Λειτουργικά Συστήματα

Εργασία 2

Για να αυξήσω το μέγιστο μέγεθος ενός αρχείου στο xv6 από 268 blocks σε 65803 blocks, τροποποίησα τον τρόπο με τον οποίο ορίζονται και διαχειρίζονται τα blocks ενός αρχείου. Το πρόβλημα πηγάζει από το γεγονός ότι το xv6 χρησιμοποιεί έναν inode με 12 άμεσες διευθύνσεις block και μία απλά-έμμεση διεύθυνση, η οποία μπορεί να αναφέρεται σε 256 blocks. Αυτό περιορίζει το μέγιστο μέγεθος αρχείου στα 12 + 256 = 268 blocks. Για να επιτύχω τον στόχο, εισήγαγα έναν μηχανισμό διπλά-έμμεσου block, ο οποίος επιτρέπει τη χρήση επιπλέον 256 έμμεσων blocks, καθένα από τα οποία μπορεί να δείχνει σε 256 blocks δεδομένων.

Τροποποίηση της bmap

Αρχικά, προσάρμοσα τη συνάρτηση bmap, η οποία είναι υπεύθυνη για την απεικόνιση των λογικών αριθμών block ενός αρχείου στους φυσικούς αριθμούς block στον δίσκο. Η συνάρτηση αυτή πλέον χειρίζεται τα εξής:

1. Τα πρώτα 11 blocks χρησιμοποιούν τις άμεσες διευθύνσεις στο ip->addrs.
2. Το 12ο στοιχείο του πίνακα ip->addrs χρησιμοποιείται για την αποθήκευση του απλά-έμμεσου block. Εάν το block αυτό δεν υπάρχει, δημιουργείται δυναμικά.
3. Το 13ο στοιχείο χρησιμοποιείται για το διπλά-έμμεσο block. Εάν το διπλά-έμμεσο block δεν υπάρχει, δημιουργείται. Αυτό το block περιέχει δείκτες σε απλά-έμμεσα blocks. Αντίστοιχα, κάθε απλά-έμμεσο block περιέχει δείκτες σε blocks δεδομένων.

Τροποποίηση της itrunc

Για να διασφαλίσω ότι τα blocks που αποδεσμεύονται κατά τη διαγραφή ενός αρχείου καθαρίζονται σωστά, τροποποίησα τη συνάρτηση itrunc. Εκτός από την απελευθέρωση των άμεσων και των απλά-έμμεσων blocks, πρόσθεσα κώδικα για την απελευθέρωση όλων των blocks που εμπλέκονται στο διπλά-έμμεσο block. Αυτό γίνεται μέσω διπλού βρόχου, όπου ο εξωτερικός βρόχος διαχειρίζεται τα απλά-έμμεσα blocks και ο εσωτερικός τα blocks δεδομένων που αυτά περιέχουν. Κάθε block που απελευθερώνεται αποδεσμεύεται μέσω της bfree, εξασφαλίζοντας την ορθή λειτουργία του συστήματος αρχείων.

Τροποποίηση των δομών δεδομένων

Τροποποίησα τις δομές inode(file.h) και dinode(fs.h) για να υποστηρίξουν το νέο σχήμα διευθύνσεων. Μείωσα τον αριθμό των άμεσων blocks από 12 σε 11 (με την αλλαγή της σταθεράς NDIRECT/fs.h) και πρόσθεσα το διπλά-έμμεσο block, διατηρώντας το συνολικό μέγεθος της δομής αμετάβλητο.

Για να προσθέσω υποστήριξη για συμβολικούς συνδέσμους (symbolic links) στο xv6, έπρεπε να επεκτείνω τις λειτουργίες του συστήματος αρχείων και να προσαρμόσω κρίσιμα σημεία του κώδικα. Οι συμβολικοί σύνδεσμοι είναι ειδικά αρχεία που περιέχουν το όνομα μονοπατιού ενός άλλου αρχείου ή φακέλου. Η υλοποίηση αυτή απαιτούσε τη δημιουργία της κλήσης συστήματος symlink(char *target, char *path) και την τροποποίηση της open() για τη σωστή διαχείριση των μονοπατιών που περιέχουν συμβολικούς συνδέσμους. Η λογική βασίζεται στην αποθήκευση του στόχου του συνδέσμου στο block δεδομένων ενός νέου inode με τύπο T_SYMLINK. Επίσης, χρειάστηκε να αλλάξω το Makefile , όπου προσέθεσα το symlinktest, γαι τον έλεγχο των αλλαγών.

symlink

Για τη δημιουργία της κλήσης symlink, προστέθηκε στο **stat.h** ένας νέος τύπος αρχείου T_SYMLINK για την αναγνώριση των συμβολικών συνδέσμων. Στο **user.h** προσέθεσα την αντίστοιχη κλήση συστήματος και στο **usys.pl** προσέθεσα την entry("symlink"), ώστε να μπορεί να καλείται από προγράμματα χρήστη (user programs). Στη συνέχεια, υλοποιήθηκε η λειτουργία της κλήσης συστήματος στο **sysfile.c**. Αυτή δημιουργεί έναν νέο inode με τύπο T_SYMLINK και αποθηκεύει το μήκος και το μονοπάτι του στόχου στα blocks δεδομένων του inode. Ο σχετικός κώδικας γράφει πρώτα το μήκος του μονοπατιού και ακολούθως το ίδιο το μονοπάτι. Αν η δημιουργία του inode αποτύχει, η κλήση επιστρέφει σφάλμα. Αυτό επιτρέπει τη διαχείριση των συμβολικών συνδέσμων χωρίς να χρειάζεται ο στόχος να υπάρχει κατά τη στιγμή της δημιουργίας.

Τροποποίηση της open

Η κλήση open() τροποποιήθηκε ώστε να αναγνωρίζει και να ακολουθεί αναδρομικά τους συμβολικούς συνδέσμους. Όταν ένα αρχείο τύπου T_SYMLINK συναντηθεί, η open() διαβάζει το αποθηκευμένο μονοπάτι του στόχου από το block δεδομένων του inode και το αντικαθιστά με το αρχικό μονοπάτι. Αυτή η διαδικασία επαναλαμβάνεται αναδρομικά μέχρι να βρεθεί ένα πραγματικό αρχείο ή να εντοπιστεί κύκλος. Για την αποφυγή άπειρων αναδρομών, τέθηκε όριο βάθους 10 συμβολικών συνδέσμων. Επιπλέον, προστέθηκε η σημαία O_NOFOLLOW στο **fcntl.h**, επιτρέποντας τη διαχείριση του ίδιου του συμβολικού συνδέσμου αντί του στόχου του, όταν είναι απαραίτητο. Οι αλλαγές αυτές εξασφαλίζουν ότι η open() μπορεί να διαχειριστεί μονοπάτια με συμβολικούς συνδέσμους με ασφάλεια και ακρίβεια.

Επαλήθευση και έλεγχος

Η υλοποίηση επαληθεύτηκε μέσω της εντολής bigfile, η οποία κατάφερε να δημιουργήσει αρχείο 65803 blocks και της symlinktest. Επιπλέον, όλες οι δοκιμές του usertests πέρασαν επιτυχώς, επιβεβαιώνοντας ότι το σύστημα αρχείων λειτουργεί σωστά με τα νέα χαρακτηριστικά. Ωστόσο, για τη bigfile στο σύστημά μου απαιτείται πολύς χρόνος.