Technische Universität Ilmenau

Fakultät für Informatik und Automatisierung

Studienarbeit

Entwurf und Implementation eines Bootmanagers für MSYS

Holger Jödicke

Studiennummer 23352

Betreuer: Dr. Schindler

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Ziel

1.1 Allgemeine Zielstellung

Ziel dieser Arbeit ist das komfortable Booten von MSYS und anderen Betriebssystemen von einer oder mehreren Festplatten. Um diese Aufgabe zu erfüllen, ist es nötig, einen Bootmanager zu benutzen. Damit die im Kapitel 4.1 formulierten Anforderungen erfüllt werden, mußte ein neuer Bootmanager entworfen und implementiert werden.

1.2 MSYS

MSYS ist der Name eines Testbetriebssystems. Der Autor schreibt in [6]:

> „MSYS ist in der Version ab 2.0x ein 32-Bit-Testbetriebssystem. Es läuft auf PC-Systemen, die mindestens mit dem INTEL-Prozessor 80386 ausgestattet sind (genauso also auch auf 80486er und Pentium-Systemen). Das "M" in MSYS steht für eine ganze Reihe von (Multi-) Worten wie Multi-Tasking. . . , Multi-Prozessor, Multi-User, Multi-Media usw. "SYS" ist die Abkürzung für System."

MSYS ist noch im Entwicklungsstadium. Es existieren große Teile des Mikrokernels und einige rudimentäre Treiber, um das System zu testen. Damit MSYS auch mit anderen Betriebssystemen parallel auf einem PC betrieben werden kann, wird ein Bootmanager benötigt.

1.3 Was ist ein Bootmanager

Auf vielen PCs ist nur ein Betriebssystem installiert, dieses wird immer gestartet und es kann damit gearbeitet werden. In dieser Konfiguration ist der Bootvorgang festgelegt und es besteht nicht die Notwendigkeit diesen normalen Ablauf zu verändern. Sind mehrere Betriebssysteme auf einem Rechner installiert, dann sollen diese auch gestartet werden. Das abwechselnde Starten der verschiedenen Betriebssysteme ist aber nicht ohne zusätzliche Software möglich. Ein Bootmanager ist ein solches Programm. Mit ihm ist es möglich, während des Bootvorgangs auszuwählen, welches Betriebssystem gestartet werden soll.

2 Grundlagen

2.1 Aufbau einer Festplatte

Eine Festplatte besteht in der Regel aus mehreren magnetischen Scheiben. Für jede dieser Scheiben gibt es zwei Schreib- /Leseköpfe. Einen für die Oberseite und einen für die Unterseite. Auf

jeder Scheibenseite sind in konzentrischen Ringen die einzelnen Spuren angelegt. Jede Spur wird wiederum in einzelne Sektoren unterteilt. Um einen bestimmten Sektor zu finden, genügt es zu wissen auf welcher Spurnummer, mit welchem Kopfnummer und mit welcher Sektornummer auf die Festplatte zugegriffen werden muß. Auf heutigen PC-Systemen wird üblicherweise mit 512 Bytes großen Sektoren geareitet. Der erste Sektor einer Festplatte hat die Adresse: Spur 0, Kopf 0 und Sektor 1. Es ist zu sehen, daß die Spuren und Köpfe von 0 an gezählt werden, im Gegensatz zu den Sektoren, bei denen mit 1 begonnen wird.

Durch den oben beschriebenen Festplattenaufbau, hat sich die CHS-Notation herausgebildet. CHS ist die Abkürzung für Cylinder, Head und Sector. Damit wird die Adresse eines Sektor beschrieben. Für moderne Festplatten reicht diese Adressierung nicht mehr aus. Mit der CHS-Notation lassen sich nur maximal 8 GByte auf einer Festplatte ansprechen. Bei modernen Festplatten ist der pysikalische Aufbau transparent. Das heißt es wird nach außen ein bestimmter Aufbau „vorgegaukelt", welcher aber intern ganz anders ist. Da die wirkliche Geometrie einer Festplatte keine Rolle mehr spielt, hat sich eine neue Adressierungsart entwickelt: die LBA-Notation. Dabei werden alle Sektoren einer Festplatte, von Null beginnend, durchnummeriert. Damit können bis zu 2 TByte Daten auf einer Festplatte adressiert werden.

Um mehrere Betriebssysteme auf einer Festplatte unterzubringen, muß diese partitioniert werden. Das heißt es werden auf der Festplatte Bereiche festgelegt, welche durch die einzelnen Betriebssysteme exklusiv benutzt werden können. Solch ein Bereich wird Partition genannt. Eine Konvention beim Anlegen von Partitionen ist, daß diese immer ganze Zylinder belegen. Zu einem Zylinder werden alle Spuren einer Festplatte mit der gleichen Nummer zusammengefaßt. Das bedeutet: ein Zylinder beginnt mit Kopf 0, Sektor 1 und endet beim letzten Kopf und dem letzten Sektor pro Spurnummer.

Diese Aufteilung der Festplatte in einzelne Partitionen wird in der Partitionstabelle vermerkt. Die Partitionstabelle befindet sich am Ende des MBRs. Dieser wiederum belegt den ersten Sektor einer Festplatte. Damit wird sichergestellt, daß bei einem Wechsel der Festlatte in einen anderen Rechner, die Daten der einzelnen Partitionen immer noch vorhanden sind. Der Aufbau des MBRs und der Partitionstabelle wird in Abbildung 1 dargestellt.

Der überwiegende Teil des MBRs wird durch ein Programm belegt. Am Ende stehen 64 Bytes für die Partitionstabelle bereit, daran schliessen sich noch einmal 2 Bytes mit einer Kennung an. Mit dieser Kennung kann überprüft werden ob ein gültiger MBR vorliegt. Ist die Kennung nicht 55AAh, dann ist der MBR ungültig und der Rechner kann nicht gebootet werden. Die Partitionstabelle hat eine feste Länge und bietet damit für genau vier Partitionseinträge Platz. Jeder dieser Einträge kann die Lage und Größe einer Partition beschreiben. Damit auch mehr als vier Partitionen pro Festplatte angelegt werden können, gibt es ab DOS 3.3 erweiterte Partitionen. In diesen Partitionen können dann weitere Partitionen angelegt werden. In [5] wird auf diese Möglichkeit näher eingegangen.

Ein Partitionseintrag beginnt mit dem Bootindikator. Ist dieser gesetzt, also 80h, dann wird

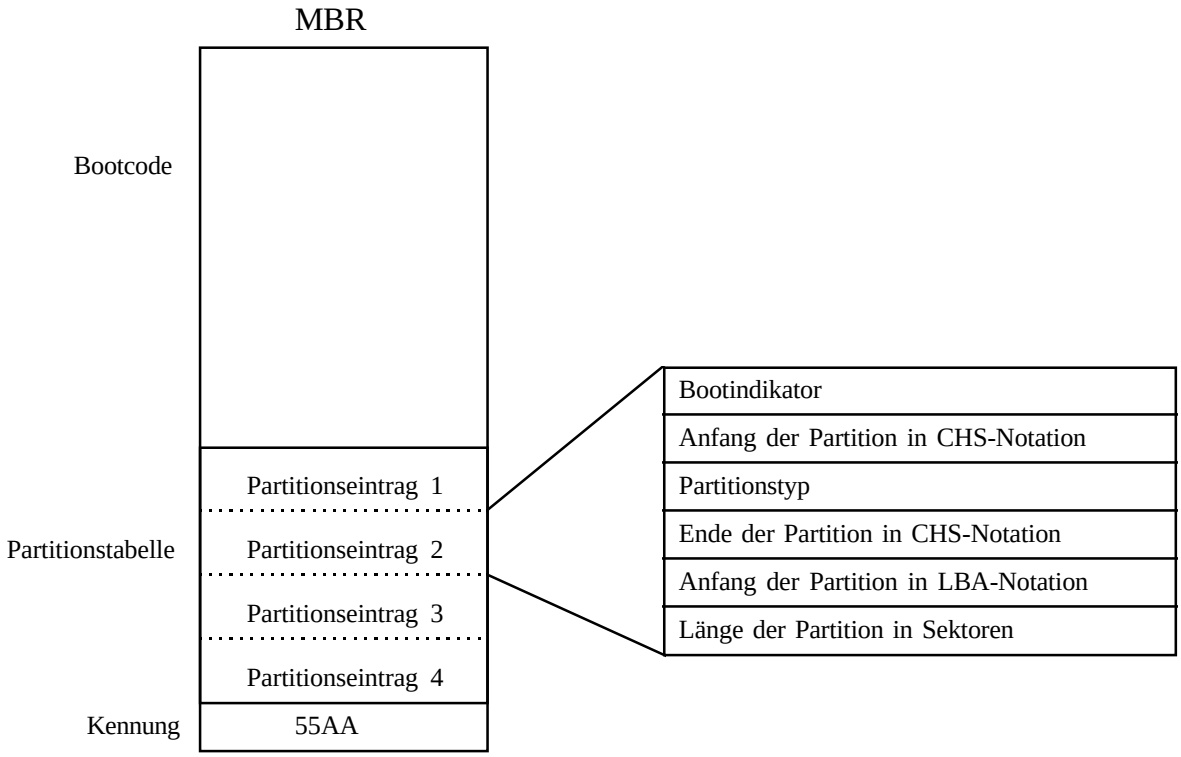

Abbildung 1: MBR mit Partitionstabelle

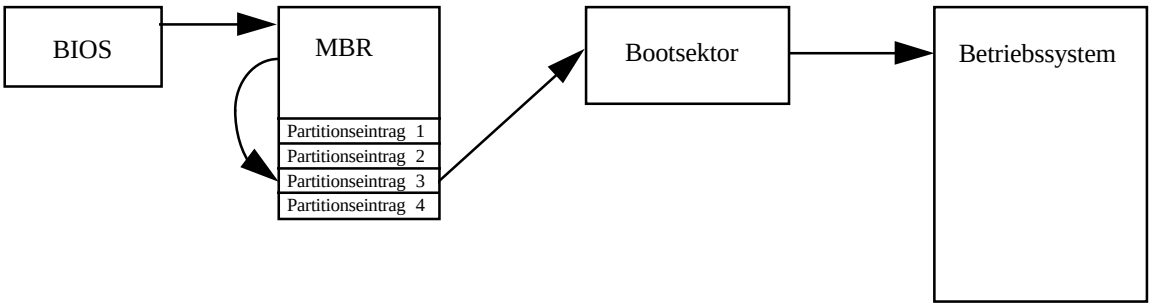

Abbildung 2: Ablauf des Bootvorgangs

diese Partition gebootet. Von den vier Partitionseinträgen darf nur einer auf 80h gesetzt sein, alle anderen müssen Null sein. Eine Partition mit gesetztem Bootindikator wird auch als aktive Partition bezeichnet. Mit dem 2. Wert in einem Partitionseintrag, wird der erste Sektor der entsprechenden Partition angegeben. Anhand des Partitionstyps kann das Filesystem der Partition bestimmt werden. Der folgende Wert legt die Größe der Partition fest. Die beiden letzten Werte legen noch einmal die Position und die Größe einer Partition fest, durch die LBA-Notation lassen sich damit aber größere Festplatten partitionieren als mit der CHS-Notation.

Auf das im MBR enthaltene Programm, wird im folgenden Kapitel eingegengen.

2.2 Der Bootvorgang am PC

Während des Bootens eines PC geschehen mehrere Dinge. In [5] wird dieser Vorgang ausführlich beschrieben. Hier soll zum besseren Verständnis nur auf die grundlegenden Dinge eingegangen werden.

Nach dem Einschalten wird ein Programm im BIOS des PCs abgearbeitet. Am Ende des Programms wird als letzter Schritt der MBR der ersten Festplatte in den Speicher geladen und gestartet. Das Programm im MBR schaut in der Partitionstabelle nach, welche Partition als aktiv markiert ist. Von dieser Partition wird dann der entsprechende Bootsektor geladen. Danach wird das in dem Bootsektor enthaltene Programm ausgeführt. Seine Aufgabe ist es, daß dazu gehörende Betriebssystem zu laden und zur Ausführung zu bringen. In Abbildung 2 wird dieser Vorgang schematisch dargestellt. Aus diesem Vorgehen ergeben sich vier Ansatzmöglichkeiten für einen Bootmanager, welche in den folgenden Kapiteln vorgestellt werden.

2.2.1 Integration von BIOS und Bootmanager

Eine Möglichkeit, einen Bootmanager zu installieren, wäre im BIOS. Es müßte nur nach dem Abarbeiten der Initial-Routinen dem Anwender ein Auswahlmenü präsentieren. Dort wird das zu bootende Betriebssystem ausgewählt. Danach wird der dazu gehörende Bootsektor geladen

und gestartet. Dieser kann dann alles Weitere übernehmen. Für das Auswahlmenü müßten einige Daten gespeichert werden, aber auch das sollte keine Probleme bereiten. Das könnte zum Beispiel im CMOS-RAM des PCs geschehen.

Dieses Verfahren müßte aber von den Hestellern des entsprechenden BIOS' implementiert werden. Von dritter Seite ist es kaum möglich einen Bootmanager nachträglich im BIOS zu installieren. Damit ist diese Alternative für den MSYS Bootmanager nicht nutzbar.

2.2.2 Installation des Bootmanagers im MBR

Es ist sinnvoll einen Bootmanager direkt in den MBR zu integrieren, da an dieser Stelle die Auswahl getroffen wird, welcher Bootsektor geladen wird. Der-MBR trifft seine Auswahl immer anhand des Bootindikators in der Partitionstabelle. Sind auf der Festplatte mehrere primäre Partitionen installiert, so kann durch das Verändern der Bootindikatoren von verschiedenen Partitionen gebootet werden. Dieses Vorgehen funktioniert nicht mit erweiterten Partitionen, da diese keinen Bootsektor haben, welcher gebootet werden kann. Daraus resultierend gab es für eine Reihe von Betriebssystemen Programme, die diesen Bootindikator verändern konnten. Damit konnte dann beim nächsten Booten des PCs von einer anderen primären Partition gebootet werden. Dieses Vorgehen ist aber umständlich, da oft erst das alte Betriebssystem gebootet werden muß. Dort wird dann eingestellt, was als nächstes zu starten ist und dann muß der Rechner noch einmal neu gebootet werden, um mit dem ausgewählten Betriebssystem arbeiten zu können.

Ein im MBR installierter Bootmanager sollte zur Bootzeit den Nutzer fragen, welches System gebootet werden soll und dann dieses System starten. Ein weiterer Vorteil gegenüber dem einfachen Verändern des Bootindikators ist die Möglichkeit, auch von anderen Festplatten oder von erweiterten Partitionen zu booten.

Problematisch bei einer Installation im MBR ist der geringe Platz. Mögliche Lösungen werden im Kapitel 2.3 aufgezeigt.

2.2.3 Installation des Bootmanagers in einen Bootsektor

Auch diese Variante ist oft anzutreffen. Sie wird verwendet, wenn zwischen dem Bootmanager und dem Betriebssystem, in dessen Bootsektor der Bootmanager installiert ist, eine enge Verbindung besteht. Dies ist zum Beispiel bei Windows NT[1] der Fall. Ein anderer Fall kann sein, daß bereits ein weiterer Bootmanager im MBR installiert ist und dieser nicht entfernt werden darf. Bei Lilo[2] besteht die Möglichkeit ihn so zu installieren, daß er im Bootsektor einer Linuxpartition integriert ist.

[1] Der Windows NT Bootmanager wird im Kapitel 3.2 vorgestellt.
[2] Lilo wird im Kapitel 3.3 vorgestellt.

Ist der Bootmanager im Bootsektor untergebracht, muß er auch die Aufgaben des Bootsektors mit übernehmen. Das bedeutet, er muß auch das entsprechende Betriebssystem direkt booten können. Um den Bootmanager in verschiedenen Bootsektoren unterzubringen, müßten für jedes dieser Betriebssysteme entsprechende Routinen entwickelt werden. Dadurch wird bei dieser Methode der Aufwand für einen unabhängigen Bootmanager sehr hoch.

Auch hier gibt es das Platzproblem, auf das weiter unten eingegangen wird.

2.2.4 Booten eines Betriebssystems von einem anderen aus

Diesem Verfahren sind durch die verschiedenen Betriebssysteme enge Schranken gesetzt. Dem Autor ist nur Linux bekannt, welches von dieser Möglichkeit Gebrauch macht. Dort existiert ein Programm, welches unter DOS ausgeführt wird und dann Linux bootet.

Bei höher entwickelten Betriebssystemen besteht die Gefahr des Datenverlustes, wenn während des laufenden Betriebes einfach ein anderes gestartet wird. Diese Systeme müssen immer kontrolliert wieder heruntergefahren werden, um einen Datenverlust zu verhindern. Weiterhin wird durch die Sicherheitsmechanismen des 'Protected Mode' ein solches Vorgehen weitgehend verhindert. Da dies bei DOS nicht der Fall ist, kann einfach der Kernel oder der Bootsektor eines anderen Betriebssystems geladen und gestartet werden.

Von DOS aus ist das Starten von anderen Systemen möglich. Damit ist aber wieder eine Abhängigkeit gegeben, die nicht gewollt ist, da in diesem Fall immer DOS benötigt wird. Hier kann auf den OS/2-Bootmanager[3] verwiesen werden, er verhält sich ähnlich. Große Teile des Bootmanagers liegen wie ein Betriebssystem in einer eigenen Partition. Da er aber direkt auf seine Aufgabe zugeschnitten ist, gibt es die Probleme mit dem 'Protected Mode' nicht.

Diese Möglichkeit eignet sich nicht, um einen voll funktionsfähigen Bootmanager zu implementieren, wenn auf ein vorhandenes Betriebssystem aufgesetzt wird. Bei Verwendung einer extra Partition treten wieder dieselben Probleme auf wie beim OS/2-Bootmanager. Daher wurde diese Methode nicht verwendet, um den Bootmanager zu entwickeln.

2.3 Die Teilung des Bootmanagers

Weiter oben wurden verschiedene Orte vorgestellt, an denen der Bootmanager installiert werden kann. Dabei zeigte es sich, daß eine Unterbringung des Bootmanagers im MBR oder im Bootsektor auf Platzprobleme stößt. Da der MSYS-Bootmanager im MBR untergebracht ist, werden im Folgenden Lösungen für das Platzproblem des MBRs vorgestellt. Ähnliche Lösungen sind auch für den Bootsektor möglich.

Der MBR besitzt eine Größe von 512 Bytes. Nach Abzug des Platzes für die Partitionstabelle und die Kennung sind noch 446 Bytes übrig. Dieser Platz kann für den Bootmanager verwendet

[3] Der OS/2 Bootmanager wird im Kapitel 3.1 vorgestellt.

werden. Auch bei effizienter Programmierung reicht der Platz nicht für einen voll funktionsfähigen Bootmanager aus. Daher besteht die Notwendigkeit, den Bootmanager in zwei Teile aufzutrennen. Der erste Teil wird im MBR installiert und hat nur die Funktion, den Hauptteil von der Festplatte nachzuladen. Dafür gibt es mehrere Möglichkeiten, welche im Folgenden vorgestellt werden.

Die erste Variante wird auch von einigen Viren und Festplattenmanagern[4] praktiziert. Da im Normalfall die erste Partition auf einer Festplatte erst auf Spur 0, Kopf 1 und Sektor 1 beginnt, sind alle Sektoren auf Spur 0 und Kopf 0, bis auf den MBR, frei. Alle bekannten Festplatten besitzen mindestens 17 Sektoren pro Spur, damit sind mindestens noch 16 Sektoren frei. Das entspricht 8 KByte Daten die dort untergebracht werden können. Dieser Platz reicht für einen kleinen in Assembler geschriebenen Bootmanager durchaus aus. Es gibt aber keinen allgemeingültigen Standard, welcher festlegt, wie Partitionen angelegt werden müssen. Daher kann nicht mit letzter Bestimmtheit angenommen werden, daß diese 16 Sektoren immer frei sind. Zum Beispiel war es mit älteren Versionen von Linux möglich, Partitionen anzulegen, die direkt nach dem MBR begannen, so daß es hinter dem MBR keinen freien Platz gab. Aus diesem Grund wurde eine derartige Implementation unterlassen.

Eine weitere Möglichkeit ist der direkte Zugriff auf das Dateisystem einer Partition. Das wird zum Beispiel vom Windows NT-Bootmanager genutzt. Auf diese Art und Weise werden auch einige Betriebssysteme von ihren Bootsektoren geladen. Dabei befindet sich der Hauptteil des Bootmanagers in einer Datei, welche auf dem Dateisystem einer Partition untergebracht ist. Die Aufgabe des MBRs oder wie bei Windows NT des Bootsektors ist es, diese Datei zu laden und auszuführen. Die Schwierigkeit besteht vor allem darin, in einem Sektor alle Funktionen unterzubringen, die nötig sind, um auf einem Dateisystem alle benötigten Operationen durchzuführen. Da dies im Allgemeinen nicht möglich ist, müssen gewisse Einschränkungen getroffen werden. Häufig anzutreffen ist die Festlegung, daß die nachzuladende Datei im Hauptverzeichnis des Dateisystems stehen muß, damit kann das „Hangeln" durch die Verzeichnisstruktur eingespart werden. Bei älteren DOS-Versionen wurde weiterhin festgelegt, daß die nachzuladenden Dateien im Hauptverzeichnis die ersten beiden Verzeichniseinträge belegen.

Bei dieser Herangehensweise ergibt sich das Problem, daß für die verschiedene Dateisysteme entsprechende Routinen geschrieben werden müßten. Damit der Platz von einem Sektor ausreicht, müssen alle Dateioperationen in Assember geschrieben sein und auch dann ist immer nur Platz für die Unterstützung von einem bestimmten Dateisystem. Um eine gewisse Unabhängigkeit zu gewährleisten, muß ein nicht unerheblicher Aufwand bei der Entwicklung und bei der Verwaltung von mehreren unterschiedlichen MBRs betrieben werden. Um den Aufwand nicht unnötig zu erhöhen, ist diese Methode nicht verwendet worden.

Die jetzt folgende Möglichkeit zum Nachladen des Hauptteils des Bootmanagers wurde verwirklicht und in den MSYS-Bootmanager eingebaut. Auch bei dieser Variante liegt der Hauptteil in

[4]Damit ist es möglich eine Festplatte mit mehr als 504 MByte in einem Rechner zu betreiben, auch wenn das BIOS diese nicht unterstützt. Der Festplattenmanager übernimmt in diesem Fall die Aufgabe des BIOS.

einer Datei vor. Es wird aber nicht versucht die Datei über das Dateisystem zu laden, sondern unabhängig vom Dateisystem wird dem Ladeteil im MBR nur der Ort, das heißt die einzelnen Sektornummern, mitgeteilt an dem die Datei zu finden ist. Von dort wird die Datei dann Sektorweise eingelesen. Durch dieses Vorgehensweise benötigt der Ladeteil keine weiteren Informationen über den Aufbau des Dateisystems. Damit ist die Unabhängigkeit vom verwendetem Betriebssystem und Dateisystem für den Bootmanager gewährleistet. Die Schwierigkeit bei dieser Methode ist, daß dem MBR immer der Ort der nachzuladenden Datei mitgeteilt werden muß. Wenn sich der Ort unerwartet verändert, ist der MBR nicht mehr in der Lage den Bootmanager nachzuladen. Der Aufenthaltsort des Bootmanager muß daher fest sein und darf sich nicht verändern.

3 Übersicht über ausgewählte Bootmanager

Es gibt die verschiedensten Bootmanager. Im Folgendem sollen drei näher vorgestellt werden. Diese werden am häufigsten verwendet und repräsentieren alle hauptsächlichen Verfahren und Methoden, um ein Betriebssystem zu booten. Es soll aufgezeigt werden, wie diese Bootmanager vorgehen und worin die Stärken und Schwächen liegen.

3.1 Der OS/2-Bootmanager

Dieser Bootmanager wird zusammen mit dem Betriebssystem OS/2 ausgeliefert. Er wird automatisch installiert, wenn sich OS/2 nicht auf der ersten Festplatte in einer primären Partition befindet. Mit ihm ist es möglich, OS/2 und andere Betriebssysteme von beliebigen Festplatten oder Partitionen zu booten.

Nur wenn OS/2 auf der ersten Festplatte in einer primären Partition installiert ist, wird ein korrekter Bootsektor installiert, der dann OS/2 starten kann. Ist OS/2 nicht dort installiert, können andere Bootmanager nicht einfach den Bootsektor laden und dann ausführen. In diesem Fall müssen diese den OS/2-Bootmanager starten, welcher dann seinerseits direkt die OS/2-Betriebssystemdateien in den Speicher lädt und ausführt. Wird doch versucht den Bootsektor zu laden und zu starten, stürzt das Programm, welches sich im Bootsektor befindet, ab.

Bei der Installation wird für den Bootmanager eine extra Partition angelegt. Die Größe dieser Partition ist abhängig von der jeweiligen Festplattengeometrie[5]. Sie kann bis zu mehreren MByte betragen. Außerdem muß es eine primäre Partition sein, welche sich auf der ersten Festplatte befindet. Dadurch wird ein Partitionseintrag in der Partitionstabelle belegt. Davon wird oft noch ein Eintrag für eine erweiterte Partition benötigt, damit sind insgesamt nur noch zwei primäre Partitionen möglich. Das genügt zwar im Allgemeinen, es kann aber Fälle geben, in denen drei

[5]Da eine Partition immer vielfache von ganze Zylindern belegen muß, kann es, in Abhängigkeit von der Kopfanzahl und der Sektoranzahl, zu viel Verschnitt kommen.

primäre Partitionen und eine erweiterte benötigt werden. Dann kann der OS/2-Bootmanager nicht mehr verwendet werden. Mehrere primäre Partitionen werden benötigt, um Betriebssysteme zu installieren, die unbedingt von primären Partitionen gestartet werden müssen[6] oder wenn einige Partitionen verdeckt sein sollen. Dies kann zum Beispiel aus Sicherheitsgründen nötig sein.

Beim Booten des Rechners wird der Bootmanager gestartet und dem Nutzer wird ein Auswahlmenü präsentiert in dem er selektieren kann, welches Betriebssystem er booten möchte. Nach der Bestätigung wird dann dieses System gestartet. Zusätzlich kann auch nach einer bestimmten Zeit, ein Betriebssystem automatisch gebootet werden.

Die Einstellungen des Bootmangers können bei der Installlation und nachträglich von OS/2 aus verändert werden. Dazu dient das Programm „Fdisk" von OS/2. Mit ihm ist es möglich Festplatten zu partitionieren und einzurichten, anschließend können diese Partitionen in den Bootmanager aufgenommen werden. Dieser Vorgang ist relativ einfach und benutzerfreundlich.

Der Bootmanager kann leider nicht frei verwendet werden, da dies die Lizenzbestimmungen nicht zulassen.

Abschließend kann gesagt werden, daß der Bootmanager gut auf seine Aufgabe zugeschnitten ist. Er erledigt diese zuverlässig und bietet auch eine komfortable Benutzer-Schnittstelle. Sein einziger Kritikpunkt ist die Verwendung einer eigenen Partition. Zusätzlich kommt aber noch hinzu, daß er nicht als Bootmanager für MSYS eigesetzt werden kann, da es die Lizenzbestimmungen nicht zulassen.

3.2 Der Windows NT Bootmanager

Bei der Installation von Windows NT wird der systemeigene Bootmanager installiert. Er ist nicht nur für den Start von anderen Betriebssystemen zu verwenden, sondern er wird auch benötigt, um Windows NT zu starten, wenn es nicht auf der ersten Festplatte in einer Primären Partition installiert ist. Dies ist ähnlich wie bei OS/2.

Der Bootmanager wird in einer primären Partition auf der ersten Festplatte installiert. Auf dieser Partition muß entweder Windows NT mit seinem eigenen Dateisystem oder ein FAT-Dateisystem installiert sein. In den Bootsektor dieser Partition wird ein entsprechender Startcode geschrieben. Dieses Programm lädt dann von der Partition den Code für den Bootmanager in den Speicher und startet ihn.

Auch bei diesem Bootmanager bekommt der Anwender ein Auswahlmenü präsentiert, aus dem er dann das gewünschte Betriebssystem auswählen kann, welches danach gestartet wird. Die Einstellungen des Bootmanagers können von Windows NT aus mit Hilfe der Sytemsteuerung verändert werden.

[6] Bei Windows 95 ist das zum Beispiel der Fall.

Außerdem besteht die Möglichkeit mit einem Texteditor die Datei „boot.ini" zu bearbeiten und dort Änderungen vorzunehmen. Beim Starten von Windows NT können vom Bootmanager aus Parameter an das Betriebsystem übergeben werden. Damit kann das Verhalten von Windows NT in gewissen Grenzen beeinflußt werden.

Der Windows NT-Bootmanager besitzt einen ähnlichen Funktionsumfang wie der OS/2-Bootmanager. Aber auch bei ihm ist der Einsatz für MSYS nicht möglich, da es die Lizenbestimmungen nicht erlauben.

3.3 LILO der Linux Loader

LILO ist der gebräuchlichste Bootmanager unter Linux. Er ist bei den meisten Distributionen enthalten. Er kann alle gebräuchlichen Betriebssysteme oder deren Bootmanager starten. Außerdem wird er benötigt um Linux selbst zu starten.

Lilo muß unter Linux installiert sein, da er dort gleichzeitig die Funktionen des Bootsektors übernimmt, um dann den Kernel zu booten. In diesem Fall genügt es, Lilo nur im Bootsektor der Linuxpartition zu installieren. Damit gibt es keine Beeinflussung von anderen Betriebssystemen und deren Bootmanagern. Ist kein anderer Bootmanager installiert und es sind mehrere Betriebssysteme auf verschiedenen Partitionen und Festplatten zu starten, so kann Lilo auch im MBR der ersten Festplatte installiert werden. Dann übernimmt er das Management über alle vorhandenen Systeme.

Wie die anderen Bootmanager lädt Lilo immer die zugehörigen Bootsektoren in den Speicher und startet diese. Es ist mit Lilo zusätzlich möglich, auch während des Bootvorgangs die Festplattenreihenfolge zu vertauschen. Damit ist es möglich OS/2 oder eine DOS-Version von der zweiten Festplatte zu booten. Der verwendete Mechanismus funktioniert mit sogenannten 'Chain Loadern'. Lilo übergibt nach seiner erfolgreichen Ausführung die Kontrolle an den benötigten 'Chain Loader', welcher dann alle nötigen Handlungen veranlaßt, um das gewünschte Betriebssystem zu starten. Damit ist es auch möglich, exotische Betriebssysteme auszuführen, wenn diese sich nicht an die Normen halten. Es muß nur ein entsprechender 'Chain Loader' geschrieben werden, welcher die erforderliche Funktionalität besitzt.

Eine weitere Funktion von Lilo ist die Übergabe von Optionen über eine Kommandozeile an das zu startende Betriebssystem. Von dieser Möglichkeit macht aber nur Linux Gebrauch. Damit ist es möglich die Konfiguration des zu bootenden Kernels zu modifizieren. Dies kann zum Beispiel nötig sein, um exotische Hardware anzusprechen und entsprechend zu konfigurieren.

Während des Bootvorganges zeigt Lilo nur ein Promt an. Wenn nicht bekannt ist, welche Bootmöglichkeiten vorhanden sind, kann mit einer Spezialtaste angezeigt werden, welche Betriebssysteme zu Auswahl stehen. Danach kann der Name des zu startenden Betriebssystems eingegeben und bestätigt werden. An dieser Stelle können auch die oben besprochenen Kernelparameter

angegeben werden. Dieses Verfahren ist nicht so komfortabel wie die der anderen beiden Bootmanager.

Die Konfiguration von Lilo geschieht unter Linux. Es existiert eine Konfigurationsdatei. In dieser Datei können die zur Auswahl stehenden Betriebssystem und eventuelle Optionen eingetragen werden. Dieses Verfahren ist zwar recht mächtig, aber dafür nicht benutzerfreundlich, da der Nutzer die benötigte Syntax kennen muß. Mittlerweile existieren aber auch die ersten Oberflächen zur Konfiguration von Lilo.

Ein weiterer Vorteil von Lilo ist, daß die Quelltexte vorhanden sind. Dadurch wird eine mögliche Nutzung und Anpassung an MSYS vereinfacht. Demgegenüber ist aber zu sehen, daß die Quelltexte und auch die Funktionalität sehr auf Linux zugeschnitten sind.

4 Der MSYS Bootmanager

4.1 Entwicklungsziele für den MSYS Bootmanager

Bei der Vorstellung der obigen Bootmanager ergaben sich für jeden von ihnen einige Vorteile und auch Nachteile. Für die beiden Bootmanager von OS/2 und Windows NT ergab sich als großer Nachteil die Lizenzbestimmungen. Dadurch ist der Einsatz dieser beiden nicht für MSYS möglich. Für Lilo spricht die große Funktionalität. Im Gegensatz dazu ist die geringe Benutzerfreundlichkeit und der enge Verbund mit Linux zu sehen. Das hat zu der Entscheidung geführt, einen neuen Bootmanager zu entwickeln.

Aufbauend auf die weiter oben vorgestellten Bootmanager entstanden folgende Ziele für den MSYS Bootmanager:

1. Starten von MSYS.
2. Starten von anderen Bertriebssystemen.
3. Er soll frei vom Copyright anderer sein.
4. Er soll einfach zu bedienen sein.
5. Er soll klein und kompakt sein.
6. Er soll weitgehend unabhängig von einem Betriebssystem sein.

Die ersten beiden Punkte sind für einen modernen Bootmanager selbstverständlich. Mit dem dritten Punkt ist nicht gemeint, daß auf dem zu benutzenden Bootmanager kein Copyright im herkömlichem Sinne liegt, sondern er muß nur frei verfügbar sein und ohne lizenzrechtliche

Einschränkungen oder Lizenzgebühren benutzbar sein. Damit entfallen alle kommerziellen Bootmanager. Der vierte und fünfte Punkt sind natürlich nicht unbedingt notwendig, sie sind aber auch nicht zu vernachlässigen. Mit dem sechsten Punkt soll darauf hingewiesen werden, daß er ohne das Betriebssystem auskommt, mit dem er mitgeliefert wird. Primär sollte natürlich ein Bootmanager entstehen, der in der Lage ist, MSYS und andere Betriebssysteme zu booten und damit Benutzern die Möglichkeit zu geben, mehrere Betriebssysteme zu benutzen. Da die weitere Entwicklung von MSYS nicht feststeht, sollen die verwendeten Konzepte so allgemein gehalten sein, daß der Bootmanager auch ohne MSYS auskommen kann.

4.2 Konzepte des Bootmanagers

Im Folgenden sollen Möglichkeiten und die dazu verwendeten Konzepte des Bootmanagers zusammengefaßt werden.

Damit der Bootmanager MSYS und andere Betriebssysteme booten kann, ist er in der Lage, die Bootsektoren von verschiedenen Partitionen zu laden. Eine Unterstützung von mehreren Festplatten ist auch implementiert. Damit können alle Betriebssysteme gebootet werden, die einen funtionsfähigen Bootsektor in ihrer Partition installiert haben. Bei einigen System, die ihren eigenen Bootmanager besitzen, ist das nicht der Fall. Dazu zählen beispielsweise Windows NT und OS/2. Bei diesen Betriebssystemen kann aber der entsprechende Bootmanager gestartet werden, welcher dann das Betriebssystem lädt und startet. Damit müssen zwangsläufig mehrere Bootmanager installiert werden. Das ist ein Nachteil, läßt sich aber nicht umgehen. Um das zu vermeiden, müßte ein einheitlicher Standard entstehen, an den sich alle Betriebssysteme halten. Dieser Standard ist aber nicht in Sicht.

- Starten von beliebigen Betriebssystemen über ihre Bootsektoren.

Eine Frage bei der Entwicklung war die Entscheidung, an welcher Stelle der Bootmanager von MSYS in den Bootvorgang eingreifen soll. In Kapitel 2.2 wurden mehrere Möglichkeiten vorgestellt. Die beiden praktikabelsten sind die Eingriffe in MBR und Bootsektor. Da ein Ziel bei der Entwicklung die Unabhängigkeit von einem bestimmten Betriebssystem war, wurde der Bootmanager so entwickelt, daß er im MBR in den Bootvorgang eingreift. Damit entfällt die Notwendigkeit verschiedene Bootsektoren zu entwickeln, um die Unabhängigkeit des Bootmanagers zu gewährleisten.

- Der Bootmanager greift vom MBR aus in den Bootvorgang ein.

Wie im Kapitel 2.3 schon angedeutet, muß der Bootmanager geteilt werden, wenn er im MBR installiert werden soll. Er unterteilt sich in zwei Teile. Der erste Teil wird im MBR installiert, die dazu gehörende Datei hat den Name „MBR.bin". Diese Datei übernimmt das Nachladen und Starten des Hauptteils. Dieser erste Teil wird im Folgenden immer als Ladeteil bezeichnet. Der zweite

Teil enthält den eigentlichen Bootmanager. Diese Datei heißt „Bootman.bin". Mit der Dateiendung „.bin" soll gegenzeichnet werden, daß diese Dateien Binärdateien sind und Programmcode enthalten. Sie können aber nicht unter einem normalem Betriebssystem gestartet werden. Der Hauptteil wird geladen, in dem die von ihm belegten Sektoren direkt über ihre Sektornummern geladen werden.

- Der Bootmanager besteht aus zwei Teilen, dem Ladeteil und dem eigentlichen Bootmanager.
- Das Nachladen des Hauptteils wird durch direktes Laden der einzelnen Sektoren durchgeführt.

Die Entwicklung des MSYS Bootmanagers wurde unter DOS durchgeführt. MSYS war zu diesem Zeitpunkt noch nicht soweit entwickelt, um eine gute Programmentwicklung zu unterstützen. Als Programmiersprache wurde Assembler genutzt. Damit ist das Ziel einen kleinen und kompakten Bootmanager zu erstellen, am einfachsten zu erfüllen. Ein anderer wichtiger Grund ist, daß sich mit einem Assembler relativ leicht Binärdateien erzeugen lassen. Die meisten höheren Programmiersprachen unterstützen unter DOS nur die Erzeugung von EXE-Dateien. In diesen Dateien sind zusätzlich zum Programm noch weitere Informationen enthalten, welche von Betriebssystem benötigt werden. Mit diesen Informationen kann das BIOS aber nicht umgehen, so daß EXE-Dateien nicht für den Bootmanager eingesetzt werden können. Durch den Einsatz von Assembler wurde aber auch der Programmieraufwand erhöht.

- Einsatz von Assembler, um einen kleinen und kompakten Bootmanager zu entwickeln.

Damit der Bootmanager möglichst einfach zu bedienen ist, wird dem Anwender ein Menü präsentiert. In diesem Menü kann das gewünschte Betriebssystem ausgewählt werden.

- Anzeige eines Menüs zur Auswahl des Betriebssystems.

4.3 Aufbau des Bootmanagers

In Abbildung 3 wird der Aufbau des Ladeteils und des Bootmanagers gezeigt. In dem gezeigten Fall ist der Bootmanager auch auf der ersten Festplatte installiert. Es besteht aber auch die Möglichkeit den Bootmanager auf einer anderen Festplatte einzurichten.

Der MBR mit installierter Laderoutine ist ähnlich aufgebaut wie ein Standard-MBR, wie er zum Beispiel mit dem „Fdisk" von DOS installiert wird. Zuerst kommt der Programmcode des Bootmanagers und zum Schluß die Partitionstabelle. Zusätzlich gib es einen Eintrag von vier Bytes. Das ist ein Verweis auf den ersten Sektor des Bootmanagers. Das erste Byte enthält die Nummer der Festplatte, auf der sich der Bootmanager befindet. Die restlichen drei Bytes

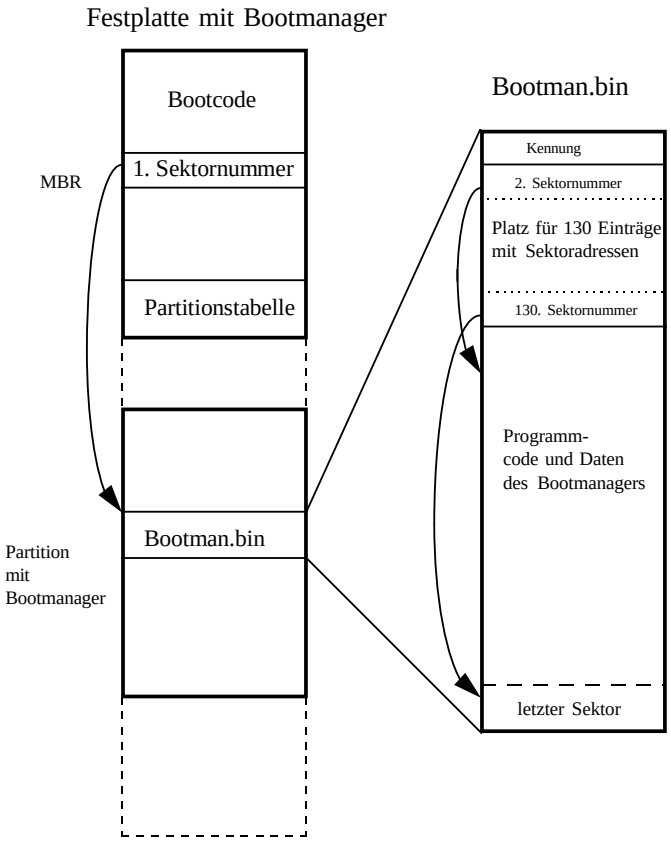

Abbildung 3: Aufbau des Bootmanagers und seines Ladeteils

enthalten die Adresse des Sektors auf der entsprechenden Festplatte in der Reihenfolge Kopf, Sektor und Spur. Mit diesen Angaben ist es möglich, den entsprechenden Sektor in den Speicher zu laden.

In diesem ersten Sektor des Bootmanagers befindet sich am Anfang eine Kennung, mit der geprüft wird, ob der korrekte Sektor geladen wurde. Es kann zum Beispiel passieren, daß die Position des Bootmanagers durch ein anderes Programm verändert wurde. In diesem Fall kann und darf der Ladeteil nicht den Bootmanager nachladen, sondern er muß einfach die Partition booten, welche in der Partitionstabelle als aktiv markiert ist. Am sinnvollsten ist es, eine Partition als aktiv zu kennzeichnen, von der aus die richtige Adresse wieder im MBR eingetragen werden kann. Bei dem jetzigen Entwicklungsstand ist das eine DOS-Partition.

An die Kennung schließt sich ein Bereich an, in dem Platz für 130 Einträge zu je drei Bytes ist. Das sind Verweise auf die restlichen Sektoren des Bootmanagers. In ihnen ist wieder die Adresse von jeweils einem Sektor gespeichert. Da auf die Festplattennummer verzichtet werden kann, sind nur noch Kopf, Sektor und Spur angegeben. Die Festplattennummer kann weggelassen werden, da normalerweise eine Datei nicht über mehrere Festplatten verteilt wird. Falls in Zukunft Dateisysteme entstehen, bei denen das doch möglich ist, muß dafür gesorgt werden, daß die Datei des Bootmanagers nicht auf mehrere Festplatten aufgesplittet wird. Bei den zur Zeit unterstützten Dateisystemen ist das Aufsplitten einer Datei auf mehrere Festplatten nicht möglich. Mit einem Platz von 130 Einträge für jeweils einen Sektor kann der Bootmanager eine Größe von etwa 64 KByte erreichen. Diese Größe sollte bei weitem ausreichen. Momentan hat der Bootmanager eine Größe von etwas weniger als zwei KByte, dazu kommen noch einmal etwa ein bis zwei KByte Daten, abhängig von der Konfiguration des Bootmanagers. Somit sollte der Platz auch in Zukunft genügen.

Im Normalfall werden nicht alle 130 Einträge benötigt, da der Bootmanager kleiner ist. Die übrigen Einträge werden mit Null aufgefüllt. Da es keine Sektoradresse mit drei Nullen gibt, kann vom Ladeteil erkannt werden, dass die restlichen Sektoren nicht mehr belegt sind.

Im Anschluß an diese Einträge beginnen der Code und die Daten. Bei diesen Daten handelt es sich um normale Variablen und Konstanten. Im Anschluß daran folgen die Daten der einzelnen Partitionen und eventuell vorhandene Optionen. In diesen Daten sind zum Beispiel die Namen enthalten, die vom Menü angezeigt werden und die Adressen der entsprechenden Bootsektoren.

4.4 Funktionsweise des Bootmanager

In Abbildung 4 ist der vereinfachte Programmablaufplan des Ladeteils zu sehen. Darin werden die dynamischen Abfolgen deutlicher gemacht. Am Anfang werden einige Initialisierungen durchgeführt. Dabei werden einige Register und Speicherbereiche mit notwendigen Werten gesetzt. Danach wird mit Hilfe der gespeicherten Sektornummer der erste Sektor des Bootmanagers in den Speicher geladen. Wenn die Kennung stimmt und der Sektor ohne Fehler geladen werden

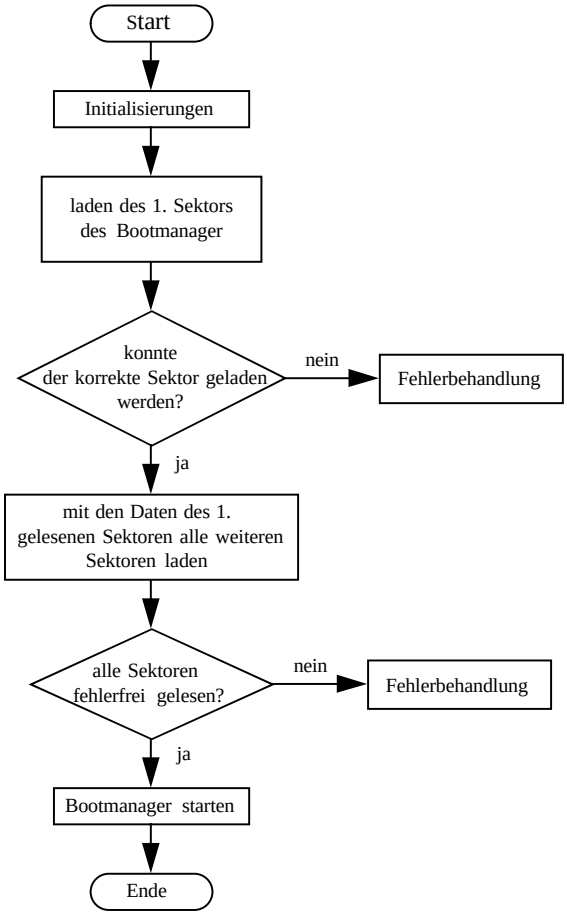

Abbildung 4: Programmablaufplan des Ladeteils

konnte, stehen nun die Nummern der nächsten Sektoren zur Verfügung. Mit diesen Informationen werden alle noch benötigten Sektoren des Bootmanagers geladen. Wenn auch dies ohne Fehler vonstatten ging, kann der eigentliche Bootmanager gestartet werden. Ist während der Abarbeitung ein Fehler aufgetreten, wird zuerst versucht die entsprechenden Sektoren noch einmal zu laden. Ist dies nicht möglich, wird versucht normal von der Festplatte zu booten. Dabei wird die Partitionstabelle nach aktiven Partitionen durchsucht. Von der erste Partition, bei der das der Fall ist, wird dann der Bootsektor geladen. Wurde dieser korrekt geladen, wird ihm die Kontrolle übergeben.

Die Abbildung 5 stellt den Programmablaufplan des Bootmanages dar. Der Bootmanager wird durch den Ladeteil gestartet. Daraufhin führt dieser zuerst einige Initialisierungen durch. Dabei werden die Partitionsdaten ausgewertet und es wird eine Systemtabelle aufgebaut, in der die Namen, die Festplattennummern und die Adressen der Bootsektoren in CHS-Notation der zu startenden Partitionenen stehen. In Abhängigkeit von den eingestellten Optionen wird eine Interruptroutine für die Uhr installiert. Weiterhin wird der Typ der Grafikkarte untersucht, um auch Monochromkarten zu unterstützen. Aus Sicherheitsgründen wird der Festplatteninterrupt untersucht, damit läßt sich feststellen ob eventuell ein Bootvirus im System vorhanden ist. Zusätzlich werden einige programminteren Variablen gesetzt.

Nach diesen Initialisierungen kommt die im Bild gezeigte Hauptschleife. Darin wird zuerst die Bildschirmanzeige auf den aktuellen Stand gebracht. Diese Funktion stellt das Menü zur Auswahl der einzelnen Partitionen dar. Danach kommt in dieser Schleife eine Funktion zur Tastaturabfrage. Wird eine für das Programm relevante Taste gedrückt, wird diese ausgewertet. Solange keine Auswahl einer Partition getroffen wurde, wird die Schleife von neuem durchlaufen. Wurde die Interruptroutine gestartet, arbeitet diese parallel zu der Hauptschleife. Ist die Uhr abgelaufen, wird die Auswahl einer Partition simuliert. Damit kann durch den Anwender oder das Ablaufen der Uhr ein Abbruch der Schleife bewirkt werden.

Nach dem Abbruch der Hauptschleife wird die Interruptroutine wieder deinstalliert. Außerdem wird der Bildschirm und der Cursor wieder hergestellt. Danach wird normalerweise das ausgewählte Betriebssystem gebootet. Es besteht aber auch die Möglichkeit von Diskette zu booten oder einen Neustart des Rechners auszulösen.

Um das ausgewählte Betriebsystem von seiner Partition zu starten, wird der zum ausgewählten Menüpunkt passende Eintrag in der Systemtabelle gesucht. Darin steht schon aufbereitet die Festplattennummer und die Sektoradresse des Partitionsanfangs. Da ein Bootsektor immer der erste Sektor einer Partition ist, muß dieser nur noch geladen und gestartet werden. Im Fehlerfall wird eine Fehlermeldung ausgegeben und der Rechner angehalten.

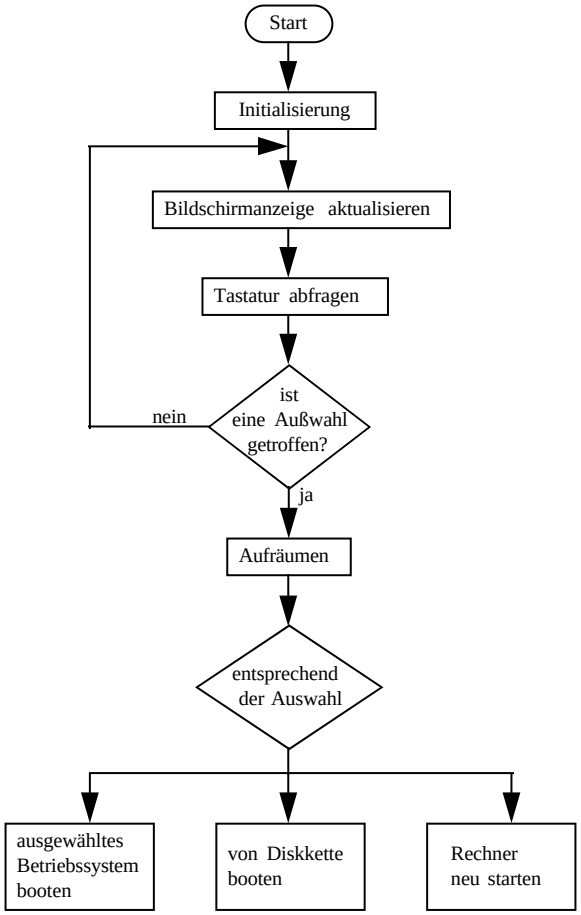

Abbildung 5: Programmablaufplan des Bootmanagers

4.5 Beurteilung des Bootmanagers

Der Bootmanager ist in der Lage MSYS und andere Betriebssysteme zu booten. Damit sind die beiden Hauptziele erreicht. Da er neu entwickelt wurde, bestehen auch keine Copyrightansprüche von dritter Seite, so dass er ohne Einschränkung für MSYS zur Verfügung steht. Der Bootmanager ist mit einer Größe von weniger als 5 KByte sehr klein. Durch das Menüsystem ist auch die Bedienung recht einfach und unkompliziert gehalten. Auch der letzte noch offene Punkt, die Unabhängigkeit von jeglichen Betriebssystemen, ist gegeben. Für das Nachladen des Hauptteils wird ein eigenständiges Verfahren verwendet und nicht das Dateisystem eines Betriebssystems. Damit wird eine größtmögliche Unabhängikeit erreicht.

Damit sind alle im Kapitel 4.1 gestellten Ziele erreicht. Für den Bootmanager selbst trifft das auch zu. Bei dieser Betrachtungsweise bleibt aber die Installation des Bootmanagers außen vor. Dafür wird ein extra Programm benötigt. Dieses Programm wird auch weiterhin immer vom einem oder mehreren Systemen abhängen. Das wird sich auch in Zukunft nicht vermeiden lassen. Auf diese Aussage und die sich daraus ergebenden Konsequenzen wird in Kapitel 5 weiter eingegangen und dabei auch das entsprechende Setup- und Installationsprogramm vorgestellt.

4.6 Mögliche weitere Entwicklungen

Ein Ziel für die weitere Entwicklung der Bootmanagers könnte es sein, Methoden zu entwickeln und zu implementieren, mit denen auch Betriebssysteme gestartet werden können, die einen fehlerhaften Bootsektor installieren. Das setzt eine genaue Kenntnis des entsprechenden Bootvorgangs voraus und ist meist nicht für andere Betriebssysteme verwendbar. Eine Möglichkeit dazu ist ein speicherresistentes Programm, welches die Festplattennummern vertauscht. Damit kann dem zu startenden Betriebssystem vorgegaukelt werden, es würde von der ersten Festplatte gebootet, statt von der zweiten.

Das Laden der einzelnen Sektoren wird immer über das BIOS abgewickelt. Damit ist der Bootmanager vom jeweiligem BIOS abhängig. Dadurch treten einige Einschränkungen auf. Bei IDE-Festplatten kann nur auf die ersten 512 MByte und bei EIDE-Festplatten nur auf die ersten 8 GByte zugegriffen werden. Bei einem neueren BIOS ist es möglich mit Hilfe einer BIOS-Erweiterung des Interrupt 13H auch Festplatten von mehr als 8 GByte anzusprechen. Dazu ist es nötig die entsprechenden Dokumentationen zu finden und auszuwerten, dann kann der Bootmanager entsprechend erweitert werden.

5 Das Setup des Bootmanagers

5.1 Aufgaben des Setups

Zu einem Bootmanager gehört immer noch ein Programm, welches diesen installiert und administriert. Daher wurde ein separates Programm entwickelt. Das Programm „Setup.exe", im weiteren immer als Setup bezeichnet, soll diese beiden Aufgaben lösen. Bei einigen Bootmanagern werden diese Aufgaben auch durch das Betriebssystem wahrgenommen. Da der Bootmanager aber nicht von MSYS unabhängig sein sollte und MSYS auch noch nicht soweit entwickelt ist, um so etwas zu unterstützen, wurde auf eine derartige Unterstützung verzichtet.

Die beiden Hauptaufgaben, die Installation und die Administration des Bootmanagers, lassen sich noch unterteilen. Im Folgenden ist die Aufteilung zu sehen.

- Vom Benutzer die Einstellungen des Bootmanagers abfragen.
- Diese abgefragten Einstellungen und Optionen im Bootmanager speichern.
- Die genaue Position und die Sektoren des Bootmanagers auf der Festplatte finden.
- Die gefundenen Sektornummern im Kopf des Bootmanagers eintragen.
- Den Ladeteil zusammen mit der Nummer des 1. Sektors des Bootmanagers und der Partitionstabelle im MBR abspeichern.

Der Benutzer kann sich im Setup anschauen, wieviele Partitionen von welchem Typ in seinem PC vorhanden sind. Den einzelnen Partitionen können Namen gegeben werden. Diese Namen werden dann vom Bootmanager angezeigt. Weiterhin können die Partitionen angegeben werden, die später vom Bootmanager als Menü präsentiert werden. Zusätzlich kann eine Zeit vorgegeben werden, nach der automatisch ein bestimmtes Betriebssystem gebootet werden soll. Alle vom Benutzer vorgenommenen Einstellungen werden zusammen mit den Partitionsinformationen an den Bootmanager angehangen.

Danach wird das Dateisystem untersucht, um die genaue Position des Bootmanagers zu bestimmen. Da die Position einer Datei nach außen transparent ist, muß das Dateisystem auf unterster Ebene mit Hilfe des BIOS durchsucht werden. Dazu ist eine genaue Kenntnis des entsprechenden Dateisystems nötig. Daran ist auch erkennbar, warum das Setup nicht system-unabhängig sein kann. Bis auf die Nummer des ersten Sektors wird die Liste mit den Sektornummern in den Kopf des Bootmanagers eingetragen (siehe Abbildung 3). Zum Schluß wird der Ladeteil gemeinsam mit der Partitionstabelle der Festplatte und der ersten Sektornummer in den MBR geschrieben.

5.2 Konzept des Setups

Beim Entwurf des Programms mußte festgelegt werden, für welches Betriebssystem es entwickelt wird. Die Entscheidung fiel zugunsten von DOS. Daher ist das Programm unter dem Betriebssystem DOS lauffähig. Das Setup ist damit nicht mehr vom Betriebssystem unabhängig, wie der Bootmanager. Die Entscheidung das Setup für DOS zu entwickeln, wurde auf Grund der weiten Verbreitung von DOS und dessen Dateisystems getroffen. Außerdem unterstützt MSYS momentan nur das FAT-Dateisystem, welches von DOS verwendet wird. Eine weitere Tatsache ist, daß unter DOS der direkte Zugriff auf das BIOS ohne Weiteres möglich ist. Bei anderen Betriebssystemen ist damit meist ein höherer Aufwand verbunden.

- Entwicklung des Setup für DOS.

Aus der Unterstützung von DOS resultiert, daß das Setup in der Lage ist, den Bootmanager auf einem FAT12 oder FAT16 Dateisystem einzurichten. Da die Algorithmen zum Zugriff auf das Dateisystem existieren, ist beim Vorhandensein einer gut funktionierenden Hochsprache eine Portierung auf ein anders Betriebssystem, wie MSYS, relativ einfach durchzuführen. Größere Probleme dürfte eine Portierung auf ein anderes Dateisystem mit sich bringen. Für die Unterstützung eines anderen Dateisystems müßte ein Teil der Algotithmen neu entworfen und implementiert werden. Daher ist in diesem Fall der Aufwand höher.

- Unterstützung von FAT12 und FAT16 Dateisystemen.

Da es beim Setup nicht auf die Gößе und Geschwindigkeit ankam, sondern mehr darauf zu zeigen, wie ein mögliches Installationsprogramm aussehen und funktionieren könnte, wurde auf Pascal zurückgegriffen. Damit waren auch zwei angenehme Nebeneffekte verbunden. Zum einen ist die Lesbarkeit und Verständlichkeit des Quelltextes besser als bei anderen Programmiersprachen, wie zum Beispiel C und zum anderen konnten auch schon vorhandene Programmteile eingebunden werden.

- Verwendung von Pascal als Programmierstprache.

Das Setup ist stark modularisiert. Unterstützt wird das auch durch das Unit-Konzept von Pascal. Dadurch ist es möglich, bestimmte Teile des Setups einfach auszutauschen. Nach einer ersten einfachen Version der Benutzeroberfläche, wurde eine zweite Version geschrieben. Da die gesamte Funktionalität der Oberfläche in einer eigenen Unit liegt, war ein einfaches Wechseln der Oberflächen möglich.

5.3 Beurteilung des Setups und mögliche weitere Entwicklungen

Die primäre Aufgabe, das Installieren des Bootmanagers, wird durch das Setup ordentlich durchgeführt. Auch das Einstellen aller notwendigen Angaben und das Zusammenwirken mit dem

eigentlichen Bootmanager ist damit möglich. Damit sind alle Anforderungen erfüllt.

Da der zeitliche Umfang für die Studienarbeit beschränkt ist, mußte beim Setup der Schwerpunkt auf der Implementation der Funktionalität liegen. Bei der weiteren Entwicklung des Setups sollte ein Schwerpunkt die Benutzeroberfläche und die Benutzerführung sein. Zusätzlich ist es auch sinnvoll, eine Online-Hilfe einzubauen.

6 Weitere Ergebnise

Bei der Durchführung dieser Arbeit ergaben sich einige weitere Ergebnise. Im Folgendem sollen diese kurz vorgestellt werden.

6.1 Einige Worte zu Sicherheitsaspekten

Bei der Durchführung dieses Projektes mußte besonders auf den Schutz der Festplatte und der darauf befindlichen Daten Wert gelegt werden. Da beim Verändern oder sogar Löschen des MBRs mit seiner Partitionstabelle der Computer nicht mehr startet, ist darauf ein besonderes Augenmerk zu richten. Aber nicht nur der MBR enthält sensitive Daten, es gibt noch mehr Orte an denen auch schon kleine Veränderungen zum Datenverlust führen können.

Um das Risiko möglichst klein zu halten, wurden mehrere kleine Programme geschrieben. Die meisten bauen direkt auf der Bootunit auf. Einige dienen nur der Information des Nutzers, mit anderen ist auch eine Manipulation von Festplatten möglich.

Am Anfang der Entwicklung des Bootmanagers war es zu Testzwecken oft nötig einen neuen MBR zu installieren. Wenn noch Fehler in diesem MBR enthalten waren, wurde es nötig den alten MBR wieder zu installieren. Dazu wurden zwei kleine Programme geschrieben, die diese Aufgabe wahrnehmen.

- Save_MBR.exe
- Load_MBR.exe

Diese beiden Programme sind auch für normale Nutzer gedacht. Mit dem Programm „Save_MBR.exe" ist es möglich den momentan installierten MBR in eine Datei zu schreiben (zu sichern). Genau das Gegenteil ist mit „Load_MBR.exe" möglich. Damit kann ein neuer MBR aus einer Datei in den MBR gespeichert werden. Damit lassen sich auch im normalem PC-Betrieb Sicherheitskopien anlegen.

Als reine Informationsprogramme und als kleine Hilfen sind die folgenden drei Programme gedacht.

- HDInfo.exe
- LBA2CHS.exe
- CHS2LBA.exe

Mit „HDInfo.exe" lassen sich die Festplattenparameter abfragen und als zusätzliche Information wird auch die Speicherkapazität mit angegeben. Die anderen beiden Programme rechnen Adressen von LBA-Notation in CHS-Notation um und umgekehrt. Diese Umrechnung ist immer von der entsprechend angegebenen Festplatte abhängig. Für andere Systeme haben diese Angaben keine Gültigkeit. Diese Programme werden benötigt, da die beiden Notationen aus Kompatibilitäts-Gründen gleichberechtigt nebeneinader existieren und immer wieder ineinander umgerechnet werden müssen.

Die folgenden beiden Programme sind nur für Nutzer gedacht die ganze Speicherbereiche einer Festplatte sichern oder verändern wollen.

- Sek2File.exe
- File2Sek.exe

Um einen ganzen Bereich von Sektoren in eine Datei zu sichern wurde das Programm „Sek2File.exe"geschrieben. Damit lassen sich Bereiche von Festplatten kopieren, die nicht direkt als Datei vorliegen. Zum Beispiel laßt sich damit ein Backup einer oder beider FATs anlegen oder nach einem Festplattencrash lassen sich damit noch benötigte Daten in eine Datei sichern. Um das oben erwähne Backup der FATs zurückzuschreiben, wird „File2Sek.exe" benötigt. Damit muß sehr vorsichtig umgegangen werden, da es sehr leicht ist, lebenswichtige Daten damit zu überschreiben.

6.2 Bootkonzept für MSYS

Da das hier vorgestellte Bootkonzept des Bootmanagers recht universell einsetzbar ist, soll es nach entsprechenden Anpassungen auch für das Booten von MSYS eingestzt werden. In [5] wurde das entsprechende Konzept vorgestellt. Durch Ausnutzung der vorhandenen Units wurde es möglich, mit geringem Aufwand ein Programm zu entwickeln, welches für MSYS eine entsprechende Sektorliste erstellt und diese in MSYS einbindet. Damit sind die Voraussetzungen geschaffen, um das neue Konzept in MSYS einzubinden.

Programmdateien

Die einzelnen Programmteile des Setups

Durch die einzelnen Units bedingt, besteht das Setup aus mehreren Quelltextdateien, welche nachfolgend kurz vorgestellt werden.

Setupxxx.pas Das ist die Hauptdatei, in ihr sind alle weiteren Units eingebunden. An Stelle der drei „x" steht eine Versionsnummer.

Bootunit.pas Hierin sind alle grundlegenden Funktionen enthalten, um den Zugriff auf das Dateisystem und die Festplatte durchzuführen. Weiterhin sind darin alle wichtigen Datenstrukturen und globale Variablen definiert. Fast alle anderen Programmteile greifen auf diese Unit zurück.

Sche.pas In dieser Unit sind die Funktionen definiert, welche benötigt werden um die Datei des Bootmanagers zu finden und die benötigte Sektorliste aufzubauen.

Menuunit.pas Das interaktive Menü wird durch diese Unit dargestellt und bearbeitet.

Win.pas Die Unit „Menuunit.pas" greift auf diese Datei zu. Hierin sind die Funktione zur Darstellung der Fenster untergebracht. Diese Unit ist in einem Praktikum der Praktischen Informatik entstanden und konnte nach leichten Änderungen für dieses Projekt wiederverwendet werden.

Hojoe.pas Einige allgemeine Hilfsfunktionen sind darin enthalten.

Weiterhin wurde auch auf die von Turbo Pascal 6.0 mitgelieferten Units „DOS" und „CRT" zurückgegriffen.

Die Dateien des Bootmanagers

Bootman.bin Diese Datei enthält den eigentlichen Programmcode des Bootmanagers.

MBR.bin Der Ladeteil, welcher im MBR installiert wird, ist in dieser Datei enthalten.

Bootman.txt Eine Textdatei mit einer kurzen Beschreibung der Tastaturbelegung.

Die Hilfsprogramme

Save_MBR.exe Damit kann der MBR in einer Datei gespeichert werden.

Load_MBR.exe Hiermit wird der Inhalt einer Datei als MBR auf der Festplatte installiert.

HDInfo.exe Mit diesem Programm werden eineige Statusinformationen zu den eingebauten Festplatten ausgegeben.

LBA2CHS.exe Das Programm rechnet Adressen von LBA-Notation in CHS-Notation um.

CHS2LBA.exe Das Programm rechnet Adressen von CHS-Notation in LBA-Notation um.

Sek2File.exe Speichert einen oder mehrere Sektoren in einer Datei.

File2Sek.exe Damit kann ein Dateiinhalt in bestimmten Sektoren einer Festpllatte abgelegt werden.

Glosar

BIOS

Abk. für **B**asic **I**nput **O**utput **S**ystem. Das BIOS umfaßt die Systemprogramme zur grundlegenden Ein- und Ausgabeoperationen und stellt eine Software-Schnittstelle zur Hardware des PC dar. Typische BIOS-Funktionen sind der Zugriff auf Disketten, Festplatten, Schnittstellen und Grafikadapter.

CHS

Abk. für **C**ylinder **H**ead **S**ector. Mit der CHS-Notation wird die genaue Position eines Sektors auf einer Festplatte angegeben. Sie wird vom BIOS verwendet um den Zugriff auf Disketten und Festplatten durchzuführen.

CMOS-RAM

Abk. für **C**omplementary **M**etall-**O**xid-**S**emiconductor **R**andom **A**ccess **M**emory. In der Echtzeituhr der PCs gibt es einen kleinen Speicherbereich, welcher von einer Batterie oder Akku mit Stom versorgt wird. Dadurch behält er seine Daten und wird als Speicher für die Konfigurationsdaten benutzt.

DOS

Abk. für **D**isk **O**perating **S**ystem. Seit 1981 wird das von Microsoft entwickelte Betriebssystem bei den meisten IBM-kompatieble PCs verwendet. Es gibt verschiedene Implementationen zum Beispiel MS-DOS von Microsoft oder PC-DOS von IBM.

EIDE

Abk. für Enhanced-IDE. Ist eine Erweiterung des IDE-Standards. Es werden Laufwerke mit wechselbarem Datenträger und eine höhere Übertragungsgeschwindigkeit unterstützt. Weiterhin wurdedie nutzbare Kapazität auf 8 GByte erweitert.

EXE

Mit dieser Dateiendung werden ausführbare (englisch **exe**cuteable) Dateien von DOS bezeichnet.

FAT

Abk. für **F**ile **A**llocation **T**able. DOS verwendet ein FAT Dateisystem. Mit der FAT wird eine Zuordnung zwischen Dateien und ihren belegten Sektoren getroffen. Es gibt 12 Bit und 16 Bit FAT Dateisysteme für DOS. Mit Windows 95 wurde eine 32 Bit breite FAT eingeführt. Diese drei arten werden kurz als FAT12, FAT16 und FAT32 bezeichnet.

GByte

Abk. für $2^{30} = 1073741824$ Bytes.

LBA

Abk. für logische Blockadressierung. Die Festplatte wird als ein kontinuierlicher Strom von Sektoren angesehen, welche einfach durchnummeriert sind.

IDE

Abk. für Integrated Drive Electronics. Ein Standard für die Anbindung von intelligenten Festplatten oder anderen Laufwerken mit integrierten Controller an den AT-Bus. Die IDE-Schnittstelle wird auch als AT-Bus- oder ATA-Schnittstelle bezeichnet. Mit dieser Schnittstelle können bis zu 503 MByte angesprochen werden.

KByte
Abk. für $2^{10} = 1024$ Bytes.

LILO

Abk. für **Li**nux **Lo**ader. Ist der am weitesten verbreitete Bootloader unter Linux.

MBR

Abk. für **M**aster **B**oot **R**ecord. Ist der erste Sektor einer Festplatte, darin ist ein Programm und die Partitionstabelle enthalten. Das Programm bootet normalerweise die, in der Partitionstabelle, als aktiv markierte Partition.

MByte

Abk. für $2^{20} = 1048576$ Bytes.

MSYS

Das M steht für Multi-Tasking, Multi-Prozessor, Multi-User, Multi-Media usw. "SYS" ist die Abkürzung für System.

OS/2

Abk. für Operating Sytem /2. Ist ein Mulitasking fähiges Betriebssystem. Es wurde von IBM entwickelt, um die Nachfolge von DOS anzutreten.

Partition

Ist ein logischer Teil einer Festplatte. Dadurch lassen sich mehrere Betriebssysteme auf einer Festplatte unterbringen, da jedes Betriebssystem in einer eigenen Partition residiert.

Partitionstabelle

Sie befindet sich am Ende des MBR. Darin sind wichtige Daten über die Lage und Größe der einzelnen Partitionen enthalten.

PC

Abk. für Personal Computer. IBM entwickelt den ersten PC auf Basis eines 8088 Prozessors.

Protected Mode

Ein fortgeschrittener Betriebsmodus ab dem Intel 80286, jeder Zugriff auf Daten, Code und IO-Bereich wird selbständig vom Prozessor geprüft, um Verletzungen festzustellen.

TByte

Abk. für $2^{40} = 1099511627776$ Bytes.

Literatur

[1] Born, Günter, MS-DOS Programmierhandbuch, Markt- und Technik-Verlag, 1990

[2] Hans- Peter- Messmer, PC-Hardwarebuch, 3. Auflage, Addison- Wesley, 1995

[3] Harald Bögeholz, Vorletzte Rettung, Pfriemeln mit Diskedit und Debug, c't 5/97, S. 188

[4] Werner Almesberger, LILO Generic boot loader for Linux, Version 19, LILO Documentation, 1996

[5] Holger Jödicke, Bootstrategien von Betriebssystemen, Hauptseminar Betriebssysteme, 1998

[6] Steffen Albrecht, Entwurf von MSYS, Praktikumsdokumentation zum Ingenieurpraktikum, Fachgebiet Betriebssysteme. TU-Ilmenau, 1997

[7] Steffen Albrecht, Entwurf eines Netzwerk-Servers zur Realisierung von Verteilungsaspekten des Testbetriebssystems MSYS unter Berücksichtigung der Eigenschaften von Multimedia-Daten, Diplomarbeit Fachgebiet Betriebssysteme, TU-Ilmenau, 1998

[8] Turbo Pascal 6.0 Programmierhandbuch, 4. Auflage, Borland GmbH Starnberg, 1992