Inhaltsverzeichnis

1 Einleitung

1.1 Ziel

Als Ergänzung zu den Inhalten der Vorlesung "Betriebsysteme" und zur Vertiefung des Systemnahen Programmierens unter C++, wurde das Dateisystem MyFs erstellt.

1.2 Technische Umgebung

Da Dateisystemverwaltung auf Kernel Ebene stattfindet, wurde File System In User Space (Fuse) als Schnittstelle zwischen dem Kernel und MyFs benutzt. Somit stand auch das Zielbetriebsystem fest, es wurde Linux aufgrund der besseren Handhabung von Fuse unter Linux. Entwickelt wurde das Dateisystem mit hilfe von XCode unter MacOS in C++. Zur Versionsverwaltung und zur Projektverwaltung wurde GitHub benutzt.

1.3 Struktur der Containerdatei

Die Containerdatei wurde intern in 5 Abschnitte unterteilt.

1. Superblock ⇒ *Informationen über das Dateisystem*
2. DMAP ⇒ *Übersicht freier Datenblöcke*
3. FAT ⇒ *Verlinkung von Datenblöcken zu Dateien*
4. Root ⇒ *Informationen über Dateien*
5. Datenblöcke ⇒ *Dateien in MyFs geteilt in Blöcke*

1.4 Projektstruktur

Bei der Projektplanung haben wir uns an dem MVC Pattern orientiert. Wir haben getrennte Klassen für die einzelnen Abschnitte (siehe 1.3), welche die Logik/Modell stellen. Dazu gibt es die Klasse "FilesystemIO" welche Methoden für das Schreiben in die Containerdatei, sowie das Lesen daraus zur Verfügung stellt und in dem Pattern als View angesehen werden kann. Objekte dieser Klassen werden von einer übergeordneten Klasse "MyFS" erzeugt. Diese ruft Methoden aus den Klassen auf, um die Ausführung des Programms zu steuern und kann als Control in dem MVC Pattern angesehen werden.

2 Implementierung

2.1 Superblock

Der Superblock enthält wichtige Informationen über das Dateisystem an sich. In ihm werden sowohl Informationen die für das Betriebssystem und den Benutzer relevant sind wie Größe des Dateisystems, der freie Speicherplatz und maximale Speichergröße. Als auch für den Betrieb des Dateisystems wichtige Daten wie die jeweiligen Startadressen der Bestandteile und die Größe dieser. Von der Implementierung handelt es sich um eine Struct mit 16 bit unsigned Integer werten.

2.2 DMAP

Die DMAP ist dazu da, einen Überblick zu schaffen, welche Datenblöcke frei sind. Im Endeffekt handelt es sich um einen Integer Array dessen Länge, der Anzahl vorhandenen Datenblöcke im Dateisystem entspricht. Dabei bedeutet eine "0" an Stelle x im Array, dass der x-te Datenblock frei ist, dementsprechend eine "1", dass der Datenblock belegt ist.
Die Klasse DMAP stellt 5 öffentliche Methoden, es können Blöcke als belegt und frei gesetzt werden. Außerdem gibt die Methode "getFreeBlock" den nächsten freien Block zurück. Wir haben uns dazu entschlossen, den Wert für den nächsten freien Block jederzeit im Hintergrund als private Variable zu halten und erst nachdem dieser Block beschrieben wurde, den neuen nächsten freien Block zu berechnen. Für die Initialisierung, wurden zwei Methoden implementiert "getAll" welchen den kompletten DMAP Array zurückgibt, und "setAll" welche den DMAP Array mit dem übergebenen Array überschreibt. Diese zwei Methoden werden benutzt, um die komplette DMAP in die Containerdatei auf der Festplatte zu schreiben bzw. sie wiederherzustellen.

2.3 FAT

Die Aufgabe der FAT ist die Verbindung zwischen einzelner Datenblöcke zu einer Datei herzustellen. Die FAT realisieren wir als einen Array. Die Größe entspricht dabei der Menge an Datenblöcke die unser Dateisystem zur Verfügung hat. In unserem Fall 65535. Somit ist gewährleistet, dass zu jeder Datei ein Index in der FAT vorhanden ist. Die Verknüpfung erfolgt dabei indem der Wert des Arrays an einer bestimmten Stelle denn Index des nächsten Datenblocks beinhaltet. Hinzu kommt noch ein Terminierendes Zeichen, welches zeigt, dass die Datei zu ende ist, bei uns erfüllt diese Aufgabe der Index 65535 da dieser außerhalb des Arrays liegt. Zum auslesen der

FAT wurde eine Methode "iterateFat" implementiert, diese bekommt den Index des ersten Datenblockes, sowie eine leere Liste übergeben. Die Methode geht die Verkettung beginnend mit dem ersten Datenblock durch und speichert die Indexe aller zusammenhängenden Datenblöcke chronologisch in die Liste. Zum Löschen einer Datei aus der FAT wird nur der erste Datenblock an die Methode "deleteFromFAT" übergeben. In dieser wird durch die Verkettung iteriert und alle zusammenhängende Einträge gelöscht. Zum einfügen neuer Informationen in die FAT stellt diese zwei verschiedene Methoden zur Verfügung. Zum einen "addNextToFAT" dieser werden zwei Indexe übergeben, die Methode stellt eine Verkettung zwischen den ersten und dem zweiten Datenblock her. Und die zweite Methode "addLastToFAT", welche an den übergebenen Index den Terminalen Index schreibt.

2.4 ROOT

Im ROOT Abschnitt werden Dateien spezifische Informationen gespeichert wie zum Beispiel der Name, letzter Änderungszeitpunkt, Index des ersten Blocks, Größe der Datei und weitere Metadaten. Gespeichert werden die Informationen in einem Array, dessen Größe abhängig von der Maximalen Anzahl Dateien in dem Dateisystem ist, sodass es zu jeder möglichen Datei genau einen Index im Rootarrray gibt. Als Datentyp des Arrays haben wir von uns geschriebenes Struct "filestats" gewählt. Zum Auslesen der Informationen stellt die Klasse eine Methode "get" welche überladen ist, einmal kann die Methode mit dem Parameter Dateiname und einer leeren fileStats Struct aufgerufen werden und einmal mit dem Parameter Index im ROOT Array und einer leeren fileStats Struct dabei werden jeweils die Korrekte Struct aus dem Array gefunden, entweder durch Vergleich des Dateinamens oder direkt durch die Position im Array und in die übergebene leere Struct reingeschrieben. Des weiteren wurden Methoden für zum Löschen von Einträgen, zum Überschreiben einer FileStat im Array, eine Überladene Methode zum erstellen neuer leerer Einträge dabei muss der Dateiname eingegeben werden, die Eingabe der Lese/Schreibrechte ist optional.Es besteht auch die Möglichkeit zu Überprüfen, ob ein Array Eintrag unter dem übergebenen Index vorhanden ist, als auch zum auslesen des Dateinamens. Eine wichtige Methode stellt dabei die "set" Methode dar, ihr wird ein Index im Array und der Pfad zu einer Datei übergeben. Die Methode liest die Eigenschaften der Datei unter dem gegeben Pfad aus und speichert diese in einer FileStat unter dem gegebenen Index ab.