INSTITUT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Konzeption einer Managementinformationsbasis für das Management von UNIX-Endsystemen

Uwe Krieger

Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Dr. Bernhard Neumair, Markus Gutschmidt
Abgabedatum:	15. Mai 1994

Ich versichere, daß ich diese Diplomarbeit selbständig verfaßt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, 15. Mai 1994

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Einordnung des Systemmanagements . . . 1

1.2 Motivation der Aufgabenstellung und wichtigste Ergebnisse . . . 2

2 Analyse des Themenkreises **4**

2.1 Eignung von SNMP für Systemmanagement . . . 4

2.2 Das OSI-Systemmanagement und seine Bedeutung für die Diplomarbeit . . . 6

2.2.1 Das OSI-Managementmodell . . . 7

2.2.2 Modellierung von Systemressourcen . . . 8

2.3 Untersuchung ausgewählter MIBs und Überführung in ein einheitliches Klassifikationsschema . . . 8

2.3.1 Auswahl der MIBs . . . 8

2.3.2 Klassifikationsschema . . . 10

2.3.3 Überführung der MIBs in das Klassifikationsschema und Bewertung . 12

3 Anforderungen eines Rechenzentrumbetreibers (LRZ) **14**

3.1 Entwicklung eines integrierten Systemmanagements am LRZ . . . 16

3.1.1 Vorarbeiten des LRZ . . . 16

3.1.2 Planung des Systemmanagements unter Berücksichtigung der Forderungen des LRZ . . . 16

3.2 Anforderungen der Systemverwalter an ein integriertes Systemmanagement . 18

3.2.1 Integration der Systemverwalter . . . 18

3.2.2 Ergebnisse der Systemverwalterinterviews . . . 19

3.2.3 Ableitung der Anforderungen an die MIB . . . 20

4 Modellierung der Hardware **21**

4.1 Allgemeine Information über das zu managende System 22

4.2 Prozessor 22

4.2.1 Übersicht 22

4.2.2 Modellierung des Prozessors für das Systemmanagement 23

4.2.3 Leistungsbewertung eines Prozessors 24

4.2.4 Abschließende MIB-Betrachtung der Prozessorklasse 28

4.3 Speicher 30

4.3.1 Primärspeicher 30

4.3.2 Modellierung der Speicher für die MIB 32

4.4 Massenspeicher 32

4.4.1 Übersicht über die gebräuchlichsten Massenspeicher 32

4.4.2 Modellierung der Massenspeicher für das Systemmanagement 33

4.5 Bussysteme 33

4.5.1 Struktur und Hierarchie 34

4.5.2 Modellierung des Busses 34

4.6 Anordnung der modellierten Hardware innerhalb der MIB 35

4.7 Modellierung eines Peripheriegerätes: Der Drucker 36

4.7.1 Installation von Druckern 36

4.7.2 Verwaltung von Druckern 36

4.7.3 Modellierung für die MIB 36

5 Management von UNIX-Prozessen **39**

5.1 Der UNIX-Prozeß 39

5.2 Der Prozeßkontext 40

5.2.1 Status eines Prozesses 40

5.2.2 Priorität eines Prozesses 42

5.2.3 Ressourcenverbrauch eines Prozesses 42

5.3 Management mittels der MIB 43

5.4 Auslastung eines UNIX-Systems 44

6 Partitionen und Filesysteme **45**

6.1 Partitionen 45

6.1.1 Modellierung der Partitionen für die MIB 45

6.2 Filesysteme 45

6.2.1 Aufbau von UNIX-Filesystemen 46

6.3 Systemadministratortätigkeiten bei Filesystemen 47

6.4 Modellierung der MIB für Filesysteme 48

7 Benutzer- und Gruppenverwaltung **49**

7.1 Benutzerverwaltung 49

7.2 Gruppenverwaltung 51

7.3 Sicherheitsaspekte der Benutzer- und Gruppenverwaltung 51

7.4 Die Who-Tabelle 52

8 Zusammenfassung und Ausblick **53**

8.1 Zusammenfassung 53

8.2 Ausblick 55

A Auswertung der MIBs mittels Klassifikationsschema **56**

B Fragen der Systemverwalterinterviews und Auswertung der Ergebnisse **72**

B.1 Fragebogen und Interviews 72

B.2 Zusammenfassung der Ergebnisse 73

C Managementinformationsbasis für das Management von UNIX-Endsystemen (UNIX-LRZ-MIB) **75**

D Implementierungshinweise **115**

D.1 Ebenen der Informationsgewinnung 115

D.2 UNIX-LRZ-MIB 117

D.2.1 System Group 117

D.2.2 Storage Group 118

D.2.3 Device Group 118

D.2.4 Processor Group 119

D.2.5 Printer Group 119

D.2.6 Disk Group 120

D.2.7 Partition Group 120
D.2.8 Filesystem Group 121
D.2.9 Process Group 121
D.2.10 User Group 122

Abkürzungen **124**

Literaturverzeichnis

Kapitel 1

Einleitung

1.1 Einordnung des Systemmanagements

Bereits seit einigen Jahren gewinnt das Netzmanagement durch Entwicklungen wie beispielsweise die Einführung von Client-Server-Architekturen und der Übergang zu Workstationverbunden ("Downsizing") zunehmend an Bedeutung ([Neum92]). In letzter Zeit wurde jedoch deutlich, daß ebenso wie beim Netzmanagement auch beim Systemmanagement Mechanismen, die unabhängig von der unterliegenden Hardware arbeiten, sowohl den Verwaltungsaufwand reduzieren, als auch die Betriebssicherheit erhöhen.

Systemmanagement umfaßt die Bereiche Planung, Konfigurierung, Steuerung, Überwachung, Fehlerbehebung und Verwaltung von Rechnerendsystemen. Während zum Netzmanagement alle Managementaspekte zählen, bei denen primär die Kommunikation im Vordergrund steht, sieht das Systemmanagement alle Aspekte, die den Betrieb sowohl einzelner Rechensysteme als auch den Betrieb eines Verbundes von Rechensystemen als verteiltes System mit entsprechend verteilter Diensterbringung betreffen ([Hege93]). Ziel des Systemmanagements ist es, Systembenutzer und Systembetreiber während Planung und Betrieb von verteilten Systemen und ihrer Endsysteme zu unterstützen, um eine gewünschte Güte des Betriebs zu gewährleisten ([Hege93]). Der Gesamtkomplex der Aufgaben des Systemmangements kann in die Funktionsbereiche Fehlermanagement, Konfigurationsmanagement, Accountingmanagement, Performancemanagement und Sicherheitsmanagement eingeteilt werden (siehe z.B. [ISO 7498-4]).

Zentraler Bestandteil einer Managementarchitektur ist das Informationsmodell. Es liefert die Basis für die Definition der sogenannten Managementinformation, die Syntax und Semantik der für die Zwecke des Netz- und Systemmanagements zwischen offenen Systemen austauschbaren Information definiert. Die Gesamtmenge der von einem offenen System nach außen zur Verfügung gestellten Information wird konzeptionell als Managementinformationsbasis [1] bezeichnet ([Neum92], [ISO 10040]).

Durch zunehmende Vernetzung von Arbeitsplatzrechnern und Verwendung einzelner Systeme für Spezialaufgaben ist ein Rechnerumfeld nicht mehr als Ansammlung von Einzelsystemen

[1] Im folgendem auch als MIB bezeichnet

zu sehen sondern eher als ein verteilter Rechner. Dadurch ist die klassische Trennung von Systemmanagement (alle Aktivitäten und Ressourcen im Rechner) und Netzmanagement (rechnerübergreifende Kommunikation und die dafür notwendigen Ressoucen) nicht mehr möglich. Für die Diplomarbeit werden alle Kommunikationsressourcen (z.B. Netzwerkkarten) und Netzdienste (z.B. ftp) ausgeschlossen. Hingegen sind die Rechner selbst und alle Dienste, die es auch auf alleinstehenden Rechnern gibt, Bestandteil der Untersuchungen.

1.2 Motivation der Aufgabenstellung und wichtigste Ergebnisse

In Kapitel 2 werden neben einer kurzen Betrachtung der Eignung von SNMP für das Systemmanagement, sowie über grundsätzliche Überlegungen zur Modellierung von Systemressourcen, bestehende Veröffentlichungen zu diesem Themenkreis analysiert und in ein einheitliches Klassifikationsschema (Anhang A) überführt. Mit Hilfe dieses Klassifikationsschemas wird die Angemessenheit und Vollständigkeit der untersuchten Management Information Bases überprüft. Ergebnis dieser Untersuchung wird sein, daß in diesem neuen Themengebiet leider sehr viele "Schnellschüsse" existieren, die den Anspruch an ein vollständiges Systemmanagement nicht erfüllen können.

Für die Bearbeitung des Themas wurde ein "Top-Down"-vorgehen gewählt. Zusammen mit den Betreibern eines großen Rechenzentrums (LRZ[2]) wurden die Ergebnisse des Kapitel 2 nochmals erörtert und mit den Anforderungen des Rechenzentrumbetreibers abgeglichen. Zur Konsolidierung der gewonnenen Information wurden neben Systemverwalterinterviews auch noch verschiedene Vorträge gehalten und eine Fragebogenaktion durchgeführt. Diese Aktivitäten werden in Kapitel 3 und Anhang B beschrieben.

Um den Anforderungen der Diplomarbeit, eine Managementinformationsbasis für das Management von UNIX-Endsystemen zu konzeptionieren, gerecht zu werden, müssen die wesentlichen Merkmale eines UNIX-Rechners herausgearbeitet und für die MIB modelliert werden. In Kapitel 4 werden zusätzlich Aussagen zur Beurteilung der Leistungsfähigkeit eines Rechners gemacht, sowie Kennzahlen, die Systemengpässe andeuten können, entwickelt. Weiterhin wird stellvertretend für die große Anzahl von existierenden Peripheriegeräten der Drucker für das Systemmanagement modelliert.

In Kapitel 5 werden die Prozesse von UNIX-Systemen untersucht. Die im Verlauf dieses Kapitels definierte Managementinformation ermöglicht die Überwachung und Steuerung des Benutzerverhaltens, das aktive Management von Prozessen, sowie eine Aussage über die Auslastung eines Rechensystems.

In Kapitel 6 werden Filesysteme und ihre Bedeutung für das Systemmanagement betrachtet. Es wird Information modelliert, die neben der bloßen Verwaltung von Filesystemen auch die Überwachung der Benutzer bezüglich Plattenspeicherverbrauch ermöglicht. Weiterhin wird die Datensicherung für die MIB modelliert.

[2]Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften

Bereits in Kapitel 5 wird Bezug auf den Benutzer des UNIX-Endsystems genommen. Die Managementinformation zur Verwaltung von Benutzern und Benutzergruppen wird in Kapitel 7 vorgenommen. Zudem erfolgt Informationsdefinition über eingeloggte Benutzer auf Rechensystemen.

Kapitel 8 betrachtet die in den vorangegangenen Kapiteln definierte Managementinformation aus Sicht der klassischen Systemadministratorenbereiche und stellt die Schnittstellen für zukünftige Erweiterungen der MIB vor. Zudem wird der Stand der Implementierung eines integrierten Systemmanagements am LRZ erläutert.

Wie bereits erwähnt enthält Anhang A das entwickelte Klassifikationsschema sowie die Auswertungen der untersuchten MIBs.

In Anhang B befinden sich der Fragebogen, mit dessen Hilfe die Systemverwalterinterviews durchgeführt wurden und eine Auswertung der Interviews.

Im Anhang C ist die definierte MIB in ASN.1-Notation für den CMU-Agenten[3] abgedruckt.

Anhang D enthält detaillierte Implementierungshinweise für die in dieser Diplomarbeit entwickelte MIB.

[3]nähere Information über verwendete Version des Agenten sowie Tools finden sich ebenfalls in Anhang C

Kapitel 2

Analyse des Themenkreises

Dieses Kapitel gibt einen kurzen Überblick über die zwei verbreitesten und für diese Diplomarbeit wichtigen Managementwelten, das Internet-Management und das OSI-Management, und beleuchtet kurz ihre Eignung für das Systemmangement. Es wird eine Auswahl veröffentlichter Management Information Bases auf die von ihnen definierte Managementinformation hin untersucht. Um den Vergleich der MIBs zu ermöglichen und die gewonnenen Erkenntnisse sowohl für die Diplomarbeit als auch für zukünftige systemmanagementrelevante Betrachtungen verwenden zu können wird ein Klassifikationsschema entwickelt und die untersuchten MIBs in dieses übergeführt.

2.1 Eignung von SNMP für Systemmanagement

Das Simple Network Management Protokoll (SNMP) hat sich für das Netzmanagement in der Internetwelt zum de-facto Standard entwickelt. Es wurde von einer Vielzahl von Herstellern auf deren Hardwareplattformen implementiert. Daher liegt es nahe, auch ein Systemmanagement auf der Basis von SNMP aufzubauen.

Das SNMP-Management basiert auf einem Manager-Agenten-Modell. Die Managementanwendung, die im Pollingbetrieb wie ein Client agiert, schickt eine Anfrage an den Agenten. Der SNMP-Agent bearbeitet die Anfrage und sendet das Ergebnis zurück. Diese Technologie wird durch drei RFCs[1] definiert. Der RFC *Structure of Management Information* [RFC 1155] legt das Informationsmodell (vgl. Abbildung 2.1) fest. In ihm werden Strukturierung und Namensgebung durch den Internet-Registrierungsbaum, in dem sämtliche Managementinformation eingeordnet und dadurch weltweit identifizierbar ist, sowie die Sprachelemente, die aus ASN.1 Makros bestehen, zur Beschreibung der Managementinformation vereinbart. Obwohl die Internet-Beschreibungstechnik weder Klassenkonzept noch Vererbungstechnik beinhaltet, können die Managed Objects einer MIB doch als Instantiierungen der Internet-MIB betrachtet werden. Eine detaillierte Beschreibung findet sich in [Hege93].

Jede MIB kann bildhaft als Baum dargestellt werden, der wiederum in diesem Namensraum eingeordnet ist. Die Blätter dieses Baums enthalten die Managed Objects, die Knoten dienen

[1]Request for Comments ist die wichtigste Dokumentform im Standardisierungsprozeß des IAB

lediglich der Strukturierung. Die Objekttypen, definiert durch ASN.1 Makros beschreiben folgende Information:

- Name und Objektidentifikator des Knotens,
- Syntax der in dem Knoten gehaltenen Managementinformation in Form eines ASN.1 Datentyps,
- Informelle Beschreibung der Semantik dieser Managementinformation,
- Implementierungsinformation, die festlegt, ob die Managementinformation obligatorisch oder optional ist.

Da jedes Objekt wieder nur eine Instantiierung haben kann, gibt es die Möglichkeit Tabellen zu bilden, die dann mehrere Instantiierungen enthalten können.

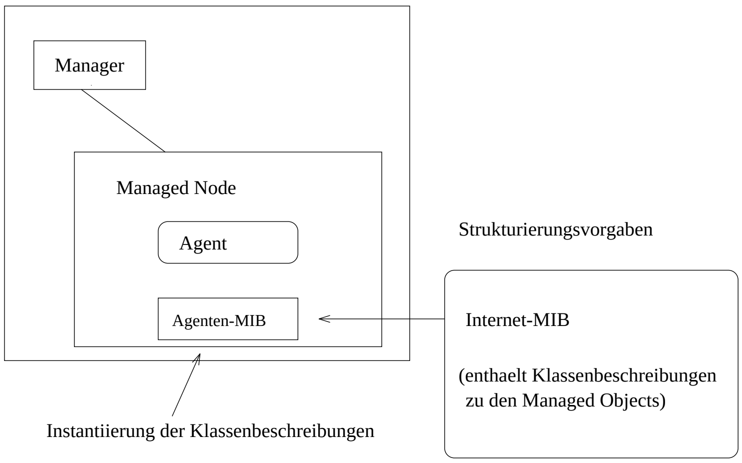

Abbildung 2.1: Internet Informationsmodell

Den zweiten Teil der Technologie stellt das Managementprotokoll SNMP dar. Es wird im RFC *Simple Network Management Protokol (SNMP)* [RFC 1157] definiert. Bei SNMP handelt es sich um ein asynchrones Request-Response-Protokoll. Bild 2.2 gibt eine Übersicht über die vier Grundoperationen des SNMP-Protokolls Version 1. *GetRequest* fragt den Wert einer Objektinstanz ab, *GetResponse* liefert den angeforderten Objektvariablenwert. *GetNextRequest* liefert den nächsten vorhandenen Wert gemäß Object-Identifier. Werte können in einer MIB durch *SetRequest* gesetzt werden. Schließlich können noch Ereignisse spontan mittels Trap durch den Agenten an die Managementstation gesendet werden. Seit Ende 1992 existiert das Protokoll in der Version 2 (SNMPv2), das gegenüber der Version 1 einige Erweiterungen beinhaltet. So können nun große Datenmengen mit einem einzigen *GetBulkRequest* geholt werden. Die Fehlerbehandlung wurde verbessert, der Informationsgehalt von Traps erweitert, sowie ein neues Sicherheitskonzept eingeführt. Informationen hierüber finden sich in [RFC 1448] und [RFC 1452].

Dritter Teil des Internet-Managements ist schließlich die Festlegung einer Standard MIB, der sogenannten MIB-II, die im [RFC 1213] definiert wird. Sie löst die MIB-I ab.

Beschränkungen von SNMP

SNMP weist in Version 1 einige Schwächen auf, die das Systemmanagement unmittelbar betreffen. Das Fehlen eines expliziten CREATE-Mechanismus zur Objektinstanzenerzeugung zwingt die Managementstation dazu, die jeweilige Realisierung im Agenten zu kennen. Wird eine Benutzerverwaltung im Systemmanagement zur Verfügung gestellt und soll ein neuer User eingetragen werden, so muß die Managementstation eine unbenutzte UID, die gleichzeitig die Objektinstanz spezifiziert, finden und diese dann mittels eines SNMP *SetRequest* eintragen. Eine Agentenimplementation, die selbst eine UID sucht, steht dem Internet-Grundsatz, den Agenten möglichst klein zu halten, entgegen ([RFC1157], Kap.3.1).

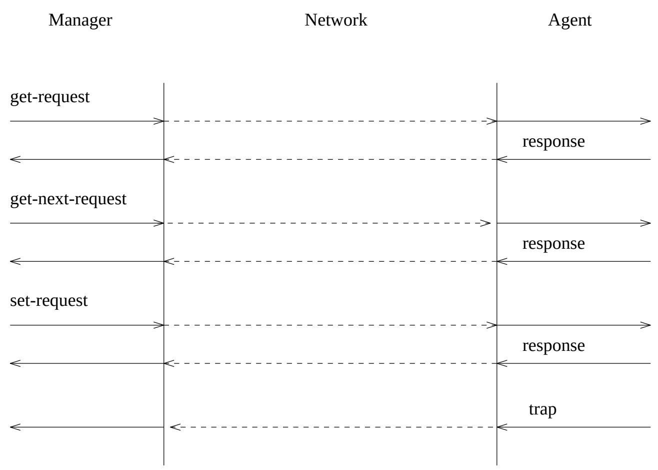

Abbildung 2.2: Ablauf der SNMP-Grundoperationen

Ebenso fehlt eine DELETE-Operation. In vielen MIB-Definitionen wird deshalb in Tabellen eine zusätzliche Spalte eingefügt. Das Schreiben eines vordefinierten Wertes in die Spalte der betreffenden Objektinstanz veranlaßt den Agenten sie zu löschen. Ein bestätigter Trap-Dienst, wünschenswert für komplexere Aktionen, steht nicht unmittelbar zur Verfügung. Einige SNMP-Agenten bilden diesen Dienst nach, indem die Management-Stationen einen bestimmten Wert in eine dafür angebotene Objektinstanz schreiben und damit den Erhalt des Traps bestätigen.

2.2 Das OSI-Systemmanagement und seine Bedeutung für die Diplomarbeit

Das OSI-Managementmodell wird allgemein als der mächtigere und für das Management von komplexen Netzen oder verteilten Systemen geeignetere Ansatz angesehen ([Hege93], [Garb91]). Jedoch ist die Akzeptanzschwelle des OSI-Managements durch die Komplexität und den Abstraktionsgrad der Konzepte sehr hoch, so daß Bereiche außerhalb der Telekommunikation fast immer zuerst mit den Mitteln des Internet-Managements abgedeckt werden.

2.2.1 Das OSI-Managementmodell

Im Gegensatz zum Internet-Management sind bei OSI alle vier geforderten Teilmodelle - sie werden nachfolgend kurz beschrieben - einer Managementarchitektur vollständig ausgeprägt ([ISO 7498]). Das Informationsmodell wird im *Structure of Management Information*-Dokument ([ISO 10165]) beschrieben. Der objektorientierte Ansatz erlaubt Datenabstraktionen und Vererbungsmechanismen. Die Managementobjekte[2] sind Abbildungen realer Ressourcen, die als Black Box modelliert sind. Das innere Verhalten der Objekte wird nur an wohldefinierten Schnittstellen, der MO-Boundary, sichtbar.

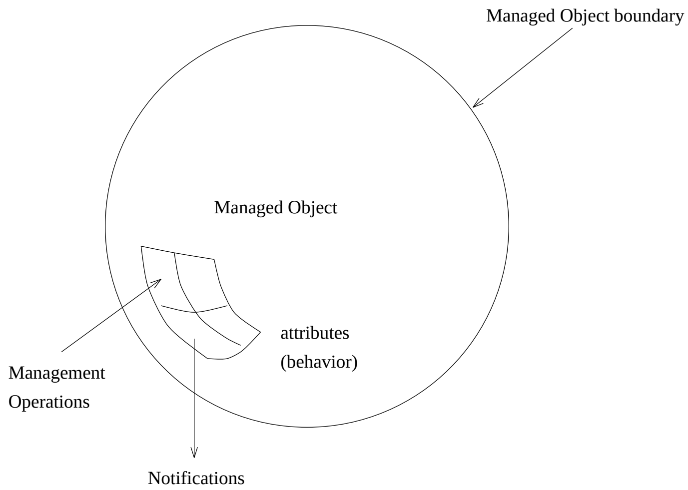

Abbildung 2.3: Modell eines Managed Objects

Dazu bieten die Objekte Methodenaufrufe an, die auf das Objekt wirken. Der Zustand eines Objekts wird durch Attribute beschrieben. Diese sind beobachtbar und/oder veränderbar. Das Organisationsmodell [ISO 10040] bezieht sich eindeutig auf ein Netz von offenen Systemen. Es unterscheidet Manager- und Agentenrolle, wobei ein System auch beide Rollen annehmen kann. Der Datenaustausch zwischen Manager und Agent erfolgt über die Managementprotokolle. Sie sind Bestandteil des Kommunikationsmodells. Weiter wichtige Dokumente sind [ISO 8649], [[ISO 9595] und [ISO 9596]. Als viertes Teilmodell untergliedert das Funktionsmodell den Gesamtkomplex in Configuration, Fault, Performance, Accounting und Security auf und beschäftigt sich mit der Ableitung generischer Funktionen [ISO 10164-nn].

[2] im folgendem Text auch als Managed Objects oder MO bezeichnet

2.2.2 Modellierung von Systemressourcen

Bei der Definition von proprietärer Managementinformation wird in der Diplomarbeit das ISO-Informationsmodell bevorzugt, da es die mächtigeren Modellierungskonzepte bereitstellt. Vorhandene, standardisierte Managementinformation wird, soweit möglich, berücksichtigt. Entscheidend sind insbesondere die RFCs 1213, 1514, 1271 sowie die ISO-Dokumente 10164-2, -5, -6, -11 und -13.

Als Beispiel für die hohe Qualität der ISO-Modelle wird die von der ISO definierte Statusinformation eines Managed Objects, mit der der momentane Zustand eines Objekts beschrieben wird, vorgestellt. So unterscheidet das OSI-Management zwischen allgemeiner Statusinformation, auch als Managementstatus bezeichnet, und speziellen Statusaspekten, die in Abhängigkeit des jeweiligen MOs modelliert werden müssen [ISO 10164-2]. Die allgemeinen Statusinformationen umfassen den operationellen, den administrativen und den Nutzungsstatus. Die speziellen Statusaspekte enthalten einen Reparatur-, einen Installations-, einen Verfügbarkeits- und einen Steuerstatus. Die in Abbildung 2.4 aufgeführten Informationen sind vor allem für physische Ressourcen geeignet.

Die einzelnen Statusmodelle werden bei der Erstbetrachtung während der Modellierung der Managed Objects jeweils erläutert und für jedes MO auf ihre Verwendbarkeit hin überprüft (ab Kapitel 4).

2.3 Untersuchung ausgewählter MIBs und Überführung in ein einheitliches Klassifikationsschema

2.3.1 Auswahl der MIBs

Um sich einen Überblick über die bereits erfolgten Arbeiten zu schaffen wurden aus der Vielzahl der veröffentlichten MIBs sechs Arbeiten ausgewählt und näher betrachtet. Die Eignung der MIBs für das Management von UNIX-Endsystemen war das erste Auswahlkriterium. Von den sechs MIBs definiert lediglich die Host Ressources MIB[3] Managementinformation, die unabhängig vom Betriebssystem, für "... any computer that communicates with other similar computers ..." ([RFC 1514], S. 1) relevant ist. Jedoch bezieht der Autor auch "... systems that run variants of UNIX" ([RFC 1514], S.1) in seine Modellierung mit ein. Das zweite Auswahlkriterium sollte sicherstellen, daß ein möglichst breites Spektrum der MIBs abgedeckt wird. So wurden neben vier Arbeiten aus dem wissenschaftlichen Bereich auch zwei herstellerspezifische MIBs mit in die Untersuchung aufgenommen. Es wurden nur solche MIBs ausgewählt, die eine hohe Güte der definierten Managementinformation erwarten ließen. So fiel die Wahl - die Erwartungen an diese MIBs sind nachfolgend kurz geschildert - auf folgende Management Information Bases:

[3] im folgendem Text auch als HRM bezeichnet

- Host Resources MIB (HRM) ([RFC 1514])

  Seit September 1993 ist die HRM zu einem RFC gworden und wird damit sehr wahrscheinlich zu einem Internet-Standard werden. Es ist davon auszugehen, daß Agenten, die von Rechnerherstellern mitgeliefert werden, oftmals auch die HRM implementiert haben werden.

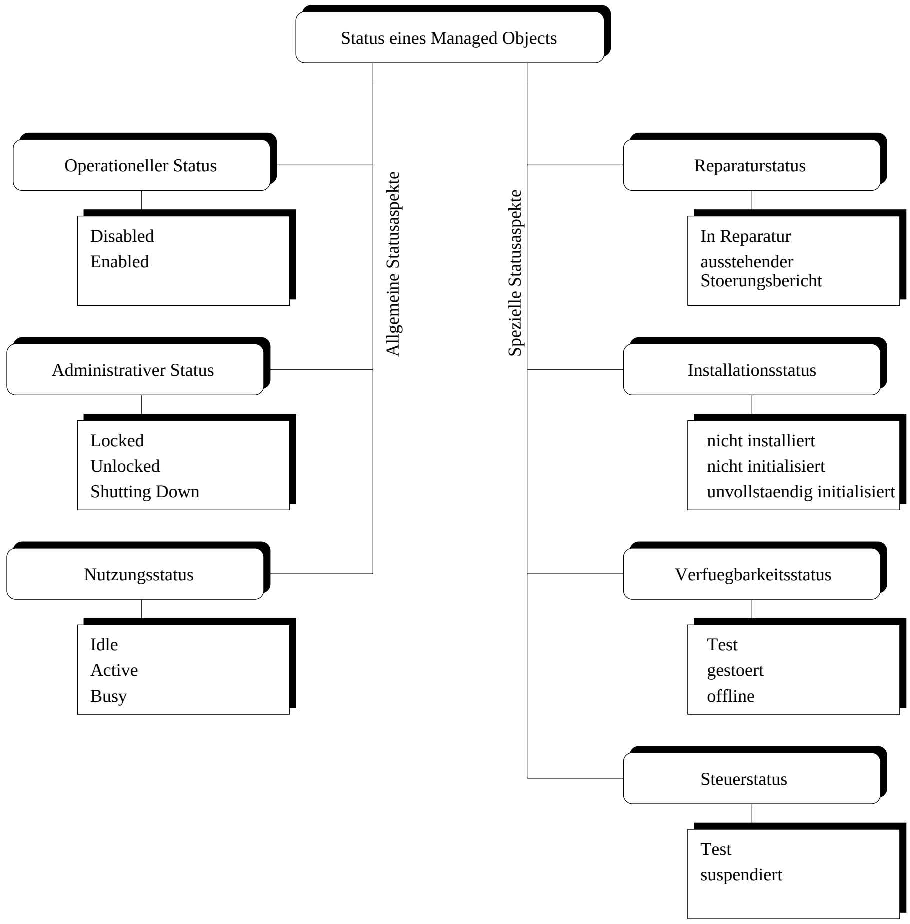

Abbildung 2.4: Zustände eines Managed Objects

- Krupczak-MIB ([Krup93a])

  Diese MIB wurde mit einem sehr ausführlichen Diskussionspapier [Krup93] über die Eignung von SNMP für das Systemmanagement veröffentlicht. Dies läßt auf eine hohe Qualität der Information unter Berücksichtigung der Eigenheiten von SNMP schließen.

- CMU-MIB ([CMU93])

  Als einzigste Management Information Base trägt diese MIB den Namen der Universität, an der sie entwickelt wurde. Dies läßt eine wissenschaftlich fundierte Arbeit erwarten.

- BSD-MIB ([BSD92])

  Diese MIB wurde speziell für das Systemmanagement von BSD-UNIX-Systemen entworfen. Da jedoch beispielsweise SUNOS ein Derivat von BSD-UNIX ist, kann die dort definierte Information sicherlich auch für andere Systeme hohen Nutzen haben. Da Marshall T. Rose, bekannt durch eine Vielzahl von Arbeiten bzgl. des Internet Managements, Mitautor dieser Management Information Base ist, sollte auch diese MIB qualitativ hochwertig sein.

- UNIX-SUN-MIB ([SUN92])

  Als großer Hardwarehersteller ist auch SUN gezwungen, sich mit dem Thema des Systemmangements auseinanderzusetzen. Der Umfang der MIB deutet auf viel Information hin.

- UNIX-HP-MIB ([HP92])

  Ebenso wie SUN ist auch HP Mitglied in vielen Standardisierungsgremien. Umfangreiche Aktivitäten auf dem Gebiet des Netzmanagements (HP Node Manager) lassen auch hohes Engagement auf dem Gebiet des Systemmanagement erwarten.

Alle diese MIBs sind in der ASN.1 Notation verfaßt und für das Internet-Management bestimmt.

2.3.2 Klassifikationsschema

Die Analyse der MIBs muß durch einen Vergleich der definierten Managementinformation vorgenommen werden. Wegen des großen Umfangs der Management Information Bases muß dieser Vergleich strukturiert erfolgen. Deshalb ist ein geeignetes Klassifikationsschema zu entwickeln. Zwar ordnen alle MIBs ihre Variablen in Gruppen an, jedoch sind diese Gruppen weder streng disjunkt noch bedeuten gleiche Gruppennamen zwangsläufig gleiche Information. Dadurch ist eine bloße Aufzählung der Gruppen als Klassifikationschema ungeeignet. Auch eine strikte Gliederung nach den klassischen Managementbereichen aus Administratorensicht (vgl. Kapitel 3) erweist sich als ungeeignet, da sich im Bereich Hardwareadministration zu viel Information sammeln würde. Deshalb wurden die einzelnen Ressourcen der UNIX-Systeme als Grundlage des Klassifikationsschemas gewählt. Für Benutzer- und Softwareverwaltung wurden zwei weitere Klassen definiert. Eine zusätzliche Klasse, die Error Group, wurde zur erweiterten Fehlerbehandlung mit in das Schema aufgenommen.

Das Klassifikationsschma definiert sieben Hauptgruppen:

- System Group

  Diese Gruppe enthält Information, die einen schnellen Überblick über die wichtigsten Daten des Systems und seinen Zustand verschafft. Dazu gehören Aussagen über die Hardware, das verwendete Betriebssystem sowie über Konfiguration und momentane Nutzung.

- Storage Group

  In dieser Gruppe wird Information über die physischen Speichermedien definiert. Diese umfassen Typ, Größe und Füllungsgrad. Die unterschiedenen Speichertypen sind beispielsweise RAM, Festplatten, CD-ROMS.

- Device Group

  Die Deviceklasse umfaßt die meisten Variablen und wurde deshalb mit Untergruppen hierarchisch aufgebaut. Eine Übersichtsklasse zählt die Devices, ihren Typ und Status auf. Weiterhin können Name und Produktidentität des Devices eingesehen werden. Die zweite Unterklasse, Prozessorklasse, stellt Information über den oder die verwendeten Prozessor(en) zur Verfügung und liefert Werte, die die Auslastung des Systems charakterisieren. Von den peripheren Geräten fand in den MIBs lediglich der Drucker Beachtung. Diese Unterklasse liefert neben Statusinformtion über den jeweiligen Drucker auch Managementinformation über die Druckjobs und die Druckschlangen. Die wichtigsten Speichermedien sind momentan Plattenlaufwerke. Deshalb werden in einer eigenen Untergruppe der Plattentyp, die Auswechselbarkeit, die Größe und wichtige Belastungsdaten beschrieben. Viele verwendete Platten beherbergen wiederum verschiedene Partitionen. Deshalb liefert die fünfte Untergruppe Daten über Partitionen. Die sechste Untergruppe ermöglicht detaillierte Aussagen über die verwendeten Filesysteme eines Systems. Neben dem Mountpunkt finden sich auch Informationen zu Inodes und Files, sowie zum Datum des letzten Backups.

- Process Group

  Prozesse sind ein wesentlicher Bestandteil eines UNIX-Systems. Sie beeinflussen maßgeblich die Auslastung eines Rechners. Jedes UNIX bietet heute umfangreiche Möglichkeiten das Benutzerverhalten durch Konfigurationfiles (Größe eines Prozesses, Anzahl der Prozesse) zu steuern.

- User Group

  Als eigener Managementbereich ist die Benutzerverwaltung zu sehen. Neben Zugangsberechtigung und Gruppenzugehörigkeit wird auch Information über den eingeloggten Benutzer definiert.

- Error Group

  Nicht alle auftretenden Fehler können unmittelbar über das Systemmanagement abgefangen werden, da beispielsweise die Schwellwertüberwachung durch einen Agenten sehr hohe Anforderungen an das System stellt. Deshalb bietet eine eigene Fehlerklasse, die die Möglichkeit definiert an der Konsole gemeldete Fehler mitzuprotokollieren, großartige Möglichkeiten zur Ursachensuche bei fehlerhaften Zuständen des Systems.

- Installed Software Group

  Bestandteil eines integrierten Systemmanagements ist die Softwareadministration. Sie umfaßt die Überwachung bestehender Installationen sowie die Konfiguration installierter Pakete. Durch Bildung einer eigenen Klasse wird der OSI Forderung entsprochen, daß Softwarekomponenten als eigenständige Managementobjekte zu behandeln sind.

2.3.3 Überführung der MIBs in das Klassifikationsschema und Bewertung

Mit dem im vorherigen Kapitel definierten Klassifikationsschema ist es möglich, die in den MIBs festgelegte Information den einzelnen Gruppen eindeutig zuzuordnen. Anhang A enthält die detaillierte Auswertung und Gliederung der sechs untersuchten MIBs gemäß dem Klassifikationsschema. Nicht mitaufgenommen wurde Managementinformation, die eher dem Netzmanagement zuzuordnen ist (vgl. Kapitel 1.1). Ebenfalls nicht betrachtet wurden Variablen zur Speicherverwaltung für die Interprozeßkommunikation, da sie lediglich von der BSD-MIB angeboten werden und durch die unsaubere Definition nicht für ein Systemmanagement geeignet sind. Ausgeschlossen wurde auch Information, die der Verwaltung eines Agenten dient, da sie nicht unmittelbar Bestandteil eines Systemmangements ist, sondern vom jeweiligen Agenten vorgesehen werden muß.

Betrachtet man die Vereinigung aller MIBs, so stellt man fest, daß viel Managementinformation festgelegt wird. Die Abbildung 2.5 zeigt die Anzahl der Variablen, die die untersuchten MIBs für die einzelnen Klassen des Klassifikationsschemas definieren. Die MIBs einzeln betrachtet weisen jedoch deutliche Schwächen auf. So kann die CMU-MIB die an sie gestellten Erwartungen nicht erfüllen. Variablen werden nicht kommentiert und entbehren somit jeder Semantik. Der Verfasser stellt sich auf den Standpunkt, daß "the semantics of each object

Klassen	**HRM**	**Krupzak**	**CMU**	**BSD**	**SUN**	**HP**
System Group	7	6			4	6
Storage Group	7	3	3			1
Device Group - Uebersicht	5	3		1		
Device-Group - Prozessor	2	7	6		4	7
Device Group - Drucker	2		9			
Device Group - Plattenlaufwerke	4		2		5	
Device Group - Plattenpartitionen	4					
Device Group - Filesystem	8	5	6	9		10
Process Group	8	7	3		10	31
User Group		15		11		
Error Group		9				
Installed Software Group	5					
Anzahl der Variablen	52	55	29	21	23	55

Abbildung 2.5: Übersicht über ausgewertete MIBs mittels Klassifikationsschema

should be understandable given the object descriptor (and perhaps some knoledge of UNIX)" ([CMU93]). Die HP-MIB definiert zwar zum Prozeßmanagement über 30 Variablen, aber bei genauer Analyse stellt man fest, daß zwar unter anderem Größe des Text- und Datasegments eines Prozesses verfügbar sind, aber keine einzige Schnittstelle implementiert wird, mit dem es der Managementanwendung möglich wäre einen Prozeß in seiner Priorität zu ändern oder anzuhalten. Ursache dieser Informationsflut ist die Library-Funktion pstat() unter HP-UX, die genau die definierten Werte liefert. An dieser Stelle wird deutlich, daß auch im Systemmanagement gerne Information zur Verfügung gestellt wird, die leicht erhaltbar ist, anstatt der Information die wirklich benötigt wird.

Sowohl die SUN- als auch die BSD-MIB bieten in den einzelnen Klassen zu wenig Information, als daß sie die Anforderungen an ein Systemmanagement erfüllen können. Jedoch ist die BSD-MIB neben der Krupczak-MIB die einzige Management Information Base, die durch alle von ihr definierten Klassen hindurch aktives Management per Definition ermöglicht. So kann beispielsweise durch das Setzen der Variable printQaction eine Druckerschlange gestartet, angehalten, gesperrt und auch eliminiert werden. Die HRM definiert in ihrer aktuellen Version solche Variablen leider nicht. Jedoch ist die klare Gliederung der einzelnen Klassen vorbildhaft. So findet man zwischen den einzelnen Gruppen keine Überschneidungen. Weiterhin sind umfangreiche Möglichkeiten implementiert um Querverweise auf andere Klassen vorzunehmen. Dadurch ist unter anderem eine sofortige und eindeutige Zuordnung von Plattenpartitionen zu Filesystemen möglich.

Die System Group wird zwar nur von vier der sechs MIBs implementiert, aber man kann argumentieren, daß die Zuordnung einzelner Variablen, auch wenn sie geeignet sind einen schnellen Überblick über das System zu geben, doch besser in den jeweiligen Klassen, denen sie semantisch zugehören, aufgehoben sind. Denn für ein Batch-System ist beispielsweise die Länge einer Job-Schlange eine wichtige Aussage, während für einen Arbeitsplatzrechner die momentane Auslastung bedeutsamer ist. Zur Storage und Device Klasse definieren alle MIBs Managementinformation. Jedoch macht außer der HRM keine weitere MIB eine wirklich deutliche, strukturierte Unterscheidung zwischen physikalischen und logischen Speichermedien. Sie begnügen sich mit einer Aussage entweder zum Filesystem oder zu den Pattenlaufwerken ohne jedoch eine Verknüpfung, die ja für das Systemmangement wichtig ist, herzustellen. Zu der Prozeßklasse definieren alle Management Information Bases außer der BSD-MIB Managementinformation, während zur Benutzeradministration lediglich die Krupczak-MIB und die BSD-MIB Variablen vorsehen. Die Krupczak-MIB hat als einzige MIB eine Error-Klasse implementiert. Diese Klasse soll Fehlermeldungen, die an der Konsole auflaufen, protokollieren und festhalten, von welchem Prozeß die Fehlermeldung verursacht wurde. Lediglich die HRM unternimmt den Versuch eine Schnittstelle zu dem bis heute unzureichend gelösten Problem der Softwareverwaltung zu definieren. Einerseits ist klar, daß eine Systemmanagement Information Base dieses komplexe Thema, zu dem eigenständige Softwareprodukte wie HP Software Distribution Utilities oder SNI SAX (nähere Information findet sich in [Kauf92]), nicht vollständig lösen kann, andererseits ist der praktische Nutzen einer Klasse, die lediglich den Namen, Produktbezeichnung und Installationsdatum von Softwarepaketen enthält, anzuzweifeln. Selbst eine Versionsnummer kann eigentlich nur durch Anhängung an den String des Namen abgespeichert werden und steht somit einer Weiterverarbeitung, wie es die Funktionalität von Softwareverteilungs- oder Softwarelizenzmechanismen fordert, nicht zur Verfügung.

Kapitel 3

Anforderungen eines Rechenzentrumbetreibers (LRZ)

Das LRZ (Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften) nimmt für die Münchener Hochschulen (TUM, LMU, FH) die Aufgabe eines Hochschulrechenzentrums wahr. Es unterstützt die Angehörigen der Münchener Hochschulen in allen Fragen des Einsatzes von Rechnern, betreibt das Rechnerkommunikationsnetz der Münchener Hochschulen mit nationalen und internationalen Verbindungen und stellt Rechenleistung auf verschiedensten Systemen zur Verfügung.

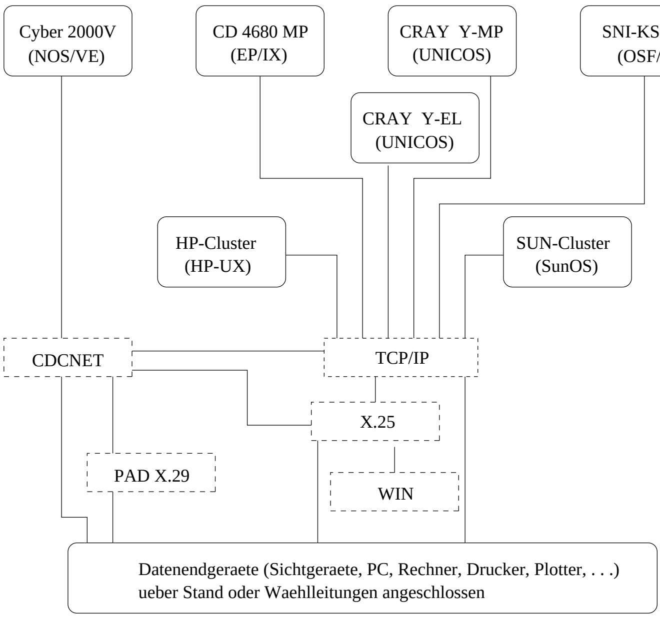

Abbildung 3.1: Schematische Anlagenkonfiguration des LRZ

Das LRZ verfügt über eine umfangreiche Hardwareausstattung. Die Maschinen lassen sich in Hochleistungsrechner, Server, Workstation-Cluster und Arbeitsplatzrechner mit peripheren Geräten unterteilen.

Hochleistungsrechner

Als bayerischer Landesvektorrechner ist eine Cray Y-MP (Typ 8/8128) installiert, die über acht Zentralprozessoren und einen Hauptspeicher von 128 Millionen Worten zu 64 Bit (1024 MByte) und einem Hintergrundspeicher von ca. 80 GByte verfügt. Der Rechner arbeitet unter dem von Cray entwickelten UNIX-ähnlichem Betriebssystem UNICOS. Eng gekoppelt mit diesem Rechner ist der Vektorrechner Cray Y-MP EL. Seine Ausstattung umfaßt zwei Zentralprozessoren, einen Hauptspeicher von 64 Millionen Worten zu 64 Bit (512 MByte) und einem Hintergrundspeicher von etwa 22 GByte. Weiterhin steht eine SNI-KSR1-32, ein Parallelrechner der Firma Kendall Square Research von SNI vertrieben, mit 32 Prozessoren und ca. 20 GByte Plattenspeicher zur Verfügung. Als Betriebssystem ist OSF/1 installiert, als Batchsystem wird wie auch bei den beiden Cray-Maschinen NQS eingesetzt.

Zentrale Server

Als zentrale Server werden zwei Maschinen der Firma Control Data eingesetzt. Zum einen eine Cyber 2000V unter dem Betriebssystem NOS/VE, welche jedoch zugunsten von UNIX-basierten Systemen aufgegeben werden soll, zum anderen eine CD 4680MP mit zwei Zentralprozessoren, die zukünftig als Datenarchiv-Rechner für ein robotergesteuertes Kassettensystem mit über 8000 GByte Speicherkapazität eingesetzt werden wird. Beide Maschinen werden mit dem Batchsystem NQS (im Verbund mit den Vektorrechnern und dem Parallelrechner) eingesetzt.

Workstation Cluster

Da die leistungsfähigsten UNIX-Systeme bereits in den Bereich der klassischen Universalrechner reichen, setzt auch das LRZ verstärkt auf UNIX basierten Client-Server-Systemen. Momentan werden zwei gekoppelte HP-Cluster, bestehend aus einem Server HP 750 und 6 Clients HP 730, betrieben. Die Server verfügen über 128 MByte Arbeitsspeicher und 10 GByte Plattenspeicher, die Clients über 64 MByte Arbeitsspeicher und je 1,4 GByte Plattenspeicher. Betriebssystem ist das HP eigene UNIX HP-UX. Als Filesystem wird AFS und als Batchsystem DQS ,eine Weiterentwicklung von NQS, eingesetzt. Neben den HP-Clustern existiert noch ein SUN-Cluster mit einem Server SUN 630MP (64 MByte Arbeitsspeicher, 5 GByte Plattenspeicher) sowie vier SPARCstations 10/4x und 4 Workstation des Typs ELC und IPX (32-64 MByte Arbeitsspeicher, 200-500MByte Plattenspeicher).

Arbeitsplatzrechner und periphere Geräte

Weiterhin verfügt das LRZ über eine Vielzahl von Arbeitsplatzrechnern, die u. a. unter den Betriebssystemen MS-DOS, MS-Windows, MAC und NEXTStep betrieben werden. Zusätzlich existieren noch eine Vielzahl von Peripheriegeräten. Dazu gehören Drucker, Schriftenleser und Scanner.

3.1 Entwicklung eines integrierten Systemmanagements am LRZ

Es ist offensichtlich, daß in einer solch heterogenen Umgebung ein integriertes Systemmanagement unabdingbar ist.

3.1.1 Vorarbeiten des LRZ

Bereits 1991 wurde im LRZ das Pilotprojekt "Watch - PC" durchgeführt, bei dem versucht wurde auf PC-Basis Rechner zu überwachen. Im August 1992 wurde ein Arbeitspapier veröffentlicht [SBAA92], in dem die Erwartungen, Vorgaben und Anforderungen an ein Systemmanagement formuliert wurden. Ziel ist eine zentrale Überwachung aller installierten Anlagen, unabhängig von Hardware und Betriebssystem unter Benutzung einer Managementzentrale. Als Interaktionsmechanismen zwischen Managementzentrale und dem zu überwachendem System wurden folgende Punkte aufgezählt:

- Pollingmechanismus, indem die Managementzentrale periodische Überwachungsmechanismen anstößt.
- Pollingmechanismus, allerdings werden Daten gelesen und ausgewertet, die ein Dämon in einem Logfile ablegt.
- Agent (als Überwachungsprozeß bezeichnet), der bei "auffälligen Ereignissen die Managementzentrale aktiv benachrichtigt." ([SBAA92]).

Als zu überwachende Ressourcen wurden insbesondere CPU, Hauptspeicher, Plattenlaufwerke, Prozesse, Netzverbindungen genannt. Im Fehlerfalle wurde von der Managementzentrale die Fähigkeit erwartet, den betroffenen Rechner im Überblicksfeld einer grafischen Darstellung farblich zu markieren, die Störung in einem Logfile festzuhalten und die zuständigen Personen durch akkustische Signale, E-Mail und Telefon zu benachrichtigen. Damit muß das Management aus Agenten und Manager bestehen. Die Agenten sind auf den jeweiligen Systemen installiert, sammeln dort mittels lokaler Mechanismen die benötigte Information und übermitteln sie über ein Managementprotokoll an den Manager, der die Daten aufbereitet und in verständlicher und übersichtlicher Form präsentiert.

3.1.2 Planung des Systemmanagements unter Berücksichtigung der Forderungen des LRZ

Abbildung 3.2 zeigt den strukturellen Aufbau des zu entwickelnden Systemmanagements. Dieser Aufbau wurde in Kooperationssitzungen mit Vorträgen des Verfassers zwischen der LMU-München und dem LRZ gestaltet. Es werden drei Managerschnittstellen verlangt. Die Operatorschnittstelle ermöglicht den Operatoren die Überwachung des laufenden Betriebs, sowie das Ausführen einfacher administrativer Tätigkeiten. Diese umfassen hauptsächlich standardisierbare Tätigkeiten wie Datensicherung, Bedienung von Peripherie, Teile der Benutzerverwaltung und Fehlerbehandlung. Über die Bereitschaftsdienstschnittstelle muß bei schwerwiegenden Fehlern außerhalb der normalen Arbeitszeiten der Notdienst verständigt werden. Hier können je nach Implementierung Telefon oder Eurosignal eingesetzt werden. Zusätzlich muß über diese Schnittstelle dem Notdienst mit genau definierten und auf das Nötigste beschränkten Privilegien die Möglichkeit gegeben werden den Fehlerzustand zu beseitigen. Die dritte und wichtigste Schnittstelle ist die des Systemverwalters. Sie sollte dem jeweiligen Administrator die Möglichkeit bieten alle anfallenden Tätigkeiten mittels der ihm zur Verfügung gestellten Funktionen zu erledigen.

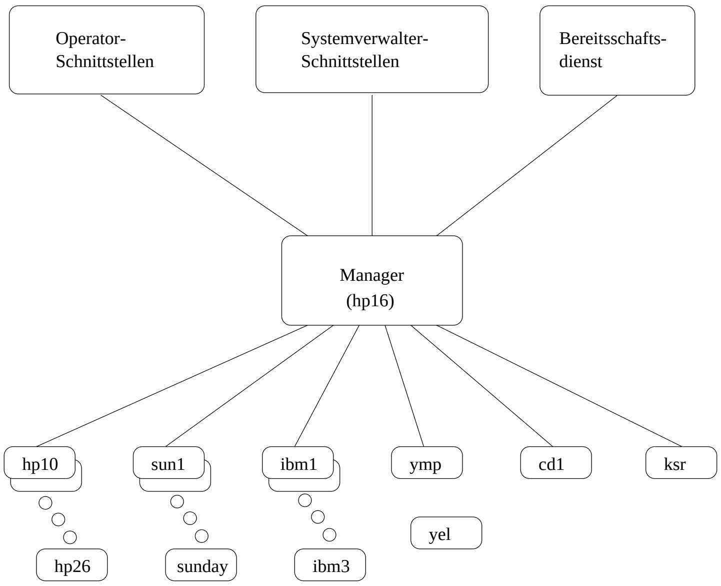

Abbildung 3.2: LRZ Systemmanagement

Der Manager soll in der Anfangsphase als zentrale Anwendung auf einer Workstation (hp16) installiert werden. Das LRZ hat verschiedene Managementanwendungen auf ihre Eignung für ein integriertes Systemmanagement hin überprüft und sich für HP-Open-View und Systems-Monitor entschieden. Die ursprüngliche Idee, die Managementinformation mittels RPC auszutauschen, wird zugunsten des standardisierten und weitverbreiteten Managementprotokolls SNMP aufgegeben. Auf den jeweiligen Systemen werden Agenten installiert. Diese Agenten stellen die benötigte Information bereit und ermöglichen in Interaktion mit dem Manager synchronen Betrieb als auch die Generierung von asynchronen Meldungen aufgrund systeminterner Ereignisse. Aus Sicht des LRZ hat die Minimierung des anfallenden Netzverkehrs absoluten Vorrang. Dies schließt ein periodisches Polling der Managementstation, wie es noch in [SBAA92] formuliert wurde, aus. Deshalb müssen die Agenten sowohl Schwellwertüberwachung als auch periodisches Logging, jeweils flexibel konfigurierbar, ermöglichen.

3.2 Anforderungen der Systemverwalter an ein integriertes Systemmanagement

Ein integriertes Systemmanagement deckt idealerweise alle anfallenden Tätigkeiten der Systemverwalter ab und stellt jede benötigte Information zur Verfügung.

3.2.1 Integration der Systemverwalter

Hauptbestandteil des Managements ist die Management Information Base, die als konzeptioneller Datenbehälter sowohl die Aktionen auf die Managed Objects als auch die Mangementinformation, die zur Verfügung steht, definiert. Wie in Kapitel 2.3.3 bereits erwähnt, wird in MIBs oftmals Information definiert, die leicht erhältlich ist, statt der Information die wirklich benötigt wird. Deshalb wurde für die Diplomarbeit ein Top-Down-Vorgehen gewählt. In Interviews mit den Systemverwaltern des LRZ wurde festgestellt welche Managementinformation wirklich benötigt wird, welche wünschenswert und welche überflüssig ist. Folgende Vorgehensweise wurde gewählt:

- Ausarbeitung der in exemplarisch ausgewählten MIBs vorhandenen Information (vgl. Anhang A) und Verteilung an die Systemverwalter.
- Interviews in denen eine Beurteilung der MIBs durch die Systemverwalter erfolgte und der Aufgabenbereich der Systemadministratoren mit folgenden Hauptfragenstellungen ermittelt wurde:
  - Verwendete Tools zur Systemverwaltung?
  - Welche Objekte und Werte werden überwacht?
  - Wieso werden sie überwacht?
  - Wie werden Fehlerzustände erkannt?
- Definierung der wesentlichen Informationsgruppen und abschließende Beurteilung durch die Systemverwalter.

Es wurden die Administratoren folgender Systeme miteinbezogen:

- Cray Y-MP, Cray Y-MP EL
- KSR
- CD1
- DEC-Workstations
- HP-, SUN-Workstations
- IBM-Workstations unter OSF DCE

3.2.2 Ergebnisse der Systemverwalterinterviews

In Anhang B finden sich zu den Hauptfragen nochmals ausgewählte Antworten, sowie eine kurze Auswertung der Ergebnisse. Im folgenden werden die Ergebnisse interpretiert.

Der Managementbereich der Hardwareadministration stellt über alle Systeme hinweg einen Schwerpunkt der Verwaltertätigkeit dar. Jedoch wird lediglich von den Workstationadministratoren detaillierte Information über den Ausbau der Geräte (CPU-Typ, Hauptspeicher, Plattenlaufwerke) erwartet. Dies hat zwei Gründe. Zum einen müssen bei einer Vielzahl von gleichartigen Systemen, falls diese Information nicht von einem Systemmanagement dynamisch abrufbar ist, Listen geführt werden und deren Aktualisierung bei Änderungen der Hardwarekonfiguration sichergestellt werden. Zum anderen haben Workstations durch den Preisverfall auch Einzug in den privaten Bereich gefunden und so sind beispielsweise Speichermodule diebstahlgefährdet. Bei Einzelrechnern wie Cray oder KSR weiß der Verwalter selbstverständlich genau über den Ausbau Bescheid, so daß diese Information nicht unbedingt benötigt wird. Auch die Verwaltung der Peripherie gehört zur Hardwareadministration. Hier interessieren sich die Administratoren nicht nur für den genauen Gerätetyp und Aufstellungsort, sondern auch für eine detaillierte Information über den Zustand und die Möglichkeit zu überprüfen, ob das Gerät richtig arbeitet. In diesem Bereich vermißten die Administratoren Information in den MIBs.

Das Performance-Tuning besteht aus den Bereichen Datenerfassung von Performancedaten und Installations- oder Konfigurationsänderungen zur Steigerung der Performance. Dies wird von allen Verwaltern als wichtig erachtet. Insbesondere werden Daten über die Auslastung von Rechensystemen und Peripheriegeräten sowie über die Plattenbelegung erfaßt. Zu den Maßnahmen des Performance-Tunings gehören beispielsweise beim Erkennen von Bottle Necks Konfigurationsänderungen, Hardwareerweiterungen und die Verlagerung von Diensten und Softwarepaketen auf andere Rechner. Auch hier sollte nach Meinung der Administratoren mehr detaillierte Information, unterstützt von Log-Mechanismen, zur Verfügung gestellt werden.

Einen weiteren Schwerpunkt ihrer Arbeit sahen die Workstationverwalter in der Datensicherung. Dabei wird insbesondere Information über das Datum des letzten Backups sowie eine Abstufung in verschiedene Backuplevel (vollständig, inkrementell) gewünscht. Für die Verwalter der Einzelsysteme ist die Datensicherung durch die Automatisierung selbstverständlich.

Eine leistungsfähige Softwareadministration wird von allen Systemverwaltern gewünscht. Bisher ist ein solches Tool im LRZ noch nicht im Einsatz. Neben Auskunft über Installationsdatum, über installierte Version, Implementierung einer Uninstall-Routine wird auch ein Mechanismus gewünscht, der sicherstellt, daß die Software richtig arbeitet.

Die Prozeßüberwachung, insbesondere das korrekte Arbeiten von Dämonen hat für alle Systemadministratoren einen hohen Stellenwert. Als wichtigste Dämonen wurden Filesystemdämonen und UNITREE-Dämonen genannt. UNITREE wird im LRZ beispielsweise an der Cray verwendet. Es ermöglicht die Handhabung großer Datenmengen. Dabei befindet sich der überwiegende Teil der Daten auf preisgünstigen Magnetbandkasetten. Ein Plattenpool wird als Cachespeicher verwendet. Ein automatisches Robotersystem führt die Übertragung der Daten von den Magnetbändern auf die Platten durch.

Das Fehlermanagement ist eine wichtige Komponente der Systemadministratorentätigkeit. Fehler können prinzipiell in allen Managementbereichen auftreten. Nach Ansicht der Systemadministratoren können Logmechanismen die Fehlersuche wesentlich erleichtern.

Die Benutzerverwaltung umfaßt neben der Vergabe von Rechnerkennungen auch die Kontingentierung von Plattenplatz und Rechenzeit. Die untersuchten MIBs beschränken sich auf Vergabe von Rechnerkennungen und Information über Namen und Gruppenzugehörigkeit. In diesem Bereich vermißten die Administratoren keine Information, obwohl natürlich die Kontingentierung der Rechenzeit als interessant erachtet wurde.

Momentan werden diese Aufgabenbereiche von den Systemverwaltern und Operateuren mittels UNIX-Systembefehlen (z.B. df für freien Speicherplatz, ps für Prozeßübersicht), Public-Domain-Tools (z.B. top für die Auslastung des Systems und Übersicht über rechenintensive Prozesse), selbstprogrammierten Shell-Scripts und rudimentären Überwachungssystemen (z.B. SICK, ein Cray-Überwachungssystem), die sich wiederum aus Shell-Scripts und Cron-Prozessen zusammensetzen.

Diese Situation ist unbefriedigend, da sie den Operateuren und Systemverwaltern keine einheitlichen Schnittstellen anbietet.

3.2.3 Ableitung der Anforderungen an die MIB

Zusammenfassend wird deutlich, daß die Systemverwalter die Vorteile eines integrierten Systemmanagements erkennen. Allerdings müssen, um die Akzeptanz dieses Managements zu gewährleisten, nicht nur detaillierte Informationen über die zu managenden Bereiche und Ressourcen zur Verfügung gestellt werden, sondern es müssen auch eindeutige Möglichkeiten existieren, die ein aktives Beeinflussen und Steuern der Ressourcen ermöglichen. Deshalb wird in den folgenden Kapiteln, in denen eine MIB für UNIX-Endsysteme entwickelt wird, bei der Modellierung von Managed Objects äußerste Sorgfalt darauf gelegt, daß die MOs alle für ein integriertes Systemmanagement benötigten Methoden anbieten. Ein zweiter Schwerpunkt liegt in dem Versuch, alle realen Ressourcen soweit wie möglich zu abstrahieren, um eine Wiederverwendung der definierten Managementinformation auch für andere Objekte sicherzustellen.

Kapitel 4

Modellierung der Hardware

In diesem Kapitel werden ausgewählte Bestandteile der Hardware für ein UNXI-Endsystem modelliert. Zuerst wird der Prozessor, dann der Arbeitsspeicher und abschließend die Massenspeicher betrachtet. Die verschiedenen Bussysteme werden ebenfalls kurz dargestellt. Pe-

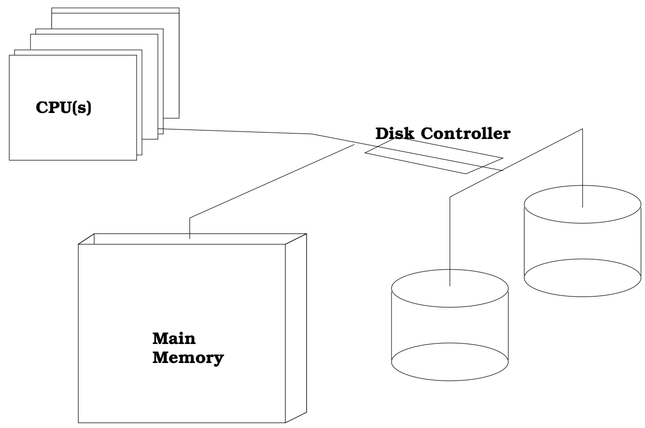

Abbildung 4.1: UNIX-System

ripheriegeräte, beispielsweise Drucker oder Scanner, werden in diesem Kapitel nicht berücksichtigt. Insbesondere wird der physische Speicher von logischen Speichern abstrahiert.

Gemeinsamkeiten der UNIX-Systeme sind die enge Kopplung einer Integer Unit (IU) mit einer Floating Point Unit (FPU), der Einsatz einer Memory Management Unit (MMU), die Verwendung großer Cache-Speicher und die Kopplung von Cache und Hauptspeicher über einen schnellen Speicherbus.

Der Kern eines jeden Rechners besteht aus der Zentraleinheit, in der die arithmetisch-logischen Verknüpfungen der Daten in Abhängigkeit vom Programm durchgeführt werden. Bei grösseren Rechnern bezeichnet man oft den eigentlichen Rechner als Zentraleinheit, an der Datensichtgeräte, Drucker, Massenspeicher und andere Peripherie angeschlossen ist. Bei Personalcomputern wird oftmals in Analogie zu den großen Rechnern die Hauptplatine als Zentraleinheit bezeichnet.

4.1 Allgemeine Information über das zu managende System

Es gibt über jeden Rechner wünschenswerte Informationen, die sich nicht unmittelbar einer der in den folgenden Kapiteln entwickelten Managementgruppen zuordnen lassen, sondern allgemeine Information liefern. Diese Information wird in einer eigenen Gruppe, der System Group, modelliert. Rechner- und Domainenname werden üblichrweise vom Netzmanagement gepflegt. Deshalb werden sie, um nicht mit dem Netzmanagement zu kollidieren lediglich als lesbarer String in die MIB aufgenommen (z.B. sunhegering5@informatik.tu-muenchen.de). Für jeden Rechner sollte eine Kontaktperson benannt werden, die für den Betrieb verantwortlich ist. Sein Name kann zusammen mit Information, wie man ihn erreichen kann, in der Variable *sysContact* hinterlegt werden. In verteilten Rechnerumgebungen ist ebenfalls die Information über den Aufstellungsort der Maschine wichtig. Dieser kann in der Variable *sysLocation* gespeichert werden. Es ist offensichtlich, daß diese Information ebenso wie *sysContact* von Hand gepflegt und bei eventuellen Änderungen aktualisiert werden muß. Ebenfalls in der allgemeinen Informationsgruppe wird eine textuelle Beschreibung der Maschine sowie das Betriebssystem aufgenommen. Das Betriebssystem wird zwar in der Implementierung dynamisch vom Rechner erfragt werden (vgl. Anhang D), es ist aber in dieser Gruppe eher als Abrundung zum allgemeinen Informationsgehalt zu sehen. Eine Releaseverwaltung, wie sie beispielsweise ein Softwaremanagement vorsehen würde, kann und soll damit nicht durchgeführt werden. Die Variable *sysUptime* nennt den vergangenen Zeitraum in Sekunden seit dem letzten Booten und unterscheidet sich damit von der änlich lautenden Variable *sysUpTime* der MIB-II, die angibt, welcher Zeitraum seit der letzten Initialisierung des Agenten vergangen ist. Die letzten zwei Werte liefern zur Abrundung die Zahl der eingeloggten Benutzer[1] sowie die Systemzeit des Rechners. Die Systemzeit wird zwar als beschreibbarer Wert modelliert, aber es ist auch hier zu beachten, daß dieser Wert keinenfalls die Mächtigkeit eines OSF DCE Distributed Time Services ([OSF DCE] hat und damit auch nicht nachgebildet werden soll.

4.2 Prozessor

Die Zentraleinheit eines Mikrocomputers besteht in der Regel aus einer Reihe komplexer hochintegrierter Bausteine und der dazu erforderlichen Verknüpfungsschaltung. Die Schaltzentrale der Zentraleinheit eines Mikrocomputers ist der Mikroprozessor.

4.2.1 Übersicht

Die heutigen Prozessoren lassen sich nach der Entwurfsphilosophie in zwei Klassen unterteilen, den RISC und den CISC-Prozessoren. Anfang der siebziger Jahre begann man mit dem Complex Instruction Set Computing, bei dem aufgrund des begrenzten und teuren Hauptspeichers ganze Algorithmen im Prozessor als Mikroprogramm implementiert wurden.

[1] Das Loggen dieses Wertes ermöglicht beispielsweise eine Statistik über die Nutzungshäufigkeit

Anfang der achtziger Jahre hatten sich die Randbedingungen für Mikroprozessoren grundlegend verändert. Hauptspeicher wurde in Halbleitertechnologie gefertigt, die Zugriffszeiten verkürzten sich und durch den Preisverfall nihilierte sich der Vorteil der Mikroprogramme den Hauptspeicherbedarf zu verringern. Zudem wuchs die Erkenntnis, daß auch Firmware selten fehlerfrei ist und Untersuchungen zeigten, daß Compiler nur einen geringen Teil der komplexen Befehle der Prozessoren nutzten ([Hopk87]). Dagegen sind die Befehlssätze von RISC-Prozessoren klein, aber elementar, und werden von speziellen RISC-Compilern voll genutzt. Die meisten Befehle werden innerhalb eines Taktzyklus abgearbeitet, nur Befehle die auf den langsameren Speicher zugreifen, benötigen zur Ausführung zwei bis vier Taktzyklen ([Bode91a]). Durch Parallelisierung von Ausführungseinheiten und Pipelining können bis zu drei Instruktionen pro Taktzyklus ausgeführt werden. Dagegen benötigte Intels 8080 bei den meisten Instruktionen 8 bis 12 Taktzyklen.

Heute existieren zwei unterschiedliche RISC-Varianten. Die Stanford- Variante, als bekanntester Vertreter ist der MIPS R3000-Prozessor zu nennen, erkennt hardwaremäßig keine Piplinehemmnisse, d.h. der Compiler muß durch Veränderung der Reihenfolge der Befehle und Einstreuung von NOOPs für die semantische Richtigkeit des erzeugten Codes sorgen. Der Prozessor ist mit 32 Allzweckregistern ausgestattet.

Die Berkely-Variante bietet durch Scoreboarding, darunter versteht man die Kennzeichnung eines Datums, dessen Wert noch nicht gültig berechnet ist, Hardwareunterstützung. Zudem verfügen diese Prozessoren über eine kompliziertere Registerorganisation. Zu dieser Gattung gehören der SUN SPARC, der AM29000, der IBM 6000

Eine ausführliche Behandlung von CISC- und RISC-Prozessoren finden sich in [Bode91a], [Bode91b] und [Muel91].

4.2.2 Modellierung des Prozessors für das Systemmanagement

Für das Systemmanagement wird die Prozessorbezeichnung und die Taktfrequenz des Prozessors durch lesbare Attribute modelliert. Sie ermöglichen dem Systemverwalter in großen Netzen den genauen Typ eines Systems zu bestimmen und können bei genügendem Hintergrundwissen bereits zu einer ersten Leistungsabschätzung des Systems verwendet werden. Kapitel 4.2.3 modelliert die Leistungsfähigkeit eines Prozessors detaillierter. Über alle Ressourcen eines Systems sollte eine Statusaussage gemacht werden. In [ISO 10164-2] werden standardisierte Modelle bereitgestellt, deren Anwendbarkeit auch auf den Prozessor hin zu untersuchen sind. Es werden drei verschiedene Stati unterschieden:

- Operability
- Usage
- Administration

Der Operability-Status gibt Auskunft, ob eine Ressource installiert und arbeitsbereit ist. Dieses Modell ist bei Ein-Prozessor-Maschinen unmittelbar ohne Modellierung implementiert, da der Agent keine Information zur Verfügung stellen kann, wenn sich die CPU im

Stadium *Disabled* befindet. Anders verhält es sich bei Multiprozessormaschinen, bei denen einzelne Prozessoren beispielsweise durch Verklemmung unbenutzbar sein können. Deshalb wird in die Management Information Base das Statusattribut aufgenommen, das den Zustand des Prozessors mitteilt. Der Usage-State unterscheidet die Zustände *Idle, Active und Busy* . Der Zustand *Idle* besagt, daß das MO nicht genutzt wird. *Busy* bedeutet, daß das MO an seiner Kapazitätsgrenze angelangt ist und keine weiteren Aufträge annehmen kann. *Active* ist der Zustand zwischen *Idle*und *Busy*. Trotz der Multiuser und Multitaskingfähigkeit von UNIX-Systemen, die einen quasiparallelen Ablauf von vielen Programmen ermöglichen, arbeitet auch in Multiprozessorsystemen der einzelne Prozessor zu jedem Zeitpunkt immer nur einen Prozeß ab. Deshalb muß per Definition ([ISO 10164-2]) das Usage-State-Diagramm auf die Zustände *Idle* (es läuft kein Prozeß auf dem Prozessor) und *Busy* (ein Prozeß wird abgearbeitet), beschränkt werden. Abbildung 4.3 zeigt das modifizierte Diagramm.

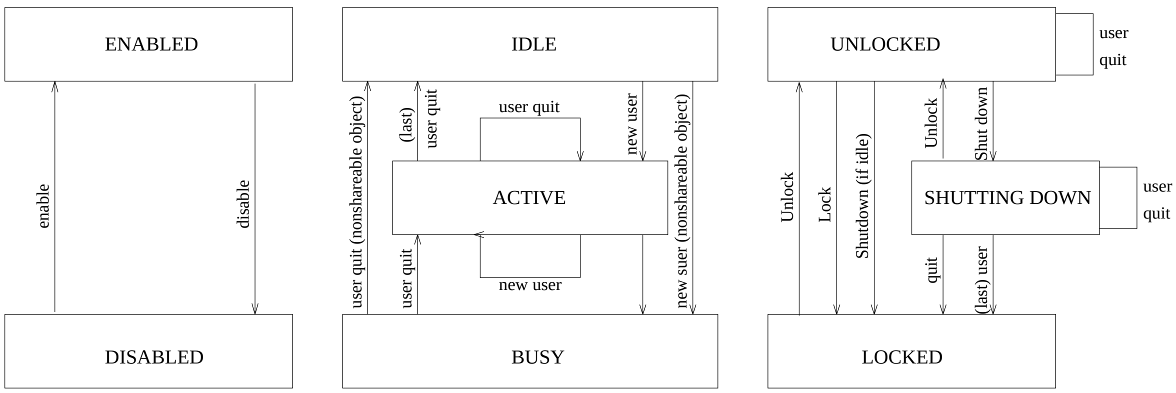

Abbildung 4.2: Operational, Usage und Administration State gemäß ISO/IEC 10164-2

Während diese Information in Single-CPU-Systemen keine Rolle spielt, sondern eher Information interessiert, wie z.B. das System ausgelastet ist, will man in Multiprozessorsytemen gerne wissen, ob einzelne Prozessoren unbenutzt sind und daher für Anwendungen allokiert werden können ([Bowen 80]). Daher wird der Operability-Status in die MIB mitaufgenommen. In Multiprozessorsystemen ist es für den Superuser typischerweise möglich einzelne Prozessoren für den Anwender zu sperren. Daher muß auch der Administration Status mitaufgenommen werden.

4.2.3 Leistungsbewertung eines Prozessors

Da der Prozessor zentrales Element eines Rechners ist, ist es natürlich auch für das Systemmanagement interessant Aussagen über seine Leistungsfähigkeit machen zu können. Die Kenntnis kann Entscheidungsgrundlage für die Einordnung eines Rechners im Netzwerk sein, sie gibt aber auch die Möglichkeit Zahlen wie zweifacher Load richtig einzustufen. Eine Workstation, die einen Loadfaktor von 2.0 aufweist, aber eine sehr leistungsfähige CPU hat natürlich noch mehr Reserven als eine Maschine mit selben Loadwert, aber schwächerer

CPU. Es existieren eine Vielzahl von Maßeinheiten, die alle versuchen eine Aussage über

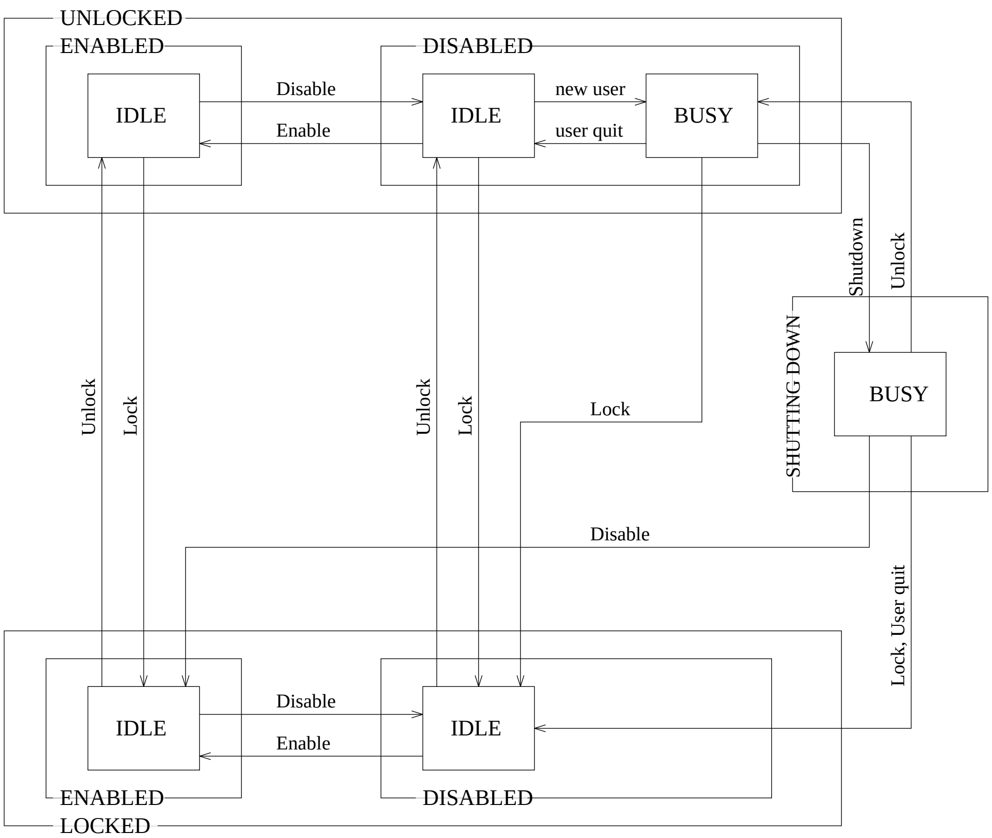

Abbildung 4.3: Kombiniertes Statusdiagramm

die Leistungsfähigkeit des Prozessors zu machen. Die bekanntesten Vertreter sind MIPS, MFLOPS, Dhrystone, Whetstone und SPECmarks. Abbildung 4.5 zeigt an welcher Stelle im Rechensystemen die verschiedenen Kennzahlen gewonnen werden. Alle diese Kennzahlen werden verwendet um verschiedene Rechensysteme zu vergleichen.

MIPS

Auf den ersten Blick erscheinen MIPS, die lediglich die Anzahl der ausgeführten Instruktionen pro Sekunde zählen, sehr geeignet, um verschiedene Prozessoren zu vergleichen.

MIPS sind, wie folgende Formel zeigt, einfach zu verstehen.

$$MIPS = \frac{InstructionCount}{ElapsedTime * 10^6}$$

MIPS sind von allen Hardwareherstellern für jede Maschine oder Prozessor leicht zu erhalten und wie man vermutet bedeuten größere Maschinen eine höhere MIPS-Zahl. Leider genügt diese Kennzahl, wie im folgenden Abschnitt gezeigt wird, wissenschaftlichen Ansprüchen nicht.

MIPS sind von Instruktionsset eines Prozessors abhängig. Die Formel für MIPS läßt sich folgendermaßen umformulieren:

$$MIPS = \frac{InstructionCount}{ElapsedTime * 10^6} = \frac{ClockRate}{CPI * 10^6}$$

Hier wird offensichtlich, daß die MIPS-Zahl von der Größe CPI abhängt. CPI besagt, wie viele Taktzyklen ein Befehl im Durchschnitt zur Abarbeitung benötigt. Der CPI-Wert in der MIPS-Formel ist der Durchschnitt über alle ausgeführten Befehle eines Testprogramms. Unterschiedliche Befehle benötigen unterschiedlich viele Taktzyklen für Ihre Ausführung.

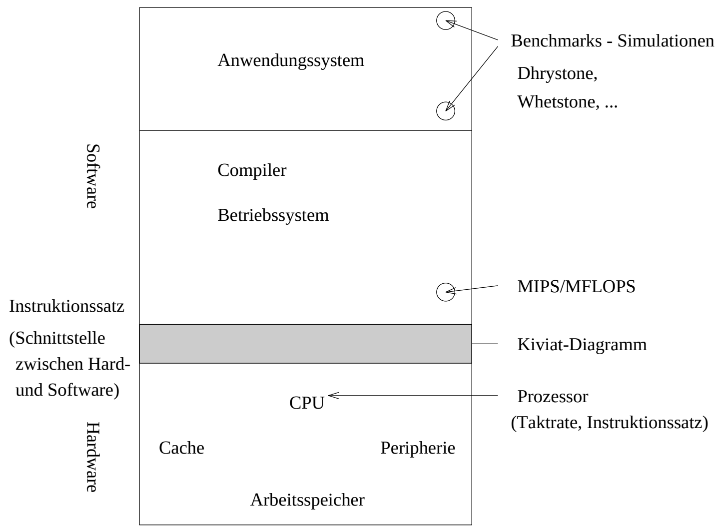

Abbildung 4.4: Vertikale Unterteilung des Systems zur Leistungsmessung

Beispiel: Es soll die Leistungsfähigkeit eines RISC-Prozessors, getaktet mit 50 MHz, festgestellt und dazu die MIPS-Zahl ermittelt werden. Es stehen zwei einfachste Testprogramme zur Verfügung. Das erste verwende 20% ALU-Operationen, sowie 40% Load- und 40% Store-Operationen. Seien nun zwei Testprogramme für die Ermittlung der MIPS auf dieser Maschine vorhanden. Das zweite Testprogramm verwende 70% ALU-Operationen und je 15% Load und Store-Operationen. Bei einer angenommenen CPI von 1 für ALU-Operationen und einer CPI von 3 für Load bzw. Store Operationen ergibt sich für Testprogramm 1 eine CPI von 2.6 und für Testprogramm 2 eine CPI von 1.6.

Eingesetzt in obige Formel ergibt sich damit unter Verwendung von Testprogramm 1 eine MIPS-Zahl von ca 19 MIPS und für Testprogramm 2 eine MIPS-Zahl von ca 31 MIPS! Diese stark voneinander abweichenden MIPS-Zahlen entstehen durch Verwendung unterschiedlicher Befehlsmixe. Um dies auszuschließen existiert der GIBSON-Befehlsmix. Dieser definiert das Verhältnis der unterschiedlichen Befehle in einem Testprogramm. Prozessoren, die die gleichen Instruction Sets besitzen, werden damit nun vergleichbar.

Wie in der Einführung zu diesem Kapitel bereits erwähnt, ist es die Eigenschaft von CISC-Prozessoren sehr mächtige Befehlsätze implementiert zu haben. Dadurch ist es möglich, daß für das selbe Resultat ein CISC, aber drei RISC-Maschinenbefehle nötig sind. Angenommen, beide Prozessoren benötigen dafür die gleiche Zeit, dann hätte nach der obigen Formel der RISC-Prozessor die dreifache MIPS-Zahl!?

Zusammenfassend läßt sich sagen, daß MIPS[2] ein denkbar schlecht geeignetes Mittel zum Leistungsvergleich sind, da ein Industriestandard fehlt.

Relative MIPS

Ein weiterer Versuch die MIPS zum Standard zu erheben, war die Definition der relativen MIPS. Dabei wird das Testprogramm auf einer VAX 11/780 gestartet und die Zeit gemessen. Die VAX hat per Definition der relativen MIPS exakt 1 MIPS. Vergleicht man nun die Zeit auf der eigenen Maschine mit der Ausführungszeit auf der VAX, so ist der Faktor, um den das Programm schneller läuft, die relativen MIPS. Allerdings ist auch bei diesem Verfahren die Gleichheit des Betriebssystems, des Compilers, sowie eventueller Optimierungen nicht gewährleistet, so daß auch diese Kennzahl nur eine schlechte Vergleichsmöglichkeit bietet. Die gleiche Problematik existiert auch für die anderen Benchmarks.

SPECmarks

Seit einigen Jahren gibt es nun eine Organisation, die Systems Perfomance Evaluation Cooperative (SPEC), die eine "non-profit organization of companies that develops standard benchmarks for measuring computer systems" ([Past91]) ist.

Diese Organisation hat es sich zum Ziel gemacht endlich vergleichbare Performancemessungen zur Verfügung zu stellen. Durch die Art der Messungen ist dies gewährleistet. Zudem finden sich die wichtigsten Hardwarehersteller im UNIX-Endmarkt wie Digital Equipment Corporation, Hewlett-Packard Company, Intel, Intergraph Corporation, IBM, MIPS Computer Systems, Motorola Microcomputer Division, Siemens-Nixdorf, Silicon Graphics, SUN Microsystems und Unisys umfaßt in der Mitgliederliste.

SPEC stellt die folgenden Anforderungen an Benchmarks

- der Benchmark muß für alle Mitglieder im Source-Code verfügbar sein.
- der Benchmark muß ohne viel Aufwand zwischen verschiedenen Hardwareplattformen portierbar sein.
- der Benchmark sollte einen Benutzer bei der Arbeit simulieren.
- der Benchmark muß Ergebnisse seiner Arbeit liefern, die leicht auf Ihre Korrektheit hin zu überprüfen sind und damit die richtige Abarbeitung des Bechmarks gewährleisten.

[2]MIPS werden manchmal auch scherzhaft als "Marketed Instruction per Second" bezeichnet

Die aktuelle Benchmark-Sammlung ist momentan in Integerintensive und Floating-Point-intensive Programme aufgeteilt. Es wird auf eine ausführliche Beschreibung der verwendeten Programme verzichtet, da sie der Diplomarbeit nicht weiter dienlich wäre. Die Integer Sammlung umfaßt espresso (Logic Design), li (Interpreter), eqntott(Logic Design), compress (Daten Kompression), sc (Spreadsheet), gcc (Compiler). Die Floating Point Sammlung beinhaltet 14 Programme, davon 12 in Fortran geschrieben und 2 in C. Im einzelnen sind dies spice2g6, doduc, mdljdp2, wave5, tomcatv, ora, alvinn, ear, mdljsp2, swm256, su2cor, hydro2d, nasa7, fpppp.

Mit den Benchmarks kann sowohl die Geschwindigkeit, als auch der Durchsatz einer Maschine gemessen werden. Die SPECRatio, mit ihr wird eine Aussage über die Geschwindigkeit der getesteten Maschine gemacht, ist das Ergebnis der Zeit, die ein bestimmter Benchmark aus der Sammlung für seine Abarbeitung benötigt hat, im Verhältnis zu der Zeit, die er auf der SPEC-Referenzmaschine, einer VAX 11/780, verbraucht hat.

$$SPECRatio(B) = \frac{SPECReferenceTime(B)}{ElapsedTime(B)}$$

Die veröffentlichte SPECint oder SPECfp ist das geometrische Mittel aller SPECRatios der entsprechenden Sammlung.

$$SPECmark = \sqrt[n]{\prod_{B \in SPEC} SPECRatio(B)}$$

Die Durchsatzkennzahl, als SPECrate bezeichnet, wird typischerweise für Multiprozessorsysteme verwendet. Sie besagt, um wieviele Einheiten ein Benchmark auf der Testmaschine öfter ausgeführt wird als auf einer VAX 11/780. Der dabei gewählte Zeitraum beträgt eine Woche. Damit gibt die SPECrate Auskunft über das Verhalten eines Systems, das mit vielen gleichartigen Jobs ausgelastet ist.

Zusätzlich zur exakten Definition der Benchmarks ist auch die Form und Inhalt des Testberichts genauestens vorgeschrieben. Wichtigstes Merkmal hierbei ist die Offenlegung der Testumgebung. So müssen in dem Bericht neben genauer Typbezeichnung der CPU, FPU, Cache-Größe auch Betriebssystemversion, Compiler, Filesystem, sowie der Background-Load und der Systemstatus (Multi/Single-User) mitgeteilt werden. Auf diese Art und Weise läßt sich die Leistungsfähigkeit einer Maschine erkennen.

Selbstverständlich müssen, um die Eignung eines Rechners für eine bestimmte Aufgabe festzustellen noch andere Parameter berücksichtigt werden (z.B. Auslastung): " users should compare the characteristics of their workload with that of the individual SPEC benchmarks and consider those benchmarks that best approximate their jobs. However, SPEC also recognizes the demand for aggregate result numbers and has defined the integer and floating-point averages"([Past91]).

4.2.4 Abschließende MIB-Betrachtung der Prozessorklasse

Im vorherigen Unterkapitel wurde erklärt, wieso SPECmarks eine hinreichende Aussage über die Leistungsfähigkeit eines Systems geben. SPECmarks sind zwischenzeitig so bekannt, daß die neuesten Ergebnisse beispielsweise über anonymous ftp (vgl. Anhang D.2.4) erhalten werden können. Diese Werte sollten auch von einem Systemmanagement zur Verfügung gestellt werden. Da Prozessortyp und Taktfrequenz vom System erfragt werden können und

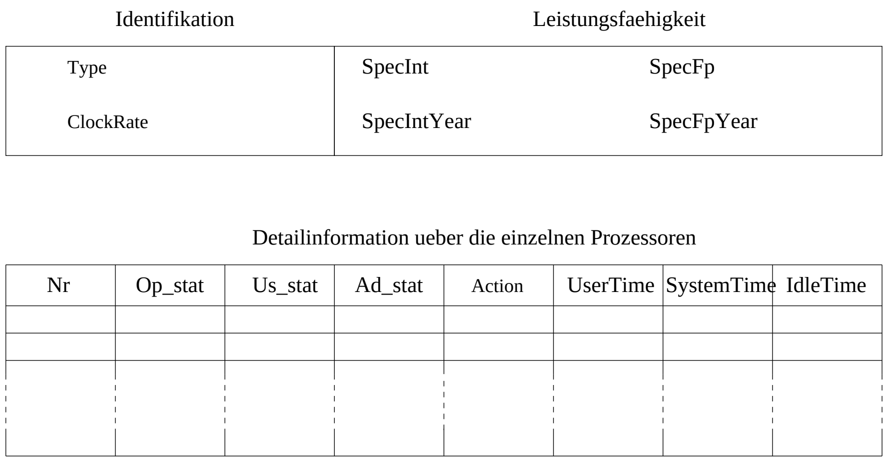

Abbildung 4.5: Prozessorklasse der UNIX-LRZ-MIB

auch für der MIB definiert worden sind, ist eine exakte Identifzierung des Prozessors möglich. Mittels dieser Identifizierung können bei geeigneter Implementierung die zwei Ergebnisse der SPECmarks, SPECint und SPECfloat zur Verfügung gestellt. Da es zwischenzeitlich mehrere SPECmark-Versionen gibt, die alle durch die Jahreszahl ihrer Entstehung unterschieden werden, sind noch zwei MIB-Variablen vorgesehen, die die Jahreszahl des Benchmarks enthalten. Weitere interessante Zahlen über die Prozessorauslastung werden ebenfalls für die MIB modelliert. Der Kernel von UNIX-Systemen besitzt Zähler, in denen er protokolliert, ob ein Prozeß den Prozessor im Systemmodus oder im Usermodus benutzt. Ebenfalls gezählt werden die Zeiteinheiten, in denen der Prozessor nicht genutzt wird[3]. Der Zähler des Usermodus wird immer dann hochgezählt wenn ein normaler Benutzerprozeß gerechnet wird. Bedient sich ein Benutzerprozeß Systemfunktionen, die der Kernel zur Verfügung stellt, so wird Kernel-Code ausgeführt und es wird der Zähler des Systemmodus hochgezählt. Werden Systemprozesse abgearbeitet, wird ebenfalls der Zähler des Systemmodus hochgezählt. Diese Werte sind für das Systemmanagement wichtig. Hat ein System beispielsweise schlechte Antwortszeiten, aber eine hohe Idlequote so kann ein find-Kommando das im Root-Verzeichnis gestartet wurde und in einem verteilten Filesystem nach einer Datei sucht[4] die Ursache dafür sein. Diese Zahlen unterstützen also das Systemmanagement bei der Lokalisation von Fehlerursachen.

[3]neben den beschriebenen Zählern existiert noch ein weiterer Zähler, der Nice genannt wird. Seine Bedeutung konnte nicht geklärt werden, da er weder in Handbüchern noch in Manualseiten erwähnt wird. Bei verschiedenen Lastversuchen auf HP und SUN Workstations blieb er immer null. Er wird deshalb auch in der MIB nicht berücksichtigt

[4]nfsd-Dämonen führen auf dem Server sehr viele I/O-Operationen aus

4.3 Speicher

Rechensysteme umfassen gewöhnlich mehrere, in ihrem Zugriff hierarchisch angeordnete Speichereinheiten. Abbildung 4.6 zeigt die vertikale Hierarchie der Speicherkomponenten. Dabei induziert die Pyramide das mengenmäßige Vorhandensein der einzelnen Speicherarten. Während die Kosten pro Byte von der Spitze zum Boden sinken, verlängern sich gleichzeitig die Zugriffszeiten. Eine zusätzliche Unterteilung läßt sich gem. [Schn91] in Primärspeicher, dies sind Register, Cache, Arbeitsspeicher, in Sekundärspeicher, das sind Plattenspeicher, und Tertiärspeicher, das sind Bandspeicher vornehmen.

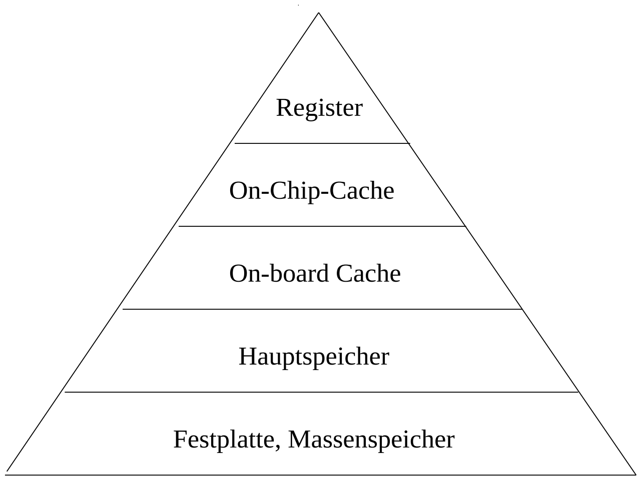

Abbildung 4.6: Vertikale Hierarchie der Speicherkomponenten

4.3.1 Primärspeicher

Die Primärspeicher weisen wiederum eine Hierarchie auf. An der Spitze stehen unmittelbar die Register, Speichereinheiten im Prozessor. Sie sind je nach Prozessorarchitektur und Typ in unterschiedlicher Anzahl und Bestimmung vorhanden. Registerinhalte werden während des Betriebs von UNIX-Systemen nur für das Debuggen von Programmen benötigt und sind deshalb für ein Systemmanagement bedeutungslos. Auch die Zahl oder Zweck der Register hat keinerlei systemmanagementrelevate Aussagekraft, so daß sie für die MIB nicht modelliert werden.

Cache

Die Technik eines schnellen Zwischenspeichers zwischen Prozessor und Hauptspeicher stammt ursprünglich von Großrechnern. Zwischenzeitlich ist der Cache auch bei kleinen Rechnern, wie PC und Workstations üblich. Er wurde durch die Takterhöhungen bei Prozessoren nötig. Erfolgte bei einem Prozessor, der mit 5 MHz getaktet war, ein Speicherzugriff, so verblieben 200 ns um das angeforderte Datum bereitzustellen, bevor der nächste Befehl abgearbeitet wurde. Wird ein Prozessor mit 50 MHz betrieben, so müßte der Arbeitsspeicher jeden Zugriff innerhalb 20 ns befriedigen, ohne daß der Prozessor Wait-States einlegen muß. Da die mittleren Zugriffszeiten des Hauptspeichers ca. 70 ns betragen, wird der Prozessor mit aktuell benötigten Programmteilen und Daten lokal vom aus schnellen S-RAM Bausteinen bestehenden Cache versorgt. Dabei unterscheidet man zeitliche Lokalität, Befehle oder Daten auf die vor kurzen zugegriffen wurde werden wieder benötigt (z.B Schleifendurchläufe), und räumliche Lokalität (z.B. Arrays). An dieser Stelle wird auch das Ziel eines hierarchischen Speicheraufbaus deutlich. Idealerweise liefert die nächstlangsamere Stufe die Daten so schnell, wie die schnellere Stufe, aber ist eben wesentlich billiger.

Allgemein kann man wegen des oben beschriebenen Verhaltens die Aussage treffen, daß je größer ein Cache ist, desto mehr Code und Daten können in ihm gehalten werden und desto größer ist die Wahrscheinlichkeit einen Treffer (Cache Hit) zu erzielen ([Coyw92]). Allerdings ist der reale Leistungsgewinn durch Verwendung eines Caches von sehr vielen Randbedingungen abhängig. Dazu gehören neben Betriebssystemen (vgl [Stil94]) auch das bereits erwähnte Lokalitätsverhalten. Deshalb werden Leistungsaussagen nicht für die MIB modelliert. Für die Mangement Information Base wird lediglich die Größe des First-Level-Cache, der Cache-Speicher im Prozessor, und die Größe des Second-Level-Cache aufgenommen. Da bei Hardwareinkopatibilitäten oftmals der Cache abgeschalten werden muß, wird der Administration State mit in die MIB aufgenommen.

Hauptspeicher

Der Hauptspeicher ist in der Speicherhierarchie das Bindeglied zu den Sekundärspeichern. Konnten vor einigen Jahren Prozessoren erst 64 KByte Speicher adressieren, so sind es heute bereits Terabytes. Übliche Hauptspeicherausbauten bei Workstation liegen heute zwischen 4 und 256 MByte Arbeitsspeicher.

Wurden früher, um Speichermangel zu umgehen, Overlay-Techniken verwendet, bei denen bestimmte Programmteile erst bei Bedarf geladen wurden, verwenden heute die meisten Betriebssysteme virtuellen Speicher. Der virtuelle Speicher ist für Anwender und Programm transparent und vergrößert den Abeitsspeicher.

Die einzige Information, die von einem physischen Hauptspeicher für das Systemmanagement wichtig ist, ist seine Größe. Sie zeigt die Eignung einer Maschine für speicherintensive Anwendungen.

Virtueller Speicher

Unter einem virtuellen Speicher versteht man die Abstraktion des für Prozesse verfügbaren adressierbaren Speichers vom tatsächlich vorhandenen Arbeitsspeicher. Dieser entsteht durch die Einbeziehung von Plattenspeicher. Der virtuelle Speicher wird ebenfalls für die MIB modelliert. Er steigert, natürlich auch in Abhängigkeit von der Geschwindigkeit des Hintergrundspeichers, die Leistungsfähigkeit des Systems. Interessant beim virtuellen Speicher ist seine Größe und seine Belegung.

4.3.2 Modellierung der Speicher für die MIB

Um die Anzahl der definierten Variablen gering zu halten, und da die vier beschriebenen Speicherarten sich ähnlich modellieren liesen, wird eine Storage-Tabelle in der MIB eingeführt. Die Tabelle enthält Angaben zu Typ (Cache, RAM und virtueller Speicher), eine textuelle Beschreibung, die Größe der kleinsten Speichereinheit, die Gesamtgröße, die benutzte Größe und eine Statusinformation. Man beachte, daß nicht alle Variablen von allen Speichern genutzt werden. So will man beim Systemmanagement üblicherweise nicht wissen,

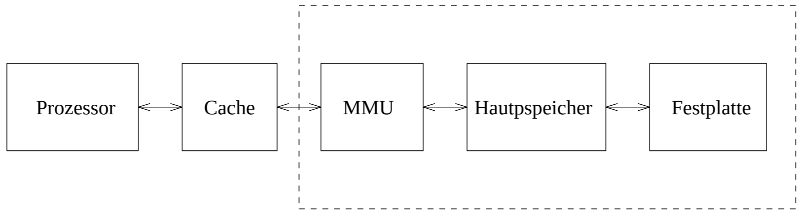

Abbildung 4.7: Der virtuelle Speicher

wieviel KByte vom Cache benutzt sind. Interessanter hingegen ist die Abbildung des Statusflags beim Cache. Ein Cache, der eine Größe von 0 KByte in der MIB eingetragen hat, existiert nicht. Ein Cache, der hingegen eine Größenangabe besitzt, aber im Zustand disabled ist, ist entweder defekt oder ausgeschalten. Diese Statusinformation genügt und erspart so die Aufnahme des Operability States.

4.4 Massenspeicher

Dienten bis vor einem Jahrzehnt Massenspeicher nur zur Speicherung von Daten und zum Datentransport, so werden sie heute im Rahmen von virtuellen Speichern in den Rechenprozess einbezogen. Deshalb ist auch Massenspeichern im Systemmanagement besondere Aufmerksamkeit zuzuwenden. In diesem Unterkapitel werde ich eine kurze Übersicht über die heute am häufigsten verwendeten Massenspeichen geben und sie für das Systemmanagement modellieren.

4.4.1 Übersicht über die gebräuchlichsten Massenspeicher

Diskettenlaufwerke sind die kleinsten Massenspeicher. Sie dienen dem Datenaustausch zwischen unvernetzten Rechnern sowie im eingeschränkten Maße dem Anlegen von Sicherungskopien. Übliche Speicherkapazitäten liegen heute zwischen 700 KByte und 3 MByte, jedoch existieren bereits Disketten, die 20 MByte unter Verwendung eines Lasers bei der Formatierung speichern können.

Festplatten sind das am weitest genutzte Medium. Durch große Speicherkapazität, die mehrere GByte betragen kann, und kurze Zugriffszeiten im Millisekundenbereich können sie auch als Erweiterung des Hauptspeichers verwendet werden. Veränderungen befassen sich mit den Abmessungen und dem Stromverbrauch der Platten, um den Einsatz in tragbaren Computern zu ermöglichen.

Optische Platten weisen sehr hohe Speicherkapazitäten auf (ca. 650 MByte), sind transportabel und sehr unempfindlich gegen äußere Einflüsse. Momentan werden drei verschiedene Arten unterschieden. Die CD-ROM kann nur gelesen werden. Die WORM (Write Once Read Many) kann genau einmal beschrieben werden. Die "Erasable Optical Disk" kann (fast) beliebig oft beschrieben werden. Die verwendete Technik basiert auf einer Magnetisierung der Platte unter zusätzlicher starker Erwärmung. Jedoch schwächt sich der sogenannte Kerr-Effekt mit der Zeit ab.

Bandlaufwerke dienen der Datensicherung und dem Datentransport. Prinzipiell unterscheidet man Start-Stop-Bänder, die während des Lesens und Schreibens auch angehalten werden können und deshalb auch als Archivierungsmedium bentuzt werden, und Streamer-Bänder, die von Anfang bis Ende durchlaufen.

4.4.2 Modellierung der Massenspeicher für das Systemmanagement

Prinzipiell treten fast alle oben beschriebenen Speicherarten in UNIX-Systemen sowohl als logischer als auch als physischer Speicher auf. Eine Ausnahme bilden die Bandlaufwerke, die kein Filesystem haben und deshalb vom Betriebssystem nicht auf Anforderung von Applikationen blockweise vergeben werden. Alle beschriebenen Massenspeicher haben, falls sie auf dem zu managenden System esitieren, einen Eintrag in der Device-Tabelle (vgl. Kapitel 4.6. Zusätzlich werden die Plattenlaufwerke verfeinert und in einer eigenen Tabelle eingetragen. Anders als in vielen MIB-Veröffentlichungen werden im diskTable lediglich Festplatten und optische Platten aufgenommen. Disketten beispielsweise aufzunehmen macht wenig Sinn, da sie im allgemeinen nur kurz zum Datenaustausch in das Laufwerk gelegt werden und somit kein MO im herkömlichen Sinn ist.

Standardmäsig wird in die Disk-Tabelle der Zugriff (nur lesen oder auch schreiben), der Typ, die Information ob es sich um ein Wechselmedium handelt und die Größe der Platte aufgenommen. Zudem interessieren im Systemmanagement (vgl. Kapitel 3.2) die Beanspuchung der Laufwerke. Deshalb werden noch die Werte *diskrs* (disk reads per second), *diskws* (disk writes per second) und *diskpu* (percentage disk utilization). Schließlich wird noch der Operability State für die Platten modelliert, da beispielsweise Platten durch Defekt oder Wechselplatten den Status disabled annehmen können.

4.5 Bussysteme

Um die verschiedene Komponenten eines Rechners, wie CPU, FPU, Caches und Speicher miteinander zu verbinden, werden Busse verwendet. Ein Bus ist ein Bündel von Leitungen, von dem Stichleitungen - die auch steckbar sein können - zu den einzelnen Komponenten abgehen. Damit werden für einen Bus pro Signal eine Verbindungsleitung und k Stichleitungen benötigt [Muel91]. Für Mehrprozessorsysteme gibt es spezielle Topologien, beispielsweise Matrixanordnung oder Hypercube.

4.5.1 Struktur und Hierarchie

Bei einem Bus sind alle Komponenten durch Signalleitungen verbunden und haben eine eindeutige Adresse. Jede Komponente empfängt alle Informationen, aber zu einer bestimmten Zeit darf nur eine Komponente senden. Verwendung von Standardbussen ermöglicht die Zusammenstellung maßgeschneiderter, preiswerter Systeme. In Rechnersystemen gibt es nach [faer87] verschiedene Klassen von Bussystemen, die auch eine Hierarchie induzieren.

- Prozessorbus, zuständig für die Verbindung zwischen Registern und ALU. Meistens der schnellste Bus.
- On-Board-Bus, verbindet beispielsweise Hauptspeicher und CPU. Bei hochwertigen Systemen oftmals ein breiter und schneller Bus.
- Backplane-Bus, auf ihm stecken verschiedene Karten, oftmals Controller oder Meßmodule.
- Peripheriebus, zuständig für den Anschluß von Festplatten, Floppy-Disk-Laufwerken und Bändern.

Nicht immer sind alle Buskomponenten in einem Rechnersystem vorhanden oder unterscheidbar. So verzichtet beispielsweise die SUN SPARC SLC vollkommen auf einen Backplane-Bus. Alle Komponenten und Schnittstellen sind auf der Hauptplatine untergebracht. Der SUN S-BUS steuert sowohl Speicher als auch Peripherie an.

4.5.2 Modellierung des Busses

Ein Bus wird durch die Struktur zu einer knappen Systemressource. So ist es heute unverständlich, wie beispielsweise Personal Computer mit der 64bit-CPU Intel-Pentium angeboten werden, das Board aber für Speicher- und Peripheriezugriffe lediglich einen 16-bit breiten, mit 7 MHz getakteten AT-Bus zur Verfügung stellt. Wie in Kapitel 4.5.1 beschrieben, werden heute häufig verschiedene Bussysteme in einem einzigen Rechner zur Verfügung gestellt. Neben durchdachtem Design, wie die Precision Architecture von Hewlett-Packard, ist oftmals auch der stufenweise Umstieg des Benutzers - neues Board, alte Karten - auf ein neues System ein Verkaufsargument. In einem allgemeinen Systemmanagement würde es sicherlich zu weit führen, Perfomancemessungen oder Funktionsüberprüfungen zur Verfügung zu stellen. Aber hier gilt ebenso wie für den Prozessor, daß Kentnisse über Art und Zusammensetzung der Busse, sowie über angeschlossene Peripherie, grobe Aussagen über die Leistungsfähigkeit und Eignung eines Systems für bestimmte Anwendungen ermöglichen. Die verwendeten Bussysteme können vom System leicht erfragt, und vom Agenten zur Verfügung gestellt werden. Busse werden ebenfalls in die Device-Tabelle aufgenommen.

4.6 Anordnung der modellierten Hardware innerhalb der MIB

In diesem Kapitel wurden die wichtigsten Hardwarebestandteile eines UNIX-Endsystems modelliert. Dies waren Prozessor, Arbeits- und Massenspeicher. Die einzelnen MIB-Variablen und Tabellen werden der Device-Gruppe zugeordnet. Zusätzlich wird nach dem Vorbild der *Horst Ressources MIB* [RFC 1514] eine Device Tabelle eingeführt. Diese Tabelle enthält

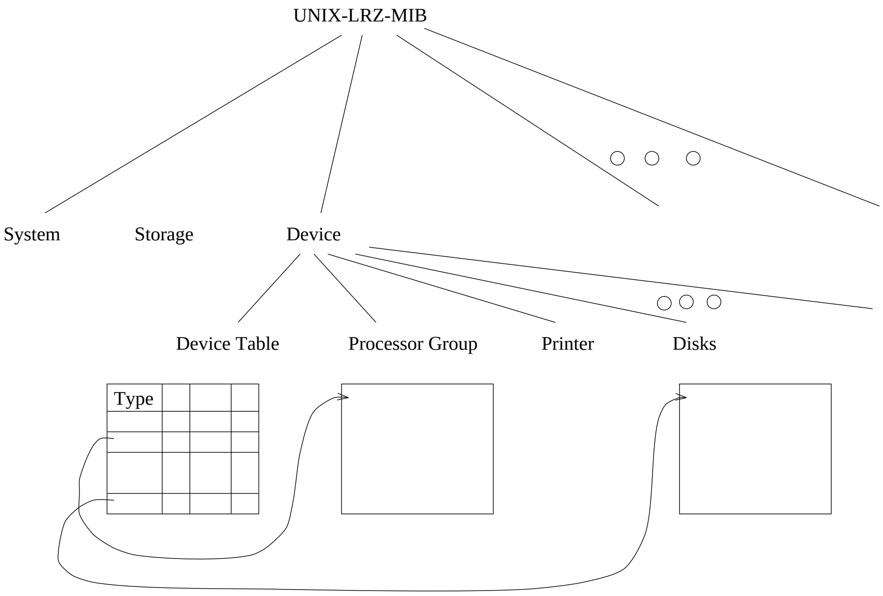

Abbildung 4.8: Die Device Tabelle in der MIB

eine Auflistung aller Devices des Systems. Dazu gehören eine Typkennung[5], eine textuelle Beschreibung des Devices und eine Statusangabe, Da in diesem Tabelle sehr unterschiedliche Device-Arten enthalten sein können, wird der Status auf 5 verschiedene Zustände beschränkt:

- unknown (der Status kann nicht abgefragt werden)
- running (Device läuft ohne Fehler)
- warning (ein Fehler ist aufgetreten oder bald zu erwarten)
- testing (Device wird getestet oder führt Selbstest durch)
- down (Device nicht einsatzbereit)

Es ist offensichtlich, daß die in dieser Tabelle definierte Managementinformtion nicht für ein vollständiges Systemmanagement hinreichend ist, aber da im Rahmen der Diplomarbeit nicht alle Devices modelliert werden können, bietet diese Tablle zum einen einen guten Überblick über die verfügbaren Devices auf dem System und zugleich die Schnittstelle für zukünftige Erweiterungen.

[5]eine vollständige Auflistung aller definierten Devicetypen findet sich in Anhang D

4.7 Modellierung eines Peripheriegerätes: Der Drucker

Stellvertretend für die vielen existierenden Peripheriegeräte, die in dieser Diplomarbeit nicht berücksichtigt werden können, wird in diesem Kapitel der Drucker für das Systemmanagement modelliert. Viele der definierten Attribute und Zustände werden sich auf andere Geräte übertragen lassen. Die Systemverwaltertätigkeit bei Druckern gliedert sich in:

- Installation von Druckern
- Verwaltung von Druckern
- Problembeseitigung bei Druckern

4.7.1 Installation von Druckern

Zur Installation eines Druckers muß dieser an einem Rechner angeschlossen werden. In einer verteilten Umgebung bezeichnet man den Rechner dann auch als Print Server, wenn er den Druckdienst mehreren anderen Rechnern zugänglich macht. Weitere Installationstätigkeiten umfassen die Einstellung der Hardware, die Baud-Rate und den Port über den er angesteuert wird. Bei der Konfiguration muß ein eindeutiger Name für den Drucker innerhalb des Rechnerverbundes, der auf ihn zugreift, vergeben werden. Die Systemverwaltung wird erleichtert, wenn die Namen aller Drucker nach einer einheitlichen Konvention vergeben werden. Beispielsweise könnte der Druckertyp immer an erster Stelle im Namen auftauchen (z.B. *psghoethe* für einen postscriptfähigen Drucker).

4.7.2 Verwaltung von Druckern

Die Verwaltung von Druckern umfaßt folgende Aufgaben:

- Informationserhalt über Drucker und Druckaufträge,
- Verwalten von Druckaufträgen,
- Drucker für Benutzer verfügbar machen und eventuelle Probleme beseitigen,
- Starten und Anhalten des Druckschedulers,
- Löschen von Loggfiles.

4.7.3 Modellierung für die MIB

Für die Managementinformationsbasis wird eine kombinierte Tabelle aus Drucker und zugehöriger Warteschlange modelliert.

Die Zustände des Druckers werden wiederum gemäß [ISO 10164-2] modelliert. Der Operational State kann wieder die Zustände *enabled* und *disabled* annehmen. Der Usage State pendelt zwischen *idle* und *busy* . Obwohl man in Abhängigkeit der Größe der Warteschlange den Zustand *active* vorsehen könnte[6], wird darauf verzichtet, da man hier bereits die Managementpolitik festlegen würde. Dies ist jedoch nach [DSIS 93] unzulässig und soll vermieden werden. Erst der Systemverwalter soll den Druckauftragseingang in Abhängigkeit der Anforderungen seiner Umgebung beispielsweise durch eine Schwellwertüberwachung steuern.

Für den Drucker wird diesmal zusätzlich zum Administration State (*unlocked* , *locked* , *shuttingdown*) der Availability State mitaufgenommen. Der Availability Status beschreibt den Verfügbarkeitszustand. Ein Drucker kann 5 verschiedene Zustände annehmen. Der Zustand *inTest* sagt aus, daß der Drucker eine Testroutine abarbeitet. *Failed* weist auf einen internen Fehler hin. Hier genügt es oftmals schon den Drucker zurückzusetzen (Warmstart). Der Availability Status *powerOff* bemängelt die fehlende Stromversorgung, während *offLine* auf den nicht empfangsbereiten Zustand des Druckers hinweist. Der letzte für den Drucker modellierte Availability State ist *notInstalled* und bedeutet, daß der Drucker keine physische Verbindung zum Druckerserver hat.

Der modellierte Fehlerstatus ermöglicht, falls diese Werte vom Drucker unterstützt werden, die Unterscheidung zwischen einem *other* Fehler[7], einem Papierstau (*jammed*), einer geöffneten Wartungsklappe (*doorOpen*), dem Fehler *noPaper* , dem Fehler *noToner* und den beiden Warnungen *lowPaper* und *lowToner* .

Wichtig ist, daß diese Fehlermeldungen unmittelbar den Operational State beeinflussen. Außer den beiden Warnungen *lowPaper* und *noToner* führen alle anderen Fehlermeldungen unmittelbar zu dem Operational State *disabled* .

Ebenfalls beeinflußt wird der Status des entsprechenden Eintrags in der Device Tabelle. Die Meldung *other* führt zum Status *unknown* , die zwei Warnungen zum Zustand *warning* und alle anderen Fehlermeldungen zu *down* .

Als zusätzliche Information wird die Variable *printerLocation* zur Verfügung gestellt, in die der Systemadministrator den physischen Aufstellungsort, und wenn der Platz ausreicht, auch noch einen zuständigen Ansprechpartner aufnehmen kann.

Für die Administration der Druckerwarteschlange wird sehr detaillierte Information modelliert. Neben der Login-Kennung des jeweiligen Jobeigentümers[8] werden die Größe des Auftrags, seine Priorität, seine Uhrzeit und Datum des Eintritts in die Schlange, und falls er schon gedruckt wird, die bereits verarbeitete Größe zur Verfügung gestellt. Obwohl nicht alle UNIX-Systeme prioritätsgesteuerte Druckauftragsverwaltungen zur Verfügung stellen wird diese Eigenschaft modelliert, da sie für den Systemverwalter eine sehr bequeme Möglichkeit darstellt die Reihenfolge der Aufträge zu ändern, ohne manuell Druckaufträge umsortieren zu müssen. Dem Verwalter werden fünf verschiedene Managementaktionen bereitgestellt. Er kann Druckschlangen für die Aufnahme neuer Jobs sperren (*reject*) und entsperren (*accept*), er kann einzelne Druckaufträge abbrechen bzw. löschen (*cancel*) und er kann einen bereits in Arbeit befindlichen Druckauftrag anhalten (*hold*) und danach fortsetzen (*resume*). Diese Aktionen beeinflussen unmittelbar den Status der Druckschlange. Sie bewegt sich zwischen

[6] weniger als 10 Druckaufträge bedeuten *active* , mehr bedeuten *busy*

[7] diese Fehlermeldung wird vom Drucker nicht unterstützt

[8] in Verbindung mit der in Kapitel 7.1 modellierten Benutzertabelle ist die Abbildung auf die UID sichergestellt

enabled , *disabled* , *waiting* und *running* . Diese Statusdefinition ist unmittelbar aus dem Operational State mit den Zuständen *enabled* und *disabled* , sowie dem Usage State mit den Zuständen *idle* und *busy* abgeleitet. Die Zusammenführung der beiden Zustandsarten ist zulässig, da jeder Zustand nur alleine auftreten kann.

Kapitel 5

Management von UNIX-Prozessen

Ein Prozeß ist ein in Ausführung befindliches Programm, daß vom Betriebssystem gesteuert wird. Ein Prozeß benötigt zur Ausführung verschiedene Betriebsmittel. Dies sind insbesondere Arbeitsspeicher und Rechenzeit. Da diese beiden Ressourcen in jedem System beschränkt sind und im allgemeinen in einer Rechenanlage mehrere Prozesse konkurrieren, muß das Betriebssystem mittels Zuteilungsstrategien einen möglichst fairen Ablauf gewährleisten.

Ein Systemmanagement stellt idealerweise Mechanismen zur Verfügung, die es neben der bloßen Überwachung der Prozesse auch ermöglichen, den Ressourcenverbrauchs der Prozesse zu erweitern oder zu minimieren. Die Modellierung des UNIX-Prozesses für das Systemmanagement erfolgt in diesem Kapitel.

5.1 Der UNIX-Prozeß

In diesem Unterkapitel werden kurz die wichtigsten Identifikationsmerkmale eines UNIX-Prozesses beschrieben. Weitere Informationen finden sich in [Glas93], [Dree88] und [Leff89].

Beim Start von UNIX existiert im System genau ein sichtbarer Prozeß, der "init" genannt wird (vgl. Abbildung 5.1). Neue Prozesse werden nicht durch die Anweisung "create a new process to run program x" gestartet, sondern es muß ein bestehender Prozeß dupliziert werden. Dadurch hat UNIX einen hierarchischen Prozeßaufbau und "init" ist der Vorfahre aller anderen Prozesse. Jedem Prozeß wird eine im System eindeutige Prozeßidentifikationsnummer (PID) zugeordnet. Sie ist der Referenzmechanismus für Signale oder Interprozeßkommunikation. Weitere Identifkationsnummern sind die PID des Vaters (PPID) und die Gruppen-PID (GPID). Sie ermöglicht es beispielsweise der Managementanwendung ein Broadcast-Signal an Prozesse zu senden, die über eine Pipe[1] verbunden sind, da diese Prozesse die gleiche GPID haben. Ebenfalls für das Prozeßmanagement werden die UID, das ist die systemweit eindeutige Benutzeridentifikationsnummer, und die GID[2], die Gruppenidentifikationsnummer, des Prozeßeigentümers zur Verfügung gestellt. Sie sind nötig um Prozesse

[1]Eine Pipe(line) lenkt die Standardausgabe eines Prozesses zur Standardeingabe eines Folgeprozesses

[2]Jeder Benutzer ist mindestens einer Gruppe zugeordnet und hat damit bestimmte Rechte beim Dateizugriff. Weitere Informationen finden sich in Kapitel 7.2.

einem Benutzer des Systems zuordnen zu können oder das Verhalten einzelner Benutzers zu überwachen. Diese Werte werden durch einfach lesbare Attribute modelliert.

5.2 Der Prozeßkontext

Jeder Prozeß im System ist durch seinen Prozeßkontext eindeutig definiert. Dazu gehören neben den beschreibenden Attributen aus 5.1 auch der Programmadreßraum, die geöffneten Dateien, die Betriebsmittelkonten, Arbeitszustand und Rechte.

5.2.1 Status eines Prozesses

Jeder UNIX-Prozeß nimmt während seines Lebenszyklus verschiedene Zustände ein, die unterschiedliche Auswirkungen auf das Rechensystem haben. Die beschriebenen Zustände, oftmals auch als Status bezeichnet, sind in allen UNIX-Systemen gleich, nur die Bezeichnungen können sich unterscheiden.

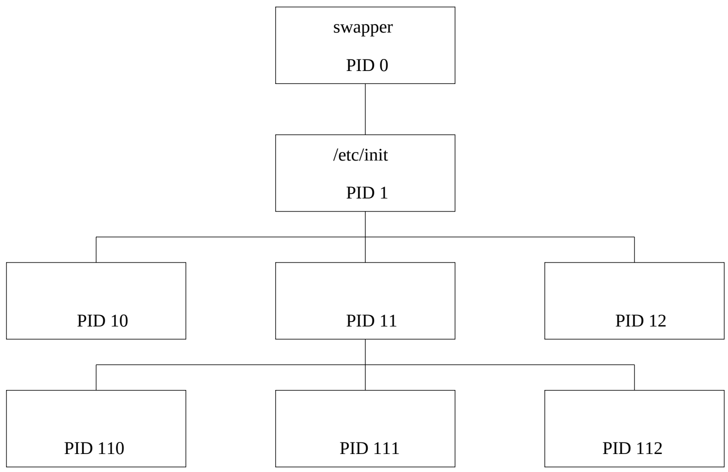

Abbildung 5.1: Prozeßhierarchie

Abbildung 5.2 zeigt ein verallgemeinertes Zustandsübergangsdiagramm, das von speziellen UNIX-Begriffen abstrahiert wurde. Der Trivialzustand "nicht existent" ist für ein Systemmanagement nur insofern entscheidend, als daß die Differenz zwischen maximal möglichen Prozessen auf einem System zu den geladenen Prozessen einen Anhaltspunkt über die Systemreserven geben kann (vgl Kapitel 5.2.3). Allerdings kann die Erzeugung neuer Prozesse vom Betriebssystem bereits unterhalb der Maximalzahl verweigert werden, wenn beispielsweise die übrigen Prozesse zuviele Dateien geöffnet haben oder nicht mehr genügend Arbeitsspeicher zur Verfügung steht. Der Zustand "wartend" wird von UNIX-Systemen differenziert. Wird ein Prozeß durch einen fork oder execve-Aufruf erzeugt, so befindet er sich in einem

Anfangswartezustand (SIDL)[3] bis er genügend Systemressourcen allokiert hat. Ebenfalls in den Wartezustand gerät der Prozeß wenn er auf Beendigung eines Ereignisses, beispielsweise den Abschluß einer I/O-Operation, wartet (SSLEEP) oder durch ein Signal angehalten wurde (SSTOP). Verfügt ein Prozeß über alle benötigten Betriebsmittel und existiert keine Bedingung, daß er im Wartezustand bleiben muß, so geht er automatisch in den Zustand rechenbereit (SRUN) über und bewirbt sich mit den anderen rechenbereiten Prozessen um den Rechenkern (vgl. Kapitel 5.2.2). Bei Zuteilung wird sein Zustand als rechnend (RUN) markiert. Den rechnenden Zustand verläßt der Prozeß entweder, weil ihm der Rechenkern vom Betriebssystem entzogen wird, er ein Signal erhalten hat oder auf ein Ereignis warten muß. Der Prozeßzustand wird in diesem Kapitel genau beschrieben, da er für das Systemmanagement bedeutend ist. Die Zustandsinformation wird in dem lesbaren Attribut processState modelliert, das für den jeweiligen Prozeß die oben beschriebenen Zustände zur Verfügung stellt.

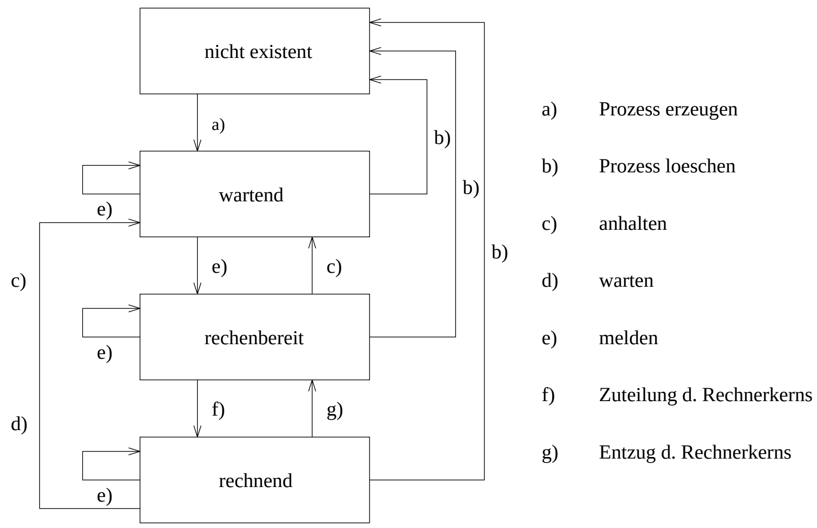

Abbildung 5.2: Prozeßzustände im UNIX-System

Ein weiterer, besonderer, Zustand ist der Zombie-Status (SZOMB), der ein Zwischenstadium bei der Prozeßbeendigung darstellt. Er entsteht, wenn der Vaterprozeß den Return Code des abgearbeiteten Sohnes nicht akzeptiert. Dies passiert, wenn der Vaterprozeß niemals einen wait()-Aufruf[4] ausführt und der Sohn verbleibt als Zombie im System. Obwohl er weder Code-, Data- oder Stacksegment belegt und deshalb wenig Systemressourcen verbraucht, belegt er Platz in der Prozeßtabelle des Systems und erniedrigt damit die Zahl der noch startbaren Prozesse. Deshalb ist es vorteilhaft sie zu entfernen.

Weitere Zustandsinformation findet sich in dem Attribut processFlags. Obwohl UNIX-Systeme oft mehr als zwanzig Flags zur Verfügung stellen, werden hier nur die wichtigsten für

[3] Die Abkürzungen in Klammern sind BSD-UNIX spezifisch, finden sich aber unter anderen Namen auf allen anderen UNIX-Systemen

[4] Dieser Aufruf dient speziell dem Erhalt einer Beendigungsmeldung des Sohnprozesses

das Systemmanagement modelliert. So muß eine Managemenanwendung erkennen können, ob es sich um einen Benutzer- oder Systemprozeß (SSYS) handelt (vgl. Kapitel. 5.3), ob sich ein Prozeß im Hauptspeicher (SLOAD) befindet oder ausgelagert (SSWAP) ist (vgl. Kapitel 6.4). Wenn sich ein Prozeß trotz eines SSLEEP-Zustands noch im Hauptspeicher befindet so kann dies auch daran liegen, daß der Benutzer das Swappen unterbunden hat (SULOCK). Ebenfalls in die MIB wird das Flag SPHIO aufgenommen, das anzeigt, daß der Prozeß Plattenoperationen betreibt. Mit den modellierten Flags ist es möglich Systemprozesse von Benutzerprozessen zu unterscheiden und zudem Prozesse zu erkennen die das System durch viele I/O-Operationen belasten oder Arbeitsspeicher unnötig belegen.

5.2.2 Priorität eines Prozesses

Für die weitere Modellierung der MIB muß kurz die Prioritätsberechnung von UNIX sowie die Möglichkeiten des Systemverwalters diese zu beeinflussen, beschrieben werden.

Ein Teil des Kernels, genannt Scheduler, übernimmt die Rechnerkernzuteilung an die einzelnen Prozesse. Dabei verwenden die UNIX-Systeme ein prioritätsgesteuertes Round-Robin-Verfahren. Jeder Prozeß verfügt über einen Prioritätswert zwischen -20 und +19. Prozesse die den Rechenkern zugeteilt bekommen haben, sinken sehr schnell in ihrer Priorität, während Prozesse, die ihre Zeitscheibe nicht nutzen, ihre Priorität steigern. Dadurch können auch in ausgelasteten Systemen interaktive Programme wie beispielsweise Editoren gute Antwortszeiten haben, da sie den Rechenkern wenig beanspruchen. Zusatzlich zum Prioritätswert kennt UNIX den Nice-Wert, der die Grundpriorität darstellt. Dieser kann vom Prozeßeigentümer und Systemverwalter geändert werden um auf die Prozeßverwaltung Einfluß zu nehmen. Der Nice-Wert wird unmittelbar zum Prioritätswert hinzugezählt. So werden für jeden Prozeß zwei Attribute hinzugefügt. Das Atrribut processPri, das den aktuellen Prioritätswert enthält und das Nice-Attribut, welches beschreibbar ist und so dem Systemverwalter und/oder der Managementanwendung die Möglichkeit gibt die Grundpriorität des Prozesses zu ändern.

5.2.3 Ressourcenverbrauch eines Prozesses

Wie eingangs erwähnt, benötigen Prozesse hauptsächlich Speicher und Rechenzeit. Da UNIX-Systeme sowohl mehrere Benutzer gleichzeitig bedienen und viele Prozesse quasiparallel abarbeiten können, kann es schnell zu einer Überlastung des Systems kommen. Zwar werden maximale Anzahl der gleichzeitig unterstützten User-Sessions, systemweite Prozeßanzahl und Prozeßanzahl pro Benutzer in Konfigurationfiles festgelegt, diese sind jedoch meistens sehr großzügig bemessen. Um Überlastung eines Systems und damit beispielswiese unakzeptable Antwortszeiten zu vermeiden, müssen verfeinerte Mechanismen zur Verfügung gestellt werden. Sie müssen es ermöglichen, Prozesse, die das System zu stark belasten, zurückzudrängen. In diesem Unterkapitel werden die die dafür nötigen Managementinformationen modelliert.

Das Attribut processSize nennt die Größe eines Prozesses in Kilobyte. Es ermöglicht Prozesse zu lokaliseren, die durch ihren Speicherbedarf Swapaktivitäten hervorrufen und damit unter Umständen trotz geringer CPU-Beanspruchung lange Antwortzeiten verursachen. Eine Unterscheidung in Text-, Data- und Stacksegment erfolgt hierbei nicht, da für die Systembelastung die Gesamtgröße entscheidend ist. Das Attribut processCPUtime nennt die gesamte verbrauchte Rechenzeit des Prozesses seit seinem Start.

5.3 Management mittels der MIB

Für das Prozeßmanagement werden verschiedene Informationsstufen bereitgestellt. Neben der Tabelle, die alle Prozesse auf dem System enthält, geben die Variablen maximale Anzahl gleichzeitig geladener Prozesse auf dem System, Anzahl gerade geladener Prozesse und Speicherlimit eines einzelnen Prozesses einen schnellen Überblick über das zu managende UNIX-System. Gleichzeitig werden zur Überwachung von Prozeßgrößen und Rechenzeit

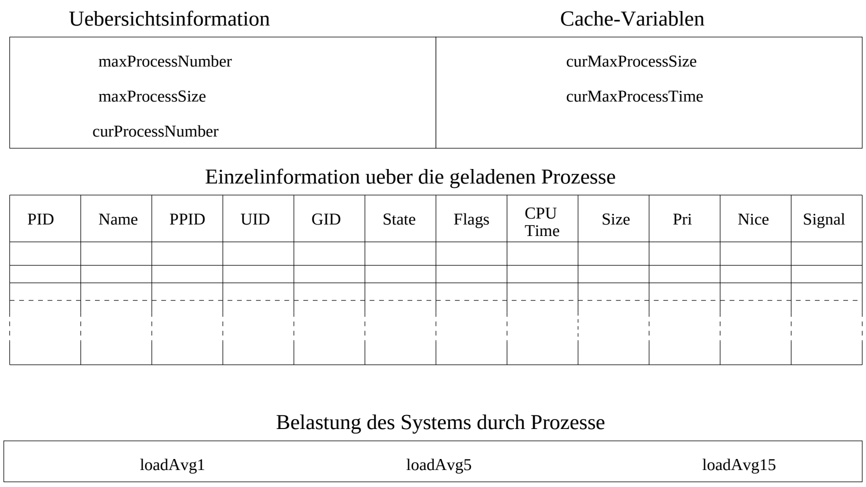

Abbildung 5.3: Prozessmanagement nach Informationsschwerpunkten gegliedert

zwei Cache-Variablen modelliert, die eine effiziente Schwellwertüberwachung ermöglichen. Die Variable curMaxProcessTime beinhaltet die momentan längste Rechenzeit aller Benutzerprozesse in Sekunden auf dem System und die Variable curMaxProcessSize beinhaltet den größten Speicherbedarf in Kilobytes von allen Benutzerprozessen auf dem System. Erst bei Überschreiten eines eventuell gesetzten Schwellwertes muß der betreffende Prozeß lokalisiert und Gegenmaßnahmen eingeleitet werden. Per Definition werden für die Cache-Variablen nur Benutzerprozesse berücksichtigt, da Systemprozesse oftmals durch die Lebenszeit lange Rechenzeiten haben.

Exemplarisch wird nun ein Überwachungsprogramm für Benutzerprozesse vorgestellt, wie es in der Informatik-Halle der TU-München eingesetzt wird. Die Informatik-Halle ist ein Workstationverbund von 120 Rechnern, und durch die Anzahl der Benutzer eine extreme Herausforderung an ein Systemmanagement. Dabei sehen die Regeln folgendermaßen aus:

- Ist der Prozeßeigentümer nicht eingeloggt, die verbrauchte CPU-Zeit aber bereits größer als 3 Minuten und nice ≤ 10, so wird der Prozeß auf die niedrigste Priorität gesetzt.

- Ist der Eigentümer noch eingeloggt, die verbrauchte CPU-Zeit beträgt mehr als 5 Minuten, so wird der Prozeß schrittweise zurückgestuft.
- Zwischen 9:00 und 16:00 Uhr werden alle Prozesse mit einem Hauptspeicherbedarf von mehr als 2 MByte und einer Rechenzeit größer 2 Minuten gestoppt.
- Alle Prozesse mit mehr als 60 Minuten Rechenzeit werden terminiert.

Über jede Änderung des Prozeßtatus erhält der Eigentümer eine e-Mail, in dem ihm mitgeteilt wird, was getan wurde. Dieser Überwachungsmechanismus ist mit der in diesem Kapitel definierten Information, sowie der Kenntniss, ob ein Benutzer eingeloggt ist (vgl. Kapitel 7) jederzeit nachbildbar.

5.4 Auslastung eines UNIX-Systems

Eine häufig zur Beurteilung der Auslastung eines UNIX-Systems herangezogene Größe ist der Loadwert. Auf allen UNIX-Derivaten findet sich das bekannte "xload", das diesen Wert graphisch anzeigt. Der Loadwert ist als die durchschnittliche Anzahl rechenbereiter Prozesse definiert, die sich um den Rechenkern bewerben und ist als Durchschnittswert über den Zeitraum der letzten Minute, der letzten 5 Minuten und der letzten 15 Minuten verfügbar. In Verbindung mit der in Kapitel 4.2.3 definierten Leistungsfähigkeit des Prozessors, der Kentniss über die Nutzung der CPU sowie Kentnisse über den generellen Ausbau des Systems läßt sich die wirkliche Belastung eines Systems abschätzen. An dieser Stelle wird auf den Versuch verzichtet, aus diesen Werten eine neue Belastungszahl zu definieren. Es sei aber auf die Möglichkeit hingewiesen sowohl eine Schwellwertüberwachung durch logische Verknüpfung der verschiedenen Werte zu modellieren als auch aufgrund dieser Zahlen beispielsweise eine Entscheidungsgrundlage zu haben, auf welcher Maschine zusätzliche Prozesse gestartet werden sollen.

Kapitel 6

Partitionen und Filesysteme

In Kapitel 4.4.2 wurden Festplatten erläutert und für das Systemmanagement modelliert. Um die Platten für den Betrieb eines UNIX-Systems bereit zu machen müssen sie als erstes formatiert werden.

6.1 Partitionen

Nach dem Formatieren werden die Platten partitioniert. Der Partitionierungsvorgang wird beispielhaft am Betriebssystem SunOS beschrieben. Sun hat acht verschiedene Spezialpartitionen. Jede dieser Partitionen kann nur ein Filesystem enthalten, gleichzeitig kann aber ein Filesystem nicht mehrere Partionen überspannen. Alleinstehende UNIX-Systeme haben mindestens eine Betriebssystempartition, einen Swapbereich, ein Verzeichnis mit ausführbaren Kommandos und eine Partition mit den Home-Verzeichnissen. Jede Festplatte verfügt über eine Partitionstabelle, die Auskunft über die Anzahl und die Größe der Partitionen gibt.

6.1.1 Modellierung der Partitionen für die MIB

Die Information über die Partitionen wird für das Systemmanagement in einer Tabelle aufgenommen. Bestandteil der Information ist der *partitionLabel* , das ist eine textuelle Beschreibung der Partition, die *partitionID* , das ist die eindeutige Nummer mit der das Betriebssystem die Partition erkennt und die Größe der Partition in KBytes (*partitionSize*). Dazu wird noch eine Variable implementiert, die auf die Instanz der Platte in der Disk-Tabelle verweist, auf der sie sich befindet (*partitionDiskIndex*).

6.2 Filesysteme

Um nun diese Speichereinheiten für das System nutzbar zu machen, müssen logische Speichereinheiten, sogenannte Filesysteme auf ihnen angelegt werden. Ein Filesystem überspannt niemals mehrere Partitionen und eine Partition beherbergt niemals mehrere Filesysteme.

6.2.1 Aufbau von UNIX-Filesystemen

UNIX-Filesysteme sind hierarchisch angeordnet. Obwohl sich die einzelnen UNIX-Derivate etwas in den Hauptverzeichnissen unterscheiden, ist der Aufbau doch immer sehr ähnlich. Nachfolgend wird der prinzipielle Aufbau des UNIX-Verzeichnisbaums am Beispiel von Solaris 2 erläutert. Eine detaillierte Aufstellung der einzelnen Verzeichnisse für *System V* findet sich in [Gulb88].

- / (Das Root-Verzeichnis): Es ist die Wurzel des Verzeichnisbaumes und beinhaltet die systemkritischen Dateien, wie z.B. /kernel/unix, Gerätetreiber und das Bootprogramm.
- /etc: Dieses Verzeichnis beinhaltet systemspezifische Dateien, die zur Systemverwaltung benötigt werden.
- /usr: In diesem Verzeichnis liegen Dateien und Unterverzeichnisse, die alle Benutzer gemeinsam nutzen. Dazu gehören beispielsweise auch die Manualseiten im Verzeichnis /usr/share.
- /home: Unter diesem Verzeichnis sind alle Home-Verzeichnisse der Benutzer angeordnet.
- /var: In diesem Verzeichnis liegen beispielsweise uucp-Files.
- /tmp: Temporäre Dateien sollten in diesem Verzeichnis abgelegt werden. Die Inhalte dieses Verzeichnisses werden bei jedem Bootvorgang gelöscht.

Es gibt einige Grundsätze um Filesysteme gut aufzubauen und damit einen möglichst reibungslosen und störungsfreien Benutzerbetrieb zu ermöglichen. Der erste Grundsatz lautet, Filesysteme, die häufig genutzt werden, sind auf verschiedenen Platten anzulegen. Dies gilt hauptsächlich für das /home und das /swap Verzeichnis[1]. Projekte und Gruppen sollten nach Möglichkeit innerhalb des gleichen Filesystems gehalten werden. Ebenso ist die Anzahl der Filesysteme pro Platte möglichst gering zu halten. Auf der Root-Platte befinden sich im Idealfall nur das Root-, das /usr und das Swap Filesystem. Geräumige Plattenbereiche weisen weniger Fragmentierung als kleine Bereiche auf, die zudem auch sehr schnell zur Überfüllung neigen.

Prinzipiell werden drei verschiedene Filesystemtypen unterschieden:

- Disk-based
- Network-based
- Pseudo

[1] *System V* arbeitet nicht mit einem Swapfilesystem sondern mit einem Swapbereich

In diesem Kapitel werden die plattenbasierten Filesysteme modelliert. Die Netzfilesysteme, NFS und RFS werden, da sie sehr komplexe Mechanismen sind, nur am Rande betrachtet. Nicht betrachtet werden die Pseudofilesysteme. Zu groß sind die Unterschiede zwischen den einzelnen UNIX-Derivaten und zu wenig haben sie mit wirklichen Filesystemen zu tun. Als Beispiel sei nur das /procfs Verzeichnis von *System V* erwähnt, welches alle Prozesse in einem virtuellen Verzeichnis auflistet. Man kann vermuten, daß zu einem späteren Zeitpunkt, wenn in diesem Verzeichnis mehr Information über die Prozesse zur Verfügung gestellt wird, beabsichtigt ist, dem Benutzer die Möglichkeit zu geben, Prozesse mit den gleichen oder ähnlichen Befehlen wie Dateien zu verwaltet.

Um ein Filesystem verfügbar zu machen, muß es gemountet werden. Es gibt verschiedene Möglichkeiten ein Filesystem zu mounten. Es kann in einer Mounttabelle gespeichert werden (SunOS 4.1: /etc/fstab), so daß es bei jedem Bootvorgang gemountet wird. Das Filesystem kann auch durch einen Automounter bei Bedarf gemountet werden oder wenn es zu selten genutzt wird, kann es bei Bedarf von der Kommandozeile aus gemountet.

6.3 Systemadministratortätigkeiten bei Filesystemen

Da der Bereich der Filesysteme ein Schwerpunkt bei den Systemverwalterinterviews war, wird die Information anhand der üblichen Tätigkeiten der Systemverwalter modelliert

- Zu den Aufgaben gehört das Anlegen von Filesystemen (UNX-Kommando mkfs), die richtige Plazierung von Filesystemen (vgl. Kapitel 6.2.1) und die ausreichende Bereitstellung von Swapplatz für die Systeme.
- Die Integrität des Filesystems muß gewährleistet sein (UNIX-Kommando fsck).
- Die Systemverwalter haben die Füllungsgrade der einzelnen Filesysteme und die Verfügbarkeit der Inodes sicherzustellen (UNIX-Kommando df).
- Sie müssen je nach Bedarf Plattenplatzbeschränkungen aktivieren oder deaktivieren. Benutzer müssen überprüft werden, ob sie Kontingente überschreiten und eventuelle Gegenmaßnahmen müssen eingeleitet werden.
- Die für den Betrieb und die Benutzer nötigen Filesysteme sind verfügbar zu machen (mounten) oder auszublenden (umount). Die Mountstrategien (Automount, manueller mount, Mount bei Booten) müssen festgelegt werden. Weiterhin müssen die Systemverwalter auf den Systemen festlegen ob und welche Filesysteme sie in einer verteilten Rechnerumgebung exportieren.
- Durch regelmäßige Backups muß auch im Fehlerfalle die Datensicherheit gewährleistet sein und verlorene Daten müssen sich wiederherstellen lassen.

6.4 Modellierung der MIB für Filesysteme

In dieser Kapitel wird eine Tabelle für die MIB modelliert, mit der sich fast alle oben beschriebenen Systemverwaltertätigkeiten durchführen lassen. Nicht modelliert wird in dieser Tabelle die Quotenregelung auf Benutzerebene. Sie wird unmittelbar der Benutzerberechtigung zugeordnet (vgl. Kapitel 7.1). Auf die Modellierung der Möglichkeit mit dem Agenten neue Filesysteme zu erzeugen, wird verzichtet. Ein neues Filesystem zu erzeugen ist eine seltene Tätigkeit eines Systemverwalters. Sie ist nur nötig, wenn Festplatten ersetzt oder hinzugefügt werden, wenn die Partitionierung einer Festplatte geändert oder die Blockgröße des Filesystems geändert wird. Da für die Implementierung dieser Funktionalität sehr systemnahe Programmierung unter Verwendung nicht dokumentierter Schnittstellen nötig wäre, ist das Risiko zu hoch, einen unbeabsichtigten Schaden oder Datenverlust zu riskieren. Eine Erweiterung der MIB um diese Funktionalität sollte angedacht werden, wenn durch Implementierung anderer systemnaher Werte genügend Erfahrung gesammelt wurde und von Seiten der Systemverwalter wirklich der unbedingte Wunsch besteht, diese Tätigkeiten mittels des Agenten auszuführen. Ebenso wird auf die zeitlich vorgezogene Aufschreibung dieser Funktionalität verzichtet, da man in einer MIB nur die Managementinformation aufschreiben soll, die man auch implementieren wird ([DSIS 93]).

Bestandteil der Filesystemgruppe ist die vollständige Unterstützung der Backupaktivitäten. Bei der Datensicherung werden die Datenbestände dupliziert, um im Fehlerfall die Daten wiederherstellen zu können. Die Datensicherung sollte periodisch vorgenommen werden und wird zumeist auf Magnetbändern durchgeführt. Allgemein wird gefordert, daß eine Datensicherung während des normalen Benutzerbetriebs erfolgen können muß. Es existieren neben der Vollsicherung, bei der alle Daten gesichert werden, auch Teilsicherungen. Teilsicherungen werden oft auch als inkrementelle Sicherungen bezeichnet, wenn die zu sichernden Daten nach ihrer letzten Änderungsdatum selektiert werden[2]. Genau dieses Verfahren wird in der MIB definiert. Nach einer Vollsicherung (Level 0) können 10 Teilsicherungen gemacht werden.

Weiterhin kann man in der Filesystem-Tabelle die Quotenregelung für ein Filesystem aktivieren und deaktivieren, sowie sich über Größe und Auslastung sowohl des Plattenplatzes als auch der Inodes informieren. Die definierte Managementinformation gibt zusammen mit einer Schwellwertüberwachung eine gute Möglichkeit, Meldunen wie "File System full" zu vermeiden.

[2] Beispiel: Sichere alle Daten, die seit dem letzten Level(n-1) Backup verändert wurden

Kapitel 7

Benutzer- und Gruppenverwaltung

Die Einrichtung und Verwaltung von Benutzer- und Gruppenkennungen ist eine rudimentäre Aufgabe des Systemmanagements. Für die Nutzung eines Rechners benötigt man auf der jeweiligen Plattform eine validierte Benutzerkennung. Mit dieser Benutzerkennung ist eine Reihe von benutzerspezifischen Daten verbunden:

- Berechtigung zur Benutzung
- zugeteilter Plattenplatz
- Passwort
- Spezialberechtigungen (Systemadministration, ...)

In diesem Kapitel werden kurz die benötigten Daten zur Einrichtung einer Benutzerberechtigung erläutert. Danach wird die Gruppenverwaltung erklärt und abschließend noch die Unterschiede zwischen lokalen und netzweiter Accountverwaltung aufgezeigt. Die Bedeutung der Benutzerverwaltung für die Betriebssicherheit von UNIX-Systemen wird ebenfalls erläutert. Gleichzeitig werden die systemmanagementrelevanten Verwaltungsdaten für die MIB modelliert.

7.1 Benutzerverwaltung

Benutzerkennung, auch als "Logins" bezeichnet, ermöglichen es dem Benutzer sich im System anzumelden. Die numerische Benutzerkennung (UID) ist mit der alphanumerischen Benutzerkennung (Login) unmittelbar verbunden. Die Begründung in der Verwendung sowohl von Namen als auch Nummern liegt darin, daß Menschen leichter mit Namen umgehen können, während für Rechner die Verarbeitung von Zahlen angebrachter ist. Die UIDs sind die Basis beispielsweise für Abrechnungsmechanismen, für die Kontrolle des vom Benutzer verbrauchten Plattenplatzes und für den Zugriff auf Filesysteme. Ebenso werden sie wie bereits in Kapitel 5.2.3 angedeutet verwendet, um den Eigentümer von Prozessen identifizeren zu können.

Unabdingbar ist, daß die gewählte Kennung eindeutig auf dem System sein muß. Wird die Kennung für eine Berechtigung auf mehreren Systemen in einem Netz gewählt, so muß sie netzweit eindeutig sein. Die Kennung sollte auch auf allen Systemen gleich sein, um von vornherein Konflikte mit Benutzerberechtigungen, beispielsweise beim Dateitransfer auszuschließen. Üblicherweise ist von den UNIX-Systemen der Aufbau vorgeschrieben, so verlangt beispielsweise SunOS 4.1 eine Länge zwischen 6 und 8 Zeichen, wobei das erste Zeichen ein Buchstabe sein muß und mindestens ein Kleinbuchstabe in der Kennung gefordert wird.

Ebenso muß die UID in der Sytemumgebung eindeutig sein[1]. Auch hier wird von den UNIX-Systemen ein bestimmter Bereich vorgegeben, innerhalb dem die IDs frei gewählt werden können. Dabei sind die ersten Nummern reserviert, beispielsweise 0 für die Kennung root.

Eine zweite numerische Kennung ist die Gruppenidentifikationsnummer (GID). Sie wird in Kapitel 7.2 modelliert.

Um unberechtigten Benutzern den Zugang zum System zu verwehren, erhält jeder Benutzer ein Passwort, welches er selbst ändern kann. Damit keine Sicherheitslücken entstehen, sollte der Systemverwalter auf die Möglichkeit, die von fast allen UNIX-Systemen angeboten wird, einen Account bis zur ersten Benutzung ohne Passwort einzurichten, verzichten.

Das in der MIB definierte Comment-Feld kann als zusätzliches Feld genutzt werden um Information über den Benutzer abzuspeichern. Es ist ein historisches Überbleibsel, das Informationen für den Zugriff auf GECOS-Mainframes speicherte. Der Status wurde mit deprecated versehen, da dieses Feld wahrscheinlich in zukünftigen UNIX-Versionen nicht mehr vorhanden sein wird und dann auch von der MIB nicht mehr unterstützt werden muß.

Jeder Benutzer benötigt ein Home-Verzeichnis. Dieses Verzeichnis ist der Teil des File-Systems, das es dem Benutzer ermöglicht, private Dateien zu speichern. Kann ein UNIX-System das Home-Verzeichnis nicht bestimmen, so wird dem Benutzer entweder der Zugang verweigert oder er befindet sich nach dem Login im Root-Verzeichnis. Wichtig ist, daß bei der Einrichtung einer neuen Benutzerkennung zwar das Home-Verzeichnis angegeben werden muß, aber das Home nicht automatisch eingerichtet wird.

Bei der Vergabe der Kennung wird ebenfalls eine Login-Shell für den Benutzer spezifiziert. Auf UNIX-Sytemen gibt es verschiedene Shells. Die bekanntesten sind die Bourne-Shell, die C-Shell und die Korn-Shell. Insbesondere unterstützt die Bourne-Shell Benutzer, die viel Shell-Programmierung betreiben.

Manchmal kann die Notwendigkeit bestehen einen User-Account vorübergehend sperren zu müssen, beispielsweise wenn der Verdacht besteht, daß die Kennung nicht nur von dem dafür berechtigten Benutzer genutzt wird. Die einfachste Möglichkeit, die mit der MIB nachgebildet werden kann, ist das Passwort zu ändern.

Entscheident ist die Funktionalität die von der MIB-Implementation gefordert wird. Eine Benutzerkennung ist solange nicht gültig, bis alle geforderten Daten angegeben sind. Dazu gehören die UID, die Login-Kennung, die GID und das Homeverzeichnis. Damit die Bentzerkennung gültig wird muß auch noch das Homeverzeichnis angelegt werden. Per Voreinstellung

[1]Eine Ausnahme kann beispielsweise die Einrichtung verschiedener Kennungen mit gleicher UID bei Schulungen sein, um nach Abschluß der Schulung die Teilnehmer leicht löschen zu können

wird einem neuen Benutzer die /bin/sh-Shell gegeben. Ebenfalls wird per Voreinstellung der Zwang zum Passwort ändern abgeschaltet.

Bei der Benutzerverwaltung wird die Möglichkeit angeboten, dem Benutzer eine "Quota" zuzuweisen. Eine Quota bestimmt das Limit wieviel Plattenplatz der Benutzer verbrauchen darf. Die Quota wird zwischen einem Hardlimit und einem Softlimit unterschieden. In der MIB wird das Softlimit gesetzt. Das Hardlimit wird dann vom Agenten automatisch gesetzt. Es beträgt 5 MByte mehr als das Softlimit. Ebenfalls mit der "Quota" kann dem Benutzer eine maximale Anzahl an zu verbrauchenden Inodes zugewiesen werden. Hier wird ebenfalls das Softlimit gesetzt, das Hardlimit wird vom Agenten selbst 10Gleichzeitig wird eine MIB-Variable modelliert, die es ermöglicht den Plattenverbrauch eines Benutzers abzufragen. Eine Schwellwertüberwachung oder ein Logmechanismus angesetzt auf diese Variable ermöglicht eine gute Möglichkit zur Benutzerüberwachung und Protokollierung von Benutzerverhalten.

7.2 Gruppenverwaltung

Alle Benutzer eines Systems sind wiederum in Gruppen organisiert. Eine UNIX-Gruppe ist eine konzeptionelle Einheit, die bestimmte Rechte beim Zugriff auf Dateien oder Programme besitzt. Eine Unix-Gruppe besteht neben einer eindeutigen Gruppennummer (GID) noch aus einem Gruppennamen und einer Ansammlung von Benutzern. Beispielsweise werden Mitarbeiter, die am gleichen Projekt arbeiten, einer Gruppe zugeordnet. Neben der erwähnten GID, dem Namen und den Benutzern wird auch noch das Gruppenpasswort für die MIB modelliert. Da es in neueren UNIX-Versionen nicht mehr genutzt wird, ist sein Status mit deprecated belegt worden. Ein Benutzer hat eine "Primary Group". Dieser wird er beim Einloggen in das System automatisch zugeordnet. Sie wird bei Einrichtung der Benutzerkennung spezifiziert (vgl. Kapitel D.2.10). Ist der Benutzer in mehreren Gruppen eingetragen, so kann er mit dem Kommando newgrp in eine Seondary Group wechseln.

An dieser Stelle sei wieder auf die MIB-Implementation hingewiesen. Ein Gruppeneintrag wird erst gültig, wenn eine unbenutzte GID und ein gültiger Name vergeben wurde.

7.3 Sicherheitsaspekte der Benutzer- und Gruppenverwaltung

Ein Systemmanagement muß ebenfalls die Sicherheit des Systems gewährleisten. Dazu gehört beispielsweise das Wissen, daß Benutzerdaten weder unberechtigt gelesen, geändert oder gelöscht werden können und daß sich keine unberechtigten Benutzer Zugang zu dem System verschaffen können. Von neueren UNIX-Betriebssystemen werden teilweise bereits sehr mächtige Sicherheitsmechanismen angeboten. Stellvertretend dafür sei OSF DCE genannt, das bereits alle wesentlichen Mechanismen, wie Authentisierung, Autorisierung und Verschlüsselung bereitstellt(vgl. [Schi93], [OSF DCE]). Zusätzlich sollte das Systemmanagement alle eventuellen Sicherheitslücken entdecken und schließen. Die erste Sicherheitsstufe ist der Login-Prozeß. Der Benutzer muß sich dem System gegenüber durch eine Kennung sowie seinem Passwort identifizieren. Die Benutzer werden zwar meistens von den Systemverwaltern aufgefordert ein sicheresPasswort zu wählen, welches beispielsweise Sonderzeichen enthält. Dies wird jedoch nicht von allen Benutzern befolgt. Der Systemverwalter kann aber in den meisten UNIX-Systemen den Benutzer zwingen, sein Passwort regelmäßig zu wechseln. Üblicherweise muß dem System gegenüber eine Mindestanzahl von Tagen angegeben werden, der zwischen dem Wechsel eines Passworts liegen muß und den maximalen Zeitraum, in dem ein Passwort ungeändert bleiben darf. In der MIB wird dies für jeden Benutzer einzeln durch die Aufnahme der Attribute userPasswdMin und userPasswdMax modelliert. Bietet das System nur die Möglichkeit (z.B. SCO-UNIX) einheitlich alle Benutzer zu einem Passwortwechsel zu zwingen, so haben die Einträge in userPasswdMax für alle Benutzer den gleichen Wert. Gleiches gilt für das Attribut userPasswdMin.

7.4 Die Who-Tabelle

Als dritter Bereich in der Benutzerverwaltung wird noch Information über die eingeloggten Benutzer gegeben. Diese Information ist wichtig um beispielsweise die in Kapitel 5 beschriebene Prozeßüberwachung, bei der berücksichtigt wird ob ein Benutzer eingeloggt ist, nachbilden zu können. Zudem ist es für einen Systemverwalter beispielsweise bei der Ursachenforschung für ein stark ausgelastetes System durchaus interessant zu sehen wieviele Benutzer auf dem System arbeiten. Dabei werden dem Systemmanagement Benutzername, Device über das der Benutzer eingeloggt ist, sowie der Zeitpunkt des Eingeloggens zur Verfügung gestellt.

	notwendige Information	zusaetzliche Information	Sicherheitsaspekte	Steuerung und Ueberwachung des Benutzerverhaltens
User Table	userID userLogin userGroupID userHome userShell	userComment userFullName userTelephone userOffice	userPasswd userPasswdMin userPasswdMax userPasswdWarn	userQuotaSoft userQuotaUsed userInodeSoft userInodeUsed
Group Table	groupName groupID	groupUsers	groupPasswd	
Who Table			userWhere	whoName whoTime who Device

Abbildung 7.1: Definierte MIB-Information zur Benutzerverwaltung, geordnet nach Informationsbeschaffenheit

Kapitel 8

Zusammenfassung und Ausblick

In dieser Diplomarbeit wurde eine Management Information Base für das Systemmanagement von UNIX-Endsystemen modelliert. Da das Systemmanagement ein sehr umfangreiches Themengebiet ist, kann und will diese Diplomarbeit das Thema selbstverständlich nicht vollständig aufarbeiten. In diesem Kapitel wird ein Überblick über die definierte Managementinformation im Vergleich zu den klassischen Managementbereichen gegeben, sowie über den Stand der Implementierung der MIB am LRZ berichtet. Des weiteren werden Hinweise für den Einsatz der MIB gegeben und erläutert, wie zukünftige Erweiterungen aufgenommen werden können.

8.1 Zusammenfassung

Der Schwerpunkt dieser Arbeit lag darin, die definierte Managementinformation semantisch vollständig zu unterlegen und die Anwendbarkeit von bereits existierenden Managementmodellen zu überprüfen. Nachfolgend findet sich eine Auflistung der verschiedenen Managementbereiche und die Gegenüberstellung inwieweit in der Management Information Base dazu Information definiert wurde.

- Die Benutzerverwaltung wurde in Kapitel 7 modelliert. Hierbei lag der Schwerpunkt in der Administration der benutzerspezifischen Objekte wie beispielsweise Rechnerkennungen und Benutzergruppen. Ebenfalls in diesem Kapitel und in Kapitel 5 wurde die Kontingentierung von Betriebsmitteln definiert. Weiterhin wurde das Verwalten von Kennungen in Netzen diskutiert.
- Das Sicherheitsmanagement wurde nicht in einem eigenen Kapitel diskutiert, sondern es wurden die einzelnen Gruppen bereits während der Modellierung auf sicherheitsrelevante Aspekte hin untersucht und falls nötig beschrieben. Allgemein läßt sich feststellen, daß zwar eine MIB prinzipiell Information definieren kann, die beispielsweise die Möglichkeit gibt sicherheitsrelevante Operationen zu protokollieren, jedoch muß der Systemverwalter aus diesen Daten selbst sicherheitsgefährdente Operationen erkennen und Gegenmaßnahmen einleiten und weitere Überwachungsmechanismen an-

stoßen. Außerdem sind die Managementdaten unbedingt vor unberechtigten Zugriff zu schützen (vgl. Kapitel 8.2).

- Zur Softwareadministration wurde außer Information zum verwendeten Betriebssystem keine weitere Managementinformation definiert. Dieser Themenkomplex ist so umfassend, daß auf einen nicht zufriedenstellenden Lösungsansatz (vgl. Kapitel 2.3.3) verzichtet wurde.

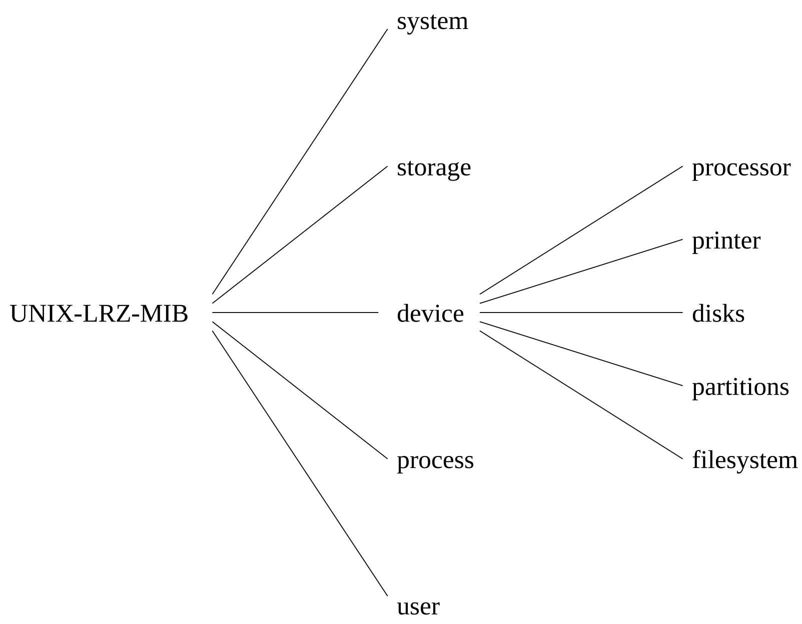

Abbildung 8.1: Haupt- und Untergruppen der definierten MIB

- Zur Hardwareadministration wurde vor allem in Kapitel 4 viel Information unter Verwendung bereits existierender Modelle definiert. Stellvertretend für die Vielzahl von Peripheriegeräten wurde der Drucker modelliert. Andere Objekte, wie beispielsweise Scanner, Plotter, Audio- und Video-Devices mußten unberücksichtigt bleiben. Dennoch wurde versucht, ein Modell zu entwickeln, das auch für andere Hardware anzuwenden ist.

- Das Performancetuning soll die Leistungsfähigkeit bestehender Hardware sicherstellen. Dazu gehören das Ändern von Konfigurationsparametern, die Hardwareerweiterung von Rechnern oder die Verlagerung von Diensten und Daten auf andere Rechner. Diese Maßnahmen können natürlich nur vorgenommen werden, wenn fundierte Information über Auslastung von Rechnern und Platten oder auch Benutzerverhalten vorliegen. Diese Information läßt sich aus der MIB ableiten (vgl. Kapitel 4 und 6).

- Für ein effizientes Fehlermanagement muß die Lokalisation und Diagnose des Fehlers sichergestellt werden. Dies kann durch das Loggen wichtiger Werte durch die Systemverwalter unterstützt werden.

Neben der unvollständigen Modellierung von Peripheriegeräten und dem Verzicht auf eine Teilmodellierung eines Softwareverwaltungsmechanismus wurden ebenfalls keine Schnittstellen zum Formatieren von Plattenlaufwerken, Einrichten von Partitionen und Erzeugen von Filesystemen definiert. Wie in Kapitel 6.4 bereits diskutiert, erfordern diese bei der Implementierung eine sehr systemnahe Programmierung unter Verwendung undokumentierter Schnittstellen. Deshalb sollten erst weitreichende Erfahrungen mit der Implementierung der MIB gesammelt werden, bevor diese Thematik angegangen wird.

8.2 Ausblick

Während der Erstellung dieser Diplomarbeit wurde bereits mit der Implementierung der MIB im Rahmen eines Fortgeschrittenenpraktikums an der TU-München begonnen. Erste ausgewählte Werte können bereits vom verwendeten CMU-Agenten zur Verfügung gestellt werden. Die Erstimplementation erfolgt auf SUN-Workstations und wird danach auf andere Systeme portiert werden. Im Rahmen anderer Fortgeschrittenenpraktika werden gleichzeitig der CMU-Agent auf andere Plattformen transferiert, sowie ein Multiplexer (S-Muxer) implementiert, der die Koexistenz mehrerer Agenten auf den zu überwachenden Systemen sicherstellt. Dadurch ist es möglich, neben gekauften Agenten auch eigene private Implementierungen, die die eigenen Bedürfnisse vollständiger befriedigen, zu betreiben, ohne daß der Systemverwalter mit der Agentenheterogenität konfrontiert wird. Die Management Information Base wurde so definiert, daß Erweiterungen problemlos aufgenommen werden können. Insbesondere in der Device-Gruppe werden wegen der Vielzahl der im LRZ vorhandenen Peripheriegeräte die zukünftigen Erweiterungsschwerpunkte liegen, deren Überwachung natürlich auch im Rahmen eines integrierten Systemmanagements gewünscht wird.

Ein letzte wichtige Bemerkung dient dem Betrieb des Systemmanagements. Im RFC 1446 werden ein Digest Authentication Protocol und ein Symmetric Privacy Protocol definiert. Während das Authentication Protocol sicherstellt, daß die empfangenen Daten auch wirklich von einem berechtigten Sender stammen, bietet das Privacy Protocol die Möglichkeit, die Daten zu verschlüsseln. Diese Sicherheitsmechanismen sind bei der Transferierung sicherheitsrelevanter Daten unbedingt einzusetzen. Es darf beispielsweise bei der Einrichtung eines neuen Benutzeraccounts unter keinen Umständen passieren, daß die Daten gelesen oder modifiziert werden können.

Anhang A

Auswertung der MIBs mittels Klassifikationsschema

Aufbau des Klassifikationsschemas

Struktur des Klassifikationsschemas:

System Group	charakterisierende Information über das System
Storage Group	logische Speicherkomponenten
Device Group	Hardware des Systems (auch Peripherie)
Übersicht	Auflistung aller Devices des Systems
Prozessor	Daten über Prozessor(en) und Auslastungswerte
Drucker	Statusinformation, Druckschlangen, Druckjobs
Plattenlaufwerke	physikalische Betrachtung der Speichermedien
Partitionen von Platten	Verfeinerung der Plattenlaufwerke
Filesystem	Information über Filesystem
Process Group	Managementinformation über Prozesse auf dem System
User Group	Benutzeradministration und Information über eingeloggte Benutzer
Error Group	Behandlung von allen Fehlerzuständen des Systems
Installed Software Group	Softwareverwaltung

Bemerkung 1:
Aus den einzelnen Management Information Bases werden nur Variablen mit managementrelevanter Information übernommen. Strukturierungselemente, wie z.B. Indizes in Tabellen oder Information die das Cachen von Einträgen für den Agenten ermöglichen sollen, werden nicht mitaufgenommen.

Bemerkung 2:
Stammen alle Variablen einer MIB, die in eine Gruppe des Klassifikationsschemas aufgenommen werden, aus derselben MIB-Gruppe, so wurde das Präfix der Variablen zur einfachen Lokalisation in die erste Zeile mitaufgenommen.

Abbildung A.1:

System Group

allgemeine Information über Rechner

Information	HRM	Krupczak	CMU	BSD	SUN	HP
	(hrSystem)					(computerSystem)
Zeit seit letztem Booten	Uptime				UpTime	UpTime
internes Datum u. Zeit des Systems	Date				unixTime	
Boot-Device des Systems	InitialLoadDevice					
Boot-Parameter des Systems (Pfad des OS)	InitialLoadParameters					
Anzahl der eingeloggten Benutzer	NumUsers					Users
Anzahl der gerade geladenen Prozesse	Processes					processNum
Anzahl der maximal möglichen Prozesse	MaxProcesses					MacProc
Name des Rechners		nodename				
Typ der CPU	-> hrProcessor	cpu				
Menge des Hauptspeichers	-> hrStorage	memory				PhysMemory
Seriennummer des Motherboards		serialno				
Versionsnummer d. OS	-> hrSW	osver				
Releasenummer d. OS des Rechners		osrel				
eindeutige Hardwarekennung					hostID	
erste Zeile von /etc/motd					motd	
max. Hauptspeichermenge, über die Benutzer verfügt						MaxUserMem

Bemerkung: Variablen, die bereits in der MIB-II definierte Information beinhalten, wurden nicht berücksichtigt

Abbildung A.2:

Storage Group

Speicheradministration (logischer Speicher)

Information	HRM	Krupczak	CMU	BSD	SUN	HP
	(hrStorage)	(dev)	(mem)			(computerSystem)
Hauptspeichergröße des Rechners	MemorySize	->System				-> System
Speichertyp (*)	Type					
Beschreibung der Speichereinheit	Descr					
Größe einer Speichereinheit	AllocationUnits	Bsize				
Größe des Speichers in Speichereinheiten	Size	Tblks				
Anzahl der benutzten Speichereinheiten	StorageUsed					
abgelehnte Speicheranforderungen	AllocationFailures					
Anzahl der unbenutzten Speichereinheiten		Fblks				
benutzter Virtueller Speicher			ActiveVirtual			
gesamter, zur Verfügung stehender virt. Speicher			TotalVirtual			
benutzter Hauptspeicher			ActiveReal			
freier Hauptspeicher						FreeMemory

(*) Bemerkung: Speichertypen des logischen Speichers sind RAM, virtueller Speicher, Festplatten, Wechselplatten, CD-ROMs u.a.

Abbildung A.3:

Device Group (1) - Übersicht

Information	HRM	Krupczak	CMU	BSD	SUN	HP
	(hrDevice)	(dev)		(fs)		
Typ des Device (*)	Type	Type				
Beschreibung d. Device	Descr	Fstr				
Produkt-ID	ID					
Status (running, testing, down,...)	Status					
Anzahl der aufgetretenen Fehler	Errors					
Name des Device		Device		Name		

Abbildung A.4:

(*) Bemerkung: DeviceTypen sind Prozessor, Netzwerkkarten, Drucker, Plattenlaufwerke, Graphikkarten, Audiokarten, Coprozessoren, Tastatur, Modem, Paralleler Port, Mause, Serieller Port, Tape u.a.

Device Group (2) - Prozessor

Information	HRM	Krupczak	CMU	BSD	SUN	HP
	(hrProcessor)		(cpu)		(rs) (*)	(computerSystem)
Produkt-ID	FrwID					
Load-Wert der letzten Minute	Load		LoadOneMinute			AvgJobs1
Endzeitpunkt der Aufzeichnung		sampleTime				
Länge des Meßintervalls		sampleIntv				
Idle-Zeit im Meßintervall		cpuIdle	UsageIdle			
User-Zeit im Meßintervall		cpuUser	UsageUser			
Kernel-Zeit im Meßintervall		cpuKernel	UsageSystem			
Wait-Zeit auf I/O im Meßintervall		cpuWait				
Sxbrk-Zeit im Meßintervall		cpuSxbrk				
Load-Wert über die letzten 5 Minuten			LoadFiveMinute			AvgJobs5
Load-Wert über die letzten 15 Minuten			LoadFifteenMinute			AvgJobs15
Zeit, in der CPU von Benutzerprozessen beansprucht					UserProcessTime	UserCPU
Zeit, in der CPU im Nice-Modus beansprucht					NiceModeTime	SysCPU
Zeit, in der CPU von Systemprozessen beansprucht					SystemProcessTime	IdleCPU
Zeit, in der CPU im Idle-Mode					IdleModeTime	NiceCPU

(*) Bemerkung: Zeitangaben sind aufsummiert seit dem letzten Boot-Zeitpunkt

Abbildung A.5:

Abbildung A.6:

Device Group (3) - Drucker

Deviceadministration

Information	HRM	Krupczak	CMU	BSD	SUN	HP
	(hrPrinter)		(print)			
Druckerstatus (idle, printing, warmup, ...)	Status					
Fehlerstatus (low Paper, no Paper, no Toner, ...)	DetectedErrorState					
Name einer Druckerwarteschlange			QName			
Status einer Schlange (Binär)			QStatus			
Status einer Schlange (Text)			QDisplay			
Einträge in der Warteschlange			QEntries			
Platz in der Warteschlange			JRank			
Jobname			JName			
Benutzerkennung des Auftraggebers			JOwner			
textuelle Beschreibung des Inhalts des Jobs			JDescription			
Größe des Jobs			JSize			

Abbildung A.7:

Device Group (4) - Plattenlaufwerke

Information	HRM	Krupczak	CMU	BSD	SUN	HP
	(hrDiskStorage)	(dev)	(diskDrive)		(rs)	
Zugriffsart (schreiben-lesen, nur lesen)	Access					
Typ der Platte (*)	Media					
Wechselmedium	Removeable					
Größe des Speichermediums	Capacity					
letzter Zugriff auf Platte			Time			
Zugriffe auf Platte			Transfers			
Zahl der eingelesenen Pages von der Disk					VPagesIn	
Zahl der ausgelesenen Pages auf die Disk					VpagesOut	
Zahl der eingeswappten Pages					VSwapIn	
Zahl der ausgeswappten Pages					VSwapOut	
Zahl der Device-Interrupts					VIntr	

(*) Bemerkung: Arten sind Festplatte, Diskette, CD-ROM, CD-WORM, RAM-Disk, ...

Device Group (5) - Partitionen von Platten

Information	HRM	Krupczak	CMU	BSD	SUN	HP
	(hrPartition)	(dev)				
textuelle Beschreibung d. Partition	Label					
Betriebssystemkennung d. Partition	ID					
Größe der Partition	Size					
Verweis auf Filesystemtable	FSIndex					

Abbildung A.8:

Device Group (6) - Filesystem

Abbildung A.9:

Information	HRM	Krupczak	CMU	BSD	SUN	HP
	(hrFS)	(dev)	(fileSystem)	(fs)		(fileSystem)
Pfadangabe des Root-Verzeichnisses	MountPoint	MntPt	MountPoint	MountPoint		Dir
Name u. Adresse d. Servers von dem FS gemountet	RemoteMountPoint					
Typ des Filesystems (*)	Type		Name	MountType		
Zugriffsart (schreiben-lesen, nur lesen)	Access					
Bootfähigkeit	Bootable					
Verweis auf StorageTable	StorageIndex					
Datum des letzten vollständigen Backups	LastFullBackupDate		DumpDate			
Datum des letzten nichtvollständigen Backups	Last Part.BackupDate					
Filesystem-ID		Fsid				ID1 / ID2
maximal unterstützte Namenslänge		MaxNameLen				
Anzahl der Files oder Inodes		Tfiles	Size	InodesCount		Files
Anzahl der freien Files oder Inodes		Ffiles	Free	InodesAvailable		Ffree
Anzahl der benutzten Files oder Inodes			Used			
Optionen des gemounteten Filesystems				MountOptions		
Blockgröße des Filesystems				BlockSize		Bsize
Gesamtzahl aller Blöcke im Filesystem				BlockCount		Block
Anzahl aller freien Blöcke im Filesystem				BlocksFree		Bfree
Anzahl d. freien Blöcke für Normalbenutzer				BlocksAvailable		Bavail
Anzahl der gemounteten Filesysteme						Mounted
Name des Filesystems						Name

(*) Bemerkung: Arten sind NFS, DFS, AFS, NTFS, u.a.

Abbildung A.10:

Process Group (1)

Information	HRM (hrSWRun)	Krupczak (process)	CMU (process)	BSD	SUN (psProcess)	HP (process)
textuelle Beschreibung des Prozesses	Name					
eindeutige Identifikationsnr. des Prozesses	Index	ID	ID		ID	PID
Produkt-ID d. Software	ID					
Pfadangabe d. Software	Path					
Typbezeichnung (OS, Driver, Application,...)	Type					
Status (running, runnable,...)	Status	State			State	Status
von Prozeß konsumierte CPU-Zeit	PerfCPU				CPUTime	CPUticks
von Prozeß allokierter Speicher	PerfMem				Size	
Prozessname (Startkommando)		Name	Command		Name	Cmd
Nice-Wert		Nice				Nice
Flags des Prozesses		Flags				Flags
UID des Besitzers		UID			UserID	UID
GID des Besitzers		GID				
Nr. der CPU, die den Prozess abarbeitet			SlotIndex			ProcNum
PID des Vaterprozesses					ParentID	PPID
Wartewert d. Prozesses					WaitChannel	Wchan
assoziiertes Terminal					TTY	TTY
Name des Besitzers					UserName	Uname
Index für pstat()-Aufrufe						Idx
Größe des Data-Segments						Dsize
Größe des Text-Segments						Tsize
Größe des Stack-Segments						Ssize
Process tty major number						Major
Process tty minor number						Minor
Prozeßgruppe, der der Prozeß angehört						Pgrp
Priorität des Prozesses						Prio
Speicheradresse des Prozesses						Addr

Information	HRM	Krupczak	CMU	BSD	SUN	HP
	(hrSWRun)	(process)	(process)		(psProcess)	(process)
Prozessor Uitlization						CPU
verbrauchte Prozessorzeit im Usermodus						Utime
verbrauchte Prozessorzeit im Systemmodus						Stime
Startzeit des Prozesses						Start
Größe einer Zeitscheibe beim Scheduling						Time
Lebenszeit des Prozesses						CPUticksTotal
Fair Share Scheduler Group						Fss
Prozent der CPU-Benutzung						PctCPU
Resident Set Size for process						Rssize
gespeicherte UID						SUID

Abbildung A.11:

User Group (1)

Information	HRM	Krupczak	CMU	BSD	SUN	HP
		(user)		(user)		
Benutzer-Login		LoginID		Name		
verschlüsseltes Passwort d. Benutzers		Passwd		Passwd		
Benutzer-ID		UID		ID		
Benutzer-Gruppen-ID		GID		Group		
Benutzername		Name		FullName		
Home d. Benutzers		HomeDir		Home		
Shell d. Benutzers		Shell		Shell		
Quota d. Benutzers				Quota		

		(group)		(group)		
Gruppenname		Name		Name		
verschlüsseltes Passwort		Passwd		Passwd		
Gruppen-ID		GID		ID		

User Group (2)

Benutzerverwaltung (eingeloggte Benutzer)

Information	HRM	Krupczak	CMU	BSD	SUN	HP
		(who)				
Name des Benutzers		Name				
Device, von welchem aus Einloggen erfolgte		Device				
PID der Login-Shell		PID				
Uhrzeit des Einloggens		Time				
Ort, an dem Benutzer sich aufhält		Where				

Abbildung A.13:

Error Group

Information	HRM	Krupczak	CMU	BSD	SUN	HP
Module-ID, von dem Fehlermeldung stammt		mid				
Zweit-ID des Moduls		sid				
Zeitpunkt der Fehlermeldung		time				
eindeutige Fehlerkennung		tag				
Typ des Fehlers		type				
Grund des Fehlers		cause				
Schwere des Fehlers		severity				
Level der Software, die Fehler meldet		level				
Name des Moduls		module				

Abbildung A.14:

Installed Software Group

Softwareübersicht

Information	HRM	Krupczak	CMU	BSD	SUN	HP
	(hrSWInstalled)					
Letzte Installationsänderung im System	LastChange					
Name, Hersteller, Versionsnr, Seriennummer	InstalledName					
Produktidentifikationsnummer	ProductID					
Typ der Software	Type					
Letzte Veränderung des Programms	Date					

Abbildung A.15:

Anhang B

Fragen der Systemverwalterinterviews und Auswertung der Ergebnisse

B.1 Fragebogen und Interviews

Die in diesem Anhang repräsentativ vorgestellten Fragen sind eine Mischung aus den Systemverwalterinterviews und der anschließend durchgeführten Fragebogenaktion.

Die vorgestellten MIBs

- Wie beurteilen Sie die HRM und die UNIX-MIB[1]?

  Es ist sehr viel Information definiert. Manche Werte sind für die Tätigkeit eines Systemverwalters interessant.

- Welche Information interessiert sie besonders?

  Als Workstationverwalter bin ich sehr an Prozeßüberwachung interessiert, insbesondere in NFS-Umgebungen am Erkennen von rekursiven find-Aufrufen. Ebenso will ich Informationen über Plattenaktivitäten und über die Anzahl der geöffneten Files.

  Als Nicht-Workstationverwalter interessiert mich besonders an der Prozeßüberwachung ob beispielsweise die UNITREE- Dämonen richtig arbeiten.

- Welche Information ist verfehlt?

  Die Softwareüberwachung ist eine gute Idee, aber sie scheint nicht vollständig zu sein. Die in den MIBs definierte Information ist zu netzlastig und deshalb für ein Systemmanagement größtenteils ungeeignet.

- Welche Funktionalität sollten einzelne Variablen bieten?

  Bei vielen Variablen wäre es wünschenswert, wenn man bei Fehler eine Meldung bekommt. Ebenso wäre eine Schwellwertüberwachung einiger Werte wohl sinnvoll.

[1] Die Systemverwalter hatten ca. 4 Wochen vor den Interviews eine Ausarbeitung über die beiden MIBs vom Verfasser der Diplomarbeit bekommen

Besonderheiten in den jeweiligen Umgebungen der Systemverwalter

- Welche Besonderheiten weist der von ihnen verwaltete Rechner auf?

  Eine Besonderheit ist das NQS (Network Queuing System), über das alle Batchjobs abgewickelt werden, die länger als fünf Minuten auf dem Rechner sein werden. Eine weitere Besonderheit ist die Verwendung von Bändern als Tertiärspeicher und ihre Einbindung über UNITREE. Bei der Cray wird eine 100%-Auslastung gewünscht, wobei die Rechenzeit, die im Systemmodus verbraucht wird, 10% nicht übersteigen sollte.

- Welche Besonderheiten weisen ihre Benutzer auf?

  Auf der Cray benötigen die Benutzer immer mehr Speicherplatz für ihre Anwendungen. Die Wartezeiten, bis ein Auftrag gerechnet wird, steigen. Auf dem SUN-Workstationcluster werden sehr viele Plattenoperationen durchgeführt.

Die Tätigkeit des Systemverwalters

- Was sind Ihre Haupttätigkeiten?

  Bei der Cray muß sichergestellt sein, daß sie eine möglichst gute Auslastung hat. Die Überwachung erfolgt differenziert duch MWS, das von einem Techniker bedient wird, sowie über OWS.Zudem wird SICK, ein Cray-Überwachungssystem eingesetzt. Weiterhin ist das Funktionieren von UNITREE sicherzustellen.

- Welche Werte müssen sie regelmäßig überwachen und welche Hilfsmittel setzen sie dazu ein?

  Auf Workstations werden insbesondere unter Verwendung von den UNIX-Kommandos iostat, vmstat, pstat die Plattenaktivitäten, die Paging-Aktivitäten sowie die Filesystemaktivitäten beobachtet. Einen guten Überblick über die Auslastung des Systems liefert das frei erhältliche xload.

B.2 Zusammenfassung der Ergebnisse

Auf eine mathematische Auswertung der Ergebnisse wurde in diesem Kapitel verzichtet, da durch die Mischung von Workstationverwaltern und Administratoren von Cray, KSR und Cyber zwei zu unterschiedliche Rechnerwelten und Managementansichten vermischt worden wären. Deshalb wird hier nochmals ein kurzer Überblick über die Ergebnisse gegeben. Von allen Verwaltern wurde eine Prozeßüberwachung gewünscht, um damit auch über die auf dem System laufenden Dämonen informiert zu sein. Ebenfalls großes Interesse beseht in Information über NQS und über die Erreichbarkeit der Maschinen. Für einen mächtigen Softwareverteilungs- und Verwalungsmechanismus sahen ebenfalls fast alle Verwalter Bedarf. Information über den Ausbau der Maschinen (Hauptspeicher, etc) wurde nur von den Workstationverwaltern gewünscht. Die CPU-Auslastung wurde wiederum von allen Verwaltern gefordert. Ebenfalls als wichtig wurde die Überwachung der Filesysteme eingestuft

(Füllungsgrad und Mountinformation). Eine Protokollierung und Überwachung von Backupaktivitäten wurde ebenfalls nur von den Workstationverwaltern als erforderlich angesehen.

Anhang C

Managementinformationsbasis für das Management von UNIX-Endsystemen (UNIX-LRZ-MIB)

In diesem Anhang befindet sich die im Rahmen der Diplomarbeit entwickelte Management Information Base für das Management von UNIX-Endsystemen. Die Aufschreibung der MIB erfolgt in ASN.1 Notation, gemäß Internet-Informationsmodell für SNMPv2 ([RFC 1442]). Der Einsatz am LRZ erfolgt in Verbindung mit dem CMU-SNMP-Agenten Version 2.1.2. Dieser Agent versteht sowohl SNMPv1 als auch SNMPv2. Da die momentan im Einsatz befindliche Version der Managementplattform *HP OpenView* noch nicht SNMPv2 unterstützt, muß unter Umständen für die Erstimplementation die MIB mit dem Tool "snmp.convert" in das SNMPv1-Format überführt werden. Das Tool gehört zum CMU-Paket, welches über anonymous ftp von dem FTP-Server *lancaster.andrew.cmu.edu* erhältlich ist.

```
-- LRZ-UNIX-MIB
-- Diplomarbeit an der TUM-Muenchen
-- Aufgabensteller: Prof. Dr. H.-G. Hegering
-- Autor: Uwe Krieger
-- e-mail: krieger@informatik.tu-muenchen.de


SNMPv2-LRZ DEFINITIONS ::= BEGIN;

    nullOID       OBJECT IDENTIFIER ::= { ccitt 0 }
    internet      OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
    directory     OBJECT IDENTIFIER ::= { internet 1 }
    mgmt          OBJECT IDENTIFIER ::= { internet 2 }
    experimental  OBJECT IDENTIFIER ::= { internet 3 }
    private       OBJECT IDENTIFIER ::= { internet 4 }
    enterprises   OBJECT IDENTIFIER ::= { private 1 }



   -- currently the LRZ-MIB is fixed under experimental.100 !
   -- with registration the tree probably has to be moved.

   LRZ OBJECT IDENTIFIER ::= { experimental 100 }

   -- FoPra Bastian Pusch (puschb@informatik.tu-muenchen.de)
   -- LRZ-LOG OBJECT IDENTIFIER ::= {LRZ 1}

   LRZ-UNIX OBJECT IDENTIFIER ::= {LRZ 2}

   -- textual conventions
   Boolean ::= INTEGER{true(1),false(2)}
   KBytes ::= INTEGER(0..2147483647)
   Bytes ::= INTEGER(0..2147483647)
   DisplayString ::= OCTET STRING (SIZE (0..255))


   -- Groups in LRZ-UNIX-MIB

   system      OBJECT IDENTIFIER ::= {LRZ-UNIX 1}
   storage     OBJECT IDENTIFIER ::= {LRZ-UNIX 2}
   device      OBJECT IDENTIFIER ::= {LRZ-UNIX 3}
   process     OBJECT IDENTIFIER ::= {LRZ-UNIX 4}
   user        OBJECT IDENTIFIER ::= {LRZ-UNIX 5}

   -- The UNIX-LRZ System Group
   -- Implementation of the System Group is mandatory for all
   -- systems. If an agent is not configured to have a value for
   -- any of these variables, a string of length 0 is returned.
```

```
sysName OBJECT-TYPE
   SYNTAX     DisplayString (SIZE (0..255))
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The system's fully qualified system and domain name,
           e.g. sunhegering5@informatik.tu-muenchen.de."
   ::={ system 1 }

sysContact OBJECT-TYPE
   SYNTAX     DisplayString (SIZE (0..255))
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
           "The textual identification of the contact
           person for this managed system, together with
           information on how to contact this person."
   ::={ system 2 }

sysLocation OBJECT-TYPE
   SYNTAX     DisplayString (SIZE (0..255))
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
           "The pysical location of this node."
   ::={ system 3 }

sysOs OBJECT-TYPE
   SYNTAX     DisplayString (SIZE (0..255))
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The textual description of the operating
           system that the managed node is running."
   ::={ system 4 }

sysHardware OBJECT-TYPE
   SYNTAX     DisplayString (SIZE (0..255))
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
           "The textual description of the managed
           node, e.g. SUN ELC."
   ::={ system 5 }

sysUptime OBJECT-TYPE
   SYNTAX     INTEGER
```

```
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The time since last boot in seconds."
   ::={ system 6 }

sysDate OBJECT-TYPE
   SYNTAX     DateAndTime
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
           "The host's notation of local date and time."
   ::={ system 7 }

sysUsers OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The number of users logged on to the system."
   ::={ system 8 }

-- The UNIX-LRZ Storage Group
-- Implementation of the Storage Group is mandatory for all
-- systems. If an agent is not configured to have a value for
-- any of these variables, a string of length 0 is returned.

stoTypes OBJECT IDENTIFIER ::= { storage 1 }
StoRam            OBJECT IDENTIFIER ::= { stoTypes 1 }
StoVirtualMemory  OBJECT IDENTIFIER ::= { stoTypes 2 }
StoFLCache        OBJECT IDENTIFIER ::= { stoTypes 3 }
StoSLCache        OBJECT IDENTIFIER ::= { stoTypes 4 }

stoTable OBJECT-TYPE
    SYNTAX SEQUENCE OF stoEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
           "The table with logical storage areas on this host."
::= { storage 1 }

stoEntry OBJECT-TYPE
    SYNTAX StoEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
           "One entry for each logical storage area
           on the host."
```

```
    INDEX { stoIndex }
    ::= { stoTable 1 }

  StoEntry ::= SEQUENCE {
        StoIndex            INTEGER,
        StoType             OBJECT IDENTIFIER,
        StoDescr            DisplayString,
        StoAllocationUnits  Bytes,
        StoSize             KBytes,
        StoUsed             INTEGER,
        StoState            INTEGER
  }

stoIndex OBJECT-TYPE
    SYNTAX     INTEGER
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The unique value of this instance."
    ::= { stoEntry 1 }

stoType OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The type of this instance."
    ::= { stoEntry 2 }

stoDescr OBJECT-TYPE
    SYNTAX     DisplayString
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "A string describing the instance, e.g. virtual memory
            if stoType is StoTypes 3."
    ::= { stoEntry 3 }

stoAllocationUnits OBJECT-TYPE
    SYNTAX     Bytes
    MAX-ACCESS read-only
    STATUS      current
    DESCRIPTION
            "The size in bytes of one chunk of this instance."
    ::= { stoEntry 4 }

stoSize OBJECT-TYPE
    SYNTAX     KBytes
```

```
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
            "The Size in kilobytes, as might been seen by the
            operating system."
   ::= { stoEntry 5 }

stoUsed OBJECT-TYPE
    SYNTAX     KBytes
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The kilobytes of used storage of this instance."
    ::= { stoEntry 6 }

stoState OBJECT-TYPE
    SYNTAX     INTEGER {
               locked(1),
               unlocked(2)
    }
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The percentage of used storage of this instance."
    ::= { stoEntry y }


-- The UNIX-LRZ DEVICE Group
-- Implementation of the Processor Group is mandatory for all
-- systems. If an agent is not configured to have a value for
-- any of these variables, a string of length 0 is returned.
-- some nice type definitions following

processor OBJECT IDENTIFIER  ::= { device 1 }
printer OBJECT IDENTIFIER    ::= { device 2 }
disk OBJECT IDENTIFIER       ::= { device 3 }
partition OBJECT IDENTIFIER  ::= { device 4 }
filesystem OBJECT IDENTIFIER ::= { device 5 }
network OBJECT IDENTIFIER    ::= { device 6 }
video OBJECT IDENTIFIER      ::= { device 7 }
audio OBJECT IDENTIFIER      ::= { device 8 }
modem OBJECT IDENTIFIER      ::= { device 9 }
pointing OBJECT IDENTIFIER   ::= { device 10 }
tape OBJECT IDENTIFIER       ::= { device 11 }
openprom OBJECT IDENTIFIER   ::= { device 12 }
bus OBJECT IDENTIFIER        ::= { device 13 }
controller OBJECT IDENTIFIER ::= { device 14 }
```

```
deviceTable OBJECT-TYPE
   SYNTAX SEQUENCE OF deviceEntry
   MAX-ACCESS not-accessible
   STATUS current
   DESCRIPTION
        "The table with all host's devices, if
        defined above."
   ::={ device 1 }

deviceEntry OBJECT-TYPE
   SYNTAX DeviceEntry
   MAX-ACCESS not-accessible
   STATUS current
   DESCRIPTION
           "One entry for each detected and defined device."
   INDEX {deviceIndex}
   ::={ deviceTable 1 }

DeviceEntry ::= SEQUENCE {
   devIndex  INTEGER,
   devType   OBJECT IDENTIFIER,
   devDescr  DisplayString,
   devStatus INTEGER
}

devIndex OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "An unique value for each device."
   ::={ deviceEntry 1 }

devType OBJECT-TYPE
   SYNTAX     OBJECT IDENTIFIER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The object identifier defined above for this instance.
           If the type is one of devProcessor, devPrinter or devDisk an
           correspondending entry exists to this device."
   ::={ deviceEntry 1 }

devDescr OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
```

```
           "A textuell description of the device."
   ::={ deviceEntry 1 }

devStatus OBJECT-TYPE
   SYNTAX     INTEGER {
              unknown(1),
              running(2),
              down(3),
              warning(4)
   }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "An unique value for each device"
   ::={ deviceEntry 1 }


-- The UNIX-LRZ Processor Group
-- Implementation of the Processor Group is mandatory for all
-- systems. If an agent is not configured to have a value for
-- any of these variables, a string of length 0 is returned.

cpuType OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "Tpye of CPU, eg. SPARC."
   ::={ processor 1 }

cpuClockRate OBJECT-TYPE
   SYNTAX     Integer
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The cpu's clock rate."
   ::={ processor 2 }

cpuTable OBJECT-TYPE
   SYNTAX     SEQUENCE OF CpuEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
           "A table with all processors in the host."
   ::={ processor 3 }

cpuEntry OBJECT-TYPE
   SYNTAX      CpuEntry
```

```
   MAX-ACCESS  not-accessible
   STATUS      current
   DESCRIPTION
          "A (conceptual) entry for each processor
          of the host. Only one entry for a single
          processor machine."
   INDEX   { cpunr }
   ::={ cpuTable 1 }

CpuEntry ::= SEQUENCE {
   cpuNr         INTEGER,
   cpuOpStat     INTEGER,
   cpuUsStat     INTEGER,
   cpuAdstat     INTEGER,
   cpuAction     INTEGER,
   cpuUserTime   INTEGER,
   cpuSystemTime INTEGER,
   cpuIdleTime   INTEGER
}

cpuNr OBJECT-TYPE
   SYNTAX     Integer
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The internal number of the processor."
   ::={ cpuEntry 1 }

cpuOpStat OBJECT-TYPE
   SYNTAX     INTEGER {
              enabled(1),
              disabled(2)
   }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The Operational Status derived from ISO-10164-2.
           Processor enabled means that the processor is
           available for the user or applications. Disabled means that the
           processor is not available to the user or applications.
           Not to be implemented for single-processor-machines."
   ::={ cpuEntry 2 }

cpuUsStat OBJECT-TYPE
   SYNTAX     INTEGER{
              idle(1)
              busy(2)
   }
```

```
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
            "The processor's Usage State (ISO-10164-2). Idle means
            that no job is running on the processor. Busy means that
            a job is running on the processor. Not to be implemented
            for single-processor-machines."
   ::={ cpuEntry 3 }

cpuAdStat OBJECT-TYPE
   SYNTAX     INTEGER {
              unlocked(1),
              locked(2),
              shuttingdown(3)
   }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
            "The processor's Administration State (ISO-10164-2).
            Unlocked means that usage is allowed, shuttingdown
            means that the usage for new users is not allowed
            and after the last user has is job finished the
            administrative state will be locked, what means that
            the processor is not available for any user."
   ::={ cpuEntry 4 }

cpuAction OBJECT-TYPE
   SYNTAX     INTEGER {
              enable(1),
              disable(2),
              lock(3),
              unlock(4),
              shuttingdown(5)
   }
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
           "The actions defined for systems management."
   ::={ cpuEntry 5 }

cpuUserTime OBJECT-TYPE
   SYNTAX     REAL
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
           "The percentage of time the processor was used
           in user mode (calculated over the last 0.5 seconds."
   ::={ cpuEntry 6 }
```

```
cpuSystemTime OBJECT-TYPE
   SYNTAX     REAL
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
           "The percentage of time the processor was used
           in system mode (calculated over the last 0.5 seconds)."
   ::={ cpuEntry 7 }

cpuIdleTime OBJECT-TYPE
   SYNTAX     REAL
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
           "The percentage of time the processeor was idle
           (calculated over the last 0.5 seconds)."
   ::={ cpuEntry 8 }

cpuSpecInt OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
          "The Result of the SPECint benchmark."
   ::={ processor 4 }

cpuSpecIntYear OBJECT-TYPE
   SYNTAX     Integer
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The version of the benchmark,
           the year (SPECint92)."
   ::={ processor 5 }

cpuSpecFp OBJECT-TYPE
   SYNTAX     Real
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The result of the SPECfp benchmark."
   ::={ processor 6 }

cpuSpecFpYear OBJECT-TYPE
   SYNTAX     Integer
   MAX-ACCESS read-only
   STATUS     current
```

```
   DESCRIPTION
           "The version of the benchmark,
           the year (SPECfp92)."
   ::={ processor 7 }


-- The UNIX-LRZ Printer Group
-- Implementation of the Printer Group is mandatory for all
-- systems. If an agent is not configured to have a value for
-- any of these variables, a string of length 0 is returned.
-- Information about File Systems

printerTable OBJECT-TYPE
    SYNTAX SEQUENCE OF printerEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The (conceptual) table of printers local to the
           host. A correspondent entry exists in the
           deviceTable with type devPrinter."
    ::= { printer 1 }

printerEntry OBJECT-TYPE
    SYNTAX PrinterEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "A (conceptual) entry for one printer local to the
           host.  The hrDeviceIndex in the index represents
           the entry in the hrDeviceTable that corresponds to
           the hrPrinterEntry."
    INDEX { hrDeviceIndex }
    ::= { printerTable 1 }

printerEntry ::= SEQUENCE {
        printerIndex    INTEGER,
        printerName     DisplayString,
        printerLocation DisplayString,
        printerOpStatus INTEGER,
        printerUsStatus INTEGER,
        printerAdStatus INTEGER,
        printerAvStatus INTEGER,
        printerAction   INTEGER,
        printerError    INTEGER,
        printerQueue    SEQUENCE OF printerQueueEntry
        printerQueueStatus INTEGER
    }
```

```
printerINDEX OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "An unique value for each known printer."
   ::={ printerEntry 1 }

printerName OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "An unique name for each known printer. It's a good
           idea, to choose a name containing manufacturer or/and
           the type of the printer."
   ::={ printerEntry 2 }

printerLocation OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
           "A String describing the pysical location of the
           printer. Perhaps you might wish to add also a
           person that is responsible to this printer."
   ::={ printerEntry 3 }

printerOpStatus OBJECT-TYPE
   SYNTAX INTEGER {
          enabled(1),
          disabled(2)
   }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "Enabled means that the printer is available for the
           users. Desabled means that the printer is not avail-
           able for users."
   ::={ printerEntry 4 }

printerUsStatus OBJECT-TYPE
   SYNTAX INTEGER {
          idle(1),
          busy(2)
   }
   MAX-ACCESS read-only
   STATUS     current
```

```
   DESCRIPTION
           "Idle means that the printer is currently not printing.
           Busy means that the printer is currently printing. The
           number of queued requests does NOT influence the Usage
           State."
   ::={ printerEntry 5 }

printerAdStatus OBJECT-TYPE
   SYNTAX INTEGER {
          unlocked(1),
          locked(2),
          shuttingdown(6)
   }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The printer's administrative state. Unlocked means that
           printing is allowed, shutting down means that new
           printing requests will be discarded and locked means that
           the printer is not available to any user."
   ::={ printerEntry 6 }

printerAvStatus OBJECT-TYPE
   SYNTAX INTEGER {
          inTest(1),
          failed(2),
          powerOff(3),
          offLine(4),
          notInstalled(5)
   }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The printer's availability status. inTest means that the
           ressource is undergoing a test procedure. The administrative
           state is locked. failed means that an internal fault has
           occured, poweroff means that the printer is not powered on,
           offLine means that the the printer is not online and not
           installed means that either the printer has no connection
           to the server and/or is physical gone."
   ::={ printerEntry 7 }

printerAction OBJECT-TYPE
   SYNTAX INTEGER {
          enable(1),
          disable(2),
          lock(3),
          unlock(4),
```

```
        shuttingdown(5),
        test(6),
        online(7)
   }
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
           "The actions defined for systems management. 1 to 5 are
           self explaining, test performs a testing routine and
           online sets the printer online."
   ::={ printerEntry 8 }

printerError OBJECT-TYPE
   SYNTAX     OCTET STRING
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "This object represents any error conditions
           detected by the printer.  The error conditions are
           encoded as bits in an octet string, with the
           following definitions:

                Condition         Bit #

                lowPaper          0
                noPaper           1
                lowToner          2
                noToner           3
                doorOpen          4
                jammed            5
                offline           6
                other             7

           Bits are numbered starting with the most
           significant bit of the first byte being bit 0, the
           least significant bit of the first byte being bit
           7, the most significant bit of the second byte
           being bit 8, and so on.  A one bit encodes that
           the condition was detected, while a zero bit
           encodes that the condition was not detected."
   ::= { printerEntry 9 }

printerQueueEntry OBJECT-TYPE
   SYNTAX     PrinterQueueEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
           "The Queue for this printer."
```

```
   INDEX {printerQueueIndex}
   ::={ printerEntry 10 }

PrinterQueueEntry ::= SEQUENCE {
   printerQueueIndex   Integer,
   printerQueueUser    DisplayString,
   printerQueueEntered DateAndTime,
   printerQueuePrio    INTEGER
   printerQueueSize    KBytes,
   printerQueuePrinted KBytes,
   printerQueueAction  INTEGER,
   printerQueueStatus  INTEGER
}

printerQueueINDEX OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "An unique value for each job."
   ::={ printerQueueEntry 1 }

printerQueueUser OBJECT-TYPE
   SYNTAX      DisplayString
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The job‘s owner."
   ::={ printerQueueEntry 2 }

printerQueueEntered OBJECT-TYPE
   SYNTAX     DateAndTime
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The Date and Time the job entered the queue."
   ::={ printerQueueEntry 3 }

printerQueuePrio OBJECT-TYPE
   SYNTAX      INTEGER
   MAX-ACCESS  read-write
   STATUS      current
   DESCRIPTION
           "The priority of the job. The systems manager
           is allowed to change the priority, writing a
           value to this field. If a system does not support
           priorities on print jobs nothing will happen."
   ::={ printerQueueEntry 4 }
```

```
printerQueueSize OBJECT-TYPE
   SYNTAX     KBytes
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The size of the job in kilobytes."
   ::={ printerQueueEntry 5 }

printerQueuePrinted OBJECT-TYPE
   SYNTAX     KBytes
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "If the job is currently printed, who many
           kilobytes have been printed yet ."
   ::={ printerQueueEntry 6 }

printerQueueAction OBJECT-TYPE
   SYNTAX     INTEGER {
              accept(1),
              reject(2),
              cancel(3),
              hold(4),
              resume(5)
   }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The actions defined for systems management."
   ::={ printerQueueEntry 7 }

printerQueueStatus OBJECT-TYPE
   SYNTAX     INTEGER {
              enabled(1),
              disabled(2),
              waiting(3),
              running(4)
   }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The queue's status. Enabled means that the operational
           state is enabled, disabled means that the operational
           state is disabled, waiting means that the usage state is
           idle and running means that the usage state is active."
   ::={ printerEntry 10 }
```

```
-- The UNIX-LRZ DISKs Group
-- Implementation of the DISKs Group is mandatory for all
-- systems. If an agent is not configured to have a value for
-- any of these variables, a string of length 0 is returned.

diskTable OBJECT-TYPE
   SYNTAX     SEQUENCE OF diskEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
           "A (conceptual) table of hard disks contained
            by the host."
   ::={ device 3 }

diskEntry OBJECT-TYPE
   SYNTAX     DiskEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
           "An entry for one hard disk"
   INDEX { diskIndex }
   ::={ diskTable 1 }

DiskEntry ::= SEQUENCE {
   diskIndex     INTEGER,
   diskAccess    INTEGER,
   diskMedia     INTEGER,
   diskRemoveble Boolean,
   diskCapacity  KBytes,
   diskRs        INTEGER,
   diskWs        INTEGER,
   diskPu        INTEGER,
   diskState     INTEGER,
   diskLabel     OCTET STRING
}

diskIndex OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "An unique value for each drive."
   ::={ diskEntry 1 }

diskAccess OBJECT-TYPE
   SYNTAX     INTEGER {
              read-write(1),
```

```
                read-only(2)
     }
     MAX-ACCESS read-only
     STATUS     current
     DESCRIPTION
             "The information about rights to write
             information to the disk"
     ::={ diskEntry 2 }

  diskMedia OBJECT-TYPE
     SYNTAX     INTEGER {
unknown(1),
                hardDisk(2),
                opticalDisks(3)
     }
     MAX-ACCESS read-only
     STATUS     current
     DESCRIPTION
             "The type of the disk."
     ::={ diskEntry 3 }

  diskRemoveable OBJECT-TYPE
     SYNTAX     BOOLEAN
     MAX-ACCESS read-only
     STATUS     current
     DESCRIPTION
             "Indicates wether the drive is removeable,
     e.g. Bernoulli."
     ::={ diskEntry 4 }

  diskCapacity OBJECT-TYPE
     SYNTAX     KBytes
     MAX-ACCESS read-only
     STATUS     current
     DESCRIPTION
             "The total size of this drive."
     ::={ diskEntry 5 }

     diskRs OBJECT-TYPE
     SYNTAX     INTEGER
     MAX-ACCESS read-only
     STATUS     current
     DESCRIPTION
             "Disk reads per second."
     ::={ diskEntry 6 }

     diskWs OBJECT-TYPE
     SYNTAX     INTEGER
```

```
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "Disk writes per second."
   ::={ diskEntry 7 }

   diskPu     OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "Percentage Disk usage."
   ::={ diskEntry 8 }

   diskState OBJECT-TYPE
   SYNTAX     INTEGER {
              enabled(1),
              disabled(2),
              }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "Percentage Disk usage."
   ::={ diskEntry 9 }

   diskLabel OBJECT-TYPE
   SYNTAX     OCTET STRING
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "Disk label is not only a nice name but it stores
           controller, geometry and partitionsss."
   ::={ diskEntry 10 }

-- The UNIX-LRZ PARTITION Group
-- Implementation of the PARTITION Group is mandatory for all
-- systems. If an agent is not configured to have a value for
-- any of these variables, a string of length 0 is returned.
-- Information about File Systems

partitionTable OBJECT-TYPE
   SYNTAX     SEQUENCE OF partitionEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
           "A (conceptual) table of all filesystems contained
            by the host."
   ::={ device 4 }
```

```
partitionEntry OBJECT-TYPE
   SYNTAX     PartitionEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
           "An entry for one partition"
   INDEX { partitionIndex }
   ::={ partitionTable 1 }

PartitionEntry ::= SEQUENCE {
   partitionIndex     INTEGER,
   partitionLabel     DisplayString,
   partitionID        OCTET STRING,
   partitionSize      KBytes,
   partitionFSIndex   INTEGER,
   partitionDiskIndex INTEGER
}

partitionIndex OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "An unique value for each partition in this table."
   ::={ partitionEntry 1 }

partitionLabel OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "A textual description of this partition."
   ::={ partitionEntry 2 }

partitionID OBJECT-TYPE
   SYNTAX     OCTET STRING
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The descriptor representing this partition.
           to the operating system."
   ::={ partitionEntry 3 }

partitionSize OBJECT-TYPE
   SYNTAX     KBytes
   MAX-ACCESS read-only
   STATUS     current
```

```
   DESCRIPTION
           "The size of the partition."
   ::={ partitionEntry 4 }

partitionFSIndex OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The pointer to the filesystemtable the
            partition belongs to."
   ::={ partitionEntry 5 }

partitionDiskIndex OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The pointer to the disk in the diskTable, the
           partition resides on."
   ::={ partitionEntry 6 }


-- The UNIX-LRZ FILESYSTEM Group
-- Implementation of the FILESYSTEM Group is mandatory for all
-- systems. If an agent is not configured to have a value for
-- any of these variables, a string of length 0 is returned.
-- Information about File Systems

fsTypes          OBJECT IDENTIFIER ::= { device 100 }
FSOther          OBJECT IDENTIFIER ::= { fsTypes 1 }
FSUnknown        OBJECT IDENTIFIER ::= { fsTypes 2 }
FSBerkelyFFS     OBJECT IDENTIFIER ::= { fsTypes 3 }
FSSys5FS         OBJECT IDENTIFIER ::= { fsTypes 4 }
FSFat            OBJECT IDENTIFIER ::= { fsTypes 5 }  -- DOS
FSHPFS           OBJECT IDENTIFIER ::= { fsTypes 6 }  -- OS/2
FSHFS            OBJECT IDENTIFIER ::= { fsTypes 7 }  -- MAC
FSMFS            OBJECT IDENTIFIER ::= { fsTypes 8 }  -- MAC
FSVNode          OBJECT IDENTIFIER ::= { fsTypes 9 }
FSNTFS           OBJECT IDENTIFIER ::= { fsTypes 10 }
FSJournaled      OBJECT IDENTIFIER ::= { fsTypes 11 }
FSiso9660        OBJECT IDENTIFIER ::= { fsTypes 12 } -- CD
FSNFS            OBJECT IDENTIFIER ::= { fsTypes 13 } -- CD
Netware          OBJECT IDENTIFIER ::= { fsTypes 14 }
FSAFS            OBJECT IDENTIFIER ::= { fsTypes 15 }
FSDFS            OBJECT IDENTIFIER ::= { fsTypes 16 } -- DFS (OSF)
FSRFS            OBJECT IDENTIFIER ::= { fsTypes 17 } --
FSBFS            OBJECT IDENTIFIER ::= { fsTypes 18 } -- SVR4 Boot FS
```

```
FSUFS            OBJECT IDENTIFIER ::= { fsTypes 19 }

fsTable OBJECT-TYPE
   SYNTAX     SEQUENCE OF fsTableEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
           "The file system table"
   ::= { filesystem 1 }

fsTableEntry OBJECT-TYPE
   SYNTAX     FsEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
          "A row in the file system table"
   INDEX { fsIdentifier }
   ::= { fsTable 1 }

fsEntry ::= SEQUENCE {
   fsIdentifier      INTEGER,
   fsType            OBJECT IDENTIFIER,
   fsName            DisplayString,
   fsMountPoint      DisplayString,
   fsMountType       OBJECT IDENTIFIER,
   fsMountOptions    DisplayString,
   fsBlockSize       KBytes,
   fsBlockCount      INTEGER,
   fsBlocksFree      INTEGER,
   fsBlocksAvailable INTEGER,
   fsInodeCount      INTEGER,
   fsInodesAvailable INTEGER,
   fsQuota           Boolean,
   fsAction          INTEGER,
   fsStatus          INTEGER,
   fsBackup          SEQUENCE OF backUpEntry
}

fsIdentifier OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
          "The descriptor representing this partition.
          to the operating system."
   ::= { fsEntry 1 }

fsType OBJECT-TYPE
```

```
   SYNTAX     OBJECT IDENTIFIER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "One of the file system types defined above."
   ::= { fsEntry 2 }

fsName OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The name of the file system."
   ::= { fsEntry 3 }

fsMountPoint OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The mount point of the file system."
   ::= { fsEntry 4 }

fsMountType OBJECT-TYPE
   SYNTAX     OBJECT IDENTIFIER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The type of the mounted file system."
   ::= { fsEntry 5 }

fsMountOptions  OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The file system specific options."
   ::= { fsEntry 6 }

fsBlockSize OBJECT-TYPE
   SYNTAX     KBytes
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The block size for this file system."
   ::= { fsEntry 7 }

fsBlockCount OBJECT-TYPE
```

```
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "Total number of block in this file system."
   ::= { fsEntry 8 }

fsBlocksFree OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The number of blocks free in the filesystem."
   ::= { fsEntry 9 }

fsBlocksAvailable OBJECT-TYPE
   SYNTAX INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The number of blocks free in the file system
           to non-privileged users."
   ::= { fsEntry 10 }

fsInodeCount OBJECT-TYPE
   SYNTAX   INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The total number of inodes on this filesystem."
   ::= { fsEntry 11 }


fsInodesAvailable OBJECT-TYPE
   SYNTAX INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The number of inodes available."
   ::= { fsEntry 12 }

fsQuota OBJECT-TYPE
   SYNTAX INTEGER {
          quotaon(1),
          quotaoff(2)
   }
   MAX-ACCESS read-write
   STATUS current
```

```
   DESCRIPTION
            "Writing the value 1 to this variable turns
            on quotas for the file sytem. If the system
            fails creating the quotafile the value will
            be 2 again and a trap will be sent to the
            manager."
    ::={ fsEntry 13 }

fsAction OBJECT-TYPE
   SYNTAX INTEGER {
          mount(1),
          unmount(2),
          fsck(3)
   }
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
          "The actions defined by systems management."
   ::={ fsEntry 14 }

   fsStatus OBJECT-TYPE
   SYNTAX INTEGER {
          enabled(1),
          disabled(2),
          unlocked(3),
          locked(4),
          shuttingdown(5)
   }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
          "The status derived from ISO10164-2."
   ::={ fsEntry 15 }

backUpEntry OBJECT-TYPE
   SYNTAX     BackUpEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
           "What is importend to tell: in this table
           exists only 11 entries (backuplevel 0 down
           to backuplevel 10."
   INDEX {backUpLevel}
   ::={ fsEntry 16 }

backUpEntry ::= SEQUENCE {
   backUpLevel INTEGER
   backUpDate  DateAndTime
```

```
   backUpAction    INTEGER
}

backUpLevel OBJECT-TYPE
   SYNTAX     INTEGER (0..10)
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The Level of the Backup; the first
           entry in the table is numbered 0 and
           defines a full backup."
   ::={ backUpEntry 1 }

backUpDate OBJECT-TYPE
   SYNTAX     DateAndTime
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The Lev
           entry in the table is numbered 0 and
           defines a full backup."
   ::={ backUpEntry 2 }

backUpAction OBJECT-TYPE
   SYNTAX     INTEGER (0..10)
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
           "The Level of backup wished to start.
           Level 0 is full backup."
   ::={ backUpEntry 3 }


-- The Process Group
-- Implementation of the Process Group is mandatory for all
-- systems. If an agent is not configured to have a value for
-- any of these variables, a string of length 0 is returned.

maxProcessNumber OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The maximum number of process contexts this
           system can support."
   ::={ process 1 }

maxProcessSize OBJECT-TYPE
```

```
   SYNTAX     KBytes
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The maximum size of memory a process can allocate
           in kilobytes."
   ::={ process 2 }

curProcessNumber OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The number of currently loaded processes on this
           system."
  ::={ process 3 }

curMaxProcessSize OBJECT-TYPE
   SYNTAX     KBytes
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The currently largest process's size of the text,
           data and stack segments in kilobytes on this
           system."
   ::={ process 4 }

curMaxProcessTime OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The largest process's cpu-time in seconds on
           this system (user and system time calculated)."
   ::={ process 5 }

processTable OBJECT-TYPE
   SYNTAX     SEQUENCE OF ProcessEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
           "The table of processes loaded on this system."
   ::={ process 6 }

processEntry OBJECT-TYPE
   SYNTAX     ProcessEntry
   MAX-ACCESS not-accessible
   STATUS     current
```

```
   DESCRIPTION
           "An entry for each process loaded on this system."
   INDEX {PID}
   ::={ processTable 1 }

ProcessEntry ::= SEQUENCE {
   processPID     INTEGER,
   processName    DisplayString,
   processPPID    INTEGER,
   processUID     INTEGER,
   processGID     INTEGER,
   processState   DisplayString,
   processFlags   INTEGER,
   processCPUTime INTEGER,
   processSize    KBytes,
   processPri     INTEGER,
   processNice    INTEGER,
   processSignal  INTEGER
}

processPID OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The unique process ID."
   ::={ processEntry 1 }

processName OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "Command name used to invoke this process."
   ::={ processEntry 2 }

processPPID OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The process ID of the process's parent."
   ::={ processEntry 3 }

processUID OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
```

```
   DESCRIPTION
           "The process owner's User ID."
   ::={ processEntry 4 }

processGID OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The process owner's group ID."
   ::={ processEntry 5 }

processState OBJECT-TYPE
   SYNTAX     INTEGER {
              SSLEEP(1),
              SRUN(2),
              RUN(3),
              SIDL(4),
              SZOMB(5),
              SSTOP(6)
   }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The process's state. SSLEEP means that the process
           is awainting an event, SRUN says that the process
           is runnable, RUN says that the process is
           currently running, SIDL is an intermediate state in
           process creation, SZOMB is an intermediate state in
           process termination and SSTOP says that the process
           is stopped."
   ::={ processEntry 6 }

processFlags OBJECT-TYPE
   SYNTAX     INTEGER {
              unknown(1)
              SSYS(2),
              SLOAD(3),
              SULOCK(4),
              SPHYSIO(5),
              SSWAP(6)
   }
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The process flags. SSYS means that the process
           was created by system, SLOAD means that the process
           is loaded in main memory, SULOCK means that the
```

```
        user requested that the process is not to be swapped,
        SPHYSIO means that the process is doing physical I/O,
        and SSWAP says that the process is being swapped out.
        All other flags are unknown."
   ::={ processEntry 7 }

processCPUTime OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
        "The sum of user and system time in seconds
        the process has consumed."
   ::={ processEntry 8 }

processSize OBJECT-TYPE
   SYNTAX     KBytes
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
        "The amount of memory allocated to the process
        including text, data and stack segment."
   ::={ processEntry 9 }

processPri OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
        "The current priority of this process."
   ::={ processEntry 10 }

processNice OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
        "Setting this value will cause a renice of
        the process to this value."
   ::={ processEntry 11 }

processSignal OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
        "Post the value of this column to the process as signal;
        after posting the value is 0 again."
```

```
   DEFVAL { 0 }
   ::={ processEntry 12 }

loadAvg1 OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The average load over the last 1 minute on this
           system. The load is defined as number of processes,
           that are ready for running."
   ::={ process 7 }

loadAvg5 OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "The average load over the last 5 minutes on this
           system. The load is defined as number of processes,
           that are ready for running."
   ::={ process 8 }

loadAvg15 OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
          "The average load over the last 15 minutes on this
          system. Th load is defined as number of processes,
          that are ready for running."
   ::={ process 9 }


--
-- USER Group (containing information about user accounts, about
--             group accounts and about logged users.)
-- Implementation of the USER Group is mandatory for all
-- systems. If an agent is not configured to have a value for
-- any of these variables, a string of length 0 is returned.

userTable OBJECT-TYPE
   SYNTAX     SEQUENCE OF UserEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
           "The users table"
   ::={ user 1 }
```

```
userEntry OBJECT-TYPE
   SYNTAX     UserEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
           "An entry in the userTable"
   INDEX  { userID, userLogin }
   ::={ userTable 1 }

UserEntry ::= SEQUENCE {
   userLogin      DisplayString,
   userPasswd     DisplayString,
   userID         INTEGER,
   userGroupID    INTEGER,
   userComment    DisplayString,
   userHome       DisplayString,
   userShell      DisplayString,
   userQuotaSoft  KBytes,
   userQuotaUsed  KBytes,
   userInodeSoft  INTEGER,
   userInodeUsed  INTEGER,
   userOffice     DisplayString,
   userTelephone  DisplayString,
   userFullName   DisplayString,
   userPasswdMin  INTEGER,
   userPasswdMax  INTEGER,
   userPasswdWarn INTEGER,
   userStatus     RowStatus
   }

userLogin OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
           "This field contains the user‘s login name."
   ::={ userEntry 1 }

userPasswd OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
           "This field contains the user‘s encrypted password."
   ::={ userEntry 2 }

userID OBJECT-TYPE
```

```
   SYNTAX     INTEGER
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
           "This field contains a user identification number (UID)
           that identifies the account by number to the operating
           system."
   ::={ userEntry 3 }

userGroupID OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
          "This field contains a group identification number (GID)
          that identifies the user‘s promary group.."
   ::={ userEntry 4 }

userComment OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-create
   STATUS     deprecated
   DESCRIPTION
           "This field was originally used to hold the login information
           needet to submit batch jobs to a mainframe running GECOS from
           Bell Labs. This field is still part of the file passwd in most
           UNIX-derivats. Therefore, if no GECOS mainframe is available
           just record personal info for each user."
   ::={ userEntry 5 }

userHome OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
           "This field identifies the user‘s home directory.
           Editing this field does NOT create the home directory!"
   ::={ userEntry 6 }

userShell OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
           "This field identifies the login shell to initially
           provide for the user when he logs in."
   DEFVAL {/bin/sh}
   ::={ userEntry 7 }
```

```
userQuotaSoft OBJECT-TYPE
   SYNTAX     KBytes
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
           "This field provides the Quota specified for the user.
           A Value of 0 means that no quota is activated for this
           user."
   DEFVAL { 0 }
   ::={ userEntry 8 }

userQuotaUsed OBJECT-TYPE
   SYNTAX     KBytes
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "This value shows the user's used disk space. Logging
           this value might be useful to detect reasons for ''disk full''
           warnings or to detect strange user behavior."
   ::={ userEntry 9 }

userInodeSoft OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
           "This field provides the inode Quota specified for the user.
           A Value of 0 means that no quota is activated for this
           user."
   DEFVAL { 0 }
   ::={ userEntry 10 }

userInodeUsed OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
           "This value shows the user's used inodes. Logging
           this value might be useful to detect reasons for ''disk full''
           warnings or to detect strange user behavior."
   ::={ userEntry 11 }

userOffice OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
```

```
        "Optional information about the user's office."
   ::={ userEntry 12 }

userTelephone OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
        "Optional information about the user's thelephonenumber."
   ::={ userEntry 13 }

userFullName OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
        "The user's full name."
   ::={ userEntry 14 }

userPasswdMin OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
        "The minimum number of days between password
        changes."
   DEFVAL { 0 }
   ::={ userEntry 15 }

userPasswdMax OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
        "The maximum number of days a password is valid.
         A Value of 0 means the password is always valid."
   DEFVAL { 0 }
   ::={ userEntry 16 }

userPasswdWarn OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
        "The number of days before a user is warned to change
        his password. A Value of 0 means that the user will
        not be warned. This Value shouldn't be 0, if userPasswdMax
        is activated."
```

```
   DEFVAL { 0 }
   ::={ userEntry 17 }

userStatus OBJECT-TYPE
   SYNTAX     RowStatus
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
           "The status column used for creating, modifying,
           and deleting user-accounts on the system. This object
           may not be set to 'active' unil the user's home directory
           is created and valid values for userID, userLogin and userGID
           and userHome are given."
   ::={ userEntry 18 }

groupTable OBJECT-TYPE
   SYNTAX     SEQUENCE OF GroupEntry
   MAX-ACCESS not-accessible
   STATUS     mandatory
   DESCRIPTION
           "The group table."
   ::= { user 2 }

groupEntry OBJECT-TYPE
   SYNTAX     GroupEntry
   MAX-ACCESS not-accessible
   STATUS     mandatory
   DESCRIPTION
           "An entry in the group table."
   INDEX   { groupID }
   ::= { groupTable 1 }

GroupEntry ::= SEQUENCE {
   groupName   DisplayString,
   groupPasswd DisplayString,
   groupID     Integer,
   groupUsers  SEQUENCE OF groupUsersEntry,
   groupStatus RowStatus
}

groupName OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-create
   STATUS     mandatory
     DESCRIPTION
             "The name of this group."
     ::= { groupEntry 1 }
```

```
groupPasswd OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-create
   STATUS     deprecated
   DESCRIPTION
           "The encrypted password for this group. In
           most UNIX-Versions it is not used anymore."
   ::= { groupEntry 2 }

groupID OBJECT-TYPE
   SYNTAX     Integer
   MAX-ACCESS read-create
   STATUS     mandatory
   DESCRIPTION
           "The group's ID in the system. It has to
           be unique."
   ::= { groupEntry 3 }

groupUsersEntry OBJECT-TYPE
   SYNTAX GroupUsersEntry
   MAX-ACCESS not-accessible
   STATUS current
   DESCRIPTION
           "All users belonging to this group."
   INDEX {UID}
   ::={ groupEntry 4 }

GroupUsersEntry ::= SEQUENCE {
   userName DisplayString
   }

groupUserName OBJECT-TYPE
   SYNTAX     DisplayString
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
           "The user's login."
   ::= { groupUsersEntry 1 }

groupStatus OBJECT-TYPE
   SYNTAX     RowStatus
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
           "The status of this group. This object
           may not be set to 'active' unil a valid groupID
           and a valid name are given."
   ::= { groupEntry 5 }
```

```
whoTable OBJECT-TYPE
   SYNTAX SEQUENCE OF WhoEntry
   MAX-ACCESS not-accessible
   STATUS current
   DESCRIPTION
           "This table represents who is currently logged
           into the node which the agent resides"
   ::= { user 3 }

whoEntry OBJECT-TYPE
   SYNTAX WhoEntry
   MAX-ACCESS not-accessible
   STATUS current
   DESCRIPTION
           "An entry in the who table"
   INDEX { whoIndex }
   ::= { whoTable 1 }

WhoEntry ::= SEQUENCE {
   whoIndex DisplayString,
   whoName DisplayString,
   whoDevice DisplayString,
   whoWhere DsiplayString,
   whoTime DateAndTime,
}

whoIndex OBJECT-TYPE
   SYNTAX INTEGER
   MAX-ACCESS not-accessible
   STATUS current
   DESCRIPTION
        "The instance of the whoEntry object."
   ::= { whoEntry 1 }

whoName OBJECT-TYPE
   SYNTAX DisplayString
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
           "The name of a user logged into this system."
   ::= { whoEntry 2 }

whoDevice OBJECT-TYPE
   SYNTAX DisplayString
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
```

```
            "The device the user is using to log into
            this system."
    ::= { whoEntry 3 }

   whoWhere OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
            "The host from which the user is logged into
            this system. '':0.0'' instead of a hostname
            means that the user is using the terminal to log in
            on the system."
    ::= { whoEntry 4 }

   whoTime OBJECT-TYPE
    SYNTAX DateAndTime
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
           "The time when the user first logged in."
    ::= { whoEntry 5 }

END
```

Anhang D

Implementierungshinweise

Während der Schlußphase dieser Diplomarbeit wurde bereits im Rahmen eines Fortgeschrittenenpraktikums damit begonnen, die definierte Managementinformation zu implementieren. Dafür wurde der SNMPv2 Agent aus der CMU SNMPv2 Distribution, Version 2.1.2 gewählt. Trotz der Top-Down-Vorgehensweise ist es Aufgabe der Diplomarbeit sicherzustellen, daß die geforderte Information auch in angemessener Weise vom System gewonnen werden kann. Deshalb finden sich nachfolgend zu allen definierten Variablen Vorschläge, wie die Werte einem Systemmanagement zur Verfügung gestellt werden können. Alle Hinweise beziehen sich auf das Betriebssystem SunOS 4.1.2 und 4.1.3, die definierte Managementinformation ist jedoch oftmals auf sehr ähnlicher Weise bei anderen UNIX-Derivaten zu erhalten.

D.1 Ebenen der Informationsgewinnung

Damit der Agent auf dem jeweiligen zu managenden System die in der MIB geforderte Information zur Verfügung stellen kann, müssen die betreffenden Funktionen in einer Programmiersprache implementiert sein. Nachfolgend werden die wichtigsten Arten der Informationsgewinnung in der Sprache C beschrieben.

C-Library-Funktion

Die Library-Funktionen sind die kompatibelste Art der Informationsgewinnung über Systemgrenzen hinweg. Soweit möglich sollte man sich auf sie abstützen, da sie für viele UNIX-Systeme implementiert sind.

System Call

Die System Calls sind bereits sehr systemnahe Funktionen. Ihre Verwendung wird bereits in den Man-Pages von SunOS als “not recommended“ bezeichnet, da ihr Verhalten oftmals nicht vorhersehbar ist. Allerdings kann durch sie oftmals detaillierte systemmanagementrelevante Information gewonnen werden.

Kernel-Zugriff

Der Kernel-Zugriff stellt die tiefste Ebene und gleichzeitig die am wenigsten genormte Art der Informationsgewinnung dar. Von den Workstationherstellern HP und SUN wurde während der Diplomarbeit vergeblich versucht über den Kernel und seine Variablen nähere Spezifikationen und Dokumentationen zu bekommen. Deshalb muß auf der Suche nach Systemwerten als erstes die Symbol-Tabelle des Kernels ausgelesen werden und mittels memotechnischen Assoziationen der Informationsträger erkannt werden. Unglücklicherweise können sich Teile des Kernels und damit auch seine Variablen bei Versions- oder Releasewechseln ändern, so daß die Information neu implementiert werden muß.

Nachfolgend wird exemplarisch der Zugriff auf Information aus dem SunOS-Kernel gezeigt, indem der Systemname ausgelesen wird. Auf eine detaillierte Fehlerbehandlung wurde, um das Beispiel kompakt zu halten, verzichtet. Hierzu sind die Manualseiten des jeweiligen UNIX-Systems zu lesen.

```
#include <stdio.h>

/* Kernel C-Header-Files */
#include <kvm.h>
#include <fcntl.h>
#include <nlist.h>
#include <sys/param.h>
#include <sys/user.h>
#include <sys/proc.h>

/* wish list for kernel symbols */
struct nlist nlst[] = {
          {"_hostnamelen"},
#define X_HOSTNAMELEN    0
          {"_hostname"},
          NULL
     };

main()
{
   kvm_t *kd;                      /* pointer to kernel           */
   long result;                    /* result of functions         */
   unsigned long hostnamelen_offset;  /* offsets                  */
   unsigned long hostname_offset;
   char *hostname;                 /* hostname                    */
   int  *hostnamelen;              /* Laenge des Hostnamen        */

   hostnamelen=(int*)malloc(sizeof(int));

   /* open the kernel */
   (kd = kvm_open(NULL, NULL, NULL, O_RDONLY, "example"));
```

```
   /* get the list of symbols we want to access in the kernel */
   result=kvm_nlist(kd, nlst);

   /* get the length of the hostname */
   hostnamelen_offset = nlst[X_HOSTNAMELEN].n_value;
   result= kvm_read(kd, hostnamelen_offset, (int*)hostnamelen,
    sizeof(hostnamelen));

   /* now get the hostname */
   hostname=(char*)malloc(sizeof(*hostnamelen));
   hostname_offset = nlst[X_HOSTNAME].n_value;
   result=kvm_read(kd, hostname_offset, (int*)hostname, (*hostnamelen));

   fprintf(stderr, " Hostname:    %s \n", hostname);

   /* close kernel */
   result=kvm_close(kd);
}
```

Der Hostname kann aber auch mittels der C-Library-Funktion gethostname() oder dem System Call syscall(SYS_gethostname) erhalten werden. Selbstverständlich ist eine Implementierung mittels C-Library-Funktion vorzuziehen.

UNIX-Kommandos

Eine weitere Möglichkeit Information zu erhalten sind UNIX-Kommandos. Diese sollten jedoch nur verwendet werden, wenn die Information auf keine andere Weise erhalten werden kann, da sie sehr zeitaufwendig sind (z.B. Weiterverarbeitung der Ausgabe nötig).

Konfigurationsfiles

Konfigurationfiles stellen eine reichhaltige Quelle an systemmanagementrelevanter Information dar, jedoch ist das Parsen von Textdateien zeitaufwendig. Deshalb werden soweit möglich C-Funktionen bevorzugt.

D.2 UNIX-LRZ-MIB

D.2.1 System Group

- sysName: Libraryfunktionen getdomainname() und gethostname().
- sysOS: Shell-Funktion uname -r.

- sysHardware: Das Kommando /usr/etc/devinfo liefert als erste Ausgabe den genauen Systemtyp. Die Information ist aber beispielsweise auch aus einem eventuell vorhandenen Boot-PROM (/dev/openprom) erhaltbar.
- Uptime: Die Kernelvariable _boottime enthält den Bootzeitpunkt. Durch Subtraktion der aktuellen Zeit (gettimeofday()) vom Bootzeitpunkt erhält man die Uptime.
- Date: Um Zeit und Datum zu erhalten sollte der System Call gettimeofday() verwendet werden. Die Zeit kann mittels adjtime() gesetzt werden. Der System Call settimeofday() sollte zur Korrektur der Zeit nicht verwendet werden, um Inkonsistenzen im Dateiensystem zu vermeiden, da er im Gegensatz zu adjtime() die Anpassung nicht in kleinen Schritten vornimmt.
- sysUsers: Zähle alle Einträge in der Who-Tabelle.

D.2.2 Storage Group

- RAM: Die Kernelvariable _physmem liefert die Anzahl der physischen Pages und wird multipliziert mit Größe einer Page (System Call getpagesize()). Das Ergebnis ist der Arbeitsspeicher in Bytes und muß noch durch 1024 geteilt werden um Kilobyte für die MIB zu erhalten.
- VirtualMemory: Die Kernelvariable _anoninfo liefert Werte der Struktur struct anoninfo. Die Werte von ani_free und ani_resv sind zu addieren und mit der Größe einer Page (System Call getpagesize()) zu multiplizieren. Um wie in der MIB gefordert, Kilobytes zu erhalten, muß der Wert noch durch 1024 geteilt werden.
- FLCache, SLCache: Obwohl für die MIB definiert und modelliert, ist momentan nicht bekannt ob diese Werte von SUN-Maschinen ausgelesen werden können. Eine Möglichkeit bietet u. U. das Boot-PROM.

D.2.3 Device Group

Die Implementierung dieser Gruppe erfordert sehr systemnahe Programmierung. Für die meisten Maschinentypen existiert Public-Domain-Software, die Informationen zu dieser Gruppe aus Kernel, Boot-Prom oder Dateien ausliest. Da diese Programme im Quelltext verfügbar sind, geben sie ein gute Anleitung zur Implementierung. Stellvertretend sei das Programm SYSINFO von Michael A. Copper erwähnt, das viel Deviceinformation über SUN-Maschinen liefert. Das Programm ist mit anonymous ftp von dem FTP-Server *usc.edu* (Verzeichnis /pub/sysinfo) erhältlich. Eine weitere Möglichkeit würde das Kommando /usr/etc/devinfo (SunOS 4.1) darstellen. Die Ausgaben dieses Kommandos wären passend weiterzuverarbeiten. Alle Blockdevices haben einen Eintrag in der Kernelstruktur _bdevsw. In dieser finden sich neben Plattenlaufwerken auch Bandlaufwerke. Für die Implementierung von Bandlaufwerken muß die Struktur cdevsw aus dem Includefile /usr/include/sys/conf.h verwendet werden. Nähere Informationen zur Implementierung von Plattenlaufwerken finden sich in Kapitel D.2.6.

D.2.4 Processor Group

- cpuType: Kernelvariable _cpu0
- cpuClockRate: Kernelvariable _hz
- cpuStat, cpuUsStat, cpuAdStat, cpuAction: Diese Werte sind in Abhängigkeit des jeweiligen Systems zu implementieren (vgl. Kapitel 4.2.2). Für Einprozessormaschinen müssen diese Werte nicht implementiert werden.
- cpuUsrTime, cpuSystemTime, cpuIdleTime, cpuNiceTime: Genaue Definitionen finden sich im Includefile /usr/include/sys/dk.h. Zur Berechnung der Prozentanteile müssen drei Arrays deklariert werden und der Wert des Kernel-Arrays _cp_time eingelesen werden. Dieser Vorgang ist eine bestimmte Zeit später zu wiederholen (0.5 Sekunde), und danach sind die einzelnen Prozentwerte wie folgt zu berechnen:

```
total_change = 0;
for (i = 0; i < CPUSTATES; i++) {
   /* calculate changes for each state and overall change */
   if (cp_time[i] < cp_old[i]) {
      /* this only happens when the counter wraps */
      change = (int)
      ((unsigned long)cp_time[i]-(unsigned long)cp_old[i]);
   }
   else {
      change = cp_time[i] - cp_old[i];
   }
   total_change += (cp_change[i] = change);
   cp_old[i] = cp_time[i];
}
for (i = 0; i < CPUSTATES; i++) {
   percent = ((double)changes[i] / (double)total) * 100.0;
}
```

- cpuSpecInt, cpuSpecIntYear, cpuSpecFp, cpuSpecFpYear: Die aktuellsten Ergebnisse der SPECmarks können via anonymous ftp vom Ftp-Server *ftp.cdf.toronto.edu* geholt werden. Das File *spectable* liegt im Verzeichnis /pub. Es ist ein Textfile und muß durch entsprechende Umformatierung (alle Zeilen gleiche Länge) für die Verarbeitung durch den Agenten vorbereitet werden. Anhand des CPU-Typs, der Taktfrequenz und der Maschinenbezeichnung ist eine eindeutige Identifikation im File möglich und die Resultate können zur Verfügung gestellt werden.

D.2.5 Printer Group

Als Hinweise zur Implementierung werden die zur Verfügung stehenden UNIX-Kommandos von SunOS erläutert. Es wird Aufgabe des Fortgeschrittenenpraktikums sein, eine geeignete API oder Libraryfunktionen zur Implementierung zu suchen.

- Drucker aktivieren: enable
- Druckauftrag stornieren: cancel
- Übersicht über den Druckservice: lpstat
- Drucker deaktivieren: disable
- Druckauftragseingang ermöglichen: accept
- Druckauftragseingang sperren: reject
- Konfiguration eines Druckers: lpadmin

D.2.6 Disk Group

- diskAccess, diskType, diskRemoveable, Wichtig für die Implementierung ist die Kernel-Variable _bdevsw (vgl. D.2.3). Über die im Includefile /usr/include/sys/conf.h sind die Plattenlaufwerke für die Implementierung zugänglich.
- diskCapacity: Die Implementierung muß von jeder Platte die Zahl der Zylinder, der Köpfe und der Sektoren abfragen. Die Multiplikation dieser Werte ergibt die Anzahl der Sektoren. Da ein Sektor 512 Byte groß ist, muß um die Kapazität der Platte zu erhalten, die Sektorenzahl durch 2 geteilt werden.
- diskrs, diskws, diskpu: Die Manualseite von SunOS 4.1 gibt nützliche Hinweise über Kernelzähler.
- diskStatus: Die in der Struktur bdevsw definierte Funktion *d_open überprüft ob das Plattenlaufwerk während der Autokonfigurationsphase identifiziert wurde, sie stellt die Integrität der Platte

D.2.7 Partition Group

- partitionIndex, partitionLabel, partitionID, partitionSize: Im Kernel existiert die Variable _bdevsw. Sie ist ein Verweis auf eine Tabelle im Kernel, die Einträge auf alle Festplatten und Bandlaufwerke enthält. Die zugehörige Struktur ist im Includefile /usr/include/sys/conf.h definiert. Der Aufruf psize() liefert auch die Größe der spezifizierten Partition in Blöcken.
- partitionFsIndex, partitionDiskIndex: Es ist ein Verweis auf das zugehörige Filesystem zu erzeugen und ein Verweis auf die Disk-Tabelle zu implementieren, der genau auf den Plattenintrag zeigt, auf der sich die Partition befindet.

D.2.8 Filesystem Group

Für die Implementierung ist die Filesystem-Tabelle die wichtigste Grundlage. Sie wird von allen wichtigen UNIX-Kommandos wie dump, mount, unmount, swapon, fsck, df, etc benutzt.

- fsType, fsName : Die Information ist aus der Systemtabelle /etc/fstab erhältlich. Auf die Tablle wird mit den C-Libraryfunktionen getmntent(), setmntent(), addmntent, endmntent() und hasmntopt zugegrifffen.
- fsMountPoint, fsMountType, fsMountOptions: Die Information ist aus der Systemtabelle /etc/mtab erhältlich. Auf die Tabelle wird mit den C-Libraryfunktionen getmntent(), setmntent(), addmntent, endmntent() und hasmntopt zugegrifffen.
- fsBlockSize, fsBlockCount, fsBlocksFree, fsBlocksAvailable, fsInodeCount fsInodesAvailable: Implementierungshinweise finden sich in /usr/include/sys/ufs.
- fsQuota: Um eine Quota zu aktivieren oder zu deaktivieren muß der System Call quotaactl verwendet werden. Eine Quota läßt sich nur aktivieren, wenn das File quotaFile existiert. Existiert es nicht, muß es mit dem Kommando quotacheck oder dem System Call qtinit() erzeugt werden. Kann die Quota nicht korrekt eingerichtet werden, muß der Manager davon unterrichtet werden.
- backUpLevel, backUpDate, backUpAction: Die einfachste Implementierung wäre es, einfach das UNIX-Kommando anzustoßen, aber vielleicht geht's ja auch eleganter.

D.2.9 Process Group

Der wichtigste Informationsträger dieser Gruppe ist die Prozeßstruktur der einzelnen Prozesse. Genauere Definitionen hierzu finden sich im Includefile /usr/include/sys/proc.h, /usr/include/sys/user.h und in [Leff89], Kapitel 4. Im folgenden wird ein Zeiger pp auf eine Prozeßstruktur (*kvm_getproc()) und eine Userstruktur u (kvm_getu()) vorausgesetzt.

- maxProcessNumber: Kernelvariable _nproc
- maxProcessSize: Information hierzu findet sich im Includefile /usr/sys/systm.h und in der Kernelvariable _maxmem
- curProcessNumber: Es bestehen prinzipiell zwei Möglichkeiten die Information zu erhalten. Entweder zählt man alle Prozeßstrukturen, deren p_stat-Eintrag ungleich 0 ist, oder man benutzt die Funktion kvm_nextproc().
- curMaxProcessSize: Hier ist der maximale Wert der Spalte processSize der Prozeßtabelle zu hinterlegen. Allerdings per Definition (vgl. Kapitel 5) bleiben Systemprozesse unberücksichtigt. Systemprozesse können über ihre Flags identifiziert werden.

- curMaxProcessTime: Hier ist der größte Wert aus der Spalte processTime der Prozeßtabelle zu hinterlegen. Systemprozesse bleiben unberücksichtigt (vgl. Implementierungshinweis zu curMaxProcessSize).
- processPID: pp->p_pid
- processName: u.u_comm
- processPPID: pp->p_ppid
- processGPID: pp->p_gpid
- processUID: pp->p_suid
- processGID: pp->p_sgid
- processState: pp->p_stat
- processFlags: pp->p_flag
- processCPUTime: Das Includefile /usr/include/sys/resource.h definiert die Struktur rusage in der sich Informationen über den Ressourcenverbrauch des Prozesses finden. Die Strukturen ru_utime und ru_stime sind in geeigneter Weise zu dereferenzieren und der Wert der folgenden Variablen zu addieren: u.u_ru.ru_utime und u.u_ru.ru_stime
- processSize: Addiere die Werte des Textsegments pp->p_tsize, des Datasegments pp->p_dsize und des Stacksegments pp->p_ssize.
- processPri: pp->p_pri
- processNice: Es sind die System Calls getpriority() und setpriority() zu verwenden.
- processSignal:
  Nähere Informationen findet sich im Includefile /usr/include/sys/signal.h. Entweder ist die Library-Funktion signal() oder der System Call sigvec() zu verwenden.
- loadAvg1, loadAvg5, loadAvg15: Dieser Werte sind aus der Kernel-Variable _avenrun[] zu erhalten. Dabei ist der Skalierungsfaktor FSCALE in /usr/include/sys/param.h beachten

D.2.10 User Group

- userLogin, userPasswd, userID, userGroupID, userComment, userHome, userShell: Nähere Informationen finden sich im Includefile /usr/include/pwd.h. Zur Implementierung sind die Library-Funktionen getpwent(), setpwent(), etc. zu verwenden. Idealerweise werden nur unbenutzte UIDs vergeben.

- userQuotaSoft, userQuotaUsed, userInodesSoft, userInodesUsed: Im Includefile /usr/include/ufs/quota.h finden sich nähere Definitionen. Die Implmentierung muß über den System Call quotactl() erfolgen. Das Hardlimit muß ebenfalls gesetzt werden. Es ist per Definition um 5 MByte höher als das Softlimit.

- userOffice, userTelephone, userFullName: Diese Information muß je nach Betriebssystem entweder in einer optionalen Datenbank implementiert werden, oder es werden bereits vorgesehene Schnittstellen genutzt (vgl. SUN-NIS).

- userPasswdMin, userPasswdMax, userPasswdWarn: Vorstellbar ist eine Implementierung über das Kommando passwd.

- userStatus: Hier ist insbesondere darauf zu achten, daß erst nach Erzeugung des entsprechenden Home-Verzeichnisses und Eingabe gültiger Werte für userLogin, userGroupID userID der Eintrag auf "active" gesetzt wird.

- groupName, groupPasswd, groupID, groupUsers: Hier ist Nähere Informationen finden sich im Includefile /usr/include/grp.h. Zur Implementierung sind die Library-Funktionen getgrent(), setgrent(), etc zu verwenden. Idealerweise werden beim Neuanlegen von Gruppen bereits bestehende GIDs erkannt.

- groupStatus: Der Status darf erst nach korrekter Angabe von Name und ID auf "aktiv" gesetzt werden.

- whoName, whoDevice, whoTime, whoWhere: Diese Informationen werden vom UNIX-System im Verzeichnis /etc/utmp zur Verfügung gestellt. Sie sind aus diesem File auszulesen und wie in der MIB definiert, zur Verfügung zu stellen.

Abkürzungen

API: Application Programming Interface

ASN.1: Abstract Syntax Notation One

CISC: Complex Instruction Set Computer

CMU: Carnegie Mellon University

DCE: Distributed Computer Environment

DRAM: Dynamic Random Access Memory

DSIS: Distributed Support Information Standards

GID: Group Identifier

IAB: Internet Activities Board

ISO: International Organization for Standardization

I/O: Input/Output

NQS: Network Queuing System

MIB: Management Information Base

MIPS: Million Instruction per Second

MFLOPS: Million Floating Point Operations per Second

NFS: Network File System

OSI: Open Systems Interconnection

OSF: Open Software Foundation

PID: Process Identifier

PPID: Parent Process Identifier

RFC: Request for Comments

RFS: Remote File Sharing

RISC: Reduced Instruction Set Computer

RPC: Remote Procedure Call

SMI: Structure of Management Information

SNMP: Simple Network Management Protocol

SRAM: Static Random Access Memory

UID: User Identifier

Literaturverzeichnis

[Bode91a] Arndt Bode, *RISC-Architekturen*, BI-Wiss.-Verlag, Mannheim, Wien, Zürich, 2. Auflage, 1990, ISBN 3-411-14752-0.

[Bode91b] Arndt Bode, „Manuskript der Vorlesung Rechnerarchitektur“, 1991, München, WS 1991/92.

[Bodo89] Richard Bodo, *Mikroprozessortechnik*, Carl Hanser Verlag, München, Wien, 1989, ISBN 3-446-15692-5.

[Bowen 80] B. A. Bowen und R. J. A. Buhr, *The Logical Design of Multiple-Microprocessor Systems*, Prentice Hall, New Jersey, 1980, ISBN 0-13-539908-4.

[BSD92] Marshall T. Rose, „BSD UNIX MIB“, MIB für Systemmanagement, Mai 1992.

[CMU93] „CMU UNIX MIB“, MIB für Systemmanagement, 1993.

[Coyw92] Wolfgang Coy, *Aufbau und Arbeitsweise von Rechenanlagen*, Vieweg, Wiesbaden, 2. Auflage, 1992, ISBN 3-528-14388-6.

[Dree88] Horst Drees, *UNIX: Kompendium für Anwender und Systemspeaialisten*, Markt-u.-Technik-Verlag, Haar bei München, 1988, ISBN 3-89090-494-7.

[DSIS 93] „Distributed Support Information Standards Requirements Specification“, Document PW017, Revision 1.0, Distributed Support Information Standards Group, Juni 1993.

[faer87] G. Färber, *Bussysteme. Parallele und serielle Bussysteme. Lokale Netze*, Oldenburg, München, 1987.

[Garb91] Klaus Garbe, *Management von Rechnernetzen*, Teubner, Stuttgart, 1991, ISBN 3-519-02418-7.

[Glas93] Graham Glass, *UNIX for Programmers and Users*, Prentice Hall, New Jersey, 1993, ISBN 0-13-061771-7.

[Gulb88] Jürgen Gulbins, *UNIX Version 7, bis System V.3*, Springer-Verlag, Heidelberg, 3. Auflage, 1988, ISBN 3-540-19248-3.

[Hege93] Heinz-Gerd Hegering, *Integriertes Netz- und Systemmanagement*, Addison-Wesley GmbH, Bonn, Paris, 1993, ISBN 3-89319-508-4.

[Hopk87] M. E. Hopkins, „A perspective on the 801/Reduced Instruction Set Computer", *IBM Systems Journal, 26, 1*, 1:107–121, 1987.

[HP92] „HP MIB", MIB für Systemmanagement, Juli 1992.

[ISO 10040] „Information Technology – Open Systems Interconnection – Systems Management Overview", IS 10040, ISO/IEC, September 1991.

[ISO 10164-11] „Information Technology – Open Systems Interconnection – Systems Management – Part 11: Metric Objects and Attributes", IS 10164-11, ISO/IEC, Oktober 1993.

[ISO 10164-13] „Information Technology – Open Systems Interconnection – Systems Management – Part 13: Summarization Function", DIS 10164-13, ISO/IEC, Oktober 1992.

[ISO 10164-2] „Information Technology – Open Systems Interconnection – Systems Management – Part 2: State Management Function", IS 10164-2, ISO/IEC, Juni 1993.

[ISO 10164-5] „Information Technology – Open Systems Interconnection – Systems Management – Part 5: Event Report Management Function", IS 10164-5, ISO/IEC, Juni 1993.

[ISO 10164-6] „Information Technology – Open Systems Interconnection – Systems Management – Part 6: Log Control Function", IS 10164-6, ISO/IEC, Juni 1991.

[ISO 10165] „Information Technology – Open Systems Interconnection – Structure of Management Information", IS 10165-X, ISO/IEC.

[ISO 7498] „Information Processing Systems – Open Systems Interconnection – Basic Reference Model", IS 7498, ISO/IEC, 1984.

[ISO 8649] „Information Processing Systems – Open Systems Interconnection – Service Definition for the Association Control Service Elements", IS 8649, ISO, 1988.

[ISO 9595] „Information Technology – Open Systems Interconnection – Common Management Information Service Definition", IS 9595, ISO/IEC, November 1991.

[ISO 9596] „Information Technology – Open Systems Interconnection – Common Management Information Protocol", IS 9596, ISO/IEC, November 1991.

[Jans93] R. Jansen und K. Lipinski, *SNMP: Konzepte, Verfahren, Plattformen*, Datacom, Berrgheim, 1993, ISBN 3-89238-077-5.

[Kauf92] F. J. Kauffels, *Netzwerk-Management*, Datacom, Bergheim, 2. Auflage, 1992, ISBN 3-89238-047-3.

[Krup93] B. Krupczak, „UNIX Systems Management via SNMP“, April 1993.

[Krup93a] Bobby Krupczak, „The Krupczak-Unix MIB“, MIB für Systemmanagement, April 1993.

[Leff89] Samuel J. Leffler, *The Design and Implementation of the 4.3 BSD UNIX Operating System*, Addison-Wesley Publishing Company, Reading, Massachusetts, New York, Bonn, 1989, ISBN 0-201-06196-1.

[Muel91] Christian Müller-Schloer, *RISC-Prozessoren*, Springer-Verlag, Berlin, Heidelberg, New York, 1991, ISBN 3-540-54050-4.

[Neum92] Bernhard Neumair, *Objektorientierte Modellierung von Kommunikationsressourcen für ein integriertes Performace Management*, PhD thesis, TU München, Dezember 1992.

[Neum93a] Bernhard Neumair, „Projekt Systemmanagement Arbeitspapier“, LMU, November 1993.

[OSF DCE] Open Software Foundation, *Introduction to OSF DCE*, 1992.

[Past91] David C. Pasterchik und Dileep P. Bhandarkar, *SPEC A New PerSPECtive on Comparing System Performance*, Digital, February 1991.

[RFC 1155] M. Rose und K. McCloghrie, „Structure and Identification of Management Information for TCP/IP-based Internets“, RFC 1155, IAB, Mai 1990.

[RFC 1157] J. D. Case, M. Fedor, M. L. Schoffstall und C. Davin, „Simple Network Management Protocol (SNMP)“, RFC 1157, IAB, Mai 1990.

[RFC 1213] K. McCloghrie und M. Rose, „Management Information Base for Network Management of TCP/IP-based internets: MIB-II“, RFC 1213, IAB, Maerz 1991.

[RFC 1271] S. Waldbusser, „Remote Network Monitoring Management Information Base“, RFC 1271, IAB, November 1991.

[RFC 1441] J. Case, K. McCloghrie, M. Rose und S. Waldbusser, „Introduction to version 2 of the Internet-standard Network Management Framework“, RFC 1441, IAB, April 1993.

[RFC 1442] J. Case, K. McCloghrie, M. Rose und S. Waldbusser, „Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)“, RFC 1442, IAB, April 1993.

[RFC 1443] J. Case, K. McCloghrie, M. Rose und S. Waldbusser, „Textual Conventions for version 2 of the the Simple Network Management Protocol (SNMPv2)“, RFC 1443, IAB, April 1993.

[RFC 1444] J. Case, K. McCloghrie, M. Rose und S. Waldbusser, „Conformance Statements for version 2 of the the Simple Network Management Protocol (SNMPv2)“, RFC 1444, IAB, April 1993.

[RFC 1448] J. Case, K. McCloghrie, M. Rose und S. Waldbusser, „Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)“, RFC 1448, IAB, April 1993.

[RFC 1452] J. Case, K. McCloghrie, M. Rose und S. Waldbusser, „Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework“, RFC 1452, IAB, April 1993.

[RFC 1514] P. Grillo und S. Waldbusser, „Host Resources MIB“, RFC 1514, IAB, September 1993.

[SBAA92] Schubring, Bötsch, Apostolescu und Abeck, „Anforderungsbeschreibung an das Projekt Rechnerüberwachung“, Technischer Bericht, August 1992.

[Schi93] Alexander Schill, *DCE - The OSF Distributed Computer Environment*, Springer-Verlag, 1993, ISBN 3-540-55335-5.

[Schn91] H.-J. Schneider, *Lexikon der Informatik und Datenverarbeitung*, Oldenburg, München, 3. Auflage, 1991, ISBN 3-486-21514-0.

[Sieg91] H.J. Siegert, *Betriebssysteme: Eine Einführung*, Oldenburg, München, Wien, 3. Auflage, 1991, ISBN 3-486-21930-8.

[Stil94] Andreas Stiller, „Chipsatz intern“, *ct'*, S. 213, Februar 1994.

[SUN92] „BSD UNIX MIB“, MIB für Systemmanagement, August 1992.

[Unix89] *UNIX System V/386 Release 3.2, System Administrator's Reference Manual*, Prentice Hall, New York, 1989, ISBN 0-13-877648-2.