Tinf-zusammenfassung Expertalk

Themen:

- FAT
- UFS
- Bit Map
- Linked List
- Own Implementation

FAT:

Datensystem: FAT32

FAT32 ist wie eine LinkedList aufgebaut. Der Nextblock ist wie ein Pointer und zeigt auf den nächsten Block

Pros:

- Ganze Disk block ist verfügbar
- Zugriff ist einfacher

Cons:

- Dauert lang wegen interation

UFS:

UFS ist eazy

UFS benützt I-Nodes System

Nachteil: Je größer die Datei desto „tiefer“ muss man gehen.

Aufbau:

Die ersten Blöcke sind Metadaten.

10 Blöcke sind direkte Daten.

Single Indirect

Double Indirect

Tripple Indirect

Ein Block = 1K oder 4K

Bit Vector (Bit Map):

1 -> allocated

0 -> free

Allokieren: von links nach rechts entlang der x-Achse.

Beispiel: Block 405 freimachen:

402/(Maximale Spalte)->32 = 12,21

12… Reihe

21… Spalte

In der Reihe 12 und in der Spalte 21 schreiben wir eine „0“ hin.

Disk -> 1TB -> ungefähr 970 000 000 Blocks

Maximale größe der Bitmap = ungefähr 970 000 000 Blocks / 8

LinkedList:

Ist wie ein Block System

Jeder großer Block ist ein kleiner Block

Eigene Implementierung:

Unsere eigene Implementierung ähnelt der Unix version. Die rechte werden genau gleich, wie in der Unix variante verwaltet (octalzahlen). Die oktal Zahlen sind einer Tabelle im System eingetragen und für jeder oktal Zahl stehen die jeweiligen rechte dazu. Es wird Von dem Head bis zu dem gewünschten Block gelesen (ineffizient).

Jeder Ordner und jede Datei hat eine Metadatei in sich.

Metadatei: Grundinformationen der Datei/Ordner.

Daten Speichern:

Jeder Block hat 2 Pointer (Prev und Next/ Doppelverkette Liste). Somit zeigt der Block auf den vorigen und nächsten Block. Es wird versucht alle Blöcke in einer Reihe zu speicher (5 Blöcke reserviert -> SpeicherAddr = 0x003 bis 0x008) falls nicht möglich, zufällige Blockauswahl.

Metadaten werden in der gleichen Datei gespeichert.

Daten Löschen:

Wenn Blöcke gelöscht werden dann werden sie wieder frei.

Wenn nur ein Teil der Datei gelöscht werden die Daten aus den nächsten Blöcken verschoben um Lücken zu vermeiden.