ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Λειτουργικά Συστήματα

Βασιλική Χριστοφιλοπούλου
1115202000216

Χειμερινό Εξάμηνο 2024-2025

Ιανουάριος 2025
Εργασία 2

1 Big files - Double indirect nodes

Για την λύση, θα πρέπει τα πρώτα 11 στοιχεία του πίνακα ip→addrs[] θα πρέπει να είναι άμεσα blocks· το 12ο θα πρέπει να είναι ένα απλό-έμμεσο block (όπως είναι το τρέχον)· το 13ο θα πρέπει να είναι το νέο σας διπλά-έμμεσο block.

Προκειμένου να υλοποιηθεί η δομή αυτή, ακολουθήθηκαν τα βήματα που δίνονται απο τις υποδείξεις και συνεπώς τροποιποιήθηκαν τα ακόλουθα αρχεία:

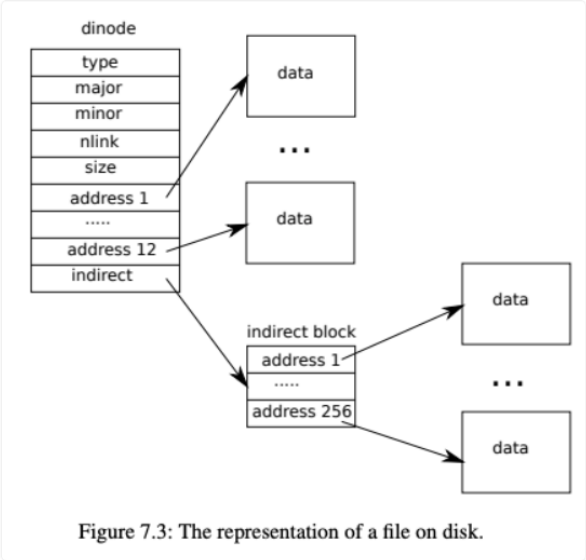

Figure 7.3: The representation of a file on disk.

Figure 1: Δομή αρχείου

1. **file.h**
   Στο αρχείο αυτό, απο την δομή inode αλλάζουμε τον ορισμό του πίνακα addrs από NDIRECT+1 σε NDIRECT+2, γιατί ο αριθμός των άμεσων blocks (NDIRECT) μειώνεται από 12 σε 11.

   ```
   uint addrs[NDIRECT+2];
   ```

2. **fs.h**
   Όμοια με παραπάνω, στην δομή dinode αλλάζουμε τον ορισμό του πίνακα addrs από NDIRECT+1 σε NDIRECT+2.
   Επιπλέον, ορίζουμε ορίζουμε τις ακόλουθες σταθερές και μακροεντολές οι οποιές χρησιμοποιούνται για την υλοποίηση των διπλά-έμμεσα blocks.

   ```
   #define NDIRECT 11                                  // Changed from 12 to 11
   #define NINDIRECT (BSIZE / sizeof(uint))            // 1024 / 4 = 256
   #define NDOUBLEINDIRECT NINDIRECT*NINDIRECT         // 256 * 256 = 65536
   #define MAXFILE (NDIRECT + NINDIRECT + NDOUBLEINDIRECT)  // 11 + 256 + 65536 = 65803
   ```

3. **fs.c**
   Στο αρχείο αυτό τροποποιήθηκαν οι συναρτήσεις:

   - **bmap(struct inode *ip, uint bn)**
     Παρατηρώντας το πώς ήταν υλοποιημένο το απλό έμμεσα block, υλοποιήσα και το διπλά έμμεσα. Αν το διπλά-έμμεσο block (στο ip → addrs[NDIRECT+1]) δεν έχει εκχωρηθεί, καλείται η συνάρτηση balloc() για να το εκχωρήσει. Στη συνέχεια, η διεύθυνση του διπλά-έμμεσου block αποθηκεύεται στο ip → addrs[NDIRECT+1].

```
if ((addr = ip->addrs[NDIRECT+1]) == 0) {
      addr = balloc(ip->dev);
      if (addr == 0)
        return 0;
      ip->addrs[NDIRECT+1] = addr;
}
```

     Το block που αντιπροσωπεύει το διπλά-έμμεσο block διαβάζεται από τον δίσκο (με τη συνάρτηση bread), και το περιεχόμενό του μετατρέπεται σε έναν πίνακα (a) από δείκτες.

```
bp = bread(ip->dev, addr); // Read the double-indirect block
a = (uint *)bp->data;
```

     Το bn / NINDIRECT υπολογίζει το index του απλά-έμμεσου block μέσα στον πίνακα του διπλά-έμμεσου block. (Ένας αριθμός block bn, μπορούμε να καταλάβουμε σε ποιο απλά-έμμεσο block βρίσκεται το block που μας ενδιαφέρει, διαιρώντας το bn με το NINDIRECT. Το NINDIRECT είναι ο αριθμός των blocks δεδομένων που μπορεί να δείξει ένα απλά-έμμεσο block)
     Αν το αντίστοιχο απλά-έμμεσο block δεν έχει εκχωρηθεί, καλείται η balloc() για να το εκχωρήσει. Η νέα διεύθυνση αποθηκεύεται στο a[bn / NINDIRECT] και καταγράφεται με τη log_write().Το διπλά-έμμεσο block απελευθερώνεται από τη μνήμη μετά την τροποποίησή του.

```
 if((addr = a[bn/NINDIRECT]) == 0)
{
  addr = balloc(ip->dev);
  if (addr == 0)
    return 0;
  a[bn/NINDIRECT] = addr;
  log_write(bp);
}
brelse(bp);
```

     Διαβάζεται το απλά-έμμεσο block (που μόλις εκχωρήθηκε ή υπήρχε ήδη). Υπολογίζεται το offset μέσα στο απλά-έμμεσο block: bn % NINDIRECT. Αν το ζητούμενο block δεν έχει εκχωρηθεί, καλείται η balloc() για να το εκχωρήσει και αποθηκεύεται στο a[bn % NINDIRECT]. (Με την πράξη bn % NINDIRECT, υπολογίζουμε το υπόλοιπο όταν διαιρούμε το bn με το NINDIRECT. Αυτό μας δίνει τη θέση του ζητούμενου block μέσα στο απλά-έμμεσο block.)

```
bp = bread(ip->dev, addr);
a = (uint*)bp->data;
     if ((addr = a[bn%(NINDIRECT)]) == 0)
     {
       addr = balloc(ip->dev);
       if (addr == 0)
         return 0;
       a[bn%(NINDIRECT)] = addr;
       log_write(bp);
     }
brelse(bp);
return addr;
}
```

   - **itrunc(struct inode *ip)**
     Με όμοιο τρόπο οπώς είχε υλοποιηθεί για το απλό έμμεσα block, υλοποιήσα και το διπλά έμμεσα. Η συνάρτηση αυτη ασχολείται με τη διαχείριση και την απελευθέρωση διπλά-έμμεσων blocks στο σύστημα αρχείων. Αυτός ο κώδικας ελευθερώνει όλα τα blocks που είναι συνδεδεμένα με το διπλά-έμμεσο

block του inode (ip): Πρώτα, ελέγχει και ελευθερώνει τα απλά-έμμεσα blocks που συνδέονται με το διπλά-έμμεσο block. Στη συνέχεια, ελευθερώνει το ίδιο το διπλά-έμμεσο block. Αυτή η διαδικασία αποδεσμεύει όλους τους πόρους (blocks) που είχαν εκχωρηθεί για την αποθήκευση του αρχείου που χρησιμοποιεί αυτό το διπλά-έμμεσο block και τα σχετικά του απλά-έμμεσα blocks.

2 Symbolic links

Για την υλοποίηση του δέυτερου ερωτήματος, ακολουθώντας τις υποδείξεις, τροποποιήθηκαν τα αρχεία:

1. **fcntl.h**
   Στο αρχείο αυτό, ακολουθώντας τις οδηγίες προστέθηκε η σημαία:
   ```
   #define O_NOFOLLOW  0x800
   ```

2. **stat.h**
   Στο αρχείο αυτό, ακολουθώντας τις οδηγίες προστέθηκε ο τύπος:
   ```
   #define T_SYMLINK  4      // Step 4: Symbolic Links
   ```

3. **user.h**
   Στο αρχείο αυτό, ακολουθώντας τις οδηγίες προστέθηκε το πρότυπο της συνάρτησης:
   ```
   int symlink(char *target, char *path);
   ```

4. **usys.pl**
   Στο αρχείο αυτό, όμοια με τις υπόλοιπες εγγραφές προστέθηκε:
   ```
   entry("symlink");
   ```

5. **syscall.h**
   Στο αρχείο αυτό, όμοια με τις υπόλοιπες εγγραφές προστέθηκε:
   ```
   #define SYS_symlink 22
   ```

6. **syscall.c**
   Στο αρχείο αυτό, όμοια με τις υπόλοιπες εγγραφές προστέθηκαν:
   ```
   extern uint64 sys_symlink(void);  // New addition
   [SYS_symlink] sys_symlink,  // New addition
   ```

7. **syscall.c**
   Στο αρχείο αυτό προστέθηκε η συνάρτηση sys_symlink(void) η οποία υλοποιεί την δημιουργία ενός symbolic link στο σύστημα αρχείων.

   (a) Αρχικά, καλείται η συνάρτηση begin_op() για να αρχίσει μια νέα λειτουργία δίσκου.
   (b) Ανακτά τις παραμέτρους target και path.
   (c) Δημιουργεί το inode για τον συμβολικό σύνδεσμο.
   (d) Γράφει τα δεδομένα στον inode: πρώτα το μήκος του target, και μετά το ίδιο το target.
   (e) Ξεκλειδώνει και αποδεσμεύει το inode.
   (f) Τερματίζει τη λειτουργία δίσκου και επιστρέφει 0 για επιτυχία ή -1 για αποτυχία.

   Επιπλέον, προκειμένου οι symbolic link να δουλέυουν σωστά, τροποποιήθηκε συνάρτηση sys_open έτσι ώστε χειρίζεται την περίπτωση όπου το path είναι συμβολικός σύνδεσμος.

   (a) Ανάκτηση των παραμέτρων.
   (b) Καλείται η συνάρτηση begin_op() για να αρχίσει μια νέα λειτουργία δίσκου.
   (c) Δημιουργία ή άνοιγμα του αρχείου.
   (d) Αν το αρχείο είναι συμβολικός σύνδεσμος, ακολουθείται ο σύνδεσμος. Για την αποφυγή ατέρμονα βρόγχου, ελέγξεται ωστε τ βάθος της αναδρομής να μην ξεπεράσει το 10.
   (e) Έλεγχος αν το αρχείο είναι φάκελος και αν η λειτουργία του ανοίγματος είναι συμβατή.
   (f) Ανάθεση ενός file descriptor και ρύθμιση των αδειών του αρχείου.
   (g) Αν απαιτείται, εκκαθαρίζεται το περιεχόμενο του αρχείου.
   (h) Ολοκλήρωση λειτουργίας και επιστροφή του file descriptor.