Hierarchische Dateisysteme

Daniel Laffling
FAU Erlangen-Nürnberg
Martensstraße 1
D-91058 Erlangen
daniel.laffling@fau.de

KURZFASSUNG

Im Zuge dieser Seminararbeit soll der Werdegang der grundlegenden Strukturierung und Ordnung von Daten auf Sekundärspeichern betrachtet werden. Das vermutlich erste System, welches ein Konzept zur hierarchischen Gliederung von Daten bot, war die MULTICS[1] Architektur. Diese Architektur wurde bereits Mitte der sechziger Jahre entwickelt und bietet, mit dem damals revolutionären Ansatz eines sicheren Mehrbenutzersystems, nicht nur im Hinblick auf das hierarchische Ablegen von Daten einen immer noch in vielen Bereichen wegweisenden Meilenstein der Geschichte informationstechnischer Systeme. Beginnend mit einer Betrachtung des damaligen Konzepts, dessen Umsetzung und Verbesserung im Laufe der Entwicklung, stellt diese Seminararbeit den Bezug zu einigen aktuellen mutmaßlichen Innovationen der Informationstechnologiegeschichte dar.

Keywords

Multics, File System, History of File Systems

1. EINLEITUNG

Als Mitte der Sechziger Jahre die Entwicklung von MULTICS am Massachusetts Institute of Technology in Kooperation mit den Bell Laboratories und General Electric begann, hatte wohl keiner der Beteiligten erwartet, dass die zu dieser Zeit getroffenen Designentscheidungen sich für die folgenden fünfzig Jahre, und vermutlich weit darüber hinaus, als wegweisend für die gesamte Geschichte der Informatik herausstellen sollten. Viele der damals erstmalig vereint genutzten und neu entwickelten Konzepte bilden heute noch den Grundstock der modernen IT-Welt.
Im speziellen sei hier die Entwicklung eines neuen Konzepts zur Speicherung von Daten auf Festspeichern erwähnt, welche zwar anfangs einige Designlücken und Probleme aufwies, jedoch im Laufe der Entwicklung immer weiter an die Bedürfnisse der Benutzer und der Hardware angepasst wurde. Schlussendlich war das System so ausgereift, dass die letzte sich im Produktiveinsatz, beim Department of National Defence in Kanada, befindliche Rechenanlage erst am 30. Oktober 2000 heruntergefahren und außer Dienst gestellt wurde.
Diese Seminararbeit beleuchtet den Werdegang der Entwicklung und Umsetzung hierarchischer Dateisysteme ab den Anfängen der Entwicklung von MULTICS. Sie beginnt bei der manifestierten Konzeption eines hierarchischen Dateisystems von Daley und Neumann [3] in Kapitel 2, berichtet in Kapitel 3 über die Implementierung mit dem Strukturellen Aufbau, Hürden bei der Umsetzung und die daraus resultierenden Verbesserungen in MULTICS , stellt damalige Konzepte und Implementierungen in den Bezug zu heutigen Architekturen in Kapitel 4 und schließt mit einem Resümee in Kapitel 5.

[1]**Mult**iplexed **I**nformation and **C**omputing **S**ervice

2. HISTORISCHE ENTWICKLUNG

Durch ansteigendes Interesse an - für Mitte der sechziger Jahre neuartigen - interaktiven Computersystemen, erschien es dringlich, die bisweilen sehr ungeschützte Datenverwaltung zu revolutionieren. Daten abzulegen erfolgte bis dato ohne jegliche Abstraktionsschicht des Systems, welches sich somit nicht um adequate Ein- und Zuteilung von Speicherbereichen bemühen konnte. Das Schreiben und Lesen von Speicherbereichen erfolgte rein adressbasiert durch den jeweiligen Programmierer selbst. Hierbei lässt sich erkennen, welche Risiken diese Art von Speicherverwaltung aufwirft, sobald man sich an die Konzeption eines Mehrbenutzer- oder Mehrkernsystems wagt. Einzelne Daten gegen unberechtigen Zugriff zu schützen, war ebenso unmöglich, wie sicherzustellen, dass keine ungewollte Manipulation von Datenbereichen im laufenden System stattfindet. Ein kleiner Programmierfehler, oder ein kippendes Bit an der falschen Stelle reichen hierbei aus, um weitreichenden Schaden an seinen eigenen Daten, den Daten fremder, oder Daten des Systems hervorzurufen. Die Schadensreichweite kann somit ins unermessliche steigen, da solche Rechenanlagen aufgrund ihres Preises meist wichtigen Organisationen mit großen Budgets, wie dem Militär vorbehalten waren.
MULTICS hingegen sollte die Effizienz solcher Organisationen durch den Einsatz von Großrechenanlagen für viele Bereiche, bei gleichbleibender oder verbesserter Sicherheit, steigern. Es sollte ein sicherer, komfortabler, paralleler Mehrbenutzerbetrieb gewährleistet werden, bei dem jeder Benutzer auch nur auf die Daten Zugriff erlangt, welche für ihn bestimmt sind. Wie bereits erwähnt, ist dies ohne ein hinreichend mächtiges Datenmanagement nicht umzusetzen.

Folglich musste ein neues Konzept zur Datenverwaltung auf Speichermedien entstehen, welches auf die Bedürfnisse eines Mehrbenutzersystems zugeschnitten war. Das eigens für MULTICS entwickelte und 1963 vorgestellte Konzept von Daley und Neumann, welches in den folgenden Abschnitten näher betrachtet werden soll, erfüllt die Kriterien einer abgesicherten Datenumgebung und löst somit auch viele aufkeimende Probleme im Ansatz, auch wenn Theorie und Praxis in diesem Fall ein wenig voneinander abweichen sollen.

2.1 Konzept

Ein solch neuartiges System zur Verwaltung und Speicherung von Daten musste einige aufkeimende Bedürfnisse befriedigen und daraus entstehende Aufgaben lösen [3]. Diese Bedürfnisse, welche in [3] aufgelistet werden, lassen sich grob in folgende vier Kategorien einteilen:

- *Priorisierung*: Schnellerer Speicher soll häufig benutzten Datenvorbehalten sein, wenig benutzte Daten sollen auf langsameren Speicher ausgelagert werden können.
- *Sicherheit*: Das Dateisystem soll Schutz vor unbeabsichtigter und unberechtigter Manipulation von Daten bieten.
- *Benutzerfreundlichkeit*: Die Daten sollen im Allgemeinen leicht zugänglich sein. Der Benutzer des Systems soll nicht mehr sehen, als er zur Bearbeitung von Daten braucht, außer er wünscht dies. Daten sollen innerhalb des Systems von mehreren Benutzern genutzt werden können
- *Portabilität & Erweiterbarkeit*: Das Dateisystem soll eine Abstraktionsebene darstellen, welche frei von den Eigenheiten der zugrundeliegenden Maschine ist und in der Lage sein nahezu unendlich großen Speicher zu verwalten.

Um diesen Bedürfnissen gerecht zu werden bedarf es einer Organisation der Daten, die sich von der bisherigen Art und Weise, Daten sequenziell auf verschiedenen Medien wie Lochkarten oder Bändern zu speichern, grundlegend unterscheidet. Da das Dateisystem eine autarke Abstraktionsebene mit systementkoppelter Zugriffsschnittstelle bieten sollte, war es nicht mehr notwendig, dass ein Benutzer die genaue physikalische Adresse der benötigten Daten auf dem Speichermedium kennt, um darauf zuzugreifen. Hierbei lag die Idee nahe, zusammenhängende Daten, wie beispielsweise ein Programm, zu einem Konstrukt zusammenzufassen, welches wir heute noch als *Datei* (siehe 2.1.1) kennen. Um die flache Struktur linearer Speicherung dieser Dateien aufzubrechen und diese anhand ihrer Zugehörigkeit gruppieren zu können, brauchte man noch eine übergeordnete Instanz, welche logisch zusammengehörende Dateien beherbergt. Diese Instanz trägt bis heute den Namen *Verzeichnis* (siehe 2.1.2) . Durch die Erschaffung dieser Meta-Konstrukte sollte es nun möglich werden, Daten hierarchisch in einer Baumstruktur (siehe Figure 1) abzulegen. Hierauf möchte ich im Folgenden detaillierter eingehen.

2.1.1 Dateien

Eine Datei ist nach Definition von R. C. Daley und P. G. Neumann [3] eine geordnete Sequenz von Elementen.

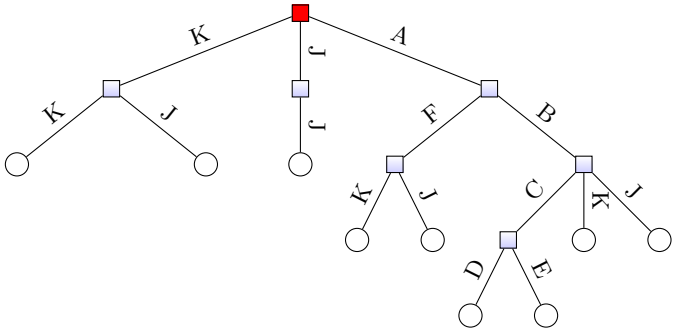

Figure 1: Baumstruktur des Dateisystems. Knoten: Dateien (Quadrate: Verzeichnisse, Kreise: Reguläre Dateien), Kanten: Symbolische Namen

Ein Element kann implementierungsabhängig ein Maschinenwort, ein Buchstabe oder ein Bit sein. Die Erstellung, Modifikation oder Entfernung einer Datei wird dem Benutzer nur durch die Verwendung der entsprechenden Schnittstelle im Dateisystem gestattet. Auf der Ebene des Dateisystems sind alle Dateien *formatlos*, also lediglich ein Zusammenschluss mehrerer Elemente, deren Interpretation übergeordneten Stufen des Systems vorbehalten bleibt. Die Adressierung einzelner Elemente einer Datei geschieht mittels derer Indices, relativ zum Beginn der Datei. Die Datei selbst erhält einen symbolischen Namen. Dieser symbolische Name kann frei gewählt werden und hat implementierungsabhängig eine definierte oder unbegrenzte Maximallänge an Zeichen. Somit kann ein Element einer Datei über den symbolischen Namen einer Datei, gefolgt von dessen linearem Index angesprochen werden. Durch die Abstraktion zu einer Datei wird somit ein logisch zusammengehörender Datensatz gekapselt [3].

2.1.2 Verzeichnisse

Ein Verzeichnis stellt eine spezielle Datei dar, welche vom Dateisystem verwaltet wird. Auch ein Verzeichnis trägt einen symbolischen Namen, beinhaltet aber eine Liste von Verzeichniseinträgen. Diese Verzeichniseinträge bestehen aus einem, vom Benutzer vergebenen, symbolischen Namen und systemrelevanten Attributen, welche in Abschnitt 2.1.4 näher betrachtet werden. Verzeichniseinträge stellen sich dem Benutzer als Liste symbolischer Namen von Dateien vor, die über dieses Verzeichnis erreichbar sind. Somit können auch Verzeichnisse über andere Verzeichnisse erreicht werden und bauen demzufolge eine hierarchische Baumstruktur auf, deren Knoten Dateien oder Verzeichnisse sein können. Den Ursprung dieses Baumes bildet ein Wurzelverzeichnis. Die Notwendigkeit der eindeutigen Benennung von Knoten beschränkt sich hiermit ebenfalls auf das Verzeichnis, dem sie selbst angehören. So kann es, wie in Figure 1 ersichtlich, im Dateisystem mehrere Einzelpfade mit dem selben symbolischen Namen geben, solange sie sich nicht auf der selben Hierarchieebene befinden [3].

2.1.3 Verknüpfungen

Die dritte Kategorie von Dateien stellen die Verknüpfungen dar. Eine Verknüpfung stellt ebenfalls eine spezielle Art von Datei dar und kann desweiteren auch einen Verzeichniseintrag darstellen. Allerdings enthält dieser Verzeichniseintrag nicht die Adresse der Datei selbst, sondern referenziert

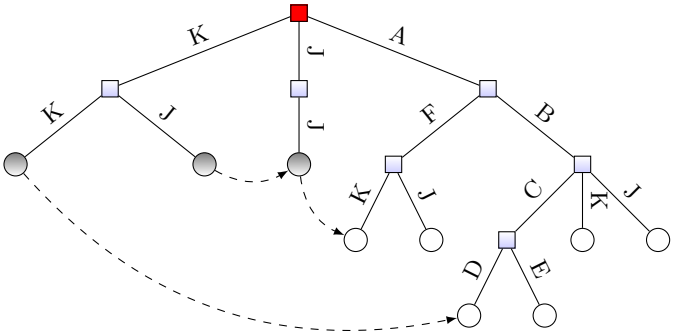

Figure 2: Baumstruktur des Dateisystems mit Verknüpfungen Wurzel:K:K nach Wurzel:A:B:C:D und Wurzel:K:J nach Wurzel:J:J nach Wurzel:A:F:K

einen anderen Verzeichniseintrag über die Verkettung von symbolischen Namen im Dateisystem. Hierbei ist es nicht wichtig für den Benutzer zu wissen, ob er gerade auf eine Verknüpfung oder auf die Datei selbst zugreift, denn sobald er einer Verknüfung folgt, erreicht er, notwendigenfalls über mehrere Verknüpfungen, schlussendlich die referenzierte Datei [3]. Siehe Figure 2.

2.1.4 Dateiattribute

Zu jedem Verzeichniseintrag müssen gewisse Metadaten abgelegt werden, welche verschiedene Informationen über die Benutzbarkeit und die aktuelle Benutzung der Einträge vorhalten. So soll zusätzlich zur physikalischen Adresse der Datei auf dem Speichermedium auch,

- der Zeitpunkt der Erstellung und letzten Modifikation,
- der Zeitpunkt des letzten Zugriffs,
- die Größe der Datei auf dem Datenträger (bei Verzeichnissen unbenutzt),
- die Zugangsbeschränkungen für Benutzer,
- der aktuelle Status der Datei (lesende Zugriffe, schreibende Zugriffe, teilende Zugriffe)

abgelegt werden [3].

2.1.5 Sicherheit und Schutz

Gerade wenn es um die Konzeption eines Computersystems geht, das von Grund auf sowohl mehrere Recheneinheiten unterstützt, als auch einen konfliktfreien Mehrbenutzerbetrieb gewährleisten soll, wird es zur zwingenden Notwendigkeit, Schutzkonzepte gegen alltägliche Gefahren zu entwickeln. Wie bereits von B. Herzog in [5] beschrieben wurde, ist dies eine Aufgabe, die interdisziplinär über alle Ebenen des Systems geschehen muss. Hierbei gilt es, neben der sicheren Ausführung von Prozeduren, auch den Blick auf die Absicherung bei der Speicherung von Daten zu richten. Die Gefahren bei der Datenablage sind sehr vielfältig. Benutzer könnten sich als andere Benutzer ausgeben um Zugriff auf Daten zu erhalten, der ihnen verwehrt bleiben soll bzw. muss. Benutzer können durch Unfall oder Böswilligkeit Daten beschädigen. Das System an sich kann Fehler hervorrufen, welche zur Beschädigung von Daten führen. Es können unvorhergesehen Hardwaredefekte auftreten.

3. IMPLEMENTIERUNG

Im Folgenden Teil dieser Arbeit soll in Abschnitt 3.1 zunächst der Strukturelle Aufbau des konzipierten Dateisystems dargelegt werden. Abschitt 3.2 berichtet daraufhin von auftauchenden Problemen bei der Umsetzung und deren Lösung im historischen Kontext der MULTICS-Entwicklung.

3.1 Struktureller Aufbau des Dateisystems

Der Umgang mit Dateien in MULTICS basiert auf dem Grundprinzip, dass jede Datei ein Segment darstellt, und jedes Segment eine Datei. Ein Segment im MULTICS-Kontext beschreibt ein direkt adressierbares Datenpaket. Die gleichbedeutende Verwendung der Begriffe Datei und Segment entstand durch die verwendete direkte Abbildung einer Datei auf dem Speichermedium in ein Segment im Adressraum des aufrufenden Prozesses. Die Datei wurde durch den Aufruf „initiate“ in den Adressraum des Prozesses als Segment abgebildet und konnte daraufhin von diesem bearbeitet werden. Um eine Datei nach vollendeter Bearbeitung zu schließen, wies man das Dateisytem mit dem Aufruf „terminate“ an, das Segment wieder aus dem Adressraum des Prozesses auszulagern und auf den Datenträger zurückzuschreiben[4]. Die hierfür verwendeten Komponenten möchte ich im Folgenden detailliert darstellen.
Der Aufbau des Dateisystems gliedert sich in mehrere feingranular aufgeteilte Module und Informationsdatenbanken. Jede Prozedur eines Benutzers besitzt sowohl prozedurlokale Daten, als auch Daten, welche mit allen weiteren Prozeduren geteilt werden. Wird eine Anfrage eines Benutzers nach einer Datei ausgeführt, beginnt diese mit dem Aufruf des Segmentverwaltungsmoduls (**Segment Management**). Das Segmentverwaltungsmodul besitzt die Aufgabe die Segmentverwaltung für den derzeit laufenden Prozess zu übernehmen. Es hält alle bis dato bekannten Segmente dieses Prozesses in der Segmentnamentabelle (**Segment Name Table - SNT**) mit

- symbolischem Namen des Segments(call name),
- dem beinhaltenden Verzeichnispfad(tree name),
- der Segmentnummer des Segments im Speicher

und weiteren Metainformationen zur prozesslokalen Segmentverwaltung vor. Segmente können entweder gerade im Hauptspeicher eingelagert sein, dann werden sie als aktiv bezeichnet, oder entsprechend als inaktiv, bei aktuell nicht vorhandener Präsenz. Aufschluss hierüber erhält das Segmentverwaltungsmodul durch die globale Segmentstatustabelle (**Segment Status Table - SST**), welche permanent im Hauptspeicher vorliegen muss. Sie enthält die Liste der derzeit aktiven Segmente im gesamten System. Sollte ein Segment erstmalig angesprochen werden (unbekannt bis zum Zeitpunkt der Referenzierung), so wird direkt eine Ausnahmeprozedur zur Auffindung (Search Module) und Einlagerung des betreffenden Segments ausgelöst. Bei der reaktivierung eines bekannten aber derzeit nicht präsenten Segments erfolgt ebenfalls eine Ausnahmeprozedur, welche vom Ablauf der eben genannten enspricht, aber auf einzelne Schritte verzichtet, da bereits durch das Vorhandensein eines Eintrags in der Segmentnamentabelle ein Ziel besteht und eine Segmentnummer für dieses Ziel bereits vergeben wurde. Das Segment muss lediglich wieder eingelagert und der entsprechende Eintrag in der Segmentstatustabelle aktualisiert werden.

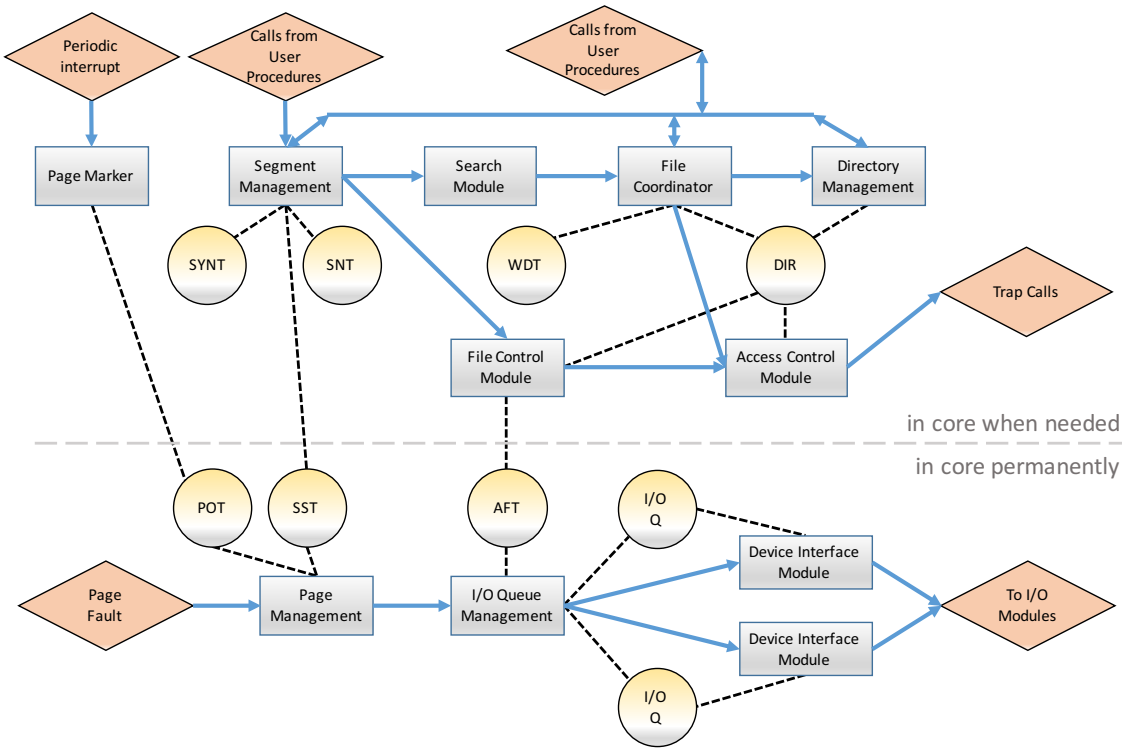

Figure 3: Überblick zu 3.1 über die beteiligten Komponenten (unvollständig) des Multics Dateisystems. Rauten: Prozeduren, Rechtecke: Module, Kreise: Datenbanken. Nachbildung aus [3]

Das Suchmodul (**Search Module**) selbst ist als Metamodul für das Auffinden von unbekannten Segmenten im Sekundärspeicher zu verstehen. Es erhält vom Segmentverwaltungsmodul Suchanfragen und traversiert nach vorgegebenem, oder durch den Benutzer zur Lade- oder Laufzeit spezifizierten Algorithmus, den Verzeichnisbaum. Um einzelne Verzeichnisse zu durchsuchen, oder während der Suche das Verzeichnis zu wechseln, benötigt das Suchmodul das Dateikoordinationsmodul.
Das Dateikoordinationsmodul (**File Coordinator**) übernimmt die Aufgabe der grundlegenden Dateisystemoperationen in einem Verzeichnis. Hiezu zählen die Erzeugung und Entfernung von Verzeichniseinträgen, Abfrage von Statusinformationen von betreffenden Verzeichnisinhalten, der Wechsel des Arbeitsverzeichnisses, die Modifikation von Zugangsbeschränkungen von Einträgen und einige weitere Aufgaben. Um diese Anforderungen zu erfüllen, benötigt das Dateikoordinationsmodul dauerhaft eine Referenz auf das derzeit gültige Arbeitsverzeichnis für jeden Prozess. Diese Information in Form des *tree names* wird in der Arbeitsverzeichnistabelle (**Working Directory Table - WDT**) vorgehalten.
Bei jeder Modifikationsanfrage an das Dateikoordinationsmodul gilt es, die Rechte des Benutzers zu wahren und Modifikationen, Statusabfragen oder Arbeitsverzeichniswechsel nur in dem Fall zu gestatten, in dem der, den Prozess ausführende, Benutzer legitimiert ist diese durchzuführen. Hierfür existiert ein eigenes Modul zur Zugangskontrolle.
Das Zugangskontrollmodul (**Access Control Module**) wird jedesmal aufgerufen, wenn die Zugangsberechtigungen zu einem Verzeichnis oder Verzeichniseintrag eingeholt werden müssen. Es erhält vom Aufrufer einen Zeiger auf den gewünschten Eintrag zusammen mit der Information über die Art des beabsichtigten Zugriffs. Der Aufrufer erhält vom Zugrangsmodul daraufhin Rückmeldung, ob der beabsichtigte Zugriff unter Berücksichtigung des ausführenden Benutzers gestattet werden kann.
Soll das Dateikoordinationsmodul eine Suche innerhalb eines Verzeichnisses durchführen, so beauftragt es das Verzeichnisverwaltungsmodul(**Directory Management**). Diese Untergliederung wurde getroffen, um eventuell aufkommende Rekursionsvorgänge während des Suchvorgangs in eine eigene Prozedur auszulagern. Die hier erwartete Rekursionswahrscheinlichkeit gründet sich auf folgenden Sachverhalt: Da das Verzeichnisverwaltungsmodul für die Suche nur einen Verzeichnispfad(*tree name*) zur Verfügung hat, stellt es Anfragen an die Segmentverwaltung, um die zu diesem Verzeichnispfad gehörige Segmentnummer zu erhalten. Wie oben beschrieben, kann dies im Falle eines nicht mehr präsenten Segments eine Ausnahmebehandlung zur Wiedereinlagerung des gesuchten Segments erforderlich machen. Dies geschieht durch eine Suche des Eintrags des derzeitigen Arbeitsverzeichnisses im übergeorneten Verzeichnis. Je nach Höhe des Verzeichnisbaumes kann es somit zu einer Verkettung von Ausnahmebehandlungen kommen. Im schlimmsten Falle bis hin zum Wurzelverzeichnis, welches aber selbst immer ein aktives Segment darstellt.
Das Dateikontrollmodul (**File Control Module**) ist als Untermodul der Segmentverwaltung für das öffnen und Schließen von Dateien zuständig und verwaltet die Dateiaktivitätsliste (**Active File Table - AFT**). Diese globale Liste beherbergt alle im System geöffneten Dateien mit den aktu-

ellen Zugriffszahlen. Der Zugriff auf eine Datei ist prinzipiell für alle zugriffsberechtigten Benutzer des Systems gleichzeitig lesend möglich. Im Modus *writable* kann sich jedoch nur ein Benutzer gleichzeitig befinden, es sei denn sie wurde von mehreren Benutzern teilend geöffnet. Somit kann sie zur Interprozesskommunikation genutzt werden.
Der Seitenmarkierer (**Page Marker**) ist eine periodisch aufgerufene Interruptroutine zur Verifikation der aktuellen Benutzung der Seiten (Pages - siehe „Paging in Multics"[8]) Er kümmert sich, basierend auf einer sich dynamisch anpassenden Aktivitätsbewertungsroutine, um die Auswahl von potenziell aus dem Haupspeicher verdrängbaren Seiten. Potenziell wenig genutzte Seiten legt er in der globalen Ersetzungstabelle (**Page Out Table - POT** ab. Bei Speicherbedarf zur Neu- oder Wiedereinlagerung werden die Seiten aus dieser Tabelle dann ersetzt.
Der Bedarf an freien Speicherseiten wird dem System durch einen Seitenfehler (**Page Fault**) signalisiert. Die Ursache für einen Seitenfehler kann entweder der Bedarf einer neuen Speicherseite, oder der Bedarf einer Speicherseite einer geöffneten Datei sein. In beiden Fällen wird Speicher für eine Seite im Hauptspeicher erforderlich. Ist dieser aktuell nicht vorhanden, so wird diejenige Seite, welche im ersten Eintrag der Ersetzungstabelle steht, aus dem Hauptspeicher verdrängt.
Selbstverständlich benötigt das Dateisystem ebenfalls eine Anbindung an die Außenwelt, die es dem Benutzer zur Verfügung stellen soll. Hierfür gibt es das E/A-Warteschlangenverwaltungsmodul (**I/O Queue Management**) mit seinen angegliederten Geräteschnittstellenmodulen (**Device Interface Module**). Über die jeweiligen Geräteschnittstellenmodule, von denen jeweils eines für einen Typ von E/A-Gerät existiert, kann das E/A-Warteschlangenverwaltungsmodul Daten aus der Außenwelt empfangen oder nach außen schreiben und diese in einer geöffneten Datei ablegen oder aus einer geöffneten Datei senden. Die Aufträge werden in Warteschlangen abgelegt (**I/O-Queue**) und daraufhin das, zu dieser Warteschlange gehörende, Geräteschnittstellenmodul über ein neues Ereignis notifiziert. Der Aufrufer des E/A-Warteschlangenverwaltungsmodules muss nach dessen Rückkehr, welche direkt nach der Benachrichtigung geschieht, entscheiden, ob er solange blockieren möchte bis die Abarbeitung erfolgt ist, oder nicht. Die Geräteschnittstellenmodule sind in der Handhabung ihrer Geräte volkommen unabhängig. Alle Operationen auf einem Gerät obliegen einzig und allein dem Geräteschnittstellenmodul und sind aus allen anderen Modulen vollständig gekapselt.
Es existieren noch einige weitere Dateisystemmodule, welche beispielsweise die Benutzerfreundlichkeit des Systems weiter verbessern sollten. Diese sind in der Abbildung 3.1 nicht aufgeführt, aber dennoch so einzigartig für dieses Zeitalter, dass ich sie im Folgenden noch anreißen möchte.
Auch damals gab es schon die Problematik, dass unterschiedliche Speicher abhängig von ihrer Geschwindigkeit und Kapazität entweder äußerst günstig oder enorm teuer waren. MULTICS hat sich bei der Entwicklung des Dateisystems von Grund auf dieser Tatsache angenommen und ein belastungsausgleichendes Speicherverwaltungssystem implementiert. Abhängig vom Grad der Benutzung einzelner Daten, verschob das System eigenständig höherpriore Daten auf schnellere Sekundärspeicher, wohingegen wenig frequentierte Datensätze auf langsamere Sekundärspeicher verlagert wurden.
Desweiteren existierten auch Datensicherungsmodule, welche sich um die eigenständige inkrementelle Sicherung und Wiederherstellung der Daten auf Festspeichern kümmerten. Die Fähigkeiten dieser Module reichten von permanenter eigenständiger inkrementeller Sicherung, eigenständigen wöchentlichen Gesamtsicherungen, über Wiederherstellung gelöschter Daten und selbständiger Überprüfung von Inkonsistenzen der Datensätze auf Band und im Sekundärspeicher, bis hin zur selbständigen Datenrückholung nach einem schwerwiegenden Systemfehler aus den inkrementellen und wöchentlichen Sicherungen.
Ebenfalls gab es weitere Module die den Umgang mit Tertiärspeichern im Falle des Kopierens von Daten auf Speichermedien und umgekehrt für den Benutzer so angenehm wie möglich machen sollten. Hier seien als Beispiel Datei an Drucker, Datei auf Lochkarten, Datei auf Band, Lochkarte an Datei und Band an Datei aufgeführt. Diese Prozeduren kehrten direkt nach dem Anstoß zurück, und ermöglichten so einen weiteren Ablauf der eigenen Arbeit ohne auf die Beendigung des Kopiervorgangs warten zu müssen [3].

3.2 Umsetzung in Multics

Das in Kapitel 2.1 beschriebene theoretische Konzept sollte in den darauffolgenden Jahren realer Teil der MULTICS Architektur werden. Hierbei wurden die Entwickler immer wieder mit Hardwareeinschränkungen und Bedürfnissen ihrer Kunden konfrontiert, die in ihrem ursprünglichen Konzept nicht oder nicht so vorgesehen waren, was sich in weiteren Verbesserungen und Umstrukturierungen niederschlug.

3.2.1 Hürden der Umsetzung

Wie so oft bei der Umsetzung von Konzepten auf reale Strukturen, ergeben sich mit der Zeit gewisse Einschränkungen. Diese Einschränkungen können sowohl auf die Leistungsfähigkeit bestehender Ressourcen, als auch auf vorangegangene Umsetzungsentscheidungen zurückgeführt werden. Hiervon blieb auch die MULTICS Entwicklung in Bezug auf die Umsetzung des Konzepts für ein mehrbenutzertaugliches, hierarchisches Dateisystem nicht verschont. Bei der Entwicklung stieß man, wie Van Vleck in [10] beschreibt, bereits relativ früh auf Hürden, welche sich nicht trivial umgehen ließen. Ihr Konzept wies bereits bei der näheren Betrachtung von Kundenanforderungen mit Blick auf Konkurrenzprodukte Defizite auf.

Austauschbare Speichermedien. Zum Einen war eine Unterstützung von austauschbaren Medien von Beginn an nicht vorgesehen. Konkurrenten, wie die IBM Rechner 2311 und 2314 welche mit dem Betriebssystem OS/360 ausgestattet waren, besaßen allerdings diese Fähigkeit, die sich auch als Benutzerbedürfnis erwies, um beispielsweise Daten außerhalb des Rechners zu verwahren oder auch bestehende Daten in das System zu integrieren ohne dabei die bestehenden Ein- und Ausgabeschnittstellen der Rechner nutzen zu müssen.

Problematik bei inkonsistenten Daten. Des weiteren hatte man sich zu Beginn der Umsetzung dazu entschieden die Laufwerksadressen einfach in aufsteigender Reihenfolge zuzuweisen, und somit alle vorhandenen Sekundärspeicher systemintern als ein großes Laufwerk zu behandeln. Diese Entscheidung, welche laut Van Vleck [10] aus Gründen der einfachen Umsetzung getroffen worden war, sollte sich in

mehrerlei Hinsicht als problematisch entpuppen. Einerseits hatte dieser Ansatz zur Folge, dass es oftmals zu einer Verteilung von zusammengehörenden Speicherseiten einer Datei auf mehrere physikalische Laufwerke kam, was zwar zu einer ausgeglichenen Lastverteilung auf den Speichermedien führte, jedoch im Falle eines Defekts eines der gekoppelten Speicher eine stundenlange Rekonfiguration aller Speicher nach sich zog. Andererseits hatte dadurch ein Ausfall einer einzigen physikalischen Speichereinheit einen mehrstündigen Wiederherstellungsprozess zur Folge, obwohl der überwiegende Teil der Daten nicht davon betroffen war.

Laufwerkskapazitäten. Eine der größten Vereinfachungen bei der Umsetzung betraf die Adressierung der Speicherseiten, welche im Detail in der Seminararbeit von Jonas Rabenstein zum Thema Paging in MULTICS [8] beschrieben wird. Dies ergab eine Limitierung von 18 Bits für eine Speicheradresse. Somit betrug die maximale Laufwerkskapazität eines MULTICS Systems 28 DSS191 Einheiten[2].

Dateigrößen. Ein weiteres größeres Problem trat auf, als die Dateien durch große Datenmengen in ihrer Größe zunahmen. Die Maximale Segmentgröße beschränkt sich durch das verwendete Addressformat [8] auf 2^{18} adressierbare Worte mit einer Wortlänge von 36 Bit auf ca. 1MiB. Da Dateien ebenfalls Segmente sind, ist auch die Dateigröße auf ca. 1MiB beschränkt. Wächst eine Datei über dieses Maß hinaus, kann sie nicht mehr trivial vom Dateisystem behandelt werden und ein Ausweichkonzept musste geschaffen werden um größere Dateien verwalten zu können. Es war prinzipiell nur durch eine weitere Indirektionsstufe in der Dateiverwaltung möglich, solch große Datenmengen gruppiert zu handhaben. Diese benötigte Indirektion ist von der Theorie her bereits im Dateisystem vorhanden, denn ein Segment, welches Informationen über weitere Segmente enthält haben wir bereits als Verzeichnis in Abschnitt 2.1.2 kennengelernt. Folglich wurde die Handhabung großer Datenmengen durch die Realisierung von Mehrsegment-Dateien (**Multi Segment Files - MSF**) bereitgestellt. Eine solche Datei stellt sich dem Benutzer grundlegend als Verzeichnis dar, dessen Inhalte (Name der Dateien aufsteigend nummeriert) der großen Datei aufgeteilt in „Segmentbrocken "entsprechen. Das sonst bei einem Verzeichnis überflüssige Attribut *bitcount*, welches bei Dateien deren Größe angibt wurde nun bei einem MSF dazu verwendet, die Anzahl der beinhalteten Segmente zu spezifizieren. Der Nachteil an dieser Implementierung bestand darin, dass erst nach und nach einzelne Programme die Fähigkeit erhielten mit diesen speziellen Dateien umzugehen. Es scheint keinen richtigen Standard seitens MULTICS für den Umgang mit MSFs gegeben zu haben, was beispielsweise dazu führte, dass einige Programme nur mit Segmentteilen der maximalen Größe umgehen konnten, andere jedoch auch mit Segmentbündeln unterschiedlicher Segmentgrößen [4].

3.2.2 Verbesserung des Multics Dateisystems - NSS

Die Hürden, die die MULTICS Systeme von Beginn an mit sich brachten (siehe Kapitel 3.2.1), sollten in einem Projekt im Jahr 1973 beseitigt werden. Man entschied sich die Schnittstelle des Dateisystems weitgehend so beizubehalten

[2]Speichereinheit der frühen siebziger Jahre. Baugleich zur Honeywell MSU0400 Einheit [1] - Glossary.

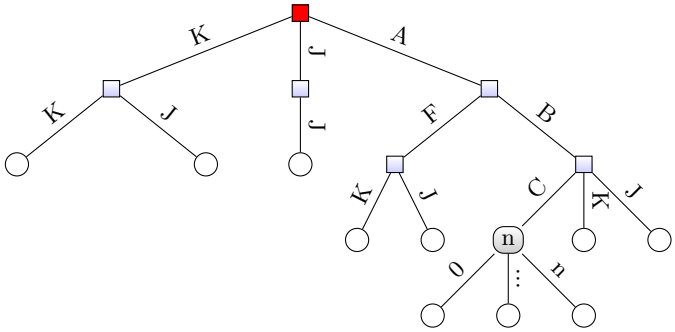

Figure 4: Baumstruktur des Dateisystems mit Multisegment Datei Wurzel:A:B:C

wie sie war, um weitreichende Systemänderungen zu vermeiden, nur die Art und Weise der Datenablage zu reformieren und weitere Kundenbedürfnisse, wie die bereits erwähnte Unterstützung für austauschbare Speichermedien, zu erfüllen. Eine der notwendigsten Veränderungen bestand in der Trennung von logischer und physikalischer Dateiverwaltung in Verbindung mit der Vermeidung der Verteilung zusammengehörender Segmente auf verschiedene physikalische Speichereinheiten. Hierzu wurde jedem Verzeichniseintrag ein logisches Wurzelverzeichnis **Root Logical Volume - RLV** zugeordnet, was dafür sorgte, dass nunmehr der Inhalt eines Verzeichnisses nur auf einer logischer Speichereinheit untergebracht werden konnte. Man entschied sich des weiteren, die vorangegangene Fähigkeit des Zusammenschlusses mehrerer Laufwerke zu einer logischen Einheit zu erhalten. Zur Verbesserung der Datenorganisation wurden Organisationsstrukturen nach dem Vorbild von OS/360 zur Dateiverwaltung in Form einer Laufwerkinhaltstabelle (**Volume Table Of Contents - VTOC**) pro logischem Laufwerk erstellt. Der Zugriff auf Verzeichniseinträge lief nun nach folgendem Schema ab:
Jeder Verzeichniseintrag enthält neben der Benutzer-ID eine Physical Volume-ID, welche dem Index des Eintrags (**Physical Volume Table Entry - PVTE**) in der globalen Tabelle physikalischer Laufwerke (**Physical Volume Table - PVT**) entspricht. der Inhalt des PVTE entspricht der Basisadresse des für dieses Verzeichnis bestimmten VTOC. Der Index des betreffenden VTOC-Eintrags wird ebenfalls im Verzeichniseintrag gesichert. Der VTOC-Eintrag enthält dann selbst eine Filemap, welche alle zugehörigen Seiten mit ihrem Speicherort auf der physikalischen Speichereinheit enthält (siehe Figure 5).

So wurde zum Einen die Möglichkeit geschaffen, Laufwerke im laufenden Betrieb ein- und auszuhängen, als auch eine Indirektionsstufe mehr ins System aufgenommen, sodass nunmehr die Anzahl der adressierbaren Laufwerke nicht wie bisher durch direkte Adressierung einer Filemap beschränkt war. Die Maximalanzahl adressierbarer Laufwerke wurde somit nur noch durch die Anzahl der möglichen Einträge in der PVT begrenzt. Zum Anderen war sichergestellt, dass im Falle eines Hardwaredefekts nur die betreffende logische Einheit rekonfiguriert werden musste, was zeitlich einen immensen Vorteil brachte.

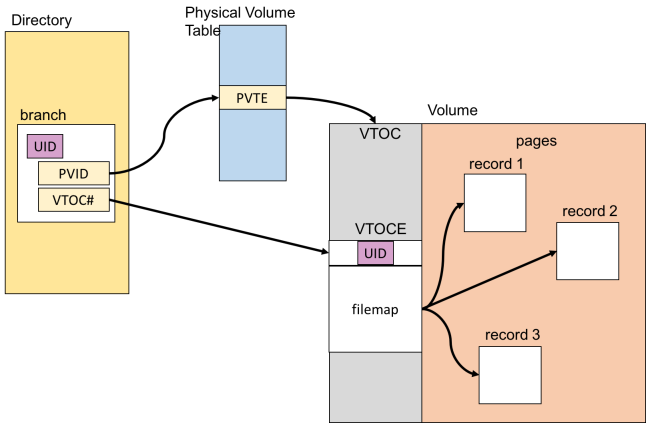

Figure 5: Veranschaulichung der 1973 eingeführten Indirektionsstufe des New Storage Systems. Aus [10].

4. MULTICS IN FOLGENDEN ENTWICKLUNGEN

Die Konzeption und Entwicklung von Multics vor nunmehr 50 Jahren stützt sich wie bereits erwähnt auf einen Entwurfsansatz welchen wir heute als Hardware-Software-Codesign bezeichnen würden [6]. Auch wenn dieser Ansatz sich nicht über die Dauer als zielführend erwies und man sich im weiteren Verlauf der Geschichte eher auf getrenntere Entwicklungswege stützte, ist man heute im Bereich der Eingebetteten Systeme wieder zu diesem Ansatz zurückgekehrt um effiziente, speziell auf eine Anwendung zugeschnittene, Systeme zu realisieren. Der heutzutage weitaus größere Bereich ist geprägt von getrennten Hard- und Softwareentwicklungen, die immer wieder Neuentwicklungen auf der jeweils anderen Seite zum Anlass nehmen, ihr eigenes Konzept weiterzuentwickeln. Auch hierbei trifft man im Verlauf der Geschichte immer wieder auf Teile des MULTICS Konzepts, welche neu aufkeimen und es schaffen den ambitionierten Benutzer zu beglücken. Im Folgenden möchte ich gezielt auf ein paar als Innovation angepriesene Exemplare der Datenverwaltung auf Hard- und Softwareseite von damals bis heute eingehen, welche in engem Zusammenhang mit tragenden Aspekten der MULTICS -Entwicklung stehen.

UNIX. Bereits während der Entwicklungszeit von MULTICS entstand in den Bell Laboratories - welche auch maßgeblich für die Entwicklung des MULTICS-Dateisystems verantwortlich waren - ein weiterer Meilenstein der Geschichte der Informatik. Dennis Ritchie und Ken Thompson, entwickelten in den Bell Laboratories ein neues Betriebssystem, welches ähnliche Ansätze hatte wie MULTICS . Wo für MULTICS spezialisierte Hardware nötig war, fokussierte sich die Entwicklung von UNIX auf die Integration auf bereits bestehenden Maschinen wie der PDP-7, PDP-9, und PDP-11. Auch in Unix sollte ein hierarchisches Dateisystem Anwendung finden, welches vom Konzept her stark an das von MULTICS erinnert. Die Struktur war ebenfalls ein Baum mit einem Wurzelverzeichnis, es gab Dateien und Verzeichnisse, welche sich in derselben Art und Weise anordnen ließen wie unter MULTICS und es gab ebenso die Möglichkeit Dateien Benutzerrechte zuzuweisen. Diese drei Grundprinzipien halten sich durchgängig in dem Gesamten Lebenslauf aller UNIX Dateisysteme und deren Ableger, zu denen auch die Linux- und Apple Dateisysteme zählen [9].

RAID-0. „Redundant Arrays of Independent Disks“lautet der heutige Begriff für die Abkürzung RAID. Eigentlich gründet sich diese Entwicklung auf das Papier “A case for redundant arrays of inexpensive disks (RAID) von Patterson et al aus dem Jahre 1988 [7]. Auch diese Entwicklung, welche wir bis heute häufig nutzen, wurde aus der Not heraus geboren. In diesem Papier werden die RAID-Level 1-5 vorgestellt, von denen wir bis heute noch Level 1 und Level 5 häufig vorfinden. Das RAID-Level 0 allerdings, welches uns heute auch als Striping bekannt ist und genaugenommen keine Redundanz sondern das Gegenteil erzeugt, ist aus Einfachheit der Implementierung ohne wirklich darüber nachzudenken [10] bereits in MULTICS implementiert worden (siehe 3.2.1).

FusionDrive. Apple Computer bietet seit 2012 seinen Kunden eine Speicherlösung bestehend aus einem Zusammenschluss eines Solid State Drives, also eines schnellen aber teuren Flash-Speichers, in Zusammenhang mit einer großen verhältnismäßig günstigen konventionellen Magnetscheibenfestplatte an. Apples *FusionDrive* ist prinzipiell Apples Implementierung des *Hystor*-Konzepts basierend auf dem Papier von Chen und Koufaty aus dem Jahre 2011[2]. Der Grundgedanke dieses Konzepts, nämlich die Verlagerung von häufig benötigten Daten auf schnellen, und die Bevorratung von selten gebrauchten Daten auf langsamem Speicher ohne Zutun (und ohne Eingriffsmöglichkeiten) des Benutzers ist aber bereits Grundgedanke der MULTICS -Implementierung gewesen, wie am Ende des Abschnitts 3.1 vermerkt wurde.

Intel Turbo Memory und Windows Vista. Im kleineren Rahmen existiert die Vorstufe von *Hystor* im Hause Intel schon länger. Bereits 2007 stellte Intel die *Intel Turbo Memory*-Bausteine vor, welche in Zusammenhang mit neuen Funktionen des Microsoft Betriebssystems Windows Vista für ein schnelleres Arbeiten sorgen sollten. Auch hier wurde ein kleiner schneller Flash-Speicher an die langsamere Festplatte angeknüpft, um in gewissen Situationen Performanzvorteile zu erzeugen. Ziel war auch hier vor Allem das Vorhalten häufig frequentierter Daten (unter anderem der notwendige Code für den Systemstart) im schnellen Zwischenspeicher. Allerdings waren diese Module von Beginn an nicht besonders groß, dafür aber besonders teuer. Der erwartete Geschwindigkeitsvorteil stellte sich in den Augen der Hersteller von Computersystemen als zu gering dar, um die hohen Kosten zu rechtfertigen. Somit erlangte diese Entwicklung nur sehr geringe Verbreitung auf dem Computermarkt.

Dies waren nur einige wenige Beispiele der fortwährenden Neuentdeckungen bestehender Dateiverwaltungskonzepte aus der MULTICS Entwicklung. Die Liste an signifikanten Ähnlichkeiten zu derzeit noch verwendeten Systemen ist lang und kann bei genauerer Betrachtung sicherlich noch weiter ausgedehnt werden.

5. FAZIT

Betrachtet man heutige Dateisysteme unter den Gesichtspunkten, die bereits 1965 bei der Konzeption von MULTICS ausschlaggebend waren, erkennt man wie zielführend und

weitreichend dieses Konzept gewesen ist. Die grundsätzlichen Gegebenheiten haben sich kaum verändert. Auch wenn damals eine Rechenanlage, wie der GE 645, noch raumfüllend und für einen Privatmann nahezu unerschwinglich gewesen ist und wir heute Personal Computer mit vielfacher Rechenleistung betreiben, gilt die damalige Grundaufgabe nach wie vor: Ein System mit mehreren Rechenkernen, welches parallel von mehreren Personen genutzt werden kann. Somit ist es nicht verwunderlich, dass wir nach wie vor auf eine hierarchische Speicherung von Daten zurückgreifen. Dabei spielen die für den Endbenutzer doch recht unterschiedlich erscheinenden Plattformen auf dem Markt keine Rolle. Die Konzepte der Speicherung im Sekundärspeicher und der Sekundärspeicher selbst haben sich selbstverständlich im Hinblick auf Performanz und Absicherung gegen Gefahren des Alltags weiterentwickelt, dennoch operieren sie im Grunde auf der selben Basis. Es gibt heutzutage eine Vielzahl von Dateisystemen, die auf den jeweiligen Anwendungsfall optimiert sind und weitaus mehr Funktionen bieten als 1965 vorstellbar waren oder benötigt wurden. Gewisse Bedürfnisse an das System erfordern auch nach wie vor eine etwas andere Umsetzung, wie sie zu damaligen Gegebenheiten der Fall war. Das Design von 1965 stellt jedoch für nahezu alle sich derzeit im Produktiveinsatz befindlichen Dateisysteme den Grundstock dar. Alle aktuellen Dateisysteme operieren immernoch als Abstraktionsschicht zwischen Benutzer und Recheneinheit. Allesamt strukturieren Daten nach wie vor durch Dateien und Verzeichnisse, welche in einem Wurzelverzeichnis aufgehängt sind. Insgesamt liegt das Augenmerk bei der Datenträger- und Dateisystementwicklung immernoch auf den vier eingangs beschriebenen Grundprinzipien,welche damals bereits bei der Entwicklung von MULTICS ausschlaggebend waren: Der Priorisierung von häufig genutzten Daten, der gebotenen Sicherheit, der Benutzerfreundlichkeit und der Portabilität und Erweiterbarkeit.
Diese Gedanken, welche vor nunmehr fünfzig Jahren durch die Köpfe einiger weniger Entwickler zu Papier gebracht worden sind, haben auch heute noch in der schnellebigen Welt der Informationstechnologie Relevanz. Dennoch gerät dieses System immer weiter in Vergessenheit, obwohl es manchmal durchaus von Vorteil wäre ausgediente Konzepte neu zu überdenken, wie in Kapitel 4 angeführte Beispiele aus der Neuzeit eindrucksvoll belegen.

6. REFERENCES

[1] http://multicians.org - aufgerufen am 02.01.2016.

[2] F. Chen, D. A. Koufaty, and X. Zhang. Hystor: making the best use of solid state drives in high performance storage systems. In *Proceedings of the international conference on Supercomputing*, pages 22–32. ACM, 2011.

[3] R. C. Daley and P. G. Neumann. A general-purpose file system for secondary storage. In *Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I*, pages 213–229. ACM, 1965.

[4] P. Green. Multics virtual memory: Tutorial and reflections. *Retrieved January*, 29:2001, 1993.

[5] B. Herzog. Schutzkonzepte in multics. In *AKSS WS15 Proceedings*. FAU Erlangen-Nürnberg - Lehrstuhl für verteilte Systeme und Betriebssysteme, 2015.

[6] M. Krüger. Der general electric 645 computer. In *AKSS WS15 Proceedings*. FAU Erlangen-Nürnberg - Lehrstuhl für verteilte Systeme und Betriebssysteme, 2015.

[7] D. A. Patterson, G. Gibson, and R. H. Katz. *A case for redundant arrays of inexpensive disks (RAID)*, volume 17. ACM, 1988.

[8] J. Rabenstein. Paging in multics. In *AKSS WS15 Proceedings*. FAU Erlangen-Nürnberg - Lehrstuhl für verteilte Systeme und Betriebssysteme, 2015.

[9] D. M. Ritchie and K. Thompson. The unix time-sharing system. *Communications of the ACM*, 17(7):365–375, 1974.

[10] T. V. Vleck. The new storage system. 1995.