ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ Ι

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2010-2011

$3^{η}$ Άσκηση

13 Δεκεμβρίου 2010

Ο στόχος της άσκησης είναι η εξοικείωση με θεμελιώδεις έννοιες και μηχανισμούς απαραίτητους για τη δημιουργία και διαχείριση αρχείων (files) και την κατανόηση των συστημάτων αρχείων (file systems). Χρησιμοποιώντας τα εργαλεία που προσφέρει το λειτουργικό σύστημα UNIX, καλείστε να υλοποιήσετε ένα σύστημα αρχείων (file system) ενός επιπέδου (flat file system) χωρίς ασφάλεια πρόσβασης (access control). Το σύστημα αρχείων θα υποστηρίζει έναν περιορισμένο αριθμό αρχείων και το μέγιστο μήκος του ονόματος ενός αρχείου επίσης θα είναι περιορισμένο. Το περιβάλλον που θα παραδώσετε πρέπει να αποτελείται από:

1) Μια προγραμματιστική διεπαφή (API)
2) Μια βιβλιοθήκη (library)
3) Μια ομάδα βοηθητικών εργαλείων (utilities)
4) Ένα υποτυπώδες κέλυφος (shell) μέσα από το οποίο θα εκτελούνται τα βοηθητικά εργαλεία.

Το σύστημα αρχείων χωρίζεται σε τρία μέρη: το Directory Service, το File Service και το Block Service. Στο υψηλότερο επίπεδο, ο χρήστης θα δίνει εντολές στο κέλυφος, το κέλυφος θα μετατρέπει τις εντολές του υψηλότερου επιπέδου σε κλήσεις συναρτήσεων των Directory και File Services, οι οποίες με τη σειρά τους θα καλούν τις χαμηλότερου επιπέδου συναρτήσεις του Block Service που αναλαμβάνουν τη μετακίνηση block από και προς τον "εικονικό" δίσκο.

Το κέλυφος θα είναι υποτυπώδες και θα χρησιμοποιείται για την αλληλεπίδραση του χρήστη με το σύστημα αρχείων. Κατά την εκτέλεσή του θα εμφανίζει ένα prompt όπου ο χρήστης θα μπορεί να εισάγει εντολές. Οι εντολές που υποστηρίζονται είναι οι εξής: `mkfs, ls, cp, rm, cat, echo, mount, umount, quit`.

Οι λειτουργίες του Directory Service θα περιορίζονται στην αντιστοίχιση των ονομάτων των αρχείων με i-nodes. Η προγραμματιστική διεπαφή του Directory Service πρέπει να προσφέρει τις ακόλουθες μεθόδους:

1) `int files(char **filenames);`
   Επιστρέφει το πλήθος των υπαρχόντων αρχείων που περιέχει το σύστημα και έναν πίνακα με τα ονόματα των υπαρχόντων αρχείων.
2) `int create(const char *filename);`

Δημιουργεί ένα νέο αρχείο με όνομα filename και μέγεθος 0. Αν υπάρχει αρχείο με το ίδιο όνομα τότε μηδενίζει το μέγεθος του (δηλ. διαγράφονται τα παλιά του περιεχόμενα). Επιστρέφει 0 σε επιτυχία.

3) `int delete(const char *filename);`
   Διαγράφει το αρχείο με όνομα filename και επιστρέφει 0 σε επιτυχία.
4) `int open(const char *filename);`
   Ανοίγει το αρχείο με το όνομα filename για διάβασμα ή/και γράψιμο και επιστρέφει το ufid του, το οποίο είναι ένας θετικός ακέραιος.

Η διεκπεραίωση των αιτήσεων πρόσβασης θα πρέπει να γίνεται χρησιμοποιώντας τις συναρτήσεις που παρέχει το File Service. Η προγραμματιστική διεπαφή του File Service αποτελείται από τις ακόλουθες μεθόδους:

1) `int close(int ufid);`
   Κλείνει ένα ανοιχτό αρχείο και επιστρέφει 0 σε επιτυχία.
2) `int file_size(int ufid);`
   Επιστρέφει το μέγεθος του αρχείου ufid σε bytes.
3) `int read(int ufid, char *buf, int num, int pos);`
   Αντιγράφει από τη θέση pos του ανοιχτού αρχείου ufid, num αριθμό από bytes μέσα στο buf. Επιστρέφει τον αριθμό των bytes που διαβάστηκαν (μπορεί να είναι μικρότερος από το num αν έχει φτάσει στο τέλος του αρχείου). (Σημ.: το πρώτο byte του αρχείου το διαβάζουμε με pos=0, ενώ το τελευταίο με pos=file_size(ufid)-1).
4) `int write(int ufid, char *buf, int num, int pos);`
   Αντιγράφει στη θέση pos του ανοιχτού αρχείου ufid, num αριθμό από bytes από το buf. Επιστρέφει τον αριθμό των bytes που γράφτηκαν. (Σημ.: με pos=file_size(ufid) κάνουμε append στο αρχείο).

Το σύστημα αρχείων θα διατηρεί τις απαραίτητες δομές για να εξυπηρετεί τις αιτήσεις που δέχεται χρησιμοποιώντας τις συναρτήσεις που παρέχει το Block Service. Για τις ανάγκες της άσκησης θα θεωρήσουμε ότι το μέγεθος του block στο filesystem μας είναι σταθερό και ίσο με 1024 bytes. Θα χρησιμοποιήσουμε τις παρακάτω βασικές δομές: Superblock, I-nodes bit map, Block bit map, Directory Table και I-nodes. Όλες αυτές οι δομές θα οριστούν παρακάτω βάσει του μεγέθους του block και θα υπολογιστεί πόσα blocks χρειάζεται κάθε μια από αυτές. Όλες οι παραπάνω δομές είναι αποθηκευμένες στο δίσκο, αλλά οι πρώτες τέσσερις διαβάζονται στην κύρια μνήμη κατά την αρχικοποίηση του συστήματος αρχείων (δηλαδή κατά το mount), ενώ ξαναγράφονται στο δίσκο με όποιες αλλαγές έχουν γίνει κατά την αποδέσμευση του συστήματος αρχείων (δηλαδή με το umount). Οι δομές αυτές θα πρέπει να περιέχουν τα πεδία που αναφέρονται στην παρακάτω λίστα.

1) Superblock: Θα δεσμεύει 1 block στον εικονικό δίσκο (συγκεκριμένα το πρώτο) και θα περιέχει τις παρακάτω πληροφορίες:
   a. Πλήθος των I-nodes.
   b. Μέγεθος κάθε block.
   c. Πλήθος των block που χρησιμοποιούνται για το I-nodes bit map
   d. Πλήθος των block που χρησιμοποιούνται για το Block bit map
   e. Πλήθος των block που χρησιμοποιούνται για το Directory Tab

Όπως θα φανεί στην συνέχεια, οι παραπάνω πληροφορίες θα είναι σταθερές στα πλαίσια της άσκησης και κατά συνέπεια δεν είναι απαραίτητη η χρήση του Superblock στην περίπτωση μας. Επειδή όμως σε πραγματικά filesystems το Superblock κρατάει τις πληροφορίες αυτές όπως έχουν οριστεί από τον χρήστη κατά την δημιουργία του filesystem, θα συμπεριλάβουμε για λόγους πληρότητας το Superblock και στην περίπτωση μας.

2) I-nodes bit map: Θα καταλαμβάνει 1 block στον εικονικό δίσκο. Κατά συνέπεια, μπορούμε να έχουμε το πολύ 8 bits/byte * 1024 bytes/block = 8192 I-nodes στο filesystem μας (και κατ' επέκταση το πολύ 8192 αρχεία). Το block αυτό θα χρησιμοποιείται ως bit map για να γνωρίζουμε ποια I-node είναι ελεύθερα και ποια χρησιμοποιούνται.
3) I-node: Θα περιέχει τις παρακάτω πληροφορίες:
    a. Μέγεθος αρχείου, (ένας int, 4 bytes).
    b. 15 indices σε blocks που ανήκουν στο αρχείο. Κάθε index θα είναι ένας int (4 bytes). Επομένως κάθε αρχείο θα καταλαμβάνει το πολύ 15 block και επειδή (λόγω του I-nodes bit map) μπορούμε να έχουμε το πολύ 8192 αρχεία, αυτά θα καταλαμβάνουν το πολύ 15 blocks/file * 8192 files = 122880 blocks.

   Από τα παραπάνω προκύπτει επίσης ότι κάθε I-node θα έχει μέγεθος 64 bytes και αφού μπορούμε (λόγω του I-nodes bit map) να έχουμε το πολύ 8192 I-nodes, το σύνολο των I-nodes που χρειαζόμαστε θα καταλαμβάνει 64 bytes/I-node * 8192 I-nodes = 512 * 1024 bytes = 512 blocks.
4) Directory Table: Θα περιέχει τις παρακάτω πληροφορίες για κάθε αρχείο:
    a. Το όνομα του αρχείου. Θα είναι ένα αλφαριθμητικό μήκους 12 χαρακτήρων (ο τελευταίος εκ των οποίων θα είναι ο χαρακτήρας τερματισμού αλφαριθμητικών '\0'), π.χ., `char filename[12];`
    b. I-node index: Ο αριθμός του I-node που περιέχει τις πληροφορίες για το αρχείο (ένας int, 4 bytes).

   Επομένως για κάθε αρχείο χρειαζόμαστε 16 bytes και επειδή (λόγω του I-nodes bit map) έχουμε το πολύ 8192 I-nodes, το σύνολο του Directory Table θα καταλαμβάνει 16 bytes/entry * 8192 entries = 128 * 1024 bytes = 128 blocks.
5) Block bit map: Σύνολο συνεχόμενων block που χρησιμοποιούνται ως bit map για να γνωρίζουμε ποια block είναι ελεύθερα και ποια χρησιμοποιούνται. Επειδή τα block του Superblock, I-nodes bit map, I-nodes και Directory Table είναι σταθερά, αρκεί να κρατάμε πληροφορίες για τα block των αρχείων. Όπως αναφέρθηκε παραπάνω, αυτά είναι 15 blocks/file * 8192 files = 122880 blocks. Άρα χρειαζόμαστε 122880 bits για να τα διευθυνσιοδοτήσουμε τα οποία είναι 122880/8 = 15360 bytes = 15 * 1024 bytes = 15 blocks.

Ο δίσκος, τον οποίο θα διαχειρίζεται το Block Service, θα είναι ένα "μεγάλο" αρχείο με μέγεθος ακέραιο πολλαπλάσιο του μεγέθους ενός block. Το αρχείο αυτό θα πρέπει να έχει προκατασκευαστεί, προκειμένου να υλοποιηθεί και να λειτουργήσει το σύστημα αρχείων που θα υλοποιήσετε. Ονομάζουμε αυτό το αρχείο "αρχείο του συστήματος" και πρόκειται για ένα συγκεκριμένο αρχείο που ορίζεται από τη βιβλιοθήκη. Το Block Service θα παρέχει τις απαραίτητες συναρτήσεις για τη διαχείριση των block στο αρχείο του συστήματος. Θα

πρέπει να υποστηρίζει συναρτήσεις για δέσμευση, αποδέσμευση, ανάγνωση και εγγραφή ενός block. Οι συναρτήσεις του Block Service θα πρέπει να χρησιμοποιούνται μόνο από τα Directory και File Services (δηλ. ο χρήστης δε θα έχει πρόσβαση σε αυτές τις λειτουργίες). Μπορείτε να χρησιμοποιήσετε τον αλγόριθμο της προτίμησής σας για τη δέσμευση blocks από το Block Service.

Για τη μεταφορά από και προς το αρχείο του συστήματος θα πρέπει να χρησιμοποιήσετε τις συναρτήσεις lseek(), read() και write() που προσφέρει το λειτουργικό σύστημα UNIX (δηλ. δε θα υλοποιήσετε ένα δικό σας υποτυπώδη device driver). Προσέξτε ότι η μεταφορά αυτή θα γίνεται μόνο σε επίπεδο block (έτσι θα κάνετε lseek() μόνο σε πολλαπλάσια του μεγέθους του block).

Σύμφωνα με τα παραπάνω, η μορφή του εικονικού δίσκου θα είναι όπως φαίνεται στο σχήμα:

Superblock 1 block	I-nodes bit map 1 block	Block bit map 15 blocks	I-nodes 512 blocks	Directory Table 128 blocks	Blocks for files 122880 blocks

Η ομάδα βοηθητικών εργαλείων που θα προσφέρει το σύστημα αρχείων, μέσα από το κέλυφος που θα υλοποιήσετε παρατίθεται στην παρακάτω λίστα. Σημειώνεται πως οι εντολές αυτές θα πρέπει να εκτελούνται από το υποτυπώδες κέλυφος που θα δημιουργήσετε και όχι από το κέλυφος του UNIX:

1) `mkfs`
   Δημιουργεί το αρχείο του συστήματος και αρχικοποιεί το σύστημα αρχείων (εγγραφή default τιμών στις δομές που είναι αποθηκευμένες στο superblock). Ουσιαστικά δημιουργεί ένα αρχείο συστήματος με την δομή που δίνεται στο παραπάνω σχήμα. Αν το αρχείο του συστήματος έχει ήδη αρχικοποιηθεί, τότε μηδενίζεται και αρχικοποιείται εκ νέου. (Σημ.: το όνομα του αρχείου του συστήματος ορίζεται από τη βιβλιοθήκη και δεν μπορεί να αλλάξει από το χρήστη).
2) `mount`
   Πριν μπορέσει να εκτελεστεί οποιαδήποτε εντολή, θα πρέπει να εκτελεστεί η εντολή mount, η οποία αναλαμβάνει να κάνει όλα τα allocations που πρέπει να γίνουν στην κύρια μνήμη και στη συνέχεια να διαβάσει από το δίσκο στην κύρια μνήμη τις βασικές δομές που ελέγχουν τη λειτουργία του συστήματος αρχείων.
3) `umount`
   Πριν τερματίσουμε το κέλυφος, η εντολή umount αναλαμβάνει να γράψει τις δομές ελέγχου από τη μνήμη στο δίσκο, να κλείσει τα ανοιχτά αρχεία και να αποδεσμεύσει τη μνήμη που έχει δεσμευτεί από το σύστημα αρχείων.
4) `quit`
   Προκαλεί τερματισμό του shell.
5) `ls`
   Εμφανίζει μια λίστα με τα αρχεία του συστήματος.
6) `cp source dest`
   Αντιγράφει το αρχείο source στο αρχείο dest.
7) `rm file`

Διαγράφει το αρχείο file.

8) `cat file`
   Εμφανίζει στην οθόνη τα περιεχόμενα του αρχείου file.
9) `echo file`
   Όταν δίνεται η εντολή echo, το κέλυφος παύει να δέχεται εντολές και ο χρήστης μπορεί να εισάγει ελεύθερα κείμενο (πχ. while(fgets(buf,sizeof(buf),stdin)) - τερματισμός με Ctrl-D).

Διαδικαστικά:

1) Οι ομάδες θα πρέπει να είναι οι ίδιες με αυτές της προηγούμενης άσκησης.
2) Η ημερομηνία παράδοσης της άσκησης είναι η Παρασκευή, 22 Ιανουαρίου 2011 και ώρα 23:59.
3) Ο κώδικας που θα παραδώσετε θα πρέπει να είναι καλά δομημένος.
4) Ο κώδικας που θα παραδώσετε θα πρέπει να είναι καλά σχολιασμένος.
5) Ο κώδικας θα παραδοθεί ηλεκτρονικά μέσω της σελίδας του μαθήματος.
6) Σε περίπτωση καθυστέρησης στην παράδοση της άσκησης θα υπάρχει μείωση 10% για κάθε 24 ώρες καθυστέρησης.
7) Σε περίπτωση που εντοπιστεί αντιγραφή, ο βαθμός στο μάθημα θα μηδενίζεται για όλους όσους εμπλέκονται (και αυτούς που έλαβαν την άσκηση και αυτούς που την έδωσαν).