισης αφιερώνονται ξεχωριστοί και αποκλειστικοί πόροι.

2.9.2 Συστατικά του συστήματος αρχείων PVFS

Υπάρχουν τέσσερα συστατικά (components) του PVFS:

- Διακομιστής metadata.
- Διακομιστής εισόδου/εξόδου.
- Βιβλιοθήκη τοπικών διαδικασίων (API)
- Υποστήριξη για τον πυρήνα του λειτουγικού συστήματος.

Τα πρώτα δύο συστατικά είναι εφαρμογές που τρέχουν στο παρασκήνιο (daemons) και φροντίζουν τη διαχείριση των metadata και των ίδιων των δεδομένων αντίστοιχα. Ο διακομιστής metadata τρέχει, προφανώς, στον κόμβο διαχείρισης και διαχειρίζεται τα metadata, δηλαδή τα χαρακτηριστικά των αρχείων (όνομα, θέση στο σύστημα αρχείων, ιδιότητες και άλλα), καθώς και το πως τα δεδομένα των αρχείων διαμοιράζονται στους κόμβους εισόδου/εξόδου κατά μήκος του συστήματος.

Ο δεύτερος, ο διακομιστής εισόδου/εξόδου τρέχει σε κάθε κόμβο εισόδου/εξόδου και διαχειρίζεται την αποθήκευση, ανάκτηση και τροποποίηση των δεδομένων των αρχείων που είναι αποθηκευμένα στον ίδιο κόμβο. Ο διακομιστής αυτός λειτουργεί ανεξάρτητα από το σύστημα αρχείων που είναι εγκατεστημένο στον κόμβο.

2.9.3 Χαρακτηριστικά του συστήματος αρχείων PVFS

Χαρακτηριστικά που προσφέρει το PVFS είναι η συμβατότητα με τις υπάρχουσες εφαρμογές, ευκολία στην εγκατάσταση και υποστήριξη πολλαπλών διαφορετικών διασυνδέσεων με το χρήστη. Το PVFS δεν απαιτεί εξειδικευμένο υλισμικό (hardware).

Το PVFS προσφέρει ένα καθολικό χώρο ονομάτων (namespace), ίδιο για όλα τα μέλη της συστοιχίας, προσβάσιμο με απόλυτη διαφάνεια από όλες τις εφαρμογές. Το PVFS μπορεί να εγκατασταθεί ταυτόχρονα σε όλους τους κόμβους της συστοιχίας επιτρέποντας στον καθένα να έχει πρόσβαση σε όλα τα αρχεία του PVFS μέσω του ίδιου σχήματος καταλόγων. Από τη στιγμή που θα εγκατασταθεί το PVFS, συνεργάζεται απόλυτα με όλες τις προϋπάρχοντες εφαρμογές χωρίς την ανάγκη εγκατάστασης κι άλλου λογισμικού.

Για να προσφέρει υψηλής ταχύτητας πρόσβαση στα δεδομένα που αποθηκεύονται στο σύστημα αρχείων από πολλούς πελάτες, το PVFS διαμοιράζει τα δεδομένα κατα μήκος πολλών υπολογιστών, οι οποίοι ονομάζονται **κόμβοι εισόδου/εξόδου** (**I/O nodes**). Με αυτή την τακτική διαμοιρασμού των δεδομένων διαμοιράζεται και το φορτίο ανάμεσα σε πολλούς κόμβους κι έτσι εξομαλύνονται τα φαινόμενα συνωστισμού (**bottleneck**) που εμφανίζονται όταν πολλοί πελάτες προσπαθούν ταυτόχρονα να ζητήσουν λειτουργίες εισόδου/εξόδου από ένα διακομιστή. Επιπλέον, αυξάνεται και η ταχύτητα με την οποία διακινούνται τα

δεδομένα λόγω της παράλληλης λειτουργίας των αποθηκευτικών μέσων.

Ο παραδοσιακός μηχανισμός των κλήσεων συστήματος για την πρόσβαση σε αρχεία είναι βολικός στη χρήση και επιτρέπει στις εφαρμογές να διαχειρίζονται με τον ίδιο τρόπο τα δεδομένα ανεξερτήτως του τύπου του συστήματος αρχείων. Ο τρόπος αυτός όμως παρουσιάζει μια καθυστέρηση γιατί η πρόσβαση γίνεται μέσω του πυρήνα του λειτουργικού συστήματος. Το PVFS κατορθώνει να ξεπερνάει το εμπόδιο αυτό. Το κατορθώνει δημιουργώντας τις δικές του τοπικές κλήσεις οι οποίες διαχειρίζονται τα συστήματα αρχείων προσπερνώντας τον πυρήνα.

2.10 Το σύστημα αρχείων ISO9660

Το ISO9660 είναι ένα σύστημα αρχείων που χρησιμοποιείται στους οπτικούς δίσκους CD-ROM. Πήρε το όνομά του από το πρότυπο ISO από το οποίο καθορίζεται.

Το ISO9660 θεσπίστηκε ως κοινό σημείο αναφοράς για να μπορούν όλα τα διαφορετικά λειτουργικά συστήματα να μοιράζονται δεδομένα μέσω των δίσκων CD-ROM.

2.10.1 Χαρακτηριστικά του συστήματος αρχείων ISO9660

Το πρότυπο καθορίζει τρία επίπεδα. Το επίπεδο 1 έχει πλήρη συμβατότητα με τους περιορισμούς στην ονοματολογία των αρχείων του FAT επιτρέποντας μόνο 8 χαρακτήρες για το όνομα και 3 για την επέκταση των αρχείων. Στα ονόματα των αρχείων επιτρέπονται μόνο κεφαλαία και μόνο αριθμητικοί χαρακτήρες μαζί με γράμματα του λατινικού αλφαβήτου και ο χαρακτήρας "_". Τα επίπεδα 2 και 3 προσφέρουν μέχρι και 31 χαρακτήρες για το όνομα. Στην πράξη, στα περισσότερα CD-ROM που κυκλοφορούν στο εμπόριο χρησιμοποιείται το επίπεδο 1.

Άλλα χαρακτηριστικά το πρότυπου είναι:

- Τα αρχεία πρέπει να είναι απαραίτητα συνεχή και όχι κατακερματισμένα.
- Επιτρέπεται μέγιστο βάθος καταλόγων ίσο με 8 συμεριλαμβανομένου του ριζικού κατάλογου.
- Οι κατάλογοι δεν επιτρέπεται να έχουν επέκταση.
- Οι καταχωρήσεις των καταλόγων είναι πάντα ταξινομημένες κατ΄ όνομα σύμφωνα με τη σειρά των χαρακτήρων ASCII.
- Οι πληροφορίες στα CD-ROM είναι γραμμένες σε τομείς (scctors) που αριθμούνται σειριακά. Δεν επιτρέπονται κενά στην αρίθμηση. Το μέγεθος του κάθε τομέα συνήθως είναι 2048 χαρακτήρες των 8 δυαδικών ψηφίων. Το πρότυπο επιτρέπει κι άλλα μεγέθη τομέα, τα οποία όμως συνήθως δε

χρησιμοποιούνται.

- Οι πρώτοι 16 τομείς (αριθμοί 0 ως 15) είναι κενοί. Οι θέσεις αυτές είναι δεσμευμένες για τον κώδικα εκκίνησης του λειτουργικού στα CD-ROM που γράφονται με την ικανότητα να εκκινούν τον υπολογιστή.

Όταν ένας αριθμός από τομείς πρέπει να διαβαστούν από το δισκάκι CD-ROM, η ανάγνωση πρέπει να γίνει με αύξουσα αριθμητική σειρά, αν αυτό είναι δυνατόν, αφού αυτή είναι η σειρά με την οποία περνούν από την κεφαλή ανάγνωσης καθώς το δισκάκι περιστρέφεται.

2.10.2 Επεκτάσεις του συστήματος αρχείων ISO9660

Υπάρχουν διάφορες επεκτάσεις του ISO9660 που χρησιμοποιούνται για να αρθούν κάποιοι από τους περιορισμούς του. Οι επεκτάσεις αυτές δεν αλλοιώνουν αναγκαστικά το πρότυπο, τα δισκάκια που είναι γραμμένα με αυτές συνήθως διαβάζονται κανονικά από όλους και όποιο λογισμικό δεν τις αναγνωρίζει, απλά τις αγνοεί μένοντας στο βασικό πρότυπο.

- **Joliet** : Χρησιμοποιείται στην οικογένεια λειτουργικών συστημάτων της Microsoft. Επιτρέπει χαρακτήρες Unicode για τα ονόματα αρχείων και την ετικέτα του δίσκου.
- **Rockridge** : Χρησιμοποιείται στην οικογένεια των λειτουργικών τύπου UNIX, όπως το Linux. Ανακαλεί τους περιορισμούς στα ονόματα και στους καταλόγους και επιτρέπει τη χρήση δικαιωμάτων πρόσβασης στα αρχεία, καθώς και την ύπαρξη ειδικού τύπου αρχείων, όπως συντομεύσεις.

2.11 Το σύστημα αρχείων Tmpfs (Temporary File System)

Η φιλοσοφία σχεδίασης του tmpfs είναι αρκετά διαφορετική από αυτή με την οποία σχεδιάστηκαν τα περισσότερα συστήματα αρχείων. Το σύστημα αρχείων tmpfs σχεδιάστηκε για την γρήγορη και αποτελεσματική διαχείριση προσωρινών αρχείων που χρησιμοποιεί ένα λειτουργικό σύστημα.

Το tmpfs μπορεί να βρίσκεται στη φυσική μνήμη του υπολογιστή ή στην περιοχή της εικονικής μνήμης. Θα μπορούσαμε να το χαρακτηρίσουμε ως το **σύστημα αρχείων της εικονικής μνήμης** του υπολογιστή.

2.11.1 Χαρακτηριστικά του συστήματος αρχείων tmpfs

Το tmpfs χαρακτηρίζεται από το γεγονός ότι προσαρμόζεται δυναμικά ανάλογα με της ανάγκες της χρήσης. Αρχικά, το μέγεθός του είναι μικρό, όσο όμως μεταφέρονται σ΄ αυτό αρχεία, δεσμεύει περισσότερη μνήμη και αυξάνει σε μέγεθος. Και καθώς διαγράφονται αρχεία ή μεταφέρονται αλλού, συρρικνώνεται πάλι.

Η πολύ μεγάλη ταχύτητα είναι ένα άλλο χαρακτηριστικό του tmpfs. Εφ΄ όσον το σύστημα αρχείων αυτό, συνήθως βρίσκεται εξ ολοκλήρου στη μνήμη RAM, της οποίας η ταχύτητα είναι εκατοντάδες ή χιλιάδες φορές μεγαλύτερη των άλλων μέσων αποθήκευσης, όπως οι σκληροί δίσκοι, η ταχύτητά του δεν είναι κάτι που πρέπει να παραξενεύει.

Τα δεδομένα του tmpfs χάνονται μετά το κλείσιμο ή την επανεκκίνηση του υπολογιστή, κι αυτό γιατί η μνήμη RAM δεν διατηρεί τα περιεχόμενά της όταν ο υπολογιστής είναι κλειστός. Αυτό το χαρακτηριστικό κάνει το tmpfs ιδανικό σύστημα αρχείων για τη φύλαξη προσωρινών αρχείων, όπως τα ενδιάμεσα αρχεία που δημιουργούν κάποιες εφαρμογές για όση ώρα εκτελούνται. Χρησιμοποιώντας το tmpfs για τέτοια αρχεία μπορεί να αυξήσει αρκετά τις επιδόσεις των εφαρμογών αυτών.

2.12 Συμπεράσματα από τη θεώρηση των συστημάτων αρχείων

Από την παράθεση των παραπάνω γίνεται σαφές ότι υπάρχει ένας μεγάλος αριθμός από συστήματα αρχείων, κάθε ένα από τα οποία σχεδιάστηκε με διαφορετικά κριτήρια και έχοντας διαφορετικές προτεραιότητες.

Υπάρχουν δηλαδή τόσα αρχεία όσα και λειτουργικά συστήματα - και περισσότερα - και κάθε ένα από αυτά καλύπτει διαφορετικές ανάγκες του λειτουργικού και των χρηστών. Μέσα σ' αυτό τον πλουραλισμό ο διαχειριστής ενός συστήματος είναι σε θέση να επιλέξει ένα ή περισσότερα συστήματα αρχείων ώστε να ικανοποιήσει τις ανάγκες του.

3. Linux και συστήματα αρχείων

3.1 Το λειτουργικό σύστημα Linux

Το Linux είναι ο πυρήνας ενός λειτουργικού συστήματος που βασίζεται στην οικογένεια των λειτουργικών του Unix και υλοποίεί το πρότυπο POSIX (που ορίζει τυποποιήσεις για λειτουργικά συστήματα τύπου Unix). Κυκλοφορεί υπό την γενική δημόσια άδεια GNU (GNU General Public License, GPL).

Το λογισμικό που καλύπτεται από την άδεια αυτή είναι **ελέυθερο λογισμικό** που δημοσιεύεται υποχρεωτικά μαζί με τον πηγαίο κώδικά του. Καθένας έχει δικαίωμα να το εισάγει στον υπολογιστή του να να το τροποποιήσει όπως θέλει και να το αναδιανέμει σε όποιον θέλει (με τον περιορισμό ότι θα συνεχίσει να καλύπτεται από την γενική δημόσια άδεια GNU).

Ο πυρήνας του Linux άρχισε να αναπτύσσεται το 1991 από ένα Φινλανδό φοιτητή, τον Linus Torvalds ο οποίος θεωρείται ο πνευματικός πατέρας του. Όταν έφτασε σε ένα σημείο λειτουργικότητας, ο Torvalds είχε την έμπνευση να διανείμει τον κώδικα του Linux μέσω του διαδικτύου. Έτσι, πολλοί προγραμματιστές διεθνώς ασχολήθηκαν με το Linux, αποσφαλματώνοντας και προσθέτοντας συνεχώς χαρακτηριστικά. Το Linux δηλαδή, είναι συλλογικό έργο χιλιάδων προγραμματιστών από όλο τον κόσμο.

Φυσικά ένα λειτουργικό σύστημα είναι πολύ περισσότερα από τον πυρήνα του. Απαιτούνται εφαρμογές για διασύνδεση με τον χρήστη (**φλοιός**), και λογισμικό για κάθε είδους εργασία που θέλουμε να εκτελέσει ο υπολογιστής. Όταν πρωτοδημιουργήθηκε το Linux υπήρχε ήδη ένα πλήθος εφαρμογών επίσης γραμμένες από εθελοντές προγραμματιστές από όλο τον κόσμο και καλύπτομενες από την γενική δημόσια άδεια GNU. Οι εφαρμογές αυτές προορίζονταν για να πλαισιώσουν το λειτουργικό σύστημα GNU μαζί με έναν πυρήνα που αναπτύσσονταν γι' αυτό το σκοπό (**GNU/Hurd**). Το Linux ως πυρήνας, «έδεσε» πολύ καλά μ' αυτές τις εφαρμογές και από την ένωσή τους δημιουργήθηκε το λειτουργικό σύστημα που ονομάζουμε **GNU/Linux**.

Επειδή, θεωρούμενο αυστηρά, Linux είναι μόνο ο πυρήνας, όταν αναφερόμαστε σε ολόκληρο το λειτουργικό (και το φλοιό και όλες τις εφαρμογές χρήστη) κανονικά πρέπει να μιλάμε για GNU/Linux. Έχει επικρατήσει όμως να αναφέρουμε και το λειτουργικό με μόνο τον όρο Linux.

Σήμερα (Φεβουάριος 2004, έκδοση 2.6.1), το Linux αποτελεί μια σταθερότατη πλατφόρμα που ανταγωνίζεται επάξια τα εμπορικά λειτουργικά συστήματα, τρέχει κάτω από σχεδόν όλους τους τύπους του υλισμικού και διαθέτει ένα τεράστιο πλήθος εφαρμογών τόσο για χρήση σε δικτυακούς διακομιστές, για προγραμματισμό (ανάπτυξη λογισμικού), αλλά και για εργασία γραφείου.

3.2 Υποστήριξη συστημάτων αρχείων στο Linux.

Ένα από τα πιο σημαντικά χαρακτηριστικά του Linux είναι ότι **υποστηρίζει ένα πολύ μεγάλο πλήθος από διαφορετικά συστήματα αρχείων**. Για την ακρίβεια, το Linux είναι, με μεγάλη διαφορά από τα άλλα, το λειτουργικό που υποστηρίζει τα περισσότερα συστήματα αρχείων. Το γεγονός αυτό το καθιστά ιδιαίτερα ευέλικτο και ικανό να συνυπάρχει με πολλά άλλα λειτουργικά συστήματα.

3.2.1 Ιστορικά στοιχεία

Στις πρώτες μέρες της ιστορίας του, το Linux ήταν απλά περισσότερο ή λιγότερο μια υλοποίηση του λειτουργικού συστήματος minix για προσωπικούς υπολογιστές (PC). Γι΄ αυτό και το μόνο σύστημα αρχείων που υποστήριζε ήταν αυτό του minix.

Παρά τη σταθερότητα και την αποδοτικότητα του συστήματος αρχείων minix, το σύστημα αυτό ήταν σχεδιασμένο με πάρα πολλούς περιορισμούς, οι οποίοι δυσχέραιναν και περιόριζαν τη λειτουργικότητα του Linux. Γι΄ αυτό έγιναν σχέδια για να υποστηρίξει το Linux όχι απλά πιο εξελιγμένο σύστημα αρχείων, αλλά να έχει υποστήριξη για όσο το δυνατόν περισσότερα γίνεται. Για το σκοπό αυτό δημιουργήθηκε για το Linux το VFS, το οποίο θα εξετάσουμε παρακάτω.

Λίγο μετά την ενσωμάτωση του VFS στο Linux την άνοιξη του 1992, ένα νέο σύστημα αρχείων σχεδιάστηκε ειδικά για το Linux και προστέθηκε σ΄ αυτό. Ονομαζόταν Εκτεταμένο Σύστημα Αρχείων (**Extended File System**) **extfs**. Με αυτό το σύστημα αναιρούνταν οι δύο σημαντικότεροι περιορισμοί του minix: Το όριο των 2 GB στο μέγεθος του συστήματος αρχείων και ο περιορισμός των μικρών ονομάτων, για τα οποία το νέο σύστημα επέτρεπε μέχρι και 255 χαρακτήρες. Ήταν μια ξεκάθαρη βελτίωση σε σχέση με το minix, υπήρχαν όμως αρκετά προβλήματα ακόμα, καθώς έλειπαν κι άλλα βασικά χαρακτηριστικά. Επίσης, το σύστημα χρησιμοποιούσε συνδεδεμένες λίστες για να διαχειρίζεται τους ελεύθερους κόμβους-δείκτες και τα ελεύθερα μπλοκ, πράγμα που οδηγούσε σε χαμηλή απόδοση. Κατά τη διάρκεια της χρήσης, οι λίστες κατέληγαν να έιναι αταξινόμητες και το σύστημα αρχείων κατακερματισμένο.

Ως απάντηση στα παραπάνω προβλήματα, τον Ιανουάριο του 1993, κυκλοφόρησαν δυο νέα συστήματα αρχείων σε δοκιμαστική έκδοση: Το σύστημα αρχείων **Xia** και το Δεύτερο Εκτεταμένο Σύστημα Αρχείων (**Second Extended File System**, **Ext2**). Το πρώτο βασιζόταν πάνω στο minix και προσέθετε χαρακτηριστικά που έλειπαν σ΄ αυτό. Κυρίως προσέθετε υποστήριξη για μεγάλα ονόματα αρχείων και για μεγαλύτερα μεγέθη του συστήματος αρχείων. Το ext2 από την άλλη, βασιζόταν στον κώδικα του ext αναδιοργανωμένο και με πολλές αλλαγές και βελτιώσεις. Σχεδιάστηκε έχοντας στο μυαλό την εξέλιξη και την υποστήριξη καινούριων χαρακτηριστικών.

Όταν πρωτοκυκλοφόρησαν τα δύο αυτά συστήματα αρχείων πρόσφεραν περίπου τα ίδια

χαρακτηριστικά. Το Xia, λόγω της πιο μινιμαλιστικής σχεδίασής του ήταν πιο σταθερό όμως, ενώ το ext2 περιέχει πολλά σφάλματα (bugs). Με τον καιρό, τα σφάλματα του ext2 διορθώθηκαν, πολλά καινούρια χαρακτηριστικά προστέθηκαν και τελικά επικράτησε αυτό ως το κατ΄ εξοχήν σύστημα αρχείων του Linux.

3.2.2 Προσαρτήσεις

Στο Linux, όπως και σε όλους τους κλώνους του Unix άλλωστε, όλα τα διαφορετικά συστήματα αρχείων που χρησιμοποιούνται, συνδυάζονται σε μια ενιαία ιεραρχική δομή καταλόγων που αναπαριστά το σύστημα αρχείων σαν μια μοναδική οντότητα. Το Linux προσθέτει κάθε νέο σύστημα αρχείων σ' αυτή την οντότητα με μια διαδικασία που λέγεται **προσάρτηση** (**mount**). Η προσάρτηση γίνεται σε έναν κατάλογο και τα αρχεία του προσαρτημένου συστήματος αρχείων βρίσκονται κάτω από αυτόν τον κατάλογο. Τυχόν αρχεία που βρίσκονταν εκεί πριν την προσάρτηση κρύβονται. Όταν το σύστημα αρχείων αποπροσαρτηθεί (**unmount**) εμφανίζονται και πάλι τα τυχόν παλιά περιεχόμενα του καταλόγου.

Ιδιαίτερη αξία έχει προφανώς ο ριζικός κατάλογος (/) που είναι ο πρώτος που προσαρτείται κατά την εκκίνηση του συστήματος. Όλοι οι άλλοι κατάλογοι βρίσκονται κάτω από αυτόν.

3.2.3 Το υποσύστημα VFS

Η πρόσβαση σε κάθε προσαρτημένο συστήμα αρχείων του Linux γίνεται μέσω του **Virtual File System** (VFS, Εικονικό Σύστημα Αρχείων). Το VFS είναι ένα **υποσύστημα** (**subsystem**) ή αλλιώς **στρώμα** (**layer**) του πυρήνα που παρέχει γενικευμένες ενέργειες και κλήσεις προς συστήματα αρχείων.

Καθώς το λειτουργικό εκκινεί, την ώρα που φορτώνεται η υποστήριξη για κάθε σύστημα αρχείων, αυτή καταχωρείται και στο VFS. Η υποστήριξη αυτή είναι είτε **ενσωματωμένη** (**embedded**) στον πυρήνα είτε με τη μορφή **αρθρώματος** (**module**). Τα αρθρώματα φορτώνονται στον πυρήνα μόνο όταν, και για όσο το σύστημα τα χρειαστεί, για όσο δηλαδή προσαρτείται το αντίστοιχο σύστημα αρχείων.

Όταν προσαρτείται ένα σύστημα αρχείων, το VFS πρέπει να διαβάσει το υπερμπλόκ του. Η διαδικασία που διαβάζει το υπερμπλοκ για κάθε τύπο συστήματος αρχείων πρέπει να εξακριβώσει την τοπολογία του συστήματος αρχείων και να τη μεταφέρει στο γενικού τύπου υπερμπλόκ που ορίζει το VFS.

Το VFS κρατάει μια λίστα με όλα τα προσαρτημένα συστήματα αρχείων μαζί με το κατά VFS υπερμπλόκ τους. Κάθε ένα από αυτά ορίζει πληροφορίες και δείκτες προς διαδικασίες που επιτελούν συγκεκριμένες λειτουργίες. Για παράδειγμα το υπερμπλόκ που αντιπροσωπεύει ένα σύστημα αρχείων ext2 περιέχει ένα δείκτη προς την ειδική ως προς το ext2 διαδικασία που διαβάζει κόμβους-δείκτες. Αυτή η διαδικασία ανάγνωσης κόμβων-δεικτών, όπως και όλες οι άλλες ειδικές ως προς συστήματα αρχείων διαδικασίες, διαβάζει και συμπληρώνει από το πραγματικό σύστημα αρχείων τα πεδία ενός κόμβου-δείκτη του VFS.

Κάθε υπερμπλόκ περιέχει ένα δείκτη προς τον πρώτο VFS κόμβο-δείκτη στο σύστημα αρχείων. Για το ριζικό σύστημα αρχείων αυτός είναι ο κόμβος-δείκτης που αναπαριστά τον κατάλογο "/". Καθώς οι διεργασίες αποκτούν πρόσβαση σε αρχεία και καταλόγους, καλούνται διαδικασίες συστήματος που διασχίζουν τους VFS κόμβους-δείκτες στο σύστημα αρχείων. Για παράδειγμα, όταν αιτούμαστε να δούμε τα περιεχόμενα ενός καταλόγου ή αρχείου, το VFS θα ψάξει ανάμεσα στους VFS κόμβους-δείκτες του συστήματος για τον αντίστοιχο κόμβο-δείκτη.

Έτσι λοιπόν, εφόσον όλοι οι κατάλογοι και τα αρχεία αντιπροσωπεύτονται από κάποιον κόμβο-δείκτη του VFS, πολλοί κόμβοι-δείκτες θα χρησιμοποιούνται αρκετά συχνά. Για παράδειγμα, ο κόμβος-δείκτης του ριζικού κατάλογου θα χρησιμοποιείται για *κάθε* αρχείο. Οι κόμβοι-δείκτες που χρησιμοποιήθηκαν πρόσφατα αποθηκεύονται στη μνήμη (**cache κόμβων-δεικτών**) για να αυξηθεί η ταχύτητα προσπέλασής τους.

Όλα τα συστήματα αρχείων χρησιμοποιούν έναν κοινό ενταμιευτή (buffer), στον οποίο αποθηκεύουν μπλοκ δεδομένων από το φυσικό μέσο για να αυξηθεί η ταχύτητα πρόσβασης σ΄ αυτό. Ο ενταμιευτής αυτός ονομάζεται **buffer cache**. Είναι ανεξάρτητος από τα συστήματα αρχείων και βρίσκεται ενσωματωμένος στο μηχανισμό με τον οποίον ο πυρήνας δεσμεύει και χρησιμοποιεί buffers. Το πλεονέκτημά του είναι ότι κάνει τα συστήματα αρχείων του Linux ανεξάρτητα από το φυσικό μέσο. Ο χειρισμός για κάθε σύστημα αρχείων γινεται με τον ίδιο τρόπο. Καθώς διαβάζονται λοιπόν μπλοκ δεδομένων από τα συστήματα αρχείων με αυτό τον τρόπο, αποθηκεύονται στην κοινή buffer cache όπου σημειώνονται με τον αριθμό μπλοκ και με τον κωδικό του συστήματος από το οποίο διαβάστηκαν. Έτσι, σε επόμενη αίτηση ανάγνωσης, διαβάζονται απ΄ ευθείας από την buffer cache.

Το VFS διατηρεί επίσης μια ξεχωριστή cache για περιεχόμενα καταλόγων, ώστε να επιταχύνεται και η ανάγνωση των καταλόγων.

3.2.4 Το υπερμπλόκ του VFS

Κάθε προσαρτημένο σύστημα αρχείων περιγράφεται πλήρως και μοναδικά από το VFS υπερμπλόκ του. Μεταξύ άλλων πληροφοριών, αυτό περιέχει:

- **Συσκευή (device)** : Ο κωδικός της συσκευής στην οποία αντιστοιχεί το φυσικό μέσο στο οποίο βρίσκεται το σύστημα αρχείων.
- **Δείκτης κόμβου-δείκτη** : Ο δείκτης **προσάρτησης** (**mounted**) δείχνει στον πρώτο κόμβο-δείκτη του συστήματος αρχείων. Ο δείκτης **κάλυψης** (**covered**) δείχνει στον κατάλογο στον οποίο προσαρτήθηκε το σύστημα αρχείων. Για το ριζικό σύστημα αρχείων δεν υπάρχει δείκτης κάλυψης.
- **Μέγεθος μπλοκ** : Το μέγεθος του μπλοκ για το συγκεκριμένη συσκευή, 1024 bytes για παράδειγμα.
- **Διαδικασίες υπερμπλόκ (Superblock operations)** : Ένας δείκτης σε ένα σύνολο από διαδικασίες γι΄ αυτό το σύστημα αρχείων. Οι διαδικασίες αυτές καλούνται για την εγγραφή και ανάγνωση δεδομένων από το σύστημα αρχείων.

- **Τύπος συστήματος αρχείων** : Δείκτης προς ένα πίνακα που διατηρείται με όλα τα υποστηριζόμενα συστήματα αρχείων που προσδιορίζει μοναδικά τον τύπο του τρέχοντος συστήματος.
- **Ειδικότερα του συστήματος αρχείων (File System Specific) : Δείκτης προς περισσότερες πλ**ηροφορίες για το σύστημα αρχείων.

3.2.5 Ο κόμβος δείκτης του VFS

Όπως στο σύστημα αρχείων ext2, στο VFS κάθε αρχείο και κατάλογος αντιπροσωπεύεται από έναν κόμβο-δείκτη. Οι πληροφορίες σε κάθε κόμβο-δείκτη του VFS είναι δομημένες γύρω από τις πραγματικές πληροφορίες του συστήματος αρχείων που το υλοποιεί. Οι κόμβοι-δείκτες του VFS υπάρχουν μόνο στη μνήμη και κρατούνται στη μνήμη cache των κόμβων-δεικτών για όσο χρόνο είναι απαραίτητοι στο σύστημα. Κάποιες από τις πληροφορίες που περιέχουν είναι οι εξής:

- **Συσκευή (device)** : Ο κωδικός της συσκευής που αντιπροσωπεύει το φυσικό μέσο στο οποίο αποθηκεύεται το αρχείο.
- **Αριθμός κόμβου-δείκτη** : Ο αριθμός του κόμβου-δείκτη, ο οποίος είναι και μοναδικός για το συγκεκριμένο σύστημα αρχείων. Ο συνδυασμός αριθμού κόμβου-δείκτη και κωδικού συσκευής είναι μοναδικός για το VFS, για όλα τα συστήματα αρχείων δηλαδή.
- **Τρόπος (mode)** : Όπως και στο ext2, το πεδίο αυτό περιγράφει τι παριστάνει ο κόμβος-δείκτης (αρχείο, κατάλογο, συμβολικό δεσμό ή άλλο) καθώς και τα δικαιώματα πρόσβασης σ΄ αυτό.
- **Κωδικός χρήστη** : Ο κωδικός του χρήστη στον οποίο ανήκει το αρχείο.
- **Ώρα** : Ημερομηνίες και ώρες τελευταίας τροποποίησης και πρόσβασης στο αρχείο.
- **Μέγεθος μπλοκ** : Το μέγεθος μπλοκ για αυτό το αρχείο, π.χ. 1024 bytes.
- **Διαδικασίες κόμβου-δείκτη (Inode operations)** : Ένας δείκτης προς ένα σύνολο από διαδικασίες οι οποίες θα χρησιμοποιηθούν για τις πράξεις εγγραφής και ανάγνωσης στο αρχείο.
- **Αριθμός αναφορών** : Πόσα στοιχεία του συστήματος χρησιμοποιούν αυτή τη στιγμή το αρχείο αυτό.
- **Κλείδωμα** : Αυτό το πεδίο χρησιμοποιείται για το κλείδωμα του κόμβου-δείκτη, για παράδειγμα, για όση ώρα γίνεται εγγραφή σ΄ αυτόν από το σύστημα.
- **Αλλαγμένο (dirty)** : Υποδηλώνει αν έχει γίνει εγγραφή στον κόμβο-δείκτη, οπότε και στο σύστημααρχείων χρειάζεται να γίνει η αντίστοιχη τροποποίηση.

3.2.6 Καταχώρηση συστημάτων αρχείων στο VFS

Όταν χτίζουμε (build) τον πυρήνα του Linux, ερωτούμαστε για ποια συστήματα αρχείων θέλουμε υποστήριξη. Όταν ο πυρήνας είναι έτοιμος, ο κώδικας εκκίνησης για τα συστήματα αρχείων περιέχει κλήσεις για τις διαδικασίες εκκίνησης για εκείνα τα συστήματα αρχείων που είναι ενσωματωμένα στον πυρήνα.

Ένας άλλος τρόπος να υποστηριχτεί ένα σύστημα αρχείων είναι με ένα **άρθρωμα** (**module**) του πυρήνα. Το άρθρωμα είναι ένα κομμάτι κώδικα που φορτώνεται δυναμικά όταν ζητηθεί η χρήση του και ομοίως ελευθερώνεται αυτόματα όταν δεν είναι πια απαραίτητο. Τα αρθρώματα κάνουν τον πυρήνα του Linux πιο ευέλικτο, αφού περιέχει κάθε στιγμή μόνο εκείνα τα στοιχεία που είναι απαραίτητα.

Στην περίπτωσή μας, όταν ζητηθεί να γίνει προσάρτηση για ένα σύστημα αρχείων που η υποστήριξή του είναι σε άρθρωμα, φορτώνεται το αντίστοιχο άρθρωμα και καταχωρείται η παρουσία του στον πυρήνα. Ταυτόχρονα, η διαδικασία εκκίνησης του συστήματος αρχείων καταχωρείται στο VFS και αντιπροσωπεύεται από έναν μοναδικό κωδικό, τον τύπο του συστήματος αρχείων. Ο κωδικός αυτός περιέχει το όνομα του συστήματος αρχείων και έναν δείκτη προς τη διαδικασία του VFS που διαβάζει το υπερμπλόκ γι΄ αυτό το σύστημα αρχείων.

Η εγγραφή Τύπος Συστήματος Αρχείων περιέχει τα παρακάτω πεδία:

- **Διαδικασία ανάγνωσης υπερμπλόκ** : Η διαδικασία αυτή καλείται από το VFS όταν ένα σύστημα αρχείων προσαρτείται.
- **Όνομα συστήματος αρχείων** : Το όνομα του συστήματος αρχείων, π.χ. ext2.
- **Απαραίτητη συσκευή** : Ο κωδικός της (συσκευής που αναπαριστά το φυσικό μέσο) που είναι απαραίτητη για την ανάγνωση και εγγραφή από αυτό το σύστημα αρχείων. Για παράδειγμα, για την ανάγνωση από την εύκαμπτη δισκέτα μπορεί να είναι η συσκευή /dev/fd0, ενώ από τον πρώτο σκληρό δίσκο, /dev/hda. Δεν έχουν όμως όλα τα συστήματα αρχείων ανάγκη μιας συσκευής. Το σύστημα αρχείων /proc για παράδειγμα, δεν χρειάζεται.

3.2.7 Προσάρτηση συστήματος αρχείων μέσω του VFS

Όταν ο διαχειριστής προαπαθεί να προσαρτήσει ένα σύστημα αρχείων, δίνει μια εντολή που περιέχει ως παραμέτρους τον τύπο του συστήματος αρχείων, κάποιες επιλογές προσάρτησης, τη συσκευή που αντιπροσωπεύει το φυσικό μέσο στο οποίο βρίσκεται το σύστημα αρχείων και τον κατάλογο κάτω από τον οποίο θα προσαρτηθεί. Παρ΄όλο που η εντολή της προσάρτησης κάνει κάποιους ελέγχους, δε γνωρίζει αν ο πυρήνας έχει υποστήριξη για το συγκεκριμένο σύστημα αρχείων.

Το πρώτο που κάνει το VFS είναι να ελέγξει αν υποστηρίζεται το συγκεκριμένο σύστημα αρχείων. Για

το σκοπό αυτό ψάχνει τη λίστα των γνωστών συστημάτων αρχείων αναζητώντας για το όνομα του δοθέντος συστήματος. Αν βρεθεί καταχώρηση, τότε σημαίνει ότι το συγκεκριμένο σύστημα αρχείων υποστηρίζεται και μάλιστα υπάρχει δείκτης προς τη διαδικασία ανάγνωσης του υπερμπλόκ. Αν δεν βρεθεί, τότε ελέγχεται μήπως υπάρχει σχετικό άρθρωμα για να φορτωθεί. Διαφορετικά επιστρέφεται λάθος με την αιτιολογία «μη υποστηριζόμενο σύστημα αρχείων».

Στη συνέχεια, το VFS πρέπει να διαβάσει τον κόμβο-δείκτη που αναφέρεται στον κατάλογο της προσάρτησης. Αυτός μπορεί να βρίσκεται στην cache των κόμβων-δεικτών ή να πρέπει να αναγνωστεί από το σύστημα αρχείων στο οποίο βρίσκεται. Από τη στιγμή που αποκτάται πρόσβαση στον κόμβο-δείκτη ελέγχεται αν πρόκειται για κατάλογο και για το αν υπάρχει ήδη σύστημα αρχείων προσαρτημένο πάνω του. (Σε πιο καινούριες εκδόσεις του πυρήνα επιτρέπεται η προσάρτηση πολλών διαφορετικών συστημάτων αρχείων πολλές φορές στον ίδιο κατάλογο, οπότε αυτό το βήμα δε χρειάζεται).

Στο σημείο αυτό, ο κώδικας προσάρτησης πρέπει να δεσμεύσει χώρο για ένα VFS υπερμπλόκ και να περάσει τη διεύθυνσή του στη διαδικασία ανάγνωσης υπερμπλόκ για το συγκεκριμένο σύστημα αρχείων. Όλα τα υπερμπλόκ των προσαρτημένων αρχείων φυλάσσονται σε ένα πίνακα του VFS και μια νέα εγγραφή πρέπει να δημιουργηθεί γι΄ αυτή την προσάρτηση. Η διαδικασία ανάγνωσης διαβάζει τις πληροφορίες από το σύστημα αρχείων, και απεικονίζει τα χαρακτηριστικά στα πεδία του υπερμπλόκ του VFS. Για το σύστημα αρχείων ext2 η απεικόνιση αυτή είναι εύκολη, αφού το VFS είναι ουσιαστικά χτισμένο πάνω στις δομές του. Για άλλα συστήματα αρχείων, όπως το FAT, δεν είναι και τόσο εύκολο, ειδικά όταν η σχεδίαση και φιλοσοφία τους απέχει πολύ από αυτή του ext2. Ανεξάρτητα από το είδος του συστήματος αρχείων όμως, η διαδικασία θα πρέπει να διαβάσει τα δεδομένα που το περιγράφουν για να συμπληρώσει το υπερμπλόκ.

Κάθε προσαρτημένο σύστημα αρχείων περιγράφεται από μια δομή σχεδιασμένη γι΄ αυτό το σκοπό και το VFS κρατάει μια λίστα αυτών των δομών για όλα τα προσαρτημένα συστήματα αρχείων. Κάθε τέτοια συσκευή περιέχει τον αριθμό συσκευής που αναπαριστά το φυσικό μέσο στο οποίο βρίσκεται το σύστημα αρχείων, τον κατάλογο στον οποίο είναι προσαρτημένο, και ένα δείκτη προς το VFS υπερμπλόκ που δεσμεύτηκε όταν προσαρτήθηκε το σύστημα σρχείων.

3.2.8 Αποπροσάρτηση συστήματος αρχείων

Για να αποπροσαρτηθεί ένα σύστημα αρχείων πρέπει να μην υπάρχουν διεργασίες που να γράφουν ή να διαβάζουν σ΄ αυτό ή να έχουν ως κατάλογο εργασίας τους κάποιον που να ανήκει σ΄ αυτό. Αν κάποιος χρησιμοποιεί το σύστημα αρχείων, τότε μπορεί να υπάρχουν κόμβοι-δείκτες του στην cache των κόμβων-δεικτών του VFS και εκείνο ελέγχει για την ύπαρξη τους ψάχνοντας τη λίστα κόμβων-δεικτών για το αν υπάρχουν κάποιοι των οποίων ο κωδικός συσκευής να είναι ίδιος με αυτόν του συστήματος αρχείων προς αποπροσάρτηση.

Αν το VFS υπερμπλόκ του συστήματος αρχείων είναι μαρκαρισμένο ως αλλαγμένο, δηλαδή έχουν γίνει αλλαγές στο υπερμπλόκ, τότε πρέπει να ξαναγραφτεί στο σύστημα αρχείων. Από τη στιγμή που έχει γίνει η εγγραφή, ελευθερώνεται η μνήμη που δεσμεύτηκε για το υπερμπλόκ. Τέλος, από τη λίστα με τα προσαρτημένα συστήματα αρχείων αφαιρείται η καταχώρηση για αυτό το σύστημα αρχείων.

3.2.9 Μνήμη cache κόμβων-δεικτών

Καθώς ήδη αναφέραμε, το VFS διατηρεί τους κόμβους-δείκτες του που προσπελάστηκαν πρόσφατα σε μια μνήμη cache για την επιτάχυνση του συστήματος. Κάθε φορά που ένας κόμβος-δείκτης διαβάζεται από την cache, το σύστημα γλυτώνει μια πρόσβαση στο πραγματικό φυσικό μέσο.

Η μνήμη cache αυτή είναι υλοποιημένη ως ένας πίνακας κατακερματισμού (**hash table**), οι εγγραφές του οποίου είναι δείκτες σε λίστες κόμβων-δεικτών του VFS που έχουν την ίδια τιμή hash. Η τιμή hash κάθε κόμβου-δείκτη υπολογίζεται από τον αριθμό του και από τη συσκευή που περιγράφει το φυσικό μέσο στο οποίο είναι αποθηκευμένο το σύστημα αρχείων στο οποίο ανήκει.

Όποτε το VFS πρέπει να διαβάσει έναν κόμβο-δείκτη, ψάχνει πρώτα στη μνήμη cache. Για να δει αν ένας κόμβος-δείκτης βρίσκεται εκεί, υπολογίζει την hash τιμή του κόμβου-δείκτη και πηγαίνει στην αντίστοιχη θέση του πίνακα της μνήμης cache, όπου βρίσκεται μια λίστα με αποθηκευμένους κόμβους-δείκτες με την ίδια hash τιμή. Με μια αναζήτηση σ΄ αυτή τη λίστα ξέρει αν ο κόμβος-δείκτης βρίσκεται ή όχι στη μνήμη cache.

Αν ο κόμβος-δείκτης βρεθεί στη μνήμη cache, τότε αυξάνεται ο μετρητής χρήσης του για να ξέρουμε ότι υπάρχει άλλη μια χρήση του και στη συνέχεια πραγματοποιείται η πρόσβαση στον κόμβο-δείκτη και στα δεδομένα του. Αλλιώς πρέπει να βρεθεί ένας ελεύθερος κόμβος-δείκτης του VFS για να διαβαστεί από το φυσικό μέσο και να καταχωρηθεί εκεί ο κόμβος-δείκτης για τον οποίο έγινε αίτηση.

Το VFS έχει διάφορες επιλογές για να βρει χώρο για καινούριους κόμβους-δείκτες. Αν είναι δυνατή η δέσμευση περισσότερης μνήμης, τότε αυτή πραγματοποιείται και δημιουργούνται νέοι κενοί κόμβοι-δείκτες που προσθέτονται στη λίστα κόμβων-δεικτών. Αν το σύστημα έχει ήδη φτάσει στο μέγιστο επιτρεπτό όριο δεσμευμένων κόμβων-δεικτών, πρέπει να βρεθεί ένας κατάλληλος υποψήφιος κόμβος-δείκτης για διαγραφή ώστε να δημιουργηθεί κενός χώρος. Καλύτεροι υποψήφιοι είναι αυτοί που έχουν μετρητή χρήσης ίσο με μηδέν, δεν χρησιμοποιούνται δηλαδή. Οι ζωτικής σημασίας κόμβοι-δείκτες όπως ο ριζικός έχουν πάντα αριθμό χρήσης μεγαλύτερο του μηδενός και δε σβήνονται ποτέ από τη μνήμη cache. Όταν βρεθεί λοιπόν ο κατάλληλος υποψήφιος κόμβος-δείκτης, καθαρίζεται και ελευθερώνεται η θέση του, φροντίζοντας να γραφτεί όποια τυχόν αλλαγή έχει γίνει σ΄ αυτόν.

Όπως και να βρεθεί η θέση για το νέο κόμβο-δείκτη, καλείται η διαδικασία η οποία διαβάζει από το σύστημα αρχείων τα στοιχεία του και τα μεταφέρει στο VFS. Όση ώρα μεταφέρονται τα δεδομένα, ο κόμβος-δείκτης είναι κλειδωμένος ώστε να μη μπορεί κανείς να τον διαβάσει.

Για να φτάσει στον κόμβο-δείκτη που πρέπει, το VFS συνήθως χρειάζεται να περάσει από πολλούς άλλους,. Για παράδειγμα, όταν διαβάζουμε ένα αρχείο ενός καταλόγου, θα πρέπει να αρχίσουμε από τον ριζικό κατάλογο κατεβαίνοντας καταλόγους ώσπου να φτάσουμε στο αρχείο που θέλουμε. Καθώς η μνήμη cache των κόμβων-δεικτών χρησιμοποιείται και γεμίζει, οι λιγότερο χρησιμοποιούμενοι κόμβοι-δείκτες απορρίπτονται, ενώ οι περισσότερο χρησιμοποιούμενοι, μένουν περισσότερο στη μνήμη cache.

3.2.10 Μνήμη cache καταλόγων

Για να επιταχύνει την πρόσβαση σε συχνά χρησιμοποιούμενους κατάλογους, το VFS χρησιμοποιεί την ίδια τακτική του caching και με τις καταχωρήσεις των καταλόγων. Καθώς γίνεται επισκέψη σε έναν κατάλογο, τα δεδομένα του τοποθετούνται στη μνήμη cache καταλόγων. Την επόμενη φορά που θα ζητηθεί

να επισκεφτεί ο κατάλογος, για να ανοιχτεί κάποιο αρχείο του ή για να δούμε τα περιεχόμενά του, ο κατάλογος θα βρεθεί στη μνήμη cache καταλόγων.

Μόνο μικρά ονόματα διαδρομής τοποθετούνται στη μνήμη cache, με μέγιστο 15 χαρακτήρες . Η προσέγγιση αυτή όμως δεν είναι παράλογη, αφού οι μικρές διαδρομές χρησιμοποιούνται πιο συχνά.

Η μνήμη cache καταλόγων αποτελείται κι αυτή από έναν πίνακα κατακερματισμού, η κάθε καταχώρηση του οποίου δείχνει σε μια λίστα από κατάλογους που έχουν την ίδια τιμή hash. Η συνάρτηση κατακερματισμού χρησιμοποιεί τον αριθμό της συσκευής που περιγράφει το φυσικό μέσο του κάθε συστήματος αρχείων και το όνομα του καταλόγου για να παράγει τον αριθμό hash.

Σε μια προσπάθεια να κρατήσει τη μνήμη cache έγκυρη, το VFS διατηρεί λίστες με λιγότερο χρησιμοποιούμενους κατάλογους (**Least Recently Used**, **LRU**). Όταν μια καταχώρηση καταλόγου πρωτομπαίνει στη μνήμη cache, πράγμα που συμβαίνει όταν λαμβάνει χώρα η πρώτη προσπέλαση σ΄ αυτόν, προστίθεται στο τέλος του πρώτου επίπεδου της λίστας LRU (μία χρήση). Σε μια γεμάτη μνήμη cache, η διαδικασία αυτή θα εκδιώξει μια καταχώρηση στην αρχή του πρώτου επίπεδου της λίστας. Αν ο κατάλογος προσπελαστεί ξανά, προβιβάζεται και περνάει στο δεύτερο επίπεδο της λίστας (δύο χρήσεις). Και πάλι, η μετακίνηση αυτή θα διώξει από το δεύτερο επίπεδο μια καταχώρηση που βρισκόταν εκεί και θα την μετακινήσει στο πρώτο επίπεδο.

Η γενική ιδέα, όπως φαίνεται, είναι να ιεραρχηθούν οι καταχωρήσεις των καταλόγων ανάλογα με το πόσες φορές έγινε πρόσβαση σ΄ αυτούς και το πόσο πρόσφατα έγινε. Οι κατάλογοι που ταξινομούνται τελευταίοι με αυτά τα κριτήρια φεύγουν από τη μνήμη cache.

3.2.11 Buffer cache

Καθώς τα προσαρτημένα συστήματα αρχείων χρησιμοποιούνται, γίνονται πολλές αιτήσεις ανάγνωσης και εγγραφής στα δεδομένα. Το VFS βλέπει τις συσκευές αποθήκευσης σαν πίνακες από μπλοκ δεδομένων. Όλες οι αιτήσεις για εγγραφή και ανάγνωση δεδομένων δίνονται στους οδηγούς των συσκευών που αναπαριστούν τα φυσικά μέσα με τη μορφή ενός συνόλου πληροφοριών. Οι πληροφορίες αυτές είναι ό,τι χρειάζονται οι οδηγοί των συσκευών: Τον κωδικό της συσκευής που την προσδιορίζει μοναδικά, και τον αριθμό του μπλοκ που χρειάζεται να γραφτεί ή να αναγνωστεί.

Για να επιταχυνθεί η πρόσβαση στα συστήματα αρχείων το VFS διατηρεί μνήμη cache και στους ενταμιευτές των συσκευών μπλοκ (block buffers), Την cache αυτή την μοιράζονται όλες οι συσκευές μπλοκ. Σε κάθε στιγμή υπάρχουν πολλοί block buffers στη μνήμη cache οι οποίοι μπορεί να ανήκουν σε πολλές διαφορετικές συσκευές και πολλές φορές σε πολλές διαφορετικές καταστάσεις. Το να υπάρχουν έγκυρα δεδομένα στη μνήμη cache γλυτώνει το σύστημα από μια χρονοβόρα πραγματική προσπέλαση στο φυσικό μέσο. Κάθε block buffer που χρησιμοποιήθηκε για να διαβάσει ή να γράψει δεδομένα πηγαίνει στη μνήμη cache των block buffers. Αργότερα, μπορεί να αφαιρεθεί για να δημιουργηθεί ελεύθερος χώρος για έναν πιο καινούριο ή πιο συχνά αιτούμενο block buffer.

Οι block buffers προσδιορίζονται μοναδικά στη μνήμη cache από τον κωδικό της συσκευής που αναπαριστά το φυσικό μέσο και τον αριθμό του block buffer. Η μνήμη cache των block buffer αποτελείται από δυο λειτουργικά κομμάτια.

- Τη λίστα των block buffers. Υπάρχει μια λίστα για κάθε υποστηριζόμενα μέγεθος buffer και οι

ελεύθεροι block buffers του συστήματος τοποθετούνται σ΄ αυτές τις λίστες όταν δημιουργούνται ή όταν ελευθερώνονται. Τα υποστηριζόμενα μεγέθη buffer είναι 512 bytes, 1, 2, 4, και 8 KB.

- Την ίδια τη μνήμη cache. Πρόκειται για έναν πίνακα κατακερματισμού που αποτελείται από έναν πίνακα από δείκτες προς αλυσίδες buffers με τον ίδιο αριθμό hash. Ο αριθμός hash κάθε block buffer υπολογίζεται από τον κωδικό της συσκευής και τον αριθμό του block buffer.

Οι block buffers θα βρίσκονται είτε σε σε μια από τις λίστες, οπότε θα είναι ελεύθεροι, είτε θα βρίσκονται στην cache αν είναι κατειλημμένοι. Όταν βρίσκονται στη μνήμη cache, τοποθετούνται επίσης σε μια λίστα LRU (λιγότερο συχνής χρήσης) στην οποία ιεραρχούνται σύμφωνα με το πόσες φορές και πόσο πρόσφατα έχουν γίνει αιτήσεις γι΄ αυτούς. Υπάρχει μια LRU λίστα για κάθε τύπο buffer.

Ο τύπος του buffer αντικατοπτρίζει την κατάστασή του και το VFS υποστηρίζει τους παρακάτω τύπους.

- **Καθαρός (clean)** : Αχρησιμοποίητος ή νέος buffer.
- **Κλειδωμένος (locked)** : Ο buffer είναι κλειδωμένος περιμένοντας να γίνει εγγραφή.
- **Αλλαγμένος (dirty)** : Έχουν γίνει αλλαγές στον buffer και περιέχει έγκυρα δεδομένα, τα οποία δεν έχουν προλάβει να εγγραφούν ακόμα.
- **Διαμοιρασμένος (shared)** : Ο buffer χρησιμοποιείται για ανάγνωση από πάνω από μία διεργασίες.
- **Αμοίραστος (unshared)** : Ο buffer χρησιμοποιείται μόνο από μια διαδικασία για ανάγνωση.

Οποτεδήποτε το σύστημα πρέπει να διαβάσει ένα buffer δεδομένων από το φυσικό μέσο, προσπαθεί πρώτα να τον βρει στην buffer cache. Αν δεν τα καταφέρει, τότε θα βρει εκεί έναν άδειο buffer από τη λίστα με το αντίστοιχο μέγεθος και αφού διαβάσει από το φυσικό μέσο τον αιτούμενο buffer, θα τον μεταφέρει και στην cache.

Όπως όλες για όλες οι μνήμες cache, έτσι και για την buffer cache πρέπει να γίνεται σωστή διαχείριση ώστε να μην μένουν εκεί πολλά δεδομένα που περιμένουν να γραφτούν. Όσο πιο πολύ καθυστερεί η εγγραφή, τόσο περισσότερο διατρέχουμε τον κίνδυνο να μη γραφτούν ποτέ τα δεδομένα αυτά λόγω αστοχίας υλικού, προβλήματος τροφοδοσίας ή και σφάλματος λογισμικού που μπορούν να οδηγήσουν σε κατάρρευση του λειτουργικού συστήματος.

4. Συγκριτική δοκιμή

Στο κεφάλαιο αυτό γίνεται μια συγκριτική δοκιμή διάφορων συστημάτων αρχείων για να διερευνηθούν και στην πράξη τα πλεονεκτήματα και τα μειονεκτήματά τους και το που υπερέχει ή υστερεί το καθένα.

4.1 Περιγραφή της δοκιμής

Χρησιμοποιήθηκε μια συστοιχία από τρία πανομοιότυπα μηχανήματα, συνδεδεμένα μεταξύ τους με κάρτες δικτύου Gigabit Ethernet.

Η βασική σύνθεση κάθε μηχανήματος ήταν:

- **Μητρική** Supermicro P3TDE6
- Δυο **επεξεργαστές** Intel Pentium III με συχνότητα λειτουργίας στα 800 Mhz.
- **Μνήμη** 1 GB SDRAM με συχνότητα λειτουργίας στα 133 Mhz.
- **Σκληρός δίσκος** IDE χωρητικότητας 10 GB, Quantum Fireball CX 10.2A

Ο κάθε σκληρός δίσκος ήταν χωρισμένος σε τρεις κατατμήσεις, μία μικρή στο κέντρο του που αποτελούσε την εικονική μνήμη (swap) κάθε συστήματος, και δυο ίδιου μεγέθους, μια στην αρχή και μία στο τέλος. Οι πρώτες χρησιμοποιήθηκαν για τα αρχεία του λειτουργικού συστήματος ενώ οι τελευταίες για να γίνουν οι μετρήσεις.

Από την πλευρά του λογισμικού, χρησιμοποιήθηκε η έκδοση 2.4.24 του πυρήνα του Linux σε διανομή Debian Sid. Για τις μετρήσεις χρησιμοποιήθηκαν τα μετροπογράμματα Bonnie και postmark μαζί με κάποια απλά σενάρια φλοιού (shell scripts). Τα σχεδιαγράμματα έγιναν με τη βοήθεια του φύλλου υπολογισμών της σουίτας γραφείου OpenOffice. Έγινε αποκλειστική χρήση ανοιχτού λογισμικού.

Εξετάστηκαν τα εξής συστήματα αρχείων:

- **ext2** (έκδοση 1.27) *
- **ext3,** (έκδοση 2.4-0.9.19) στις τρεις του καταστάσεις: **journal**, **ordered** και **writeback**
- **Reiserfs**, (έκδοση 3.11b) με και χωρίς την παράμετρο **notail**
- **JFS** (έκδοση 1.0.14)
- **XFS** (έκδοση 1.0.17)
- **vfat** (η υλοποίηση του Linux για το FAT32), (έκδοση 2.8)

* Όπως έχει προαναφερθεί, το Linux ενημερώνει την ημερομηνία τελευταίας προσπέλασης κάθε

αρχείου οποτεδήποτε κάποιος αποκτά πρόσβαση σ΄ αυτό, πράγμα που συνεπάγεται μια καθυστέρηση. Στη δοκιμή που ακολουθεί, όλα τα συστήματα αρχείων προσαρτήθηκαν με την παράμετρο **noatime** (no access time, όχι ώρα προσπέλασης) με την οποία το Linux αγνοεί την ημερομηνία τελευταίας προσπελασης και απενεργοποιεί αυτό το χαρακτηριστικό, επιτυχαίνοντας ελαφρώς καλύτερη απόδοση. Κρίθηκε όμως σκόπιμο να δοκιμαστεί ένα από τα συστήματα αρχείων και με ενεργοποιημένο το χαρακτηριστικό για να φανεί η μικρή διαφορά που προκύπτει. Για το σκοπό αυτό, επιλέχτηκε το ext2.

4.2 Μετρήσεις με το μετροπρόγραμμα Bonnie

4.2.1 Πληροφορίες για το μετροπρόγραμμα Bonnie

Το μετροπρόγραμμα Bonnie διεξάγει μια σειρά δοκιμών και αναφέρει τις επιδόσεις μαζί με τη σχετική χρήση του επεξεργαστή.

Το μετροπρόγραμμα παίρνει ως παράμετρο το μέγεθος του αρχείου με το οποίο θα γίνουν οι δοκιμές. Το μέγεθος αυτό καλό είναι να είναι αρκετά μεγαλύτερο από τη μνήμη του συστήματος για να εξουδετερωθούν οι μεγάλες επιταχύνσεις από τη χρήση της cache μνήμης και να μετρηθούν οι πραγματικές επιδόσεις των σκληρών δίσκων.

Διεξάγονται οι παρακάτω δοκιμές:

- **Σειριακή εγγραφή ανα χαρακτήρα** : Γίνεται εγγραφή χαρακτήρα προς χαρακτήρα σε ένα αρχείο μέχρι το δοσμένο μέγεθος. Εδώ, μεγάλο ρόλο παίζει η ταχύτητα του επεξεργαστή καθώς και ο τρόπος με τον οποίο το λειτουργικό σύστημα υλοποιεί τη δέσμευση χώρου.
- **Σειριακή εγγραφή ανα μπλοκ** : Γίνεται σειριακή εγγραφή κι εδώ, αλλά χρησιμοποιούνται όλόκληρα μπλοκ δεδομένων. Εδώ τον κύριο ρόλο παίζει ο τρόπος της δεύσμευσης χώρου του λειτουργικού συστήματος.
- **Επανεγγραφή** : Το αρχείο χωρίζεται σε κομμάτια (chunks) των 16 KB και κάθε για κάθε τέτοιο κομμάτι γίνεται ανάγνωση, αλλάζονται κάποια δεδομένα κι ύστερα το κομμάτι ξαναγράφεται. Εφ΄ όσον δε γίνεται δέσμευση νέου χώρου και οι συγκεκριμένες λειτουργίες εισόδου είναι καθαρά τοπικές, αυτή η δοκιμή ελέγχει τη μνήμη cache του συστήματος και την ταχύτητα των μεταφορών δεδομένων.
- **Σειριακή είσοδος ανα χαρακτήρα** : Το αρχείο διαβάζεται σειριακά, χαρακτήρα προς χαρακτήρα.
- **Σειριακή είσοδος ανα μπλοκ** : Διαβάζονται όλα τα μπλοκ του αρχείου ένα προς ένα. Η συγκεκριμένη δοκιμή δίνει μια πολύ καθαρή μέτρηση των σειριακών επιδόσεων εισόδου.
- **Τυχαίες προσπελάσεις** : Εδώ τρέχουν παράλληλα τέσσερεις ταυτόχρονες διεργασίες που διεξάγουν 4.000 τυχαίες τοποθετήσεις (seeks) στο αρχείο διαβάζοντας ένα μπλοκ δεδομένων. Σε ποσοστό 10% των περιπτώσεων γίνονται αλλαγές στο μπλοκ και στη συνέχεια αυτό ξαναγράφεται.

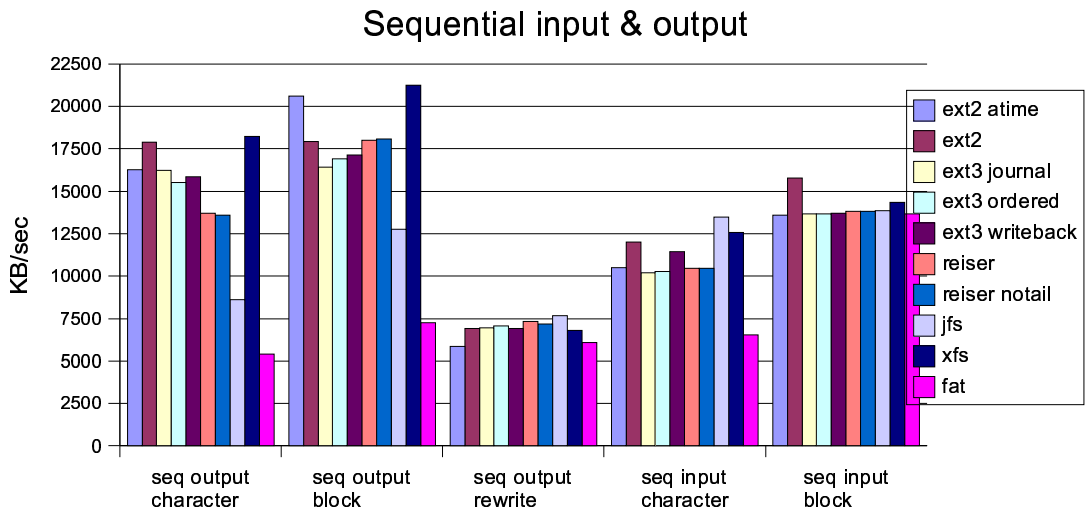
4.2.2 Διεξαγωγή των μετρήσεων

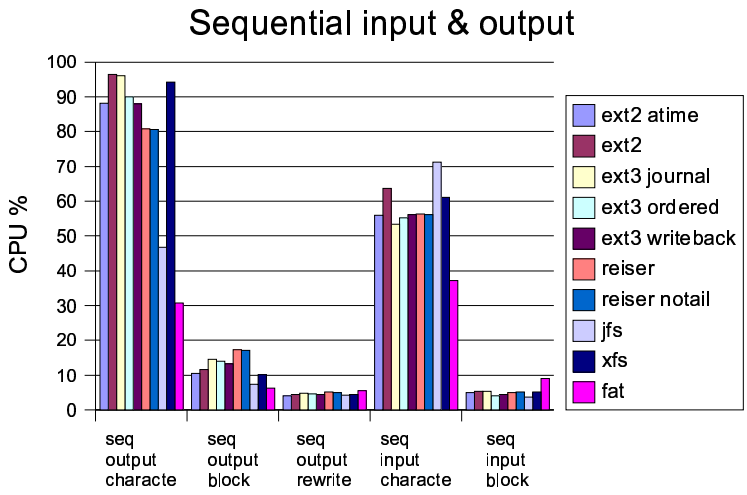
Ο παρακάτω πίνακας δίνει τα αποτελέσματα που έδωσε το μετροπρογράμμα Bonnie, όταν

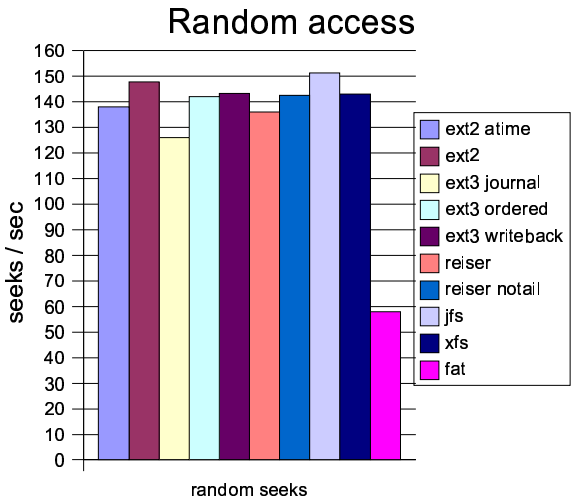
	Sequential Output						Sequential Input				Random	
	Per char		Block		Rewrite		Per char		Block		seeks	
	Kb/sec	CPU	Kb/sec	CPU	Kb/sec	CPU	Kb/sec	CPU	Kb/sec	CPU	/sec	CPU
ext2 atime	16266	88,2	20604	10,4	5853	4,1	10486	55,9	13587	5	138,2	0,6
ext2	17871	96,4	17920	11,6	6900	4,4	12001	63,7	15782	5,3	147,9	1
ext3 journal	16226	96	16424	14,5	6944	4,7	10215	53,4	13680	5,3	126	0,9
ext3 ordered	15522	90	16902	14	7086	4,6	10291	55,3	13680	4,1	142,2	0,9
ext3 writeback	15841	88	17122	13,3	6921	4,5	11453	56,2	13688	4,5	143,3	0,9
reiser	13709	80,9	18017	17,3	7339	5,2	10481	56,4	13813	5	136,2	0,7
reiser notail	13610	80,7	18074	17,2	7186	4,9	10467	56,1	13812	5,1	142,6	1,2
jfs	8621	46,8	12775	7,4	7680	4,3	13468	71,3	13871	3,6	151,4	0,5
xfs	18234	94,2	21245	10,2	6821	4,4	12581	61,2	14351	5,2	143,2	0,9
fat	5393	30,7	7254	6,2	6081	5,5	6526	37,1	13662	9	57,9	152,7

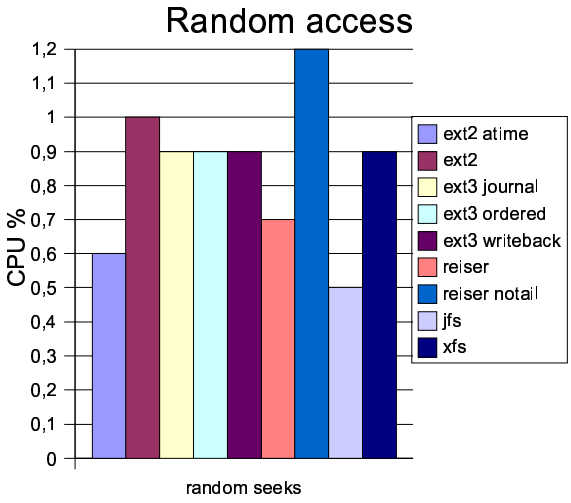
εκτελέστηκαν οι δοκιμές με μέγεθος ίσο με 2048 MB, διπλάσιο δηλαδή από τη φυσική μνήμη των υπολογιστών:

Από τον πίνακα προκύπτουν τα εξής γραφήματα:









4.2.3 Σχόλια

Από τα γραφήματα μπορούμε οπτικά να εξάγουμε τα παρακάτω:

- Σε όλες τις περιπτώσεις εκτός από εκείνης της σειριακής εξόδου ανά μπλοκ, η επιλογή της απενεργοποίησης της ημερομηνίας τελευταίας προσπέλασης των αρχείων μας δίνει μια μικρή, αλλά σε καμία περίπτωση ασήμαντη αύξηση της απόδοσης.
- Σχεδόν παντού, η κατάσταση writeback του ext3 είναι λίγο πιο γρήγορη από την ordered, η οποία με τη σειρά της είναι λίγο πιο γρήγορη από τη journal. Οι διαφορές, όμως είναι μικρές σε σχέση με την αυξημένη ασφάλεια των δεδομένων που προκύπτει. Το ext3 ελάχιστα υπολείπεται σε ταχύτητα από το ext2.
- Το JFS, παρά τη σύγχρονη σχεδίασή του, με τη χρήση προηγμένων δομών όπως τα B+δέντρα, απογοητεύει πολύ με τις άσχημες επιδόσεις του στη σειριακή έξοδο.

- Η επιλογή notail στο Reiserfs δίνει μια συνήθως μια μικρή διαφορά προς τα πάνω στις επιδόσεις.
- Το XFS τα καταφέρνει αρκετά καλά σχεδόν παντού. Ειδικά στη σειριακή έξοδο, εντυπωσιάζει με τη διαφορά του από όλα τα άλλα συστήματα αρχείων.
- Είναι ολοφάνερο ότι η παλαιολιθική σχεδίαση του FAT το καθιστά τελείως εκτός συναγωνισμού σε όλες τις δοκιμές, όπου έρχεται τελευταίο, συχνά με τεράστια διαφορά. Να σημειώσουμε ότι είναι το μόνο που στην τυχαία προσπέλαση εκτοξεύει τη χρήση του επεξεργαστή, ενώ όλα τα άλλα συστήματα μένουν κοντά στο 1%.

4.3 Μετρήσεις με το μετροπρόγραμμα Postmark

4.3.1 Το μετροπρόγραμμα postmark

Από τη μεγάλει πλειοψηφία των μετροπρογραμμάτων που κυκλοφορούν, τα περισσότερα εξετάζουν την απόδοση του συστήματος ασχολούμενα με το να γράφουν και να διαβάζουν μεγάλα αρχεία. Οι μετρήσεις αυτές είναι πράγματι πολύτιμες για τη μελέτη της απόδοσης, αλλά δεν είναι οι μόνες.

Από την άλλη πλευρά, οι διακομιστές που χρησιμοποιούνται για το ηλεκτρονικό ταχυδρομείο, τις ομάδες συζητήσεων (newsgroups), ως ενδιάμεσοι (proxy), διακομιστές ιστοσελίδων ή αρχείων, ασχολούνται ως επί το πλείστον με τη διαχείριση πολλών και μικρών αρχείων. Χιλιάδες ή δεκάδες χιλιάδες (ή ακόμα και εκατοντάδες χιλιάδες!) μικρά αρχεία που συνεχώς αλλάζουν, δημιουργούνται και διαγράφονται είναι πολύ συχνά αυτό που αναλαμβάνουν να αντιμετωπίσουν τα συστήματα αρχείων στους σύγχρονους διακομιστές.

Το postmark σχεδιάστηκε για να προσομοιώνει ακριβώς τέτοιου είδους κίνηση στο σύστημα αρχείων. Δημιουργεί μια δεξαμενή (pool) συνεχώς δυναμικά τροποποιούμενων αρχείων. Ο χρήστης μπορεί να επιλέξει;

- Το **πάνω και το κάτω όριο του μεγέθους** των αρχείων αυτών
- Τον **αριθμό** τους
- Τον αριθμό των συναλλαγών (**transactions**) που λαμβάνουν χώρα. Με τον όρο συναλλαγή, εννοούμε οποιαδήποτε πράξη τροποποίησης,ανάγνωσης, εγγραφής, επέκτασης (append), δημιουργίας ή διαγραφής στα αρχεία.
- Το ποσοστά επί των συνολικών συναλλαγών που θα καταλαμβάνουν οι εγγραφές, αναγνώσεις και άλλες συναλλαγές.
- Το μέγεθος του μπλοκ εγγραφής και ανάγνωσης.

Αρχικά, το μετροπογράμμα δημιουργεί τη δεξαμενή των αρχείων καταμετρώντας και τις επιδόσεις για τη συνεχόμενη δημιουργία τους. Ύστερα, λαμβάνει χώρα ο αριθμός και το είδος των συναλλαγών που ζητήθηκε. Κάθε συναλλαγή περιλαμβάνει ένα ζευγάρι από μικρότερες συναλλαγές: Δημιουργία ή διαγραφή, ανάγνωση ή επέκταση. Το αντικείμενο της συναλλαγής επιλέγεται τυχαία για να ελαχιστοποιηθεί η συνεισφορά της μνήμης cache και άλλων τεχνικών όπως η προανάγνωση (readaehad) στις επιδόσεις.

- Κατά τη **δημιουργία** ενός αρχείου, επιλέγεται ένα τυχαίο μέγεθος μεταξύ του ελάχιστου και του μέγιστου όριου που έχει επιλεγεί και τυχαίοι χαρακτήρες εισάγονται στο αρχείο.
- Η **διαγραφή** επιλέγει ένα τυχαίο αρχείο από τη δεξαμενή και το διαγράφει.
- Κατά την **ανάγνωση**, επιλέγεται ένα αρχείο στην τύχη και διαβάζονται εξ ολοκλήρου τα περιεχόμενά του στη μνήμη χρησιμοποιώντας το επιλεγμένο μέγεθος buffer ανάγνωσης.
- Η **επέκταση** διαλέγει ένα αρχείο στην τύχη, τοποθετείται στο τέλος του και γράφει ένα τυχαίο αριθμό από τυχαία δεδομένα, ο αριθμός των οποίων είναι μικρότερος από το επιλεγμένο μέγιστο μέγεθος αρχείου. Αν το συνολικό μέγεθος αρχείου είναι μεγαλύτερο από το επιλεγμένο μέγιστο, η επέκταση δε λαμβάνει χώρα.

Όταν έχουν πραγματοποιηθεί όλες οι ζητούμενες συναλλαγές, διαγράφονται όλα τα αρχεία της δεξαμενής, και ταυτόχρονα μετράται η ταχύτητα της διαγραφής.

Στο τέλος κάθε δοκιμής, παράγεται μια αναφορά από τα παρακάτω:

- Συνολικός χρόνος διάρκειας της δοκιμής.
- Συνολικός χρόνος διάρκειας των συναλλαγών και μέσος χρόνος συναλλαγών ανά δευτερόλεπτο.
- Συνολικός αριθμός αρχείων που δημιουργήθηκαν και μέσος ρυθμός δημιουργίας (αρχεία ανά δευτερόλεπτο).
- Αριθμός αρχείων που δημιουργήθηκαν αρχικά και και μέσος ρυθμός δημιουργίας (αρχεία ανά δευτερόλεπτο)
- Αριθμός αρχείων που δημιουργήθηκαν κατά τη διάρκεια των συναλλαγών και μέσος ρυθμός δημιουργίας (αρχεία ανά δευτερόλεπτο)
- Συνολικός αριθμός αρχείων στα οποία έγινε ανάγνωση και μέσος ρυθμός αναγνώσεων ανά δευτερόλεπτο.
- Συνολικός αριθμός αρχείων στα οποία έγινε επέκταση και μέσος ρυθμός επεκτάσεων ανά δευτερόλεπτο.
- Συνολικός αριθμός αρχείων τα οποία διαγράφτηκαν και μέσος ρυθμός διαγραφών ανά δευτερόλεπτο.
- Συνολικός αριθμός αρχείων τα οποία διαγράφτηκαν μετά το πέρας των συναλλαγών και μέσος ρυθμός διαγραφών ανά δευτερόλεπτο.
- Συνολικός αριθμός αρχείων τα οποία διαγράφτηκαν κατά τη διάρκεια των συναλλαγών και μέσος ρυθμός διαγραφών ανά δευτερόλεπτο.
- Συνολικό μέγεθος δεδομένων που διαβάστηκαν και μέσος ρυθμός ανάγνωσης κατά τη διάρκεια όλης της δοκιμής (χαρακτήρες ανά δευτερόλεπτο).
- Συνολικό μέγεθος δεδομένων που γράφτηκαν και μέσος ρυθμός εγγραφής κατά τη διάρκεια όλης της δοκιμής (χαρακτήρες ανά δευτερόλεπτο).

4.3.2 Διεξαγωγή των μετρήσεων

Έγιναν τρεις συνολικά δοκιμές με το μετροπρόγραμμα postmark:

- Μικρά αρχεία, με μέγεθος από μηδενικό ως 10.000 χαρακτήρες, αριθμός αρχείων 10.000 και 10.000 συναλλαγές
- Μεσαία αρχεία, με μέγεθος από μηδενικό ως 100.000 χαρακτήρες, αριθμός αρχείων 10.000 και 10.000 συναλλαγές
- Μεγάλα αρχεία, με μέγεθος από μηδενικό ως 1.00.000 χαρακτήρες, αριθμός αρχείων 5.000 και 5.000 συναλλαγές

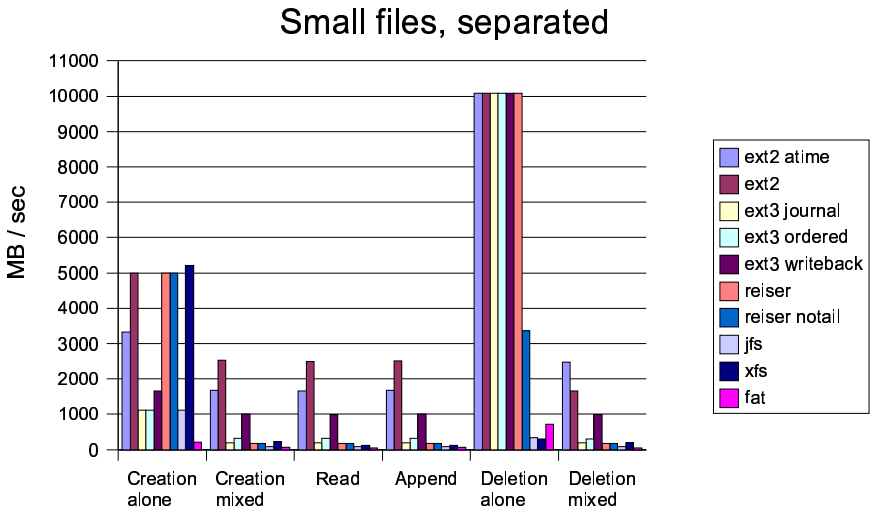
4.3.2.1 Μετρήσεις για μικρά αρχεία

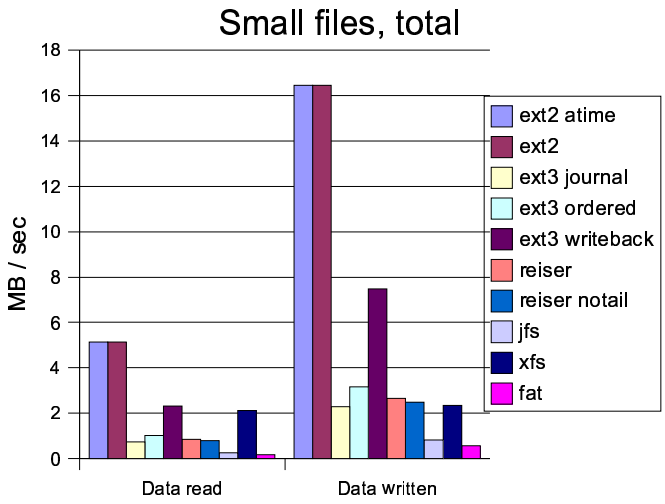
Τα αποτελέσματα της δοκιμής φαίνονται στον παρακάτω πίνακα:

size=0-10.000, number=10.000, transactions=10.000

Creation alone	Creation mixed	Read	Append	Deletion alone	Deletion mixed	Data read	Data written
3333	2528	2490	2509	10082	2479	5,14	16,44
5000	1680	1660	1673	10082	1653	5,14	16,44
1111	193	191	193	10082	190	0,73	2,28
1111	315	311	313	10082	309	1,01	3,16
1666	1008	996	1003	10082	991	2,33	7,47
5000	173	171	173	10086	170	0,85	2,65
5000	180	177	179	3360	177	0,8	2,49
1111	81	80	80	336	79	0,26	0,83
5210	235	124	123	295	191	2,12	2,35
217	58	57	58	720	57	0,18	0,57

Από τον πίνακα, προκύπτουν τα παρακάτω γραφήματα:





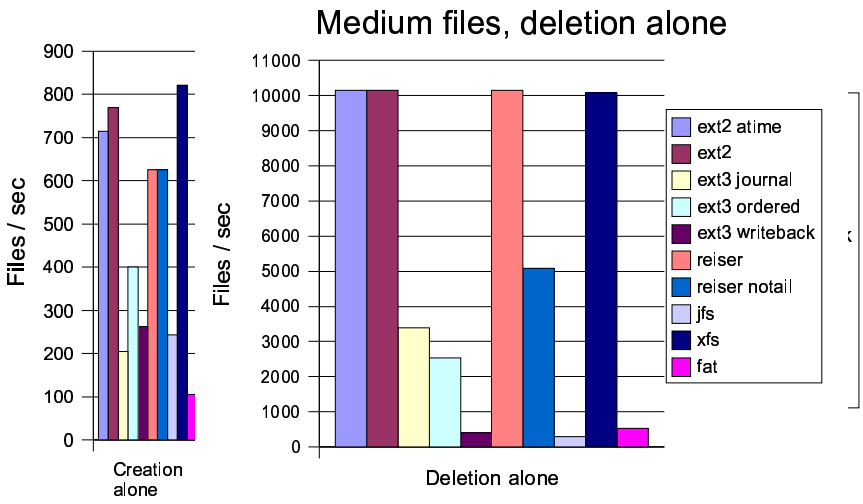
Παρατήρούμε ότι στα μικρά αρχεία, το ext2 φαίνεται να κυριαρχεί, αφήνοντας, εκτός από τη δημιουργία και τη διαγραφή πολύ πίσω όλα τα άλλα συστήματα αρχείων.

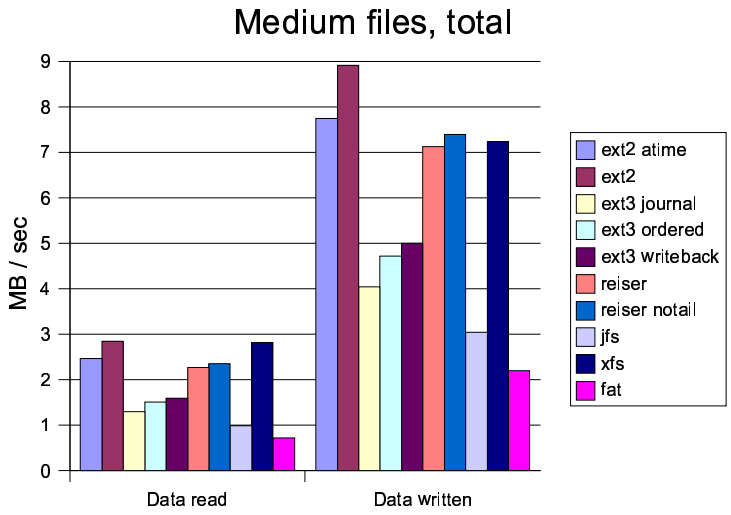
Επιπλέον, είναι ολοφάνερο το πλεονέκτημα που δίνει η παράμετρος noatime, πράγμα που είναι απόλυτα λογικό. Στα μικρά αρχεία, λόγω μικρότερου όγκου δεδομένων, η καθυστέρηση για την τροποποίηση της ημερομηνίας τελευταίας προσπέλασης είναι συγκριτικά πιο μεγάλη.

4.3.2.2 Μετρήσεις για μεσαία αρχεία

size=0-100.000, number=10.000, transactions=10.000

	Creation alone	Creation mixed	Read	Append	Deletion alone	Deletion mixed	Data read	Data written
ext2 atime	714	55	55	54	10156	54	2,46	7,74
ext2	769	65	64	64	10156	63	2,84	8,91
ext3 journal	204	33	33	33	3385	32	1,29	4,04
ext3 ordered	400	35	34	34	2539	33	1,5	4,71
ext3 writebac	263	50	49	49	406	48	1,59	5
reiser	625	51	51	50	10156	50	2,27	7,13
reiser notail	625	54	53	53	5078	52	2,35	7,39
jfs	243	26	25	25	282	25	0,99	3,04
xfs	821	71	70	70	10082	62	2,81	7,23



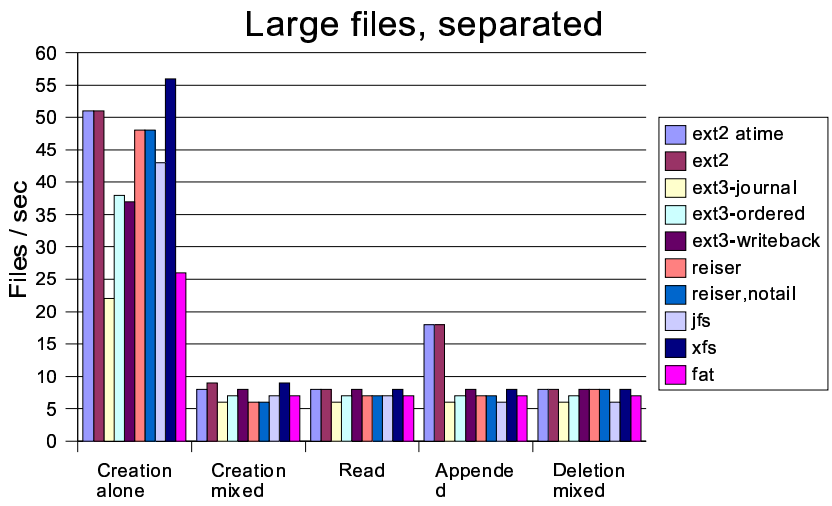


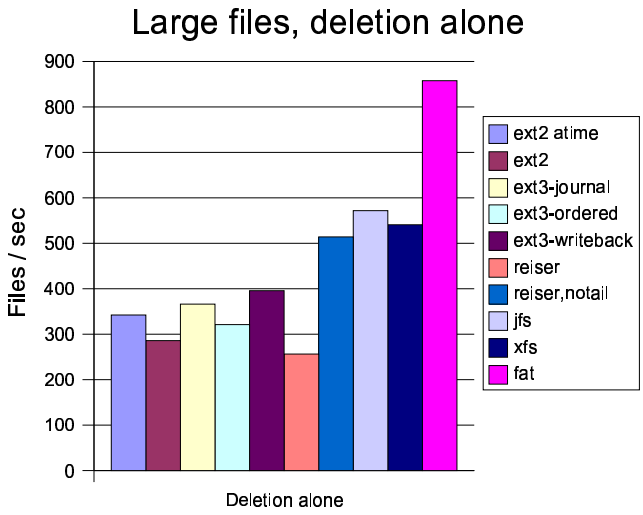
Στα μεσαία αρχεία, βλέπουμε το XFS να είναι πρώτο με διαφορά, με δεύτερο το ext2.

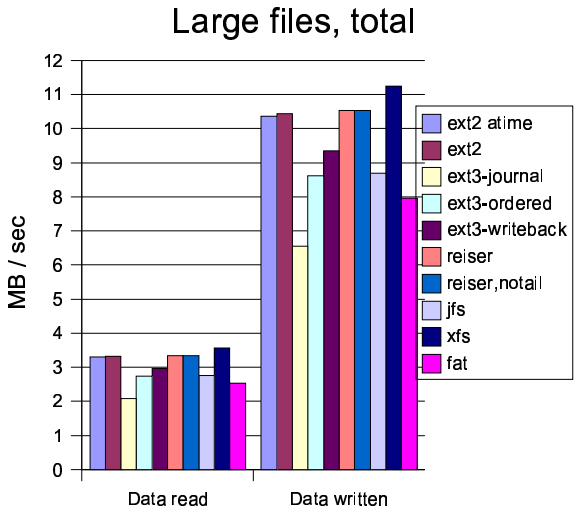
4.3.2.3 Μετρήσεις για μεγάλα αρχεία

size=0-1.00.000, number=5.000, transactions=5.000

	Creation alone	Creation mixed	Read	Appended	Deletion alone	Deletion mixed	Data read	Data written
ext2 atime	51	8	8	18	343	8	3,3	10,37
ext2	51	9	8	18	286	8	3,32	10,45
ext3-journal	22	6	6	6	367	6	2,08	6,55
ext3-ordered	38	7	7	7	321	7	2,74	8,61
ext3-writeback	37	8	8	8	396	8	2,97	9,35
reiser	48	6	7	7	256	8	3,35	10,53
reiser,notail	48	6	7	7	515	8	3,35	10,53
jfs	43	7	7	6	572	6	2,76	8,69
xfs	56	9	8	8	541	8	3,56	11,25
fat	26	7	7	7	858	7	2,53	7,97







Στα μεγάλα αρχεία, το XFS εξακολουθεί να έχει ένα προβάδισμα, μικρότερο όμως από ότι στα μεσαία.

4.3.3 Σχόλια

Βλέπουμε ότι, σε γενικές γραμμές το ext2 είναι το πιο γρήγορο σύστημα αρχείων. Ακολουθεί το XFS, ειδικά για μεσαία και μεγάλα αρχεία, και το ext3 σε writeback κατάσταση, Το Reiserfs είναι μια όχι άσχημη επιλογή ειδικά για τα μικρά αρχεία, ενώ το JFS απογοητεύει.

Οι καταστάσεις ordered και journal του ext3 είναι, στην καλύτερη περίπτωση λίγο και στη χειρότερη πολύ πιο αργές από την writeback, ενώ η επιλογή notail στο Reiserfs μας δίνει μια ελαφριά βελτίωση.

4.4 Μετρήσεις στο σύστημα αρχείων PVFS

4.4.1 Εισαγωγικά

Το PVFS ως παράλληλο σύστημα αρχείων, έχει σκοπό να βελτιώσει την ταχύτητα εισόδου/εξόδου χρησιμοποιώντας παράλληλα τους πόρους πολλών μηχανημάτων που βρίσκονται στο ίδιο τοπικό δίκτυο.

Στη δοκιμή που ακολουθεί θα δοκιμάσουμε την απόδοση του PVFS συγκρίνοντάς το με ένα απλό σύστημα αρχείων σε παρόμοιες συνθήκες. Για το σκοπό αυτό, στις κατατμήσεις της δοκιμής εγκαταστάθηκε σύστημα αρχείων ext2, πάνω από το οποίο θα λειτουργήσει το PVFS.

Εφ΄ όσον διαθέτουμε τρία όμοια μηχανήματα, θα εξεταστούν οι παρακάτω περιπτώσεις:

- Δοκιμή σε απλό σύστημα αρχείων τοπικά.
- Δοκιμή του PVFS τοπικά, με το ίδιο μηχάνημα να παίζει ταυτόχρονα το ρόλο του κόμβου διαχείρισης metadata, του διακομιστή εισόδου/εξόδου και του πελάτη.
- Δοκιμή του PVFS με το ένα μηχάνημα ως κόμβο διαχείρισης metadata και ταυτόχρονα ως πελατη, και τα άλλα δύο μηχανήματα ως διακομιστές εισόδου/εξόδου.
- Δοκιμή του PVFS με το ένα μηχάνημα ως κόμβο διαχείρισης metadata και ταυτόχρονα ως πελατη, ενώ και τα τρία μηχανήματα ως διακομιστές εισόδου/εξόδου.

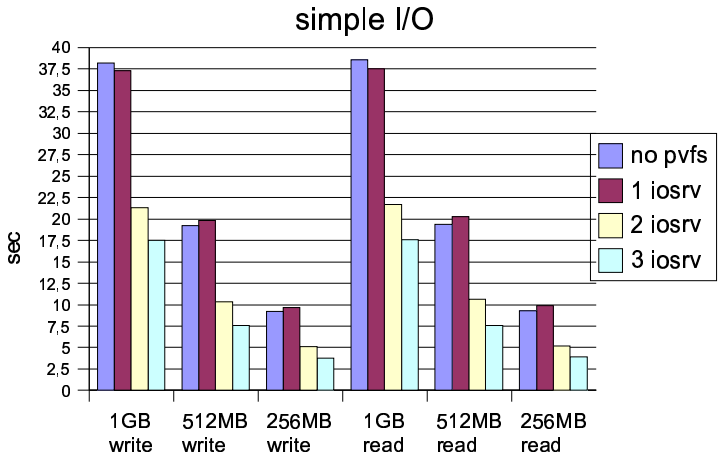
Ένα πρόβλημα που αντιμετωπίζει το PVFS είναι η χαμηλή του απόδοση όταν πρόκειται να χειριστεί buffers μικρού μεγέθους. Γι΄ αυτό και δε συνιστάται η μέτρησή του με μετροπρογράμματα όπως το Bonnie που χρησιμοποιούν τέτοιους.

4.4.2 Μετρήσεις απλής εισόδου/εξόδου

Χρησιμοποιήθηκε ένα απλό σενάριο κελύφους που κάνει χρήση της εντολής dd με μεγάλο buffer, ίσο με 1 MB για να δημιουργήσει μεγάλα αρχεία μέγεθους 256 MB, 512 MB και 1 GB. Στη συνέχεια, το αρχείο διαβάστηκε με την εντολή cat.

Τα αποτελέσματα φαίνονται στον πίνακα και στο γράφημα που ακολουθούν.

	write	write	write	read	read	read
no pvfs	38,22	19,2	9,2	38,56	19,4	9,3
1 iosrv	37,3	19,8	9,7	37,5	20,3	9,9
2 iosrv	21,3	10,3	5,1	21,7	10,6	5,2
3 iosrv	17,5	7,6	3,8	17,6	7,6	3,9



Τα αποτελέσματα είναι αρκετά λογικά και αναμενόμενα. Το PVFS τοπικά και με ένα διακομιστή εισόδου/εξόδου είναι λίγο πιο αργό από το απλό ext2. Η προσθήκη όμως δεύτερου διακομιστή εισόδου/εξόδου μειώνει το χρόνο περίπου στο μισό, ενώ η προσθήκη τρίτου, τον μειώνει περίπου στο ένα τρίτο.

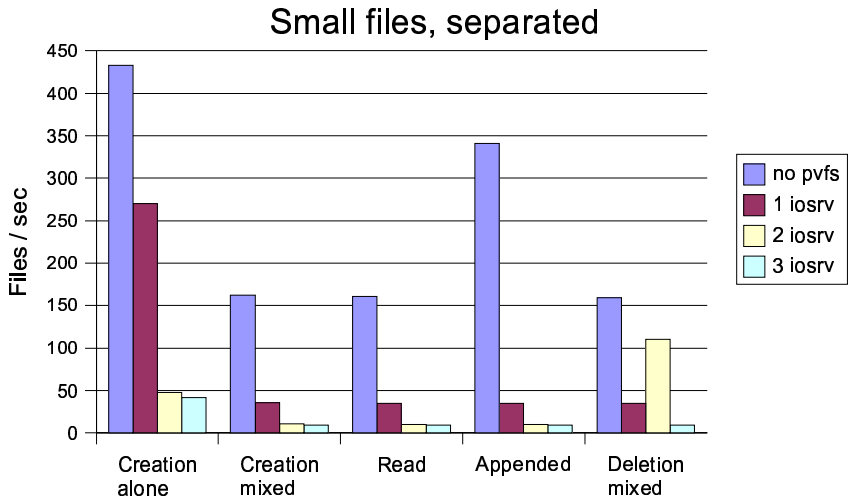
4.4.3 Μετρήσεις με το μετροπρόγραμμα postmark

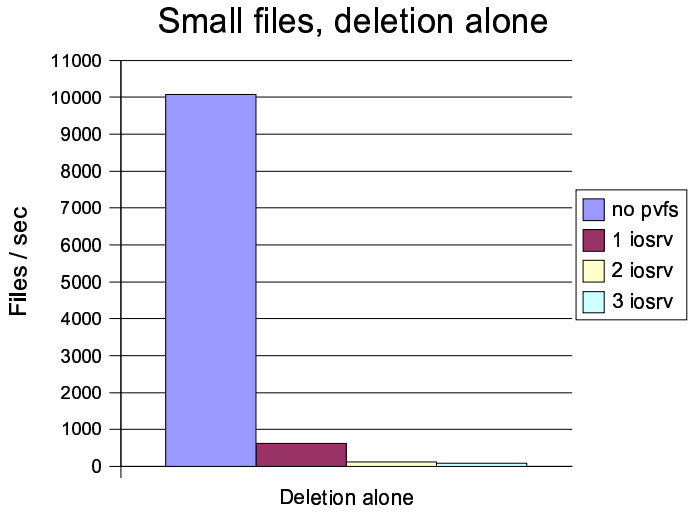
Χρησιμοποιήθηκε το ίδιο μετροπρόγραμμα με την παράγραφο 4.3 σε τρεις φάσεις:

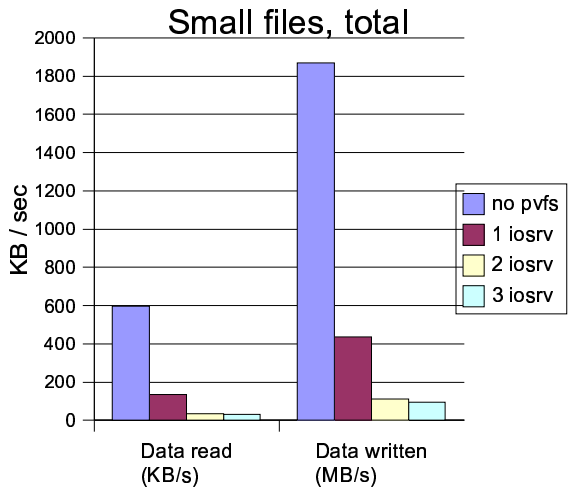
- Μικρά αρχεία, με μέγεθος από μηδενικό ως 10.000 χαρακτήρες, αριθμός αρχείων 10.000 και 10.000 συναλλαγές
- Μεσαία αρχεία, με μέγεθος από μηδενικό ως 100.000 χαρακτήρες, αριθμός αρχείων 10.000 και 10.000 συναλλαγές
- Μεγάλα αρχεία, με μέγεθος από μηδενικό ως 1.00.000 χαρακτήρες, αριθμός αρχείων 1.000 και 5.000 συναλλαγές

4.4.3.1 Μικρά αρχεία

	size=0-10.000, number=10.000, transactions=10.000						(KB/sec)	(KB/sec)
	Creation alone	Creation mixed	Read	Append	Deletion alone	Deletion mixed	Data read	Data written
no pvfs	833	162	161	341	10082	159	597,75	1870
1 iosrv	270	36	35	35	630	35	136,27	436
2 iosrv	48	11	10	10	114	110	34,93	111
3 iosrv	42	9	9	9	89	9	29,68	94,98

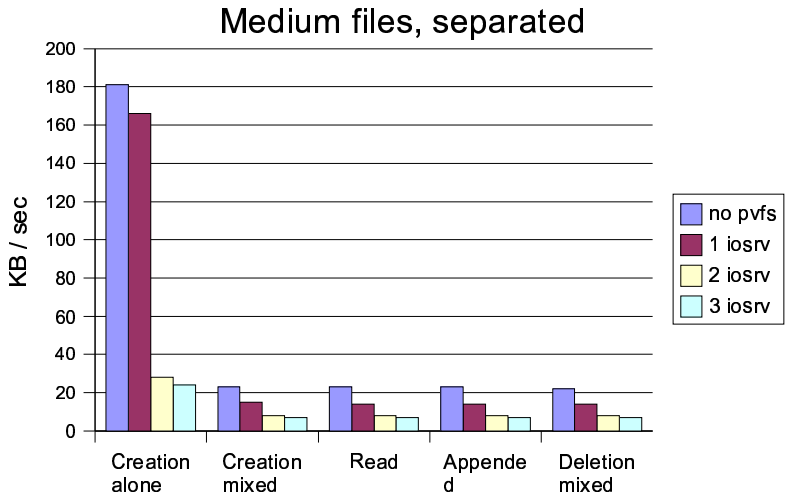


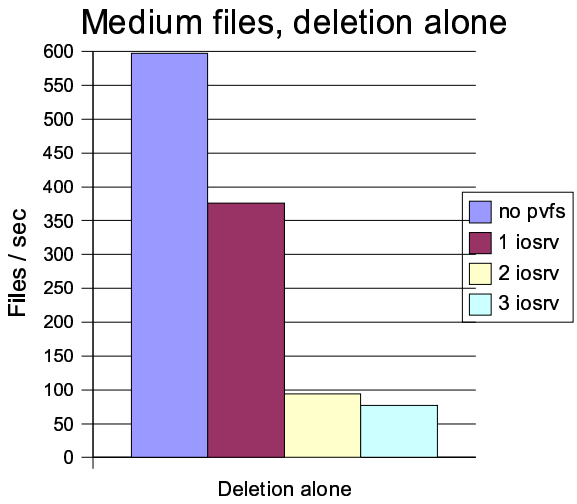


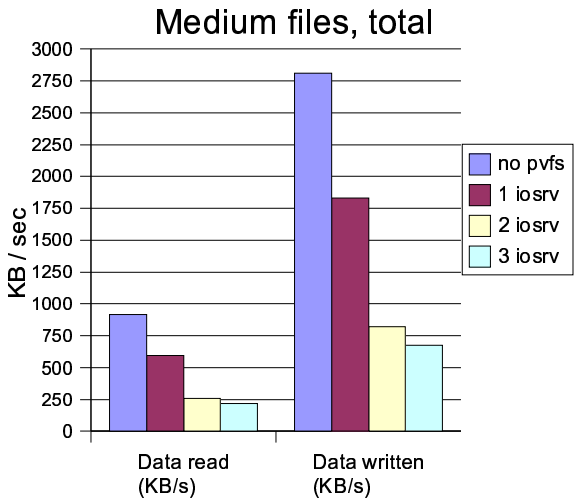


4.4.3.2 Μεσαια αρχεια

	Size=0-100.000, number=10.000, transactions=10.000						(KB/sec)	(KB/sec)
	Creation alone	Creation mixed	Read	Append	Deletion alone	Deletion mixed	Data read	Data written
no pvfs	181	23	23	23	597	22	915	2810
1 iosrv	166	15	14	14	376	14	597	1830
2 iosrv	28	8	8	8	94	8	261	822
3 iosrv	24	7	7	7	77	7	220	673

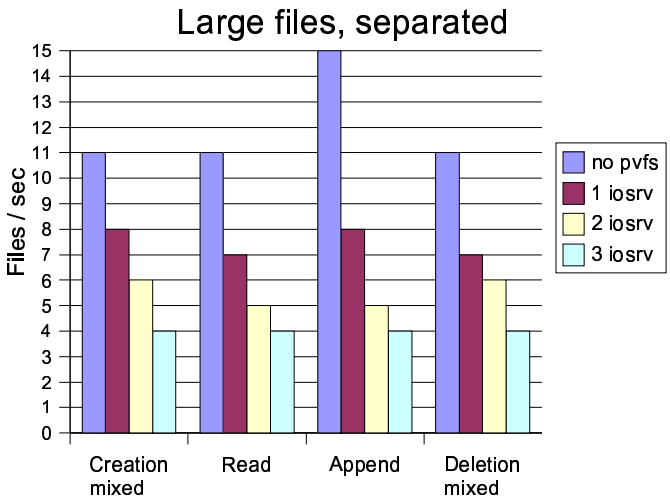


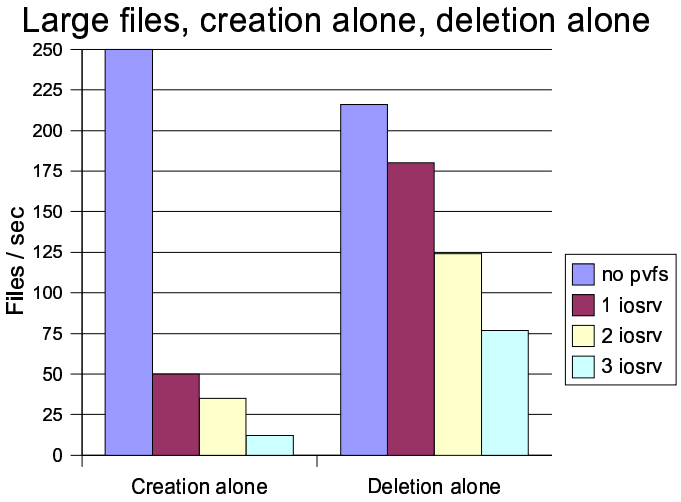


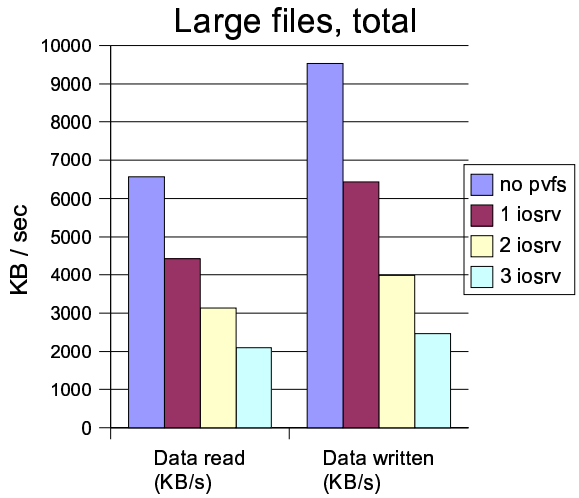


4.4.3.3 Μεγάλα αρχεία

	Size=0-1.00.000, number=1.000, transactions=5.000						(KB/sec)	(KB/sec)
	Creation alone	Creation mixed	Read	Append	Deletion alone	Deletion mixed	Data read	Data written
no pvfs	250	11	11	15	216	11	6560	9540
1 iosrv	50	8	7	8	180	7	4430	6440
2 iosrv	35	6	5	5	124	6	3127	3983
3 iosrv	12	4	4	4	77	4	2090	2459


Large files, separated
Files / sec
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Creation mixed
Read
Append
Deletion mixed
no pvfs
1 iosrv
2 iosrv
3 iosrv


Large files, creation alone, deletion alone
Files / sec
250
225
200
175
150
125
100
75
50
25
0
Creation alone
Deletion alone
no pvfs
1 iosrv
2 iosrv
3 iosrv



4.4.3.4 Σχόλια

Τα αποτελέσματα των μετρήσεων είναι πολύ απογοητευτικά για το PVFS, αφού η απόδοσή του είναι χαμηλότατη, και μάλιστα, όσο πιο πολλοί κόμβοι εισόδου/εξόδου χρησιμοποιούνται, αντί να κερδίζουμε, χάνουμε σε ταχύτητα.

Ο λόγος αυτής της χαμηλότατης απόδοσης είναι, κατά πάσα πιθανότητα, η ύπαρξη μεγάλου αριθμού αρχείων, για τα οποία χρειάζεται να φυλάσσονται πολλές αλλαγές στα metadata από τον αντίστοιχο διακομιστή, πράγμα που καθυστερούν το PVFS.

4.5 Μετρήσεις στο πρωτόκολλο NFS

Όταν πολλοί πελάτες χρησιμοποιούν ταυτόχρονα ένα διακομιστή NFS, στην καλύτερη θεωρητικά περίπτωση η ταχύτητα θα μοιράζεται σε τόσα κομμάτια όσος και ο αριθμός των πελατών, χωρίς να λαμβάνουμε υπ΄ όψη μας τη μειωμένη ταχύτητα λόγω της τυχαίας προσπέλασης στο δίσκο του διακομιστή αν οι αιτήσεις είναι πολλές και συνεχόμενες. Στην πράξη βέβαια, κάνοντας χρήση και της cache του διακομιστή, μπορεί η απόδοση να είναι κάπως μεγαλύτερη. Αυτό θα εξαρτηθεί και από το *πόσο όμοιες ως προς το περιεχόμενο είναι οι αιτήσεις*.

4.5.1 Εξαγωγή δεδομένων από αρχείο tar

Ως πρώτη δοκιμή, θα μετρήσουμε πόσο διαρκεί η εξαγωγή δεδομένων από ένα αρχείο αρχειοθέτησης τύπου tar που περιέχει τον κώδικα του πυρήνα του Linux. Τόσο το αρχείο tar όσο και τα περιεχόμενα που θα εξαχθούν θα βρίκονται στην κατάτμηση του nfs.

Δοκιμάζουμε πρώτα την εξαγωγή από ένα πελάτη, κι ύστερα από δυο πελάτες ταυτόχρονα. Στη συγκεκριμένη περίπτωση, οι εντολές που δίνονται από τους πελάτες είναι όμοιες στο κομμάτι της ανάγνωσης από το αρχείο, αλλά διαφέρουν στο κομμάτι της εγγραφής των εξαγόμενων δεδομένων.

Τα αποτελέσματα της δοκιμής φαίνονται στον παρακάτω πίνακα:

	sec
1 πελάτης	753
2 πελάτες, α	744
2 πελάτες, β	748

Δηλαδή, το NFS, όταν επρόκειτο για έναν πελάτη, έκανε χρόνο 753 sec να τον εξυπηρετήσει. Όταν επρόκειτο για δύο πελάτες έκανε πάλι περίπου τον ίδιο χρόνο και για τους δύο *ταυτόχρονα*. Αυτό δείχνει ότι λόγω των παρόμοιων αιτήσεων, έγινε caching από το NFS των πακέτων και απ΄ ευθείας εξυπηρέτηση του δεύτερου πελάτη από την cache.

4.5.2 Μετρήσεις με το μετροπρόγραμμα postmark

Στη συνέχεια, θα εκτελέσουμε το μετροπρόγραμμα postmark, με το σύνολο των 3 γνωστών δοκιμών για μικρά, μεσαία, και μεγάλα αρχεία, πρώτα από έναν πελάτη, κι ύστερα από δυο πελάτες ταυτόχρονα.

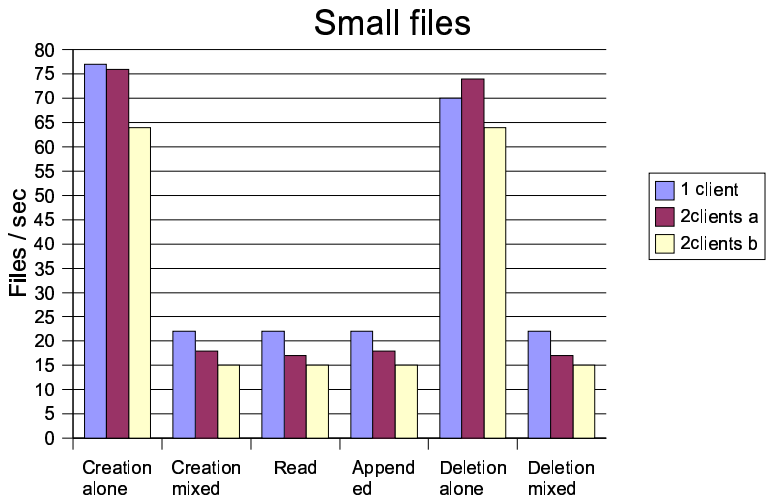
Στις δοκιμές αυτές δεν παίζει κανένα ρόλο η cache, αφού διακινείται ένα σχετικά μεγάλο πλήθος δεδομενων συγκρινόμενο με το μέγεθος της cache. Επίσης, σε καμιά περίπτωση δεν αποτελεί φραγμό το εύρος ζώνης του δικτύου μεταξύ των υπολογιστών, αφού η θεωρητική μέγιστη ταχύτητα διακίνησης δεδομένων σ΄ αυτό είναι 125 MB ανά δευτερόλεπτο (Gigabit Ethernet), δηλαδή πολύ μεγαλύτερη από τις ταχύτητες που επιτυγχάνονται.

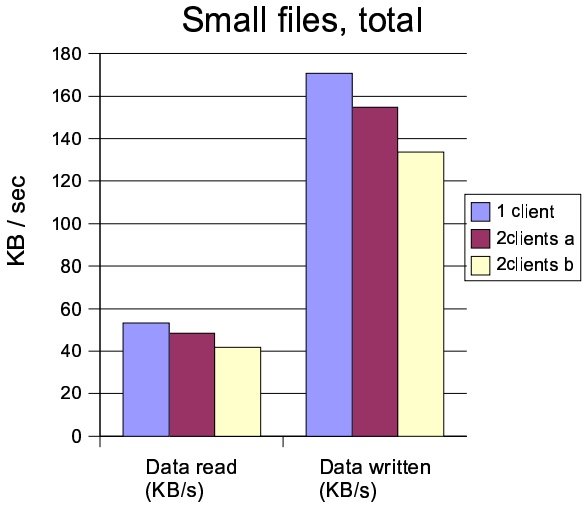
4.5.2.1 Μικρά αρχεία:

size=0-10.000, number=10.000, transactions=10.000

	Creation alone	Creation mixed	Read	Append	Deletion alone	Deletion mixed	Data read	Data written
1 client	77	22	22	22	70	22	53,35	170,69
2clients a	76	18	17	18	74	17	48,35	154,69
2clients b	64	15	15	15	64	15	41,75	133,57

Στην πρώτη γραμμή του πίνακα είναι το αποτέλεσμα της δοκιμής με έναν πελάτη, και στις 2 επόμενες το αποτέλεσμα της δοκιμής του καθενός από τους δύο που έτρεξαν ταυτόχρονα.



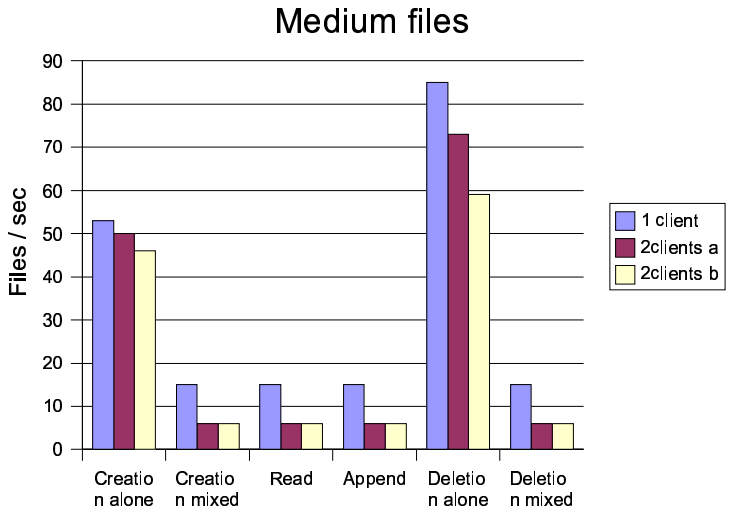


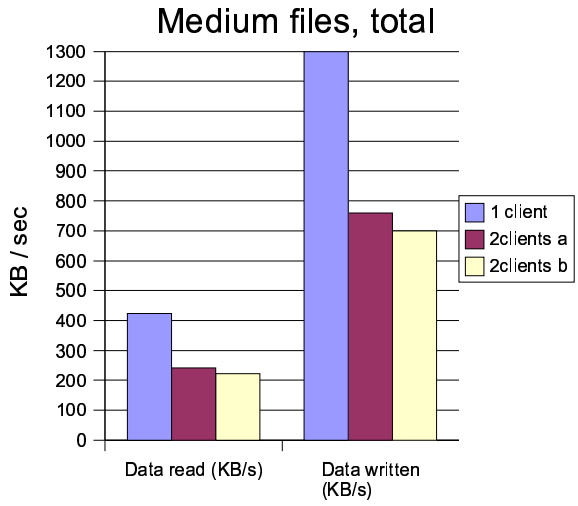
Παρατηρούμε ότι όταν τρέχουν και οι δύο πελάτες ταυτόχρονα, η μείωση των επιδόσεων είναι σχετικά μικρή σε σχέση με όταν τρέχει μόνο ένας.

Αυτό συμβαίνει γιατί στα μικρά αρχεία δεν έχουμε φτάσει ακόμα να εξαντλήσουμε τις δυνατότητες των αποθηκευτικών μέσων (σκληρών δίσκων), γιατί δε διακινούμε μεγάλες ποσότητες δεδομένων. Αυτό που καθορίζει την ταχύτητα της δοκιμής είναι καθαρά το σύστημα αρχείων και η ικανότητά του να χειριστεί πολλά μικρά αρχεία. Όταν βάζουμε παράλληλα τις αιτήσεις των δυο πελατών, είναι ουσιαστικά σαν να τρέχουμε μια δοκιμή με το διπλάσιο αρθιμό αρχείων.

4.5.2.2 Μεσαία αρχεία:

size=0-100.000, number=10.000, transactions=10.000								
	Creation alone	Creation mixed	Read	Append	Deletion alone	Deletion mixed	Data read	Data written
1 client	53	15	15	15	85	15	423,69	1300
2clients a	50	6	6	6	73	6	241,94	759,93
2clients b	46	6	6	6	59	6	222,79	699,77



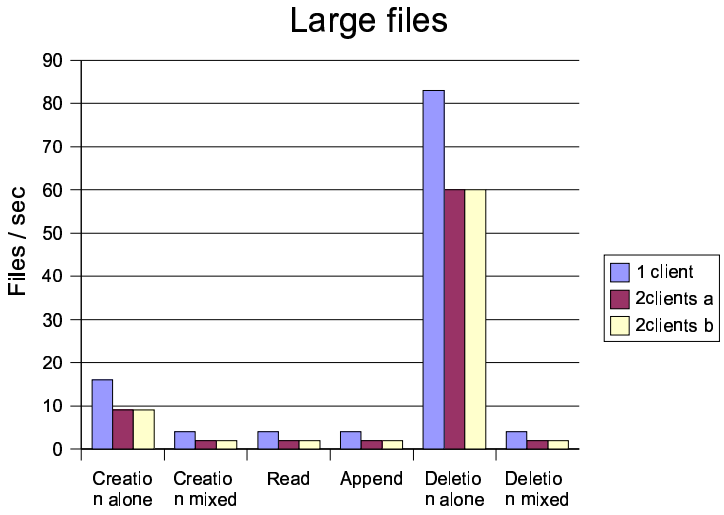


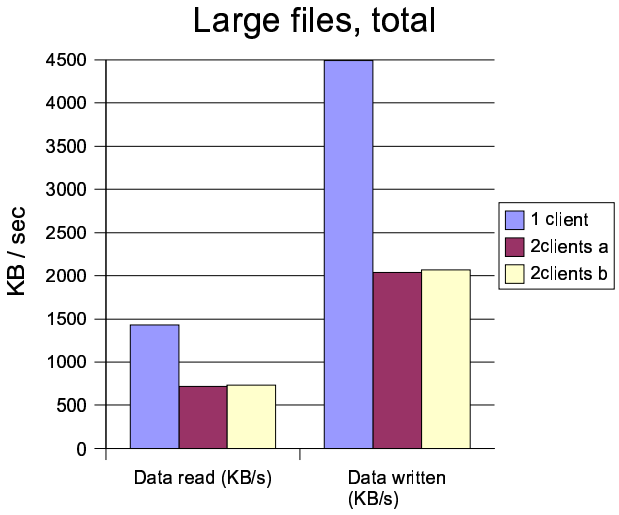
Για μεσαία μεγέθους αρχεία, αν εξαιρέσουμε τη δημιουργία και τη διαγραφή όπου και πάλι έχουμε να κάνουμε με την ικανότητα του συστήματος στο χειρισμό πολλών αρχείων, βλέπουμε ότι η προσθήκη ενός δεύτερου πελάτη παράλληλα υποδιπλασιάζει την ταχύτητα, πράγμα που είναι λογικό, καθώς εδώ ο φραγμός που αντιμετωπίζουμε είναι λόγω πλήθους δεδομένων. Ουσιαστικά, την ταχύτητα του δίσκου του διακομιστή τη μοιράζονται οι δύο πελάτες.

4.5.2.3 Μεγάλα αρχεία:

size=0-1.000.000, number=1.000, transactions=5.000

	Creation alone	Creation mixed	Read	Append	Deletion alone	Deletion mixed	Data read	Data written
1 client	16	4	4	4	83	4	1430	4490
2clients a	9	2	2	2	60	2	721	2040
2clients b	9	2	2	2	60	2	733	2065





Παρατηρούμε κι εδώ το ίδιο φαινόμενο όπως στα μεσαία αρχεία. Η ταχύτητα ανα πελάτη υποδιπλασιάζεται με την προσθήκη ενός επιπλέον πελάτη.

5. Βιβλιογραφία

Έντυπη

Andrew. S. Tanenbaum. Σύγχρονα λειτουργικά συστήματα. Εκδόσεις Παπασωτηρίου 1993.

Andrew. S. Tanenbaum. Operating systems: Design and implementation. Prentice Hall International 1987

Milan Milenkovic. Operating Systems, concepts and design, Mc Graw-Hill international editions 1992

Abraham Silberschatz & Peter Baer Galbin. Operating System Concepts. John Wiley & sons Inc, 1999

Ηλεκτρονική

Σχετικά με το Linux

About the Linux Operating System: http://www.linux.org/info/index.html

Συστήματα αρχείων γενικά

Filesystems Howto: http://www.tldp.org/HOWTO/Filesystems-HOWTO.html

Advanced filesystem implementor's guide: http://www-106.ibm.com/developerworks/library/l-fs.html

Ext2 - VFS

Design and Implementation of the Second Extended Filesystem: http://e2fsprogs.sourceforge.net/ext2intro.html

Ext3

Exploring the ext3 filesystem: http://www.linuxplanet.com/linuxplanet/reports/4136/1/

Journaling

Journaling File Systems: http://www.linux-mag.com/2002-10/jfs_01.html

Journaling File Systems: http://www.linuxgazette.com/issue55/florido.html

JFS

JFS for Linux: http://www-124.ibm.com/jfs/

JFS Overview: http://www-106.ibm.com/developerworks/library/l-jfs.html

XFS

XFS: A high-performance journaling filesystem: http://oss.sgi.com/projects/xfs/

Scalability in the XFS File System: http://oss.sgi.com/projects/xfs/papers/xfs_usenix/

Reiserfs

ReiserFS: http://www.namesys.com/v4/v4.html

An In-Depth Look at Reiserfs: http://www.linuxplanet.com/linuxplanet/tutorials/2926/1/

FAT

The FAT filesystem: http://users.win.be/W0005997/GI/fat.html

NTFS

New Technology File System (NTFS): http://www.pcguide.com/ref/hdd/file/ntfs/index.htm

PVFS

The Parallel Virtual Filesystem: http://www.parl.clemson.edu/pvfs/

NFS

NFS Backgrounder: http://www.netapp.com/tech_library/nfsbook.html

Benchmarks

textuality - Bonnie: http://www.textuality.com/bonnie/

Postmark: A new filesystem benchmark: http://www.netapp.com/tech_library/3022.html