Dateisystem-Hierarchiestandard FHS

Filesystem Hierarchy Standard (Release 2.3)
Deutsche Übersetzung (15.03.2005)

Filesystem Hierarchy Standard Group

Herausgegeben von

Rusty Russell

Daniel Quinlan

Christopher Yeoh

Dateisystem-Hierarchiestandard FHS: Filesystem Hierarchy Standard (Release 2.3) Deutsche Übersetzung (15.03.2005)

von Filesystem Hierarchy Standard Group
Herausgegeben von Rusty Russell, Daniel Quinlan und Christopher Yeoh

Veröffentlicht 28. Januar 2004

Dieses Dokument enthält eine Sammlung von Anforderungen und Richtlinien für das Plazieren von Dateien und Verzeichnissen unter UNIX-ähnlichen Betriebssystemen. Diese Richtlinien sollen die Zusammenarbeit von Anwendungsprogrammen, Systemverwaltungswerkzeugen, Entwicklungswerkzeugen und Scripten ebenso unterstützen, wie eine größere Einheitlichkeit der Dokumentation dieser Systeme.

Das dieser Übersetzung zugrundeliegende Originaldokument "Filesystem Hierarchy Standard" (Release 2.3 / Published January 28 2004) kann von *Pathname* (http://www.pathname.com/fhs/) bezogen werden.

Inhaltsverzeichnis

Anmerkungen zur Übersetzung ..... **i**

1. Einleitung **1**

1.1. Zweck 1
1.2. Benutzerhinweise 1

2. Das Dateisystem **2**

3. Das Wurzeldateisystem **3**

3.1. Zweck 3
3.2. Anforderungen 4
3.3. Optionen 4
3.4. /bin : Wesentliche Kommandos (für alle Benutzer) 5
3.4.1. Zweck 5
3.4.2. Anforderungen 5
3.4.3. Optionen 6
3.5. /boot : Statische Dateien des Laders 7
3.5.1. Zweck 7
3.5.2. Optionen 7
3.6. /dev : Gerätedateien 7
3.6.1. Zweck 8
3.6.2. Optionen 8
3.7. /etc : Rechnerspezifische Systemeinstellungen 8
3.7.1. Zweck 8
3.7.2. Anforderungen 8
3.7.3. Optionen 8
3.7.4. /etc/opt : Einstellungen für /opt 10
3.7.4.1. Zweck 10
3.7.4.2. Anforderungen 10
3.7.5. /etc/X11 : Einstellungen für das X-Window-System (optional) 10
3.7.5.1. Zweck 11
3.7.5.2. Optionen 11
3.7.6. /etc/sgml : Einstellungen für SGML (optional) 11
3.7.6.1. Zweck 11
3.7.7. /etc/xml : Einstellungen für XML (optional) 11
3.7.7.1. Zweck 11
3.8. /home : Benutzerverzeichnisse (optional) 12
3.8.1. Zweck 12
3.8.2. Anforderungen 12
3.9. /lib : Wesentliche Programmbibliotheken und Kernmodule 12
3.9.1. Zweck 12
3.9.2. Anforderungen 12
3.9.3. Optionen 12
3.10. /lib<Suffix> : Wesentliche Programmbibliotheken in alternativem Format (optional) 13
3.10.1. Zweck 13
3.10.2. Anforderungen 13
3.11. /media : Einhängepunkte für Wechselmedien 13
3.11.1. Zweck 13

3.11.2. Optionen 13
3.12. /mnt : Einhängepunkt für ein temporär eingebundenes Dateisystem 14
3.12.1. Zweck 14
3.13. /opt : Zusatzsoftware 14
3.13.1. Zweck 14
3.13.2. Anforderungen 14
3.14. /root : Benutzerverzeichnis von "root" (optional) 15
3.14.1. Zweck 16
3.15. /sbin : Wesentliche Systemkommandos 16
3.15.1. Zweck 16
3.15.2. Anforderungen 16
3.15.3. Optionen 16
3.16. /srv : Daten laufender Dienste 17
3.16.1. Zweck 17
3.17. /tmp : Temporäre Dateien 18
3.17.1. Zweck 18
4. Die /usr-Hierarchie 21
4.1. Zweck 21
4.2. Anforderungen 21
4.3. Optionen 21
4.4. /usr/X11R6 : X-Window-System Version 11 Release 6 (optional) 22
4.4.1. Zweck 22
4.4.2. Optionen 22
4.5. /usr/bin : Die meisten Benutzerkommandos 22
4.5.1. Zweck 22
4.5.2. Optionen 22
4.6. /usr/include : Verzeichnis für Include-Dateien 23
4.6.1. Zweck 23
4.6.2. Optionen 23
4.7. /usr/lib : Bibliotheken für Programmierung und Softwarepakete 24
4.7.1. Zweck 24
4.7.2. Optionen 24
4.8. /usr/lib<Suffix> : Bibliotheken in alternativem Format (optional) 24
4.8.1. Zweck 24
4.9. /usr/local : Locale Hierarchie 24
4.9.1. Zweck 25
4.9.2. Anforderungen 25
4.9.3. Optionen 25
4.9.4. /usr/local/share 26
4.10. /usr/sbin : Weniger wichtige Systemkommandos 26
4.10.1. Zweck 26
4.11. /usr/share : Architekturunabhängige Daten 26
4.11.1. Zweck 26
4.11.2. Anforderungen 26
4.11.3. Optionen 27
4.11.4. /usr/share/dict : Wortlisten (optional) 27
4.11.4.1. Zweck 27

4.11.4.2. Optionen 28
4.11.5. /usr/share/man : Online-Handbücher ("man pages") 28
4.11.5.1. Zweck 28
4.11.5.2. Optionen 29
4.11.6. /usr/share/misc : Verschiedenes 31
4.11.6.1. Optionen 31
4.11.7. /usr/share/sgml : SGML-Daten (optional) 32
4.11.7.1. Zweck 32
4.11.7.2. Optionen 32
4.11.8. /usr/share/xml : XML-Daten (optional) 32
4.11.8.1. Zweck 32
4.11.8.2. Optionen 33
4.12. /usr/src : Quellen (optional) 33
4.12.1. Zweck 33
5. Die /var Hierarchie 35
5.1. Zweck 35
5.2. Anforderungen 35
5.3. Optionen 36
5.4. /var/account : Prozesskonten (optional) 36
5.4.1. Zweck 36
5.5. /var/cache : Pufferspeicher für Anwendungen 36
5.5.1. Zweck 36
5.5.2. Optionen 37
5.5.3. /var/cache/fonts : Lokal generierte Schrifttypen (optional) 37
5.5.3.1. Zweck 37
5.5.3.2. Optionen 37
5.5.4. /var/cache/man : Lokal formatierte Handbücher (optional) 37
5.5.4.1. Zweck 37
5.6. /var/crash : Systemdumps (optional) 38
5.6.1. Zweck 38
5.7. /var/games : Variable Spieledaten (optional) 38
5.7.1. Zweck 39
5.8. /var/lib : Variable Statusinformationen 39
5.8.1. Zweck 39
5.8.2. Anforderungen 39
5.8.3. Optionen 40
5.8.4. /var/lib/<Editor> : Sicherungen und Status eines Editors (optional) 40
5.8.4.1. Zweck 40
5.8.5. /var/lib/hwclock : Status von hwclock (optional) 40
5.8.5.1. Zweck 41
5.8.6. /var/lib/misc : Verschiedene Statusinformationen 41
5.8.6.1. Zweck 41
5.9. /var/lock : Sperrdateien 41
5.9.1. Zweck 41
5.10. /var/log : Logbuchdateien und -verzeichnisse 41
5.10.1. Zweck 42
5.10.2. Optionen 42

5.11. /var/mail : Postfächer (optional) 42
5.11.1. Zweck 42
5.12. /var/opt : Variable Daten für /opt 42
5.12.1. Zweck 43
5.13. /var/run : Daten aktiver Prozesse 43
5.13.1. Zweck 43
5.13.2. Anforderungen 43
5.14. /var/spool : Spool-Dateien 43
5.14.1. Zweck 44
5.14.2. Optionen 44
5.14.3. /var/spool/lpd : Druckerwarteschlange für den lpd-Dämon (optional) 44
5.14.3.1. Zweck 44
5.14.3.2. Optionen 44
5.14.4. /var/spool/rwho : Dateien des rwhod-Dämons (optional) 44
5.14.4.1. Zweck 45
5.15. /var/tmp : Temporäre Dateien, die aber beim Systemneustart erhalten bleiben 45
5.15.1. Zweck 45
5.16. /var/yp : Datenbank des NIS ("Network Information Service") (optional) 45
5.16.1. Zweck 45
6. Betriebssystemspezifische Ergänzungen 47
6.1. Linux 47
6.1.1. / : Das Wurzelverzeichnis 47
6.1.2. /bin : Wesentliche Kommandos (für alle Benutzer) 47
6.1.3. /dev : Gerätedateien 47
6.1.4. /etc : Rechnerspezifische Systemeinstellungen 48
6.1.5. /lib64 und /lib32 : 64/32-Bit-Bibliotheken (architekturspezifisch) 48
6.1.6. /proc : Virtuelles Dateisystem für Kern- und Prozessinformationen 48
6.1.7. /sbin : Wesentliche Systemkommandos 48
6.1.8. /usr/include : Header-Dateien für C-Programme 50
6.1.9. /usr/src : Quellen 50
6.1.10. /var/spool/cron : cron- und at-Jobs 50
7. Anhang 51
7.1. Die FHS-Mailingliste 51
7.2. Was steckt hinter dem FHS ? 51
7.3. Allgemeine Richtlinien 51
7.4. Gültigkeitsbereich 51
7.5. Danksagung 52
7.6. Mitwirkende 52

Anmerkungen zur Übersetzung

Es war mir ein Anliegen, deutsche Wörter zu verwenden, und nicht, wie sonst oft in EDV-Literatur üblich, englische Fachbegriffe einfach ins Deutsche zu übernehmen. So heißt der "Boot-Vorgang" hier Systemstart, der "Kernel" ist der Kern, "Administration" ist Verwaltung und "Konfigurationen" sind Einstellungen. "Netzwerk" ist übrigens nur eine Übertragung des englischen Worts "network", die Übersetzung aber lautet: Netz.

Dahinter steht die Überlegung, dass für deutschsprachige Leser sowohl bei original englischen Begriffen als auch bei solchen die ins Deutsche übernommen werden, immer eine Verständnislücke bleibt - es fehlt nämlich der breite assoziative Hintergrund der Muttersprachler, also die Vielzahl von Bedeutungsnuancen, an die jene ihr Verständnis anknüpfen können.

Solcherlei Bemühen stößt aber gerade im Bereich EDV sehr früh an Genzen. Denn viele der abgeleiteten Begriffe wie zum Beispiel "Konfigurationsdatei" sind eingeführt. "Einstellungsdatei" hingegen wirkt sperrig. Das verlangt also in jeden Einzelfall ein Abwägen. "Beständig" und "veränderlich" sind in Kapitel 2 nützlich, um einmalig klarzumachen, was "statisch" und "variabel" bedeuten. Im restlichen Text hemmen sie aber den Lesefluss; "static files" bleiben deshalb "statische Dateien". Für "site" gibt es keine deutsche Übersetzung, die alle Bedeutungen abdeckt. Hier wird "site" meist mit "Betreiber" übersetzt.

Um ungewohnte deutsche Begriffe zu stützen, wurde hin und wieder das jeweils englische Wort in Klammern und doppelten Anführungszeichen ("quotation marks") ergänzt.

An einigen Stellen hielt ich Anmerkungen für nötig. Sie stehen in Klammern und sind mit "A.d.Ü." gekennzeichnet.

Bei den Kapiteln zu /usr/lib<Suffix>, /usr/local und zu /usr/local/share wurde ein Fehler in der Dokumentstruktur behoben.

Nürnberg, im März 2005

Hans-Werner Heinzen <hwheinzen@Bitloeffel.de>

Kapitel 1. Einleitung

1.1. Zweck

Der hier beschriebene Standard ermöglicht es sowohl Programmen als auch Benutzern, die Lage installierter Dateien und Verzeichnisse vorherzusehen.

Das erreichen wir, indem wir

- Richtlinien für jeden Bereich des Dateisystems festlegen,
- die mindestens benötigten Dateien und Verzeichnisse benennen,
- Ausnahmen zu diesen Richtlinien aufzählen und
- besondere Fälle aufzählen, wo es, historisch bedingt, Konflikte gab.

Dieses FHS-Dokument wird benutzt

- von unabhängigen Softwareherstellern, um FHS-konforme Software zu erstellen, die mit FHS-konformen Distributionen zusammenarbeiten,
- von Herstellern von Betriebssystemen, um Systeme zu erstellen, die FHS-konform sind, und
- von Benutzern, um FHS-Konformität eines Systems zu verstehen und zu warten.

Der Gültigkeitsbereich dieses FHS-Dokuments ist begrenzt.

- Lokale Plazierung lokaler Dateien bleibt lokale Aufgabe; der FHS versucht nicht, die Systemverwaltung an sich zu reißen.
- FHS behandelt die Fälle, in denen Plazierung von Dateien koordiniert werden muss zwischen verschiedenen interessierten Parteien wie lokalen Betreibern, Distributionen, Anwendungsprogrammen, Dokumentationen, usw.

1.2. Benutzerhinweise

Wir empfehlen, eher eine formatierte Version dieses Dokuments als die einfache Textversion zu lesen. In der formatierten Version sind Datei- und Verzeichnisnamen in `Schreibmaschinenschrift` gesetzt.

Variable Bestandteile von Dateinamen werden repräsentiert durch die Beschreibung ihres Inhalts eingeschlossen in spitze Klammern "<" und ">", <*also so*>. E-Mail-Adressen stehen ebenfalls in spitzen Klammern, aber in normalem Schriftfont, <also@so>.

Wahlfreie Bestandteile von Dateinamen werden eingeschlossen in eckige Klammern "*[*" und "*]*"; das kann mit den spitzen Klammern kombiniert werden. Wenn beispielsweise eine Dateiname mit oder ohne Erweiterung vorkommen kann, so wird er als *<Dateiname>[.<Erweiterung>]* geschrieben.

Variable Bestandteile von Verzeichnis- und Dateinamen werden durch "*" repräsentiert.

Die Textabschnitte die als *Hintergrund* gekennzeichnet sind, sind nur Erklärung und haben keinen normativen Charakter.

Kapitel 2. Das Dateisystem

Der hier beschriebene Standard geht davon aus, dass das hinter dem FHS-konformen Dateisystem liegende Betriebssystem die gleichen grundlegenden Sicherheitseinrichtungen unterstützt. die bei den meisten UNIX Dateisysteme angetroffen werden können.

Für Dateien kann man zwei voneinander unabhängige Unterscheidungsmerkmale festlegen: gemeinsam nutzbar ("shareable") oder nicht ("unshareable") und veränderlich ("variable") oder beständig ("static"). Grundsätzlich sollten Dateien, die sich auch nur in einem der beiden Merkmale unterscheiden, in verschiedene Verzeichnissen abgelegt werden. Das erleichtert es, Dateien mit verschiedenen Nutzcharakteristiken auf verschiedenen Dateisystemen zu speichern.

Gemeinsam nutzbare Dateien sind solche, die bei einem Rechnersystem ("host") gespeichert und von anderen genutzt werden. Nicht gemeinsam nutzbare Dateien sind die, bei denen das nicht möglich ist. Beispielsweise sind die Dateien in den Benutzerverzeichnissen gemeinsam nutzbar, Geräte-Sperrdateien ("device lock files") sind das nicht.

Beständige (statische) Dateien sind ausführbare Programme, Programmbibliotheken, Dokumentationen und andere Dateien, die sich ohne den Eingriff durch einen Systemverwalter nicht ändern. Veränderliche (variable) Dateien sind die anderen.

Hintergrund

Gemeinsam nutzbare Dateien können auf einem Rechnersystem gespeichert und von mehreren anderen benutzt werden. Typischerweise sind nicht alle Dateien der Dateisystemhierarchie gemeinsam nutzbar, und deshalb hat auch jedes (Rechner-)System einen lokalen Speicherbereich, in dem sich unter anderem die nicht gemeinsam nutzbaren Dateien befinden. Bequem ist es, wenn alle vom System benötigten Dateien, die auf einem fremden Rechnersystem gespeichert sind, zugänglich gemacht werden können, indem ein oder wenige Verzeichnisse des fremden Systems eingebunden ("mount") werden.

Beständige und veränderliche Dateien sollten deswegen voneinander getrennt werden, weil man beständige, im Gegensatz zu veränderlichen, auf einem Nur-Lese-Medium ("read-only media") speichern kann, und es auch nicht nötig ist, sie regelmäßig zu sichern ("back up").

Ältere UNIX-ähnliche Dateisystemhierarchien enthielten sowohl beständige als auch veränderliche Dateien sowohl unter `/usr` als auch unter `/etc`. Um die genannten Vorteile zu erreichen, wurde die `/var`-Hierarchie entwickelt und alle veränderlichen Dateien von `/usr` nach `/var` übertragen. Als Konsequenz kann `/usr` im Nur-Lese-Modus eingebunden werden (, wenn es ein separates Dateisystem ist). Veränderliche Dateien von `/etc` sind über einen längeren Zeitraum nach `/var` übertragen worden, so wie es die Entwicklung der Technik gerade erlaubte.

Hier folgt ein Beispiel für ein FHS-konformes System (andere Layouts sind möglich):

	gemeinsam nutzbar	**nicht gemeinsam nutzbar**
beständig (statisch)	/usr	/etc
	/opt	/boot
veränderlich (variabel)	/var/mail	/var/run
	/var/spool/news	/var/lock

Kapitel 3. Das Wurzeldateisystem

3.1. Zweck

Der Inhalt des Wurzeldateisystems muss genügen zum Starten ("boot"), zum Wiederherstellen ("restore, recover") und/oder zum Reparieren des Systems.

- Für den Systemstart muss die Wurzelpartition all das enthalten, was nötig ist, um andere Dateisysteme einzubinden. Dazu gehören Dienstprogramme, Konfigurationsdaten, Informationen für den Systemlader ("boot loader") und weitere wichtige Startdaten.
- Um Wiederherstellen ("recovery") und/oder Reparieren eines Systems möglich zu machen, müssen auf dem Wurzeldateisystem die Werkzeuge vorhanden sein, die ein erfahrener Verwalter braucht, um ein beschädigtes System zu untersuchen und wieder aufzubauen.
- Zum Wiederherstellen ("restore") eines Systems, müssen die Werkzeuge auf dem Wurzeldateisystem vorhanden sein, die zum Einspielen von Sicherungen (auf Diskette, Magnetband, usw.) nötig sind.

Hintergrund

Diesen Überlegungen, die so vieles auf dem Wurzeldateisystem haben möchten, steht vor allem das Ziel entgegen, die Wurzel so klein wie noch sinnvoll möglich zu halten. Mehrere Gründe machen es wünschenswert, das Wurzeldateisystem klein zu halten:

- Es wird manchmal von einem sehr kleinen Medium eingebunden.
- Das Wurzeldateisystem enthält viele systemspezifische Konfigurationsdateien. Das kann beispielsweise auch ein Systemkern ("kernel") sein, der spezifisch ist für dieses bestimmte System, für diesen bestimmten Systemnamen ("hostname"), usw. Das bedeutet, dass das Wurzeldateisystem nicht immer von den Systemen in einem Netz gemeinsam nutzbar ist. Indem man es klein hält, minimiert man die Menge an Speicherplatz, die wegen der nicht gemeinsam nutzbare Dateien verloren geht. Außerdem erlaubt das Arbeitsplatzrechner mit kleineren lokalen Festplatten.
- Während Sie vielleicht ihr Wurzeldateisystem auf einer großen Partition haben und dadurch in der Lage sind, es nach Herzenslust aufzufüllen, haben andere nur kleinere Partitionen zur Verfügung. Haben Sie nun mehr Dateien installiert, werden Sie wahrscheinlich Unverträglichkeiten mit den anderen Systemen entdecken, die ihr Wurzeldateisystem auf einer kleineren Partition haben. Wenn Sie ein Entwickler sind, werden ihre Hypothesen über den Inhalt des Wurzeldateisystems eventuell zu einem Problem für eine Menge Benutzer.
- Festplattenfehler im Wurzeldateisystems bedeuten ein größeres Problem als auf anderen Partitionen. Ein kleines Wurzeldateisystem ist bei einem Systemabsturz weniger fehleranfällig.

Anwendungsprogramme dürfen nie Dateien oder Unterverzeichnisse im Wurzelverzeichnis erzeugen oder dort erwarten. Andere Bereiche der FHS-Hierarchie bieten ausreichend Fexibilität.

Hintergrund

Für dieses Verbot gibt es mehrere Gründe:

- Es würde sonst Speicherplatz in der Wurzelpartition benötigt, während möglicherweise der Systemverwalter diese wegen Performanz oder Sicherheit klein und übersichtlich halten will.
- Sonst überginge man, was auch immer ein Systemverwalter an Richtlinien für Dateihierarchien über Einheitengrenzen hinweg festgelegt hat.

Auch Distributionen sollen keine neuen Verzeichnisse in der Wurzelhierarchie anlgen, ohne vorher sorgsam die Konsequenzen abzuwägen, und zwar auch für die Portabilität der Anwendungsprogramme.

3.2. Anforderungen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen werden in `/` benötigt:

Verzeichnis	Beschreibung
`bin`	Wesentliche Kommandos
`boot`	Statische Dateien des Systemladers
`dev`	Gerätedateien
`etc`	Rechnerspezifische Systemeinstellungen
`lib`	Wesentliche Programmbibliotheken und Module des Systemkerns
`media`	Einhängepunkte für Wechselmedien
`mnt`	Einhängepunkt für ein temporär eingehängtes Dateisystem
`opt`	Zusatzsoftware
`sbin`	Wesentliche Systemkommandos
`srv`	Daten laufender Dienste
`tmp`	Temporäre Dateien
`usr`	Zweite Hierarchie
`var`	Variable Daten

Jedes der oben anfgezählten Verzeichnisse wird ausführlich in eigenen Unterkapiteln, weiter unten, beschrieben. Für `/usr` und `/var` gibt es, wegen der Komplexität dieser Verzeichnisse, jeweils ein komplett eigenes Kapitel.

3.3. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/` dann vorhanden

sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
`home`	Benutzerverzeichnisse (optional)
`lib<Suffix>`	Wesentliche Programmbibliotheken in alternativem Format (optional)
`root`	Benutzerverzeichnis von "root" (optional)

Jedes der oben anfgezählten Verzeichnisse wird ausführlich in einem eigenen Unterkapitel, weiter unten, beschrieben.

3.4. /bin : Wesentliche Kommandos (für alle Benutzer)

3.4.1. Zweck

Verzeichnis `/bin` enthält Kommandos, die sowohl vom Systemverwalter als auch von Benutzern verwendet werden können, und die auch dann benötigt werden, wenn keine weiteren Dateisysteme eingebunden sind (z.B. im Einzelbenutzer-Modus). Es darf auch Kommandos enthalten, die nur indirekt von Skripten aufgerufen werden. [1]

3.4.2. Anforderungen

In `/bin` darf es keine Unterverzeichnisse geben.

Die folgenden Kommandos oder symbolischen Links zu Kommandos werden in `/bin` benötigt:

Kommando	Beschreibung
cat	Verketten von Dateien auf die Standardausgabe
chgrp	Ändern der Gruppenzugehörigkeit von Dateien
chmod	Ändern der Zugriffsrechte von Dateien
chown	Ändern von Eigentümer oder Gruppeneigentümern von Dateien
cp	Kopieren von Dateien und Verzeichnissen
date	Anzeigen oder Setzen von Systemdatum/-zeit
dd	Kopieren und Konvertieren einer Datei
df	Informieren über Dateisysteme
dmesg	Anzeigen der Systemkern-Meldungen
echo	Anzeigen einer Textzeile
false	Tue nichts und versage!
hostname	Anzeigen oder Setzen des Rechnernamens
kill	Senden von Signalen an Prozesse
ln	Erzeugen von Links zu Dateien

Kommando	Beschreibung
login	Starten einer Systemsitzung
ls	Anzeigen von Verzeichnisinhalten
mkdir	Erzeugen von Verzeichnissen
mknod	Erzeugen einer speziellen Datei
more	Blättern durch einen Text
mount	Einbinden von Dateisystemen
mv	Verschieben oder Umbenennen von Dateien
ps	Informieren über Prozesse
pwd	Anzeigen des aktuellen Verzeichnisses
rm	Löschen von Dateien oder Verzeichnissen
rmdir	Löschen leerer Verzeichnisse
sed	Stream-Editor "sed"
sh	Kommandooberfläche "Bourne shell"
stty	Anzeigen und Ändern von Terminal-Einstellungen
su	Ändern der Benutzer-ID
sync	Schreiben von Dateisystempuffern erzwingen
true	Tue nichts, aber erfolgreich!
umount	Lösen einer Dateisystem-Einbindung
uname	Anzeigen von Systeminformationen

Wenn **/bin/sh** nicht die echte "Bourne shell" ist, so muss es ein fester oder symbolischer Link auf das tatsächliche shell-Kommando sein.

Die Kommandos **[** und **test** müssen jeweils zusammen entweder unter `/bin` oder unter `/usr/bin` abgelegt werden.

Hintergrund

Beispielsweise verhält sich "bash" unterschiedlich, je nachdem ob es mit **sh** oder **bash** aufgerufen wird. Außerdem zeigt die Verwendung eines symbolischen Links dem Benutzer an, dass **/bin/sh** nicht die echte Bourne shell ist.

Die Forderung, dass **[** und **test** als Programmdateien vorhanden sein müssen, selbst wenn sie shell-intern implementiert sind, gründet sich auf den POSIX.2-Standard.

3.4.3. Optionen

Die folgenden Programme oder symbolischen Links zu Programmen müssen in `/bin` dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Kommando	Beschreibung

Kommando	Beschreibung
csh	Kommandooberfläche "C shell" (optional)
ed	Editor "ed" (optional)
tar	Archivierungsprogramm "tar" (optional)
cpio	Archivierungsprogramm "cpio" (optional)
gzip	GNU Kompressionsprogramm "gzip" (optional)
gunzip	GNU Dekompressionsprogramm "gunzip" (optional)
zcat	GNU Dekompressionsprogramm "zcat" (optional)
netstat	Netzstatistik (optional)
ping	ICMP Netztest

Wenn **gunzip** und **zcat** vorhanden sind, müssen es symbolische Links zu gzip sein. **/bin/csh** darf ein symbolischer Link entweder zu **/bin/tcsh** oder zu **/usr/bin/tcsh** sein.

Hintergrund

Die Kommandos tar, gzip und cpio sind hier ergänzt worden, um das Wiederherstellen eines Systems - mit einem noch intakten `/`-Dateisystem - möglich zu machen.

Umgekehrt sollten sie weggelassen werden, wenn ein Wiederherstellen von der Wurzelpartition aus niemals erwartet wird (z.B. bei einem ROM-Chip als Wurzel, wobei `/usr` über NFS eingebunden wird). Wenn ein Wiederherstellen des Systems über das Netz vorgesehen ist, dann müssen auch **ftp** oder **tftp** in der Wurzelpartition verfügbar sein, inclusive aller für die ftp-Verbindung benötigten Komponenten.

3.5. /boot : Statische Dateien des Laders

3.5.1. Zweck

Dieses Verzeichnis enthält alles, was für den Systemladevorgang nötig ist, ausgenommen die Konfigurationsdateien, die ja nicht zum Ladezeitpunkt gebraucht werden, und ebenso ausgenommen das Einrichtungsprogramm für die Zuordnungstabelle ("map installer"). Also enthält /boot solche Daten, die der Betriebssystemkern braucht, bevor er Benutzerprogramme ausführt. Das können z.B. auch gesicherte MBRs ("master boot records") oder Zuordnungstabellen ("map files") sein. [2]

3.5.2. Optionen

Der Betriebssystemkern muss sich entweder in `/` oder in `/boot` befinden. [3]

3.6. /dev : Gerätedateien

3.6.1. Zweck

Das Verzeichnis `/dev` ist für Gerätedateien da.

3.6.2. Optionen

Wenn es möglich sein soll, Geräte in `/dev` manuell zu erzeugen, so muss es dort auch ein Kommando namens **MAKEDEV** geben, das Geräte nach Bedarf kreieren kann. Es darf für lokale Geräte auch ein Kommando namens **MAKEDEV.local** geben.

Wenn nötig, muss **MAKEDEV** die Fähigkeit haben, jedes beliebige Gerät, das möglicherweise auf dem System zu finden ist, zu erzeugen, nicht nur die bei einer bestimmten Installation vorgesehenen.

3.7. /etc : Rechnerspezifische Systemeinstellungen

3.7.1. Zweck

Die `/etc`-Hierarchie enthält Konfigurationsdateien. Eine Konfigurationsdatei ist eine lokale Datei und wird benutzt um den Lauf von Programmen zu steuern. Sie muss statisch und darf kein ausführbares Programm sein. [4]

3.7.2. Anforderungen

Unter `/etc` sollen keine ausführbaren Programme plaziert werden.

Es wird empfohlen, Dateien in Unterverzeichnissen von `/etc` anstatt direkt in `/etc` zu speichern. (Dieser Satz erscheint im englischen Original - wohl versehentlich - als Fußnote: A.d.Ü.)

Folgende Verzeichnisse oder symbolische Links zu Verzeichnissen werden in `/etc` benötigt:

Verzeichnis	Beschreibung
`opt` (Im englischen Original waren hier - wohl versehentlich - auch noch die optionalen Verzeichnisse `X11`, `sgml` und `xml` angeführt: A.d.Ü.)	Konfiguration für /opt

3.7.3. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/etc` dann

vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
(Wohl versehentlich - war im englischen Original hier ein zweites Mal das Verzeichnis `opt` angeführt, dagegen fehlten die Verzeichnisse `X11`, `sgml` und `xml`: A.d.Ü.)	
`X11`	Konfiguration für das X-Window-System (optional)
`sgml`	Konfiguration für SGML (optional)
`xml`	Konfiguration für XML (optional)

Die folgenden Dateien oder symbolischen Links zu Dateien müssen in `/etc` vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind: [5]

Datei	Beschreibung
`csh.login`	Systemweit gültige Starteinstellungen für C-Shells (optional)
`exports`	Zugriffskontrollliste für das NFS Dateisystem (optional)
`fstab`	Statische Informationen über Dateisysteme (optional)
`ftpusers`	Zugriffskontrollliste des FTP-Dämons (optional)
`gateways`	Liste von Gateway-Rechnern (optional)
`gettydefs`	Einstellungen für getty (optional)
`group`	Liste der Benutzergruppen (optional)
`host.conf`	Einstellungen für die "resolver"-Bibliothek (optional)
`hosts`	Rechner-Adress-Zuordnungen (optional)
`hosts.allow`	Rechner-Zugriffsliste für TCP (optional)
`hosts.deny`	Rechner-Zugriffsliste für TCP (optional)
`hosts.equiv`	Vertrauenswürdige Rechner für rlogin, rsh, rcp (optional)
`hosts.lpd`	Vertrauenswürdeige Rechner für lpd (optional)
`inetd.conf`	Einstellungen für inetd (optional)
`inittab`	Einstellungen für init (optional)
`issue`	Meldung vor dem Anmelden (optional)
`ld.so.conf`	Zusätzliche Verzeichnisse für die Suche nach Programmbibliotheken (optional)
`motd`	Meldung nach dem Anmelden (optional)
`mtab`	Dynamische Informationen über Dateisysteme (optional)
`mtools.conf`	Einstellungen für mtools (optional)
`networks`	Netznamen-Adress-Zuordnungen (optional)

Datei	Beschreibung
`passwd`	Die Passwortdatei
`printcap`	lpd-Drucker
`profile`	Systemweit gültige Starteinstellungen für sh-Shells (optional)
`protocols`	Liste von IP-Protokollen (optional)
`resolv.conf`	Einstellungen für die "resolver"-Bibliothek (optional)
`rpc`	Liste von RPC-Protokollen (optional)
`securetty`	TTY-Zugriffskontrolle für root (optional)
`services`	Portnamen für Netzdienste (optional)
`shells`	Pfadnamen gültiger Kommandooberflächen (optional)
`syslog.conf`	Einstellungen für syslog (optional)

`mtab` erfüllt nicht die für `/etc` geforderte Eigenschaft "statisch": Diese Ausnahme wird aus historischen Gründen gemacht. [6]

3.7.4. /etc/opt : Einstellungen für /opt

3.7.4.1. Zweck

Rechnerspezifische Einstellungen für Zusatzsoftwarepakete müssen in einem Verzeichnis `/etc/opt/<Name>` abgelegt werden, wobei `<Name>` der Name des Teilbaums von `/opt` ist, in dem statische Informationen des jeweiligen Pakets gespeichert werden.

3.7.4.2. Anforderungen

Für `/etc/opt/<Name>` wird eine interne Struktur nicht vorgeschrieben.

Sollte es für das Funktionieren der Software nötig sein, dass die Einstellungen an anderer Stelle gespeichert werden müssen, so ist das auch erlaubt.

 **Hintergrund**

(siehe: Hintergrund zu `/opt`)

3.7.5. /etc/X11 : Einstellungen für das X-Window-System (optional)

3.7.5.1. Zweck

/etc/X11 ist die Stelle für alle rechnerspezifischen Einstellungen von X11. Nutzung dieses Verzeichnisses erlaubt lokale Kontrolle, auch wenn */usr* im Nur-Lese-Modus eingebunden ist.

3.7.5.2. Optionen

Die folgenden Dateien oder symbolischen Links zu Dateien müssen in `/etc/X11` vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Datei	Beschreibung
`Xconfig`	Einstellungen für frühe Versionen von XFree86 (optional)
`XF86Config`	Einstellungen für XFree86 Versionen 3 und 4 (optional)
`Xmodmap`	Global wirksame Anpassungen für X11 (optional)

Wenn nötig, dürfen für `xdm` oder andere Programme (beispielsweise einige Fenstermanager) Unterverzeichnisse von `/etc/X11` angelegt werden. [7] Solange es noch keine breite Akzeptanz für eine andere Namenskonvention gibt, empfehlen wir, dass ein Fenstermanager mit nur einer Konfigurationsdatei, die üblicherweise `.*wmrc` heißt, diese `system.*wmrc` nennt. Unterverzeichnisse von Fenstermanagern sollen genauso heißen, wie das lauffähige Programm.

3.7.6. /etc/sgml : Einstellungen für SGML (optional)

3.7.6.1. Zweck

Hier werden typische Konfigurationsdateien mit "high-level"-Parametern des SGML-Systems abgelegt; das zeigen Dateien mit Namen wie `*.conf`. Dateien mit Namen wie `*.cat` sind DTD-basierte Zentralkataloge, die Verweise zu allen von einer DTD benötigten Katalogen enthalten. Der Superkatalog `catalog` verweist auf alle diese Zentralkataloge.

3.7.7. /etc/xml : Einstellungen für XML (optional)

3.7.7.1. Zweck

Hier werden typische Konfigurationsdateien mit "high-level"-Parametern des XML-Systems abgelegt; das zeigen Dateien mit Namen wie `*.conf`. Der Superkatalog `catalog` verweist auf alle Zentralkataloge.

3.8. /home : Benutzerverzeichnisse (optional)

3.8.1. Zweck

Dieses Konzept ist ziemlich üblich, aber es bleibt ganz klar eine Sache der Betreiber. [8] Der Aufbau wird von Rechner zu Rechner unterschiedlich sein. Deshalb sollte kein Programm sich auf diese Plazierung verlassen. [9]

3.8.2. Anforderungen

Benutzerspezifische Einstellungen von Anwendungen werden im jeweiligen Benutzerverzeichnis in einer Datei, deren Namen mit einem Punkt beginnt ("dot file") abgelegt. Benötigt die Anwendung mehr als eine solche Datei, sollten diese in einem Unterverzeichnis abgelegt werden, dessen Name mit einem Punkt beginnt ("dot directory"). Dort sollen die Dateinamen dann aber nicht mit einem Punkt beginnen. [10]

3.9. /lib : Wesentliche Programmbibliotheken und Kernmodule

3.9.1. Zweck

Das Verzeichnis `/lib` enthält jene Programmbibliotheken, die gebraucht werden, um das System zu starten und die Kommandos im Wurzeldateisystem, d.h. die lauffähigen Programme in `/bin` und `/sbin`, zu betreiben. [11]

3.9.2. Anforderungen

Für die folgenden Dateinamensmuster muss jeweils mindestens eine Entsprechung vorhanden sein, egal ob als Datei oder symbolischer Link.

Datei	Beschreibung
`libc.so.*`	Die dynamisch eingebundene C-Bibliothek (Im englischen Original steht hier - wohl versehentlich - in Klammern "optional": A.d.Ü.))
`ld*`	Der Laufzeit-Binder/Lader (Im englischen Original steht hier - wohl versehentlich - in Klammern "optional": A.d.Ü.))

Wenn ein C-Präpozessor installiert ist, so muss, aus historischen Gründen, */lib/cpp* ein Verweis darauf sein. [12]

3.9.3. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/lib` dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
`modules`	Nachladbare Kernmodule (optional)

3.10. /lib<Suffix> : Wesentliche Programmbibliotheken in alternativem Format (optional)

3.10.1. Zweck

Auf Systemen, die mehr als ein Format ausführbarer Programme unterstützen und deshalb getrennte Bibliotheken brauchen, dürfen eine oder mehr Varianten von `/lib` vorhanden sein. [13]

3.10.2. Anforderungen

Wenn es eine oder mehrere solcher Verzeichnisse gibt, gelten für diese dieselben Anforderungen wie auch für das "normale" `/lib`; nur `/lib<Suffix>/cpp` wird hier nicht gebraucht. [14]

3.11. /media : Einhängepunkte für Wechselmedien

3.11.1. Zweck

Dieses Verzeichnis enthält Unterverzeichnisse, die als Einhängepunkte für Wechselmedien wie Diskette, CD-ROMs oder ZIP-Datenträger genutzt werden.

Hintergrund

Blickt man zurück, so wurden Wechselmedien an den verschiedensten Stellen eingehängt, so zum Beispiel unter `/cdrom`, `/mnt` oder `/mnt/cdrom`. Wenn man alle Einhängepunkte direkt im Wurzelverzeichnis plazieren würde, hätte man bald eine Vielzahl von zusätzlichen Unterverzeichnisse von `/`. Das Einrichten von Unterverzeichnissen unter `/mnt` ist auch üblich geworden, gerät aber in Konflikt mit der älteren Tradition, `/mnt` direkt als temporären Einhängepunkt zu nutzen.

3.11.2. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/media` dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
`floppy`	Diskettenlaufwerk (optional)
`cdrom`	CD-ROM-Laufwerk (optional)
`cdrecorder`	CD-Brenner (optional)
`zip`	ZIP-Laufwerk (optional)

Bei Systemen, bei denen mehrere Geräte eines Typs gibt, können Verzeichnisse zum Einhängen erzeugt werden, indem man eine Ziffer (beginnend mit "0") an den Namen anhängt; den unqualifizierten Namen muss es aber weiterhin geben. [15]

3.12. /mnt : Einhängepunkt für ein temporär eingebundenes Dateisystem

3.12.1. Zweck

Diese Verzeichnis dient einem Systemadministrator dazu, im Bedarfsfall ein Dateisystem vorübergehend einzubinden. Der Inhalt dieses Verzeichnisses liegt in lokaler Verantwortung und sollte keinen Einfluss auf das Verhalten irgendwelcher Programme haben.

Dieses Verzeichnis darf von Installationsprogrammen nicht benutzt werden; stattdessen muss ein passendes temporäres Verzeichnis benutzt werden, das nicht vom System verwendet wird.

3.13. /opt : Zusatzsoftware

3.13.1. Zweck

`/opt` ist reserviert für zusätzliche Softwarepakete.

Ein Paket, das unter `/opt` installiert wird, muss seine statischen Dateien in einem eigenen Unterverzeichnis namens `/opt/<Paket>` oder `/opt/<Hersteller>` plazieren, wobei dann `<Paket>` der Name des Softwarepakets ist und `<Hersteller>` der bei LANANA registrierte Herstellername.

3.13.2. Anforderungen

Verzeichnis	Beschreibung
<Paket>	Statische Objekte des Pakets
<Hersteller>	Bei LANANA registrierter Herstellername

Die Verzeichnisse `/opt/bin`, `/opt/doc`, `/opt/include`, `/opt/info`, `/opt/lib` und `/opt/man` wurden für den Gebrauch durch Systemverwalter reserviert. Softwarepakete dürfen dort Dateien ablegen (als Link oder Kopie), müssen aber auch in Abwesenheit dieser Verzeichnisse funktionsfähig bleiben.

Lauffähige Programme, die vom Benutzer aufgerufen werden sollen, gehören entweder nach `/opt/<Paket>/bin` oder nach `/opt/<Hersteller>/bin`. Enthält das Paket UNIX Online-Handbücher ("manual pages"), so müssen diese nach `/opt/<Paket>/share/man` oder `/opt/<Hersteller>/share/man` (Im englischen Original steht hier - wohl versehentlich - nur "`/opt/<provider>`": A.d.Ü.) abgelegt werden, mit derselben Struktur wie `/usr/share/man`.

Variable Dateien eines Softwarepakets, also solche, die sich bei normalen Operationen des Programms ändern, werden in `/var/opt` gespeichert. Genaueres dazu steht im Abschnitt über `/var/opt`.

Rechnerbezogene Konfigurationsdateien gehören nach `/etc/opt`. Genaueres dazu steht im Abschnitt über `/etc`.

Ausserhalb von `/opt`, `/var/opt` und `/etc/opt` sollen keine Dateien existieren, die von Zusatzsoftwarepaketen stammen. Ausnahmen sind nur solche Dateien, deren fehlerfreies Arbeiten von ihrer besonderen Plazierung abhängt. Beispielsweise müssen Sperrdateien für Geräte in `/var/lock` plaziert werden und Gerätedateien in `/dev`.

Distributionen dürfen Software in `/opt` installieren aber sie dürfen dort andere (vom Verwalter eingerichtete) Software ohne dessen Zustimmung weder verändern noch löschen.

Hintergrund

Die Verwendung von `/opt` für Zusatzsoftware ist weit verbreitete Praxis in der UNIX-Gemeinde. Das "System V Application Binary Interface [AT&T 1990]", welches auf "System V Interface Definition (Third Edition)" aufbaut, sieht für `/opt` eine Struktur vor, die dieser hier sehr ähnelt.

Auch der "Intel Binary Compatibility Standard v. 2 (iBCS2)" sieht eine ähnliche Struktur für `/opt` vor.

Allgemein soll gelten, dass alle Dateien, die ein Programmpaket auf einem bestimmten System benötigt, in `/opt/<Paket>` vorhanden sein müssen, also auch diejenigen, die nach `/etc/opt/<Paket>` und `/var/opt/<Paket>` kopiert werden sollen, und auch reservierte Unterverzeichnisse von `/opt`.

Die kleine Einschränkung für die Benutzung von `/opt` durch Distributionen ist nötig, weil sonst Konflikte zwischen Software, die durch die Distribution, und solcher, die vom Verwalter installiert wurde, möglich wären. Insbesondere passiert das bei Software, bei denen Pfadnamen fest einprogrammiert wurden.

Die Verzeichnisstruktur unterhalb von `/opt/<Hersteller>` bleibt dem Paketbetreuer überlassen. Allerdings wird empfohlen, Pakete nach `/opt/<Hersteller>/<Paket>` zu installieren und dort die Richtlinien für `/opt/<Paket>` zu übernehmen. Ein Grund dafür, von dieser Regel abzuweichen, sind Support-Softwarepakete, die Dateien möglicheweise in `/opt/<Hersteller>/lib` oder `/opt/<Hersteller>/bin` installieren.

3.14. /root : Benutzerverzeichnis von "root" (optional)

3.14.1. Zweck

Plazierung des Benutzerverzeichnisses von "root" kann nach lokalen Prioritäten festgelegt werden; dies hier ist aber die empfohlene Stelle. [16]

3.15. /sbin : Wesentliche Systemkommandos

3.15.1. Zweck

Hilfsprogramme für die Systemverwaltung (und andere nur für "root" vorgesehene Kommandos) werden in `/sbin`, `/usr/sbin` und `/usr/local/sbin` gespeichert. Dabei enthält `/sbin` alle notwendigen Programme für Systemstart, Reparatur und Wiederherstellung des Systems, die über die Programme in `/bin` hinausgehen. [17] Programme, die erst nach dem erfolgreichen Einbinden von `/usr` ausgeführt werden, werden üblicherweise nach `/usr/sbin` abgelegt. Lokal installierte Systemverwaltungswerkzeuge sollten in `/usr/local/sbin` plaziert sein. [18]

3.15.2. Anforderungen

Die folgenden Kommandos oder symbolischen Links zu Kommandos werden in `/sbin` benötigt:

Kommando	Beschreibung
shutdown	Herunterfahren des Systems

3.15.3. Optionen

Die folgenden Dateien oder symbolischen Links zu Dateien müssen in `/sbin` vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Kommando	Beschreibung
fastboot	Neustart des Systems ohne Prüfung der Festplatten (optional)
fasthalt	Stop des Systems ohne Prüfung der Festplatten (optional)
fdisk	Manipulieren der Partitionstabellen (optional)
fsck	Prüf- und Reparaturwerkzeug für Dateisysteme (optional)
fsck.*	Prüf- und Reparaturwerkzeug für ein bestimmtes Dateisystem (optional)

Kommando	Beschreibung
getty	Das Programm getty (optional)
halt	Anhalten des Systems (optional)
ifconfig	Einstellen einer Netzschnittstelle (optional)
init	Systeminitialisierung (optional)
mkfs	Erstellen eines Dateisystems (optional)
mkfs.*	Erstellen eines bestimmten Dateisystems (optional)
mkswap	Festlegen eines Auslagerungsbereichs (optional)
reboot	Neustart des Systems (optional)
route	Anzeigen und Ändern der IP-Routentabelle (optional)
swapon	Einschalten des Swapings (optional)
swapoff	Ausschalten des Swapings (optional)
update	Periodisches Entleeren der Dateisystempuffer (optional)

3.16. /srv : Daten laufender Dienste

3.16.1. Zweck

`/srv` enthält betreiberspezifische Daten, die von Diensten dieses Systems zur Nutzung angeboten werden.

 **Hintergrund**

Dieses Verzeichnis wurde vor allem eingeführt, um den Benutzern das Auffinden der Daten bestimmter Dienste zu erleichtern, und um Diensten, die einen einzelnen Verzeichnisbaum für Nur-Lese-Daten, Schreibdaten und Skripte (wie z.B. CGI-Skripte) brauchen, eine vernünftige Plazierung anzubieten. Dahingegen sollten Daten, die nur einen einzelnen Benutzer interessieren, im jeweiligen Benutzerverzeichnis abgelegt werden.

Es gibt zur Zeit keinen Konsens über die Namensgebung für Unterverzeichnisse von `/srv`. Eine mögliche Methode wäre, Daten unterhalb von `/srv` nach Protokollen zu ordnen, also zum Beispiel `ftp`, `rsync`, `www` oder `cvs`. Auf großen Systemen kann es sinnvoll sein, `/srv` nach Verwaltungsgesichtspunkten zu unterteilen, beispielsweise `/srv/Physik/www`, `/srv/Informatik/cvs` usw. Das wird sich von System zu System unterscheiden. Also darf sich kein Programm auf die Existenz einer bestimmten Verzeichnisstruktur unterhalb von `/srv` verlassen. Immerhin sollte es auf FHS-koformen Systemen immer ein `/srv` geben, das die Standardplazierung für solche Daten ist.

Distributionen dürfen hier lokal plazierte Dateien nicht ohne Erlaubnis des Verwalters entfernen. [19]

3.17. /tmp : Temporäre Dateien

3.17.1. Zweck

Das Verzeichnis `/tmp` muss erreichbar sein für alle Programme, die mit temporären Daten arbeiten.

Programme dürfen nicht davon ausgehen, dass Dateien oder Verzeichnisse in `/tmp` zwischen zwei Programmläufen erhalten bleiben.

Hintergrund

Der Standard "IEEE standard P1003.2 (POSIX, part 2)" stellt ganz ähnliche Anforderungen.

Auch wenn die Löschregelung für `/tmp`-Dateien der Verantwortung der Betreiber überlassen bleibt, wird hier empfohlen, Dateien und Verzeichnisse in `/tmp` bei jedem Systemstart zu löschen.

Der FHS gibt diese Empfehlung auf der Basis von Tradition und üblicher Praxis. Sie hat aber nicht den Status einer Anforderung, weil Systemverwaltung nicht Thema diese Standards ist.

Fußnoten

1. Kommandos, die nicht wichtig genug sind, um hier abgelegt zu werden, gehören stattdessen nach `/usr/bin`. Solche, die nur von Nicht-Root-Benutzern verwendet werden (das X Window System, `chsh`, usw.) sind in der Regel nicht wichtig genug, um in der Wurzelpartition gespeichert zu werden.
2. Programme, die nötig sind, um den Systemlader vorzubereiten gehören nach `/sbin`. Konfigurationsdateien für Systemlader gehören nach `/etc`.

   Der Systemlader GRUB liest seine Konfigurationsdatei vor dem Systemladen, sie gehört also nach `/boot`. Eigentlich sollte die Konfigurationsdatei aber in `/etc` sein. Die Lösung ist ein symbolischer Link in der Art `/etc/grub/menu.lst` -> `/boot/menu.lst`.
3. Auf einigen i386-Rechnern muss `/boot` wegen Hardwareeinschränkungen eventuell auf einer eigenen Partition unterhalb von Zylinder 1024 plaziert werden.

   Bestimmte MIPS-Systeme verlangen eine `/boot`-Partition, die z.B als MS-DOS-Dateisystem eingebunden ist, oder auf was auch immer diese Firmware zugreifen kann. Das kann sich als Beschränkung bei der Wahl nutzbarer Dateinamen innerhalb von `/boot` auswirken.
4. Die Scripte, die beim Systemstart aufgerufen werden, können denen von "System V", von BSD oder eines anderen Vorbilds ähneln. In künftigen Versionen dieses Standards wird eventuell Weitergehendes festgelegt.
5. Systeme, die die Shadow-Passwort-Einrichtung nutzen, haben weitere Konfigurationsdateien in `/etc` (`/etc/shadow` und andere) sowie Programme in `/usr/sbin` (**useradd**, **usermod** und andere).
6. Auf einigen Linux-Systemen kann das auch ein symbolischer Link nach `/proc/mounts` sein, so dass diese Ausnahmebedingung nicht nötig ist.

7. `/etc/X11/xdm` enthält die Einstellungen für `xdm`. Dort sind jetzt die meisten der Dateien, die früher in `/usr/lib/X11/xdm` untergebracht waren; einige lokale variable Daten für `xdm` findet man in `/var/lib/xdm`.

8. Verschiedene Leute plazieren ihre Benutzerverzeichnisse an ganz verschiedenen Orten. Das hier ist nur ein Plazierungsvorschlag. Nichtsdestoweniger empfehlen wir für alle FHS-konformen Systeme diese Plazierung als Standardeinstellung für Benutzerverzeichnisse.

   Auf kleinen Systemen ist das Benutzerverzeichnis typischerweise ein Unterverzeichnis von `/home` wie zum Beispiel `/home/smith`, `/home/torvalds` oder `/home/operator`. Auf großen System (insbesondere wenn die `/home`- Verzeichnisse mithilfe von NFS von mehreren Rechnern genutzt werden) ist es sinnvoll, die Benutzerverzeichnisse aufzuteilen. Das kann man erreichen, indem man Unterverzeichnisse wie zum Beispiel `/home/staff`, `/home/guests` oder `/home/students` anlegt.

9. Will man das Verzeichnis eines Benutzers finden, so nutze man die `getpwent(3)`-Bibliotheksfunktion anstatt sich auf `/etc/passwd` zu verlassen; Benutzerinformationen könnten mittels NIS o.ä. woanders abgelegt sein.

10. Es wird empfohlen, dass Programme in einem Benutzerverzeichnis keine Dateien oder Verzeichnisse ohne den Punkt am Beginn des Namnes ohne Benutzerintervention erzeugen. Ausnahmen sind "Autosave"- und Sperr-Dateien.

11. Programmbibliotheken, die nur von Programmen aus `/usr` (wie zum Beispiel die ganzen X-Window-Programme) gebraucht werden gehören nicht nach `/lib`. Hierher gehören nur solche, die von Programmen aus `/bin` und `/sbin` gebraucht werden. Insbesondere sollte die Bibliothek `libm.so.*` in `/usr/lib` abgelegt werden, solange sie nicht von `/bin` oder `/sbin` aus gebraucht wird.

12. Die übliche Plazierung dieses Programms ist `/usr/bin/cpp`.

13. Genutzt wird das üblicherweise für 64-Bit- und 32-Bit-Unterstützung auf Systemen mit Mehrfach-Laufzeitformaten, die aber Bibliotheken mit demselben Namen benötigen. In einem solchen Fall mögen die Bibliotheken `/lib32` und `/lib64` heißen, wobei `/lib` dann ein symbolischer Link auf eine der beiden ist.

14. Erlaubt ist `/lib<Suffix>/cpp` aber auch hier: das erlaubt den Fall, dass `/lib` und `/lib<Suffix>` dasselbe sind, also dass eines der symbolische Link auf das andere ist.

15. Auf einem FHS-konformen System mit zwei CD-ROM-Laufwerken gäbe es dann `/media/cdrom0` und `/media/cdrom1` mit `/media/cdrom` als symbolischen Link auf eins von beiden.

16. Ist das Benutzerverzeichnis von "root" nicht auf der Wurzelpartition gespeichert, so muss man sicherstellen, dass `/` als Standard genutzt wird, falls das Benutzerverzeichnis nicht gefunden wird.

    Wir empfehlen, den Benutzer "root" nicht für Aufgaben zu nutzen, die auch eine weniger privilegierter Benutzer erledigen kann; wir empfehlen also, "root" ausschließlich für Systemverwaltungsaufgaben heranzuziehen. Deshalb raten wir davon ab, Unterverzeichnisse für Post oder andere Anwendungen im Benutzerverzeichnis von "root" zu erlauben, und schlagen stattdessen vor, Post für Verwaltungsämter wie "root", "postmaster" oder "webmaster" an einen geeigneten Benutzer automatisch weiterzuleiten.

17. Ursprünglich waren die `/sbin`-Programme in `/etc` abgelegt.

18. Es ist recht einfach zu entscheiden, was ins *"sbin"*-Verzeichnis gehört: Wenn ein einfacher Benutzer (kein Systemverwalter) ein Programm irgendwann einmal direkt benutzen wird, dann gehört es in eines der *"bin"*-Verzeichnisse. Normalbenutzer sollten nicht gezwungen sein, eins der `sbin`-Verzeichnisse in ihren Programmsuchpfad einbinden zu müssen.

Beispielsweise gehört auch das vom Normalbenutzer wenig genutzte **chfn** nach `/usr/bin`. Wenn auch **ping** für "root" unentbehrlich ist, so wird es doch oft von Normalbenutzern verwendet und gehört deshalb nach `/bin`.

Wir empfehlen, dass Benutzer Lese- und Ausführungsrechte für alles in `/sbin` bekommen, ausgenommen vielleicht für Programme mit setuid- und setgid-Funktionen. `/bin` und `/sbin` wurden nicht aus Sicherheitsgründen getrennt und auch nicht, um Benutzern die Sicht auf das Betriebssystem zu nehmen. Vielmehr soll unterschieden werden zwischen den Programmen, die jeder benutzt und denjenigen, die in erster Linie für Verwaltungsaufgaben vorgesehen sind. Es bringt keine Sicherheitsvorteile, wenn man `/sbin` für Benutzer sperrt.

19. Das ist wichtig, weil gerade dieser Bereich oft sowohl durch die Distribution als auch durch den Verwalter angelegte Dateien enthält.

Kapitel 4. Die /usr-Hierarchie

4.1. Zweck

`/usr` ist der zweite wichtige Bereich des Dateisystems. Es handelt sich um gemeinsam nutzbare Daten, auf die nur lesend zugegriffen wird. Das heißt, `/usr` kann von mehreren FHS-konformen Rechnern genutzt werden und darf nicht beschrieben werden. Alle Daten, die rechnerspezifisch sind, oder solche, die sich ändern, werden woanders gespeichert.

Selbst große Softwarepakete dürfen keine eigenen direkten Unterverzeichnisse unter `/usr` haben.

4.2. Anforderungen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen werden in `/usr` benötigt:

Verzeichnis	Beschreibung
`bin`	Die meisten Benutzerkommandos
`include`	Header-Dateien für C-Programme
`lib`	Bibliotheken
`local`	Lokale Hierarchie (nach der Systeminstallation zunächst leer)
`sbin`	Weniger wichtige Systemkommandos
`share`	Architekturunabhängige Daten

4.3. Optionen

Verzeichnis	Beschreibung
`X11R6`	X-Window-System Version 11 Release 6 (optional)
`games`	Spiele und Lernprogramme
`lib<Suffix>`	Bibliotheken in alternativem Format
`src`	Quellen

Für das X-Windows-System wird hier aufgrund der bemerkenswerten Tradition und und wegen der breiten Akzeptanz eine Ausnahme gemacht.

Die folgenden symbolischen Links dürfen vorhanden sein. Das ist der Notwendigkeit geschuldet mit älteren Systemen kompatibel zu bleiben, solange bis man annehmen kann, dass alle Implementierungen die `/var`-Hierarchie nutzen.

```
/usr/spool -> /var/spool
```

```
/usr/tmp -> /var/tmp
/usr/spool/locks -> /var/lock
```

Sobald ein System einen der genannten symbolischen Links nicht mehr braucht, kann er gelöscht werden.

4.4. /usr/X11R6 : X-Window-System Version 11 Release 6 (optional)

4.4.1. Zweck

Diese Hierarchie ist reserviert für das X-Window-System Version 11 Release 6 und die zugehörigen Dateien.

Zur Vereinfachung und um XFree86 besser mit den X-Window-Systemen auf anderen Platformen kompatibel zu machen, müssen folgende symbolischen Links unterhalb von `/usr/X11R6` vorhanden sein:

```
/usr/bin/X11 -> /usr/X11R6/bin
/usr/lib/X11 -> /usr/X11R6/lib/X11
/usr/include/X11 -> /usr/X11R6/include/X11
```

Allgemein darf Software nicht auf dem Weg über die genannten Links installiert oder gesteuert werden. Die Links sind nur für die Benutzer. Das Problem hängt mit dem X-Window-Release zusammen - in Zeiten des Übergangs ist es unmöglich zu wissen, welches Release von X11 genutzt wird.

4.4.2. Optionen

Rechnerspezifische Daten in `/usr/X11R6/lib/X11` sollen nur als Muster betrachtet werden. Anwendungen, die Informationen über den aktuellen Rechner benötigen, müssen sich an eine Konfigurationsdatei in `/etc/X11` wenden, welches dann verknüpft sein kann mit einer Datei in `/usr/X11R6/lib`. [1]

4.5. /usr/bin : Die meisten Benutzerkommandos

4.5.1. Zweck

Dies ist das wichtigste Verzeichnis für Kommandos im ganzen System.

4.5.2. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/usr/bin` dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
`mh`	Kommandos des MH Postverteilsystems (optional)

Wenn `/usr/X11R6/bin` existiert, dann muss `/usr/bin/X11` ein symbolischer Link darauf sein.

Die folgenden Dateien oder symbolischen Links zu Dateien müssen in `/usr/bin` dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Kommando	Beschreibung
perl	Perl - die praktische Extraktions- und Berichtssprache (optional)
python	Python - der Interpreter (optional)
tclsh	Einfache Kommandooberfläche mit Tcl Interpreter (optional)
wish	Einfache Fensteroberfläche mit Tcl/Tk (optional)
expect	Programm für interaktiven Dialog (optional)

Hintergrund

Weil Interpreter für Shell-Skripte (aufgerufen durch `#!<Pfad>` in der ersten Zeile) sich sonst nicht auf den Pfad verlassen könnten, ist es von Vorteil, die Plazierung zu standardisieren. Für die "Bourne shell" ebenso wie für die "C shell" ist der Ort bereits festgelegt mit `/bin`. Perl, Python und Tcl aber findet man an den unterschiedlichsten Stellen. Hier dürfen sie symbolische Links auf die tatsächlichen Interpreter sein.

4.6. /usr/include : Verzeichnis für Include-Dateien

4.6.1. Zweck

Hier sollen alle systemweit genutzten Include-Dateien für die Programmiersprache C abgelegt werden.

4.6.2. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/usr/include`

dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
bsd	Include-Dateien für BSD-Kompatibilität (optional)

Wenn `/usr/X11R6/include/X11` existiert, dann muss `/usr/include/X11` ein symbolische Link darauf sein.

4.7. /usr/lib : Bibliotheken für Programmierung und Softwarepakete

4.7.1. Zweck

`/usr/lib` enthält Objektdateien, Bibliotheken und interne lauffähige Programme, die nicht für die direkte Nutzung durch Benutzer oder Shell-Skripte vorgesehen sind. [2]

Anwendungsprogramme dürfen je ein Unterverzeichnis von `/usr/lib` benutzen. Wenn ein Anwendungsprogramm das tut, müssen alle architekturabhängige Daten, die ausschließlich von diesem Programm gebraucht werden, dort gespeichert werden. [3]

4.7.2. Optionen

Historisch bedingt muss **/usr/lib/sendmail** ein symbolischer Link auf **/usr/sbin/sendmail** sein, wenn letzteres existiert. [4]

Wenn `/lib/X11` existiert, dann muss `/usr/lib/X11` ein symbolischer Link darauf sein, oder auf das, worauf ein symbolische Link `/lib/X11` zeigt. [5]

4.8. /usr/lib<Suffix> : Bibliotheken in alternativem Format (optional)

4.8.1. Zweck

`/usr/lib<Suffix>` spielt dieselbe Rolle wie `/usr/lib`, nur für alternative Binärformate. Ausnahmen: Die symbolischen Links `/usr/lib<Suffix>/sendmail` und `/usr/lib<Suffix>/X11` sind nicht vorgeschrieben. [6]

4.9. /usr/local : Locale Hierarchie

4.9.1. Zweck

Die `/usr/local`-Hierarchie benutzt der Systemverwalter, wenn er Software lokal installiert. Es muss vor dem Überschreiben im Zuge einer Aktualisierung des Systems geschützt werden. Es darf für Programme genutzt werden, die von mehreren Rechnern genutzt werden kann, die aber nicht unter `/usr` gefunden werden können.

Lokal installierte Software gehört nach `/usr/local` und nicht nach `/usr`, solange sie nicht Software in `/usr` ersetzen oder erweitern soll. [7]

4.9.2. Anforderungen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/usr/local` vorhanden sein:

Verzeichnis	Beschreibung
`bin`	Lokal installierte Kommandos
`etc`	Rechnerspezifische Einstellungen für lokal installierte Kommandos
`games`	Lokal installierte Spiele
`include`	Header-Dateien lokal installierter C-Programme
`lib`	Lokal installierte Bibliotheken
`man`	Lokal installierte Online-Handbücher ("man pages")
`sbin`	Lokal installierte Systemkommandos
`share`	Lokale architekturunabhängige Hierarchie
`src`	Quellen lokal installierter Software

Soll das System FHS-konform sein, so darf es unter `/usr/local` ausschließlich die aufgeführten Verzeichnisse geben.

4.9.3. Optionen

Wenn es ein Verzeichnis `/lib<Suffix>` oder `/usr/lib<Suffix>` gibt, so muss es auch die Entsprechung unter `/usr/local` geben.

`/usr/local/etc` darf ein symbolischer Link auf `/etc/local` sein.

Hintergrund

Die durch `/usr/local/etc` erreichte Konsistenz ist von Vorteil für alle, die installieren müssen; auf anderen Systemen wird es bereits so genutzt. Da `/usr/local` komplett gesichert werden muss, um

ein System wiederherstellen zu können, bedeutet das auch keinen zusätzlichen Wartungsaufwand. Ein symbolischer Link nach `/etc/local` genügt, wenn auf einem System alle Einstellungen in derselben Hierarchie erscheinen sollen.

Beachten Sie, dass `/usr/etc` nach wie vor nicht erlaubt ist: Programme unter `/usr` sollen ihre Einstellungen unter `/etc` speichern.

4.9.4. /usr/local/share

Die Anforderungen für dieses Verzeichnis sind diegleichen wie für `/usr/share`. Einzige zusätzliche Regel ist, dass `/usr/local/share/man` und `/usr/local/man` dasselbe bedeuten müssen, also eines davon ein symbolischer Links aufs andere sein muss. [8]

4.10. /usr/sbin : Weniger wichtige Systemkommandos

4.10.1. Zweck

Dieses Verzeichnis enthält weniger wichtige Systemkommandos für den Systemverwalter. Verwaltungsprogramme aber, die für Reparatur und Wiederherstellen des Systems, für das Einhängen von `/usr` oder andere wesentliche Aufgaben gebraucht werden, gehören stattdessen nach `/sbin`, [9]

4.11. /usr/share : Architekturunabhängige Daten

4.11.1. Zweck

Die `/usr/share`-Hierarchie ist für alle architekturunabhängigen Dateien gedacht, auf die nur lesend zugegriffen wird. [10]

Diese Hierarchie soll gemeinsam nutzbar sein von allen Architekturen unter einem Betriebssystem; also soll zum Beispiel ein Rechenzentrum ("site") mit i386-, Alpha- und PPC-Rechnern nur ein einziges `/usr/share`-Verzeichnis betreiben, das zentral eingebunden wird. Beachten Sie aber, dass `/usr/share` in der Regel nicht von verschiedenen Betriebssystemen oder verschiedenen Versionen eines Betriebssystems gemeinsam genutzt wird.

Programme oder Pakete mit Daten, die nicht verändert werden, sollten diese Daten in `/usr/share` (oder `/usr/local/share`, wenn lokal installiert) plazieren. Es wird empfohlen, dafür Unterverzeichnisse von `/usr/share` zu benutzen.

Spieledaten in `/usr/share/games` müssen rein statische Daten sein. Alle veränderbaren Dateien, wie Punkte-Dateien, Log-Dateien usw. müssen in `/var/games` gespeichert werden.

4.11.2. Anforderungen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/usr/share` vorhanden sein:

Verzeichnis	Beschreibung
`man`	Online-Handbücher
`misc`	Verschiedenes

4.11.3. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/usr/share` dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
`dict`	Wortlisten (optional)
`doc`	Verschiedene Dokumentationen (optional)
`games`	Statische Dateien für /usr/games (optional)
`info`	GNU-Info-System (optional)
`locale`	Lokalanpassungen (optional)
`nls`	Nachrichtenkataloge für NLS (optional)
`sgml`	SGML-Daten (optional)
`terminfo`	Verzeichnisse der Terminfo-Datenbank (optional)
`tmac`	troff-Makros, die nicht zu groff gehören (optional)
`xml`	XML-Daten (optional)
`zoneinfo`	Information und Konfiguration zu den Zeitzonen (optional)

Es wird empfohlen, anwendungsspezifische, architekturunabhängige Verzeichnisse hier zu plazieren. Das schließt `groff`, `perl`, `ghostscript`, `texmf`, und `kbd` für Linux oder `syscons` für BSD ein. Es liegt aber im Ermessen der Distribution, sie nach `/usr/lib` zu plazieren (wegen Rückwärtskompatibilität). Ähnlich darf eine Distribution auch zusätzlich zu `/usr/share/games` eine `/usr/lib/games`-Hierarchie benutzen.

4.11.4. /usr/share/dict : Wortlisten (optional)

4.11.4.1. Zweck

Dieses Verzeichnis ist Heimat für Wortlisten des Systems. Traditionell enthält es nur die englische `words`-Datei, die von **look(1)** und anderen Rechtschreibprogrammen benutzt wird. `words` darf englische oder amerikanische Schreibweisen enthalten.

Hintergrund

Dass hier nur Wortlisten gespeichert sind, liegt daran, dass dies die einzigen Dateien sind, die allen Rechtsschreibungs-Prüfprogrammen gemeinsam sind.

4.11.4.2. Optionen

Die folgenden Dateien oder symbolischen Links zu Dateien müssen in `/usr/share/dict` vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Datei	Beschreibung
`words`	Liste englischer Wörter (optional)

Wo sowohl amerikanische als auch englische Schreibweisen benutzt werden, kann `words` auf `/usr/share/dict/american-english` oder `/usr/share/dict/british-english` verweisen.

Wortlisten anderer Sprachen dürfen unter dem englischen Namen für diese Sprache ergänzt werden, z.B. `/usr/share/dict/french`, `/usr/share/dict/danish` usw. Wenn passend für die jeweilige Sprache, sollten diese einen ISO-8859-Zeichensatz verwenden. Wenn möglich, sogar den Latin1-(ISO 8859-1)-Zeichensatz; das ist aber nicht oft möglich.

Wenn vorhanden, müssen auch weitere Wortlisten hier abgelegt werden.

4.11.5. /usr/share/man : Online-Handbücher ("man pages")

4.11.5.1. Zweck

Dieser Abschnitt beschreibt wie die Online-Handbücher systemweit organisiert sind. Vergleiche auch den Abschnitt über `/var/cache/man`.

Das primäre `<Handbuch>`-Verzeichnis ist `/usr/share/man`. Dieses enthält die Handbücher zu Kommandos und Daten in den Dateisystemen `/` und `/usr`. [11]

Online-Handbücher werden gespeichert in `<Handbuch>/<Lokalanpassung>/man<Abschnitt>/<Architektur>`. Eine Erklärung zu `<Handbuch>`, `<Lokalanpassung>`, `<Abschnitt>` und `<Architektur>` folgt weiter unten.

Es folgt eine Beschreibung der einzelnen Abschnitte:

- `man1`: Benutzerkommandos

  Dieser Abschnitt enthält Online-Handbücher der öffentlich zugänglichen Kommandos. Das Wichtigste für einen Benutzer ist hier dokumentiert.

- `man2`: Systemfunktionen

Dieser Abschnitt dokumentiert alle Systemfunktionen (Anfragen an den Systemkern).

- `man3`: Bibliotheksfunktionen

  Abschnitt 3 dokumentiert Routinen aus Programmbibliotheken, die keine direkten Systemanfragen sind. Er und Abschnitt 2 sind nur für Programmierer interessant.

- `man4`: Gerätedateien

  Abschnitt 4 dokumentiert Gerätedateien, zugehörige Treiber und die verfügbare Netzunterstützung. Üblicherweise gehören dazu die Gerätedateien aus `/dev` und die Kernschnittstellen für Netzprotokolle.

- `man5`: Dateiformate

  In Abschnitt 5 ist die Syntax vieler Datendateien dokumentiert. Darunter fallen Include-Dateien, Programmausgaben und Systemdateien. (Unter anderem die `*.conf`-Dateien in `/etc`: A.d.Ü.)

- `man6`: Spiele

  Dieser Abschnitt dokumentiert Spiele, Demo- oder ganz allgemein: belanglose Programme. Darüber, wie wichtig das überhaupt ist, gehen die Meinungen auseinander.

- `man7`: Verschiedenes

  Online-Handbüchern, die schwer einzuordnen sind, wurde der Abschnitt 7 zugewiesen. Hier findet man Dokumentation zu troff und anderen Textverarbeitungsmakros.

- `man8`: Systemverwaltung

  Hier sind Programme dokumentiert, die der Verwalter für Betrieb und Wartung nutzt. Einzelne Programme können aber auch für den normalen Benutzer nützlich sein.

4.11.5.2. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `<Handbuch>/<Lokalanpassung>` vorhanden sein - solange sie nicht leer sind: [12]

Verzeichnis	Beschreibung
`man1`	Benutzerkommandos (optional)
`man2`	Systemfunktionen (optional)
`man3`	Bibliotheksfunktionen (optional)
`man4`	Gerätedateien (optional)
`man5`	Dateiformate (optional)

Verzeichnis	Beschreibung
`man6`	Spiele (optional)
`man7`	Verschiedenes (optional)
`man8`	Systemverwaltung (optional)

Die Komponente `<Abschnitt>` ist die Abschnittsnummer der Online-Handbücher.

Innerhalb von `/usr/share/man` muss eine Struktur vorgesehen werden, die Online-Handbücher in verschiedenen Sprachen (auch gleichzeitig) unterstützt. Wichtig in dem Zusammenhang ist außer der Sprache (mit den länderspezifischen Besonderheiten) auch der Zeichensatz.

Die Benennung der Lokalanpassungs-Unterverzeichnisse von `/usr/share/man` basiert auf dem Anhang B des POSIX 1003.1 Standards, wo der "locale identification string" beschrieben ist - die weitestgehend akzeptierte Methode, eine kulturelle Umgebung zu kennzeichnen. Dieser `<Lokalanpassung>`s-Name ist definiert durch:

```
<Sprache>[_<Region>][.<Zeichensatz>][,<Version>]
```

Der Bestandteil `<Sprache>` muss dem Standard ISO 639 entnommen werden; das ist ein Codierung der Sprachnamen. Er ist zwei Zeichen lang und besteht aus Kleinbuchstaben.

Der Bestandteil `<Region>` muss, wenn möglich, aus dem 2-Zeichen-Code des Standards ISO 3166 stammen; das ist ein Code für Ländernamen. (Die meisten von Ihnen werden den 2-Zeichen-Ländercode aus der E-Mail-Adresse kennen.) Er ist zwei Zeichen lang und besteht aus Großbuchstaben. [13]

Der Bestandteil `<Zeichensatz>` nennt den Standard, durch den der Zeichensatz beschrieben wird. Wenn `<Zeichensatz>` nur eine Zahl ist, ist das die Nummer des Internationalen Standards, der den Zeichensatz beschreibt. Es wird empfohlen, eine numerische Representation zu nutzen, wenn das möglich ist (besonders die ISO Standards), ohne weitere Trennzeichen, und wenn mit Buchstaben, dann mit Kleinbuchstaben.

Ein `<Version>`s-Bestandteil des `<Lokalanpassung>`s-Names kann an den `<Zeichensatz>`-Bestandteil durch ein Komma getrennt angehängt werden. Das kann dazu benutzt werden, um kulturelle Unterschiede zu machen, beispielsweise zwischen der alphabetischen Sortierreihenfolge und einer systematischen Ordnung. Dieser Standard empfiehlt `<Version>` nur zu benutzen, wenn es wirklich notwendig ist.

Systeme, die nur eine Sprache und nur einen Zeichensatz für alle Online-Handbücher nutzen, dürfen auf den `<Lokalanpassung>`s-Namen verzichten und stattdessen alle Handbücher direkt unter `<Handbuch>` ablegen. Zum Beispiel kann ein System, das nur englische Online-Handbücher im ASCII-Zeichensatz anbietet, diese, das heißt die `man<Abschnitt>`-Verzeichnisse, direkt in `/usr/share/man` speichern. (Das ist in der Tat das traditionelle Arrangement.)

Länder, für die es einen allgemein anerkannten Standard-Zeichensatz gibt, dürfen auf den `<Zeichensatz>`-Bestandteil verzichten; es wird aber empfohlen, ihn trotzdem zu nutzen, insbesondere für Länder mit konkurrierenden Standards.

Beispiele:

Sprache	Land	Zeichensatz	Verzeichnis
Englisch	—	ASCII	/usr/share/man/en
Englisch	Vereinigtes Königreich	ISO 8859-15	/usr/share/man/en_GB
Englisch	USA	ASCII	/usr/share/man/en_US

Sprache	Land	Zeichensatz	Verzeichnis
Französisch	Kanada	ISO 8859-1	/usr/share/man/fr_CA
Französisch	Frankreich	ISO 8859-1	/usr/share/man/fr_FR
Deutsch	Deutschland	ISO 646	/usr/share/man/de_DE.646
Deutsch	Deutschland	ISO 6937	/usr/share/man/de_DE.6937
Deutsch	Deutschland	ISO 8859-1	/usr/share/man/de_DE.88591
Deutsch	Schweiz	ISO 646	/usr/share/man/de_CH.646
Japanisch	Japan	JIS	/usr/share/man/ja_JP.jis
Japanisch	Japan	SJIS	/usr/share/man/ja_JP.sjis
Japanisch	Japan	UJIS (or EUC-J)	/usr/share/man/ja_JP.ujis

Ganz ähnlich müssen Vorkehrungen getroffen werden für architekturabhängige Online-Handbücher, wie zum Beispiel solche zu Gerätedateien oder "Low-Level"-Kommandos für die Systemverwaltung. Diese müssen im entsprechenden `<Architektur>`-Unterverzeichnis des `man<Abschnitt>`-Verzeichnisses plaziert werden. Beispielsweise könnte ein Handbuch für das i386-spezifische "ctrlaltdel"-Kommando gespeichert sein unter `/usr/share/man/<Lokalanpassung>/man8/i386/ctrlaltdel.8`.

Handbücher zu Kommandos und Daten in `/usr/local` werden unter `/usr/local/man` gespeichert. Handbücher zu X11R6 stehen unter `/usr/X11R6/man`. Es folgt daraus, dass alle Handbuchhierarchien dieselbe Struktur haben müssen wie `/usr/share/man`.

Die "cat page"-Abschnitte (`cat<Abschnitt>`), die formatierte Handbücher enthalten, dürfen ebenfalls in den `<Handbuch>/<Lokalanpassung>`-Unterverzeichnissen vorhanden sein. Sie sind aber weder zwingend notwendig, noch dürfen sie anstelle der Handbücher im nroff-Format treten.

Die numerierten Abschnitte "1" bis "8" wurden traditionell definiert. Es gilt, dass die Dateinamen der Handbücher in den jeweiligen Abschnitten mit `.<Abschnitt>` enden.

Darüberhinaus haben einige größere Anwendungshandbücher einen Suffix zum Dateinamen. Zum Beispiel ist bei den Handbüchern des Postverteilsystems MH die Buchstaben `mh` an den Dateinamen angehängt. Und alle Dateinamen der X-Window-Handbücher enden mit `x`.

Die Regel, verschiedensprachige Handbücher in dafür vorgesehene Unterverzeichnisse von `/usr/share/man` zu plazieren, gilt auch für Online-Handbücher in `/usr/local/man` und `/usr/X11R6/man`. (Ebenso gilt sie - siehe weiter unten - für `/var/cache/man`.)

4.11.6. /usr/share/misc : Verschiedenes

Dieses Verzeichnis enthält verschiedene architekturunabhängige Daten, die ohne eigene Unterverzeichnisse von `/usr/share` auskommen.

4.11.6.1. Optionen

Die folgenden Dateien oder symbolischen Links zu Dateien müssen in `/usr/share/misc` vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Datei	Beschreibung
`ascii`	ASCII-Zeichensatztabelle (optional)
`magic`	Liste "magischer Zahlen" für das Kommando **file** (optional)
`termcap`	Terminal-Daten (optional)
`termcap.db`	Terminal-Daten (optional)

Weitere (Anwendungs-)Daten können hier erscheinen. Nach eigenem Ermessen kann die Distribution diese Daten auch nach `/usr/lib` schreiben. [14]

4.11.7. /usr/share/sgml : SGML-Daten (optional)

4.11.7.1. Zweck

`/usr/share/sgml` enthält architekturunabhängige Daten, die von SGML-Anwendungen benutzt werden, wie zu Beispiel Kataloge (allerdings nicht die Zentralkataloge, siehe: `/etc/sgml`), DTDs, Entitäten oder Formatdateien ("style sheets").

4.11.7.2. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/usr/share/sgml` dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
`docbook`	docbook DTD (optional)
`tei`	tei DTD (optional)
`html`	html DTD (optional)
`mathml`	mathml DTD (optional)

Andere Dateien, die sich nicht auf eine DTD beziehen, gehören in eigene Unterverzeichnisse.

4.11.8. /usr/share/xml : XML-Daten (optional)

4.11.8.1. Zweck

`/usr/share/xml` enthält architekturunabhängige Daten, die von XML-Anwendungen benutzt werden, wie zu Beispiel Kataloge (allerdings nicht die Zentralkataloge, siehe: `/etc/xml`), DTDs, Entitäten oder

Formatdateien ("style sheets").

4.11.8.2. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/usr/share/xml` dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
`docbook`	docbook XML DTD (optional)
`xhtml`	XHTML DTD (optional)
`mathml`	MathML DTD (optional)

4.12. /usr/src : Quellen (optional)

4.12.1. Zweck

In diesem Verzeichnis kann Quellcode angelegt werden. Es ist nur zum Nachschlagen gedacht. [15]

Fußnoten

1. Beispiele solcher Konfigurationsdateien sind `Xconfig`, `XF86Config` oder `system.twmrc`.
2. Architekturunabhängige programmspezifische statische Dateien und Verzeichnisse gehören aber nach `/usr/share`.
3. Ein Beispiel wäre das Unterverzeichnis `perl5` für Perl-5-Module und Bibliotheken.
4. Auch einige Kommandos wie **makewhatis** und **sendmail** waren traditionell in `/usr/lib` plaziert. **makewhatis** ist aber ein Kommando und gehört deshalb in ein Binärverzeichnis; Benutzer rufen nur **catman**. Neuere **sendmail**-Versionen werden inzwischen standardmäßig nach `/usr/sbin` geschrieben; bei Systeme mit einem *sendmail*-kompatiblen Postverteiler muss **/usr/sbin/sendmail** ein symbolischer Link auf das entsprechende Programm sein.
5. Rechnerspezifische Daten des X-Window-Systems dürfen nicht unter `/usr/lib/X11` gespeichert sein. Rechnerspezifische Daten wie `Xconfig` oder `XF86Config` gehören nach `/etc/X11`. Das betrifft auch Konfigurationsdateien wie zum Beispiel `system.twmrc`, selbst wenn diese nur symbolische Links zu Konfigurationsdateien, evtl. unter `/usr/X11R6/lib/X11`, sind.
6. Im dem Fall, dass `/usr/lib` und `/usr/lib<Suffix>` dasselbe sind, d.h. das eine ein symbolische Link auf das andere ist, gibt es auch die genannten Dateien und das zur jeweiligen Anwendung gehörende Unterverzeichnis.
7. Software in `/` oder `/usr` darf bei Systemaktualisierung überschrieben werden. (Wir raten aber davon ab, Daten in `/etc` zu überschreiben.) Deshalb darf lokale Software nicht ohne einen guten Grund

außerhalb von `/usr/local` plaziert werden.

8. Möglicherweise wird `/usr/local/man` in zukünftigen FHS-Versionen abgeschafft, also ist es wohl vernünftig, dieses zu einem symbolischen Link zu machen.
9. Lokal installierte Systemverwaltungsprogramme wiederum gehören nach `/usr/local/sbin`.
10. Viele dieser Dateien standen ursprünglich in `/usr` (`man`, `doc`) oder `/usr/lib` (`dict`, `terminfo`, `zoneinfo`).
11. Wie man sehen kann, gibt es keine Online-Handbücher in `/`, denn sie werden weder für den Systenstart noch im Notfall gebraucht. Wirklich!
12. Hat beispielsweise `/usr/share/man/de` keine Handbücher in Abschnitt 4 (Gerätedateien), so darf `/usr/share/man/de/man4` fehlen.
13. Eine wichtige Ausnahme bildet das Vereinigte Königreich, für welches im Standard ISO 3166 als Code ’GB’ steht. Die meisten britischen E-Mail-Adressen aber enthalten das Kürzel ’UK’.
14. Solche Dateien sind etwa: `airport`, `birthtoken`, `eqnchar`, `getopt`, `gprof.callg`, `gprof.flat`, `inter.phone`, `ipfw.samp.filters`, `ipfw.samp.scripts`, `keycap.pcvt`, `mail.help`, `mail.tildehelp`, `man.template`, `map3270`, `mdoc.template`, `more.help`, `na.phone`, `nslookup.help`, `operator`, `scsi_modes`, `sendmail.hf`, `style`, `units.lib`, `vgrindefs`, `vgrindefs.db`, `zipcodes`.
15. Ganz allgemein soll nicht von diesem Verzeichnis aus kompiliert werden.

Kapitel 5. Die /var Hierarchie

5.1. Zweck

`/var` enthält veränderliche Datendateien. Es sind enthalten: Spoolverzeichnisse und -dateien, Verwaltungsdaten, Logbücher und temporäre Dateien.

Teile von `/var` können nicht von mehreren Systemen genutzt werde. Das sind beispielsweise `/var/log`, `/var/lock` und `/var/run`. Andere Teile dürfen gemeinsam genutzt werden, insbesondere gilt das für `/var/mail`, `/var/cache/man`, `/var/cache/fonts` und `/var/spool/news`.

`/var` wird hier so spezifiziert, dass es möglich wird, `/usr` im Nur-Lese-Modus einzubinden. All das, was früher in `/usr` gespeichert wurde und auf das im laufenden Betrieb, also über Installation und Wartung hinaus, schreibend zugegriffen wird, gehört jetzt nach `/var`.

Falls es nicht möglich ist, für `/var` eine eigene Partition einzurichten, so ist es trotzdem oft von Vorteil `/var` aus dem Wurzeldateisystem herauszulösen und in der `/usr`-Partition unterzubringen. (Grund dafür kann die Absicht sein, die Wurzelpartition klein zu halten, oder es mag dort Platzprobleme geben.) Auf jeden Fall darf `/var` kein Link auf `/usr` sein, weil das Schwierigkeiten bei der Trennung von `/usr` und `/var` geben würde und außerdem Namenskonflikte auftreten könnten. Machen Sie stattdessen `/var` zu einem Link auf `/usr/var`.

Anwendungsprogramme dürfen der obersten Ebene von `/var` keine Unterverzeichnisse hinzufügen. Solche Verzeichnisse sollten nur dann erstellt werden, wenn Sie systemweite Bedeutung haben, und dann auch nur in Abstimmung mit der FHS-Mailingliste.

5.2. Anforderungen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen werden in `/var` benötigt:

Verzeichnis	Beschreibung
`cache`	Pufferspeicher für Anwendungen
`lib`	Variable Statusinformationen
`local`	Variable Daten für /usr/local
`lock`	Sperrdateien
`log`	Logbuchdateien und -verzeichnisse
`opt`	Variable Daten für /opt
`run`	Daten aktiver Prozesse
`spool`	Spool-Dateien
`tmp`	Temporäre Dateien, die aber beim Systemneustart erhalten bleiben

Ein paar Verzeichnisse sind insofern "reserviert", als sie nicht willkürlich von neuen Anwendungen benutzt werden dürfen, weill sich sonst Konflikte mit älterer oder auch lokaler Praxis ergeben würden. Diese sind:

```
/var/backups
/var/cron
/var/msgs
/var/preserve
```

5.3. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/var` dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
`account`	Prozesskonten (optional)
`crash`	Systemdumps (optional)
`games`	Variable Spieledaten (optional)
`mail`	Postfächer (optional)
`yp`	Datenbank des NIS ("Network Information Service") (optional)

5.4. /var/account : Prozesskonten (optional)

5.4.1. Zweck

Dieses Verzeichnis enthält die Konten der aktuell aktiven Prozesse sowie Daten über zusammengesetzte Prozesse, wie sie in einigen UNIX-ähnlichen Betriebssystemen von **lastcomm** und **sa** benutzt werden.

5.5. /var/cache : Pufferspeicher für Anwendungen

5.5.1. Zweck

`/var/cache` ist vorgesehen als Datenpuffer für Anwendungsproramme. Das sind üblicherweise lokal generierte Daten, deren Erzeugung zeitaufwändig oder I-O-intensiv ist. Die Anwendungen müssen in der Lage sein, solche Daten neu zu erzeugen oder neu zu laden. Anders als bei `/var/spool` können Pufferdaten gelöscht werden, ohne verloren zu gehen. Pufferdaten müssen zwischen Programmaufruf und Systemneustart gültig bleiben.

Dateien unter `/var/cache` dürfen nach den Regeln der Anwendung und auch nach Maßgabe des Systemverwalters ihre Gültigkeit verlieren. Die Anwendung muss immer in der Lage sein, mit einem manuellen Löschen der Pufferdateien (in der Regel veranlasst durch Platzmangel) zurechtzukommen. Es gibt keine Vorschriften für das Datenformat.

Hintergrund

Die Existenz eines eigenen Verzeichnisses für Pufferdaten erlaubt dem Sytemverwalter, dieses bei Speicherung und Sicherung nach anderen Regeln zu behandeln, als die übrigen Verzeichnisse unter `/var`.

5.5.2. Optionen

Verzeichnis	Beschreibung
`fonts`	Lokal generierte Schrifttypen (optional)
`man`	Lokal formatierte Handbücher (optional)
`www`	Stellvertreter oder Datenpuffer für WWW (optional)
`<Paket>`	Paketspezifische Datenpuffer (optional)

5.5.3. /var/cache/fonts : Lokal generierte Schrifttypen (optional)

5.5.3.1. Zweck

Das Verzeichnis `/var/cache/fonts` soll als Speicher für alle dynamisch generierten Schrifttypen ("fonts") dienen. Insbesondere müssen alle von **mktexpk** erzeugten Schrifttypen in entsprechend benannten Unterverzeichnissen von `/var/cache/fonts` abgelegt werden. [1]

5.5.3.2. Optionen

Andere dynamisch erzeugten Schrifttypen dürfen ebenfall in passend benannten Unterverzeichnissen von `/var/cache/fonts` abgelegt werden.

5.5.4. /var/cache/man : Lokal formatierte Handbücher (optional)

5.5.4.1. Zweck

Dieses Verzeichnis stellt eine Standardplazierung für Betreiber zu Verfügung, die `/usr` als Nur-Lese-Partition eingebunden haben, aber trotzdem lokale Formatierung den Online-Handbcher erlauben wollen. Wo `/usr` beschreibbar eingebunden wird (z.B. auf einem Einzelplatzrechner), mag man sich gegen `/var/cache/man` entscheiden und stattdessen die formatierten Handbücher direkt in den `cat<section>`-Unterverzeichnissen von `/usr/share/man` ablegen. Wir empfehlen dagegen eine der folgenden Vorgehensweisen:

- Halten Sie alle Online-Handbücher nicht nur unformatiert sondern auch in der formatierten Version bereit.
- Unterbinden Sie das Abspeichern der formatierten Handbücher, so dass ein Online-Handbuch bei jedem Aufruf formatiert wird.
- Erlauben Sie, dass formatierte Online-Handbcher in `/var/cache/man` gepuffert werden.

Die Struktur von `/var/cache/man` muss zweierlei wiederspiegeln: die Vielzahl der Handbuch-Hierarchien und die mögliche Mehrsprachenunterstützung.

Ausgehend vom unformatierten Handbuch, das normalerweise in `<Pfad>/man/<Lokalanpassung>/man<Abschnitt>` untergebracht ist, wird das formatierte Handbuch in `/var/cache/man/<catPfad>/<Lokalanpassung>/cat<Abschnitt>` plaziert, wobei `<catPfad>` dadurch erzeugt wird, dass von `<Pfad>` jede führende Zeichenkette `usr` und jede schließende Zeichenkette `share` entfernt wird. (Man beachte, dass `<Lokalanpassung>` fehlen kann.) [2]

Handbücher, die nach `/var/cache/man` geschrieben wurden, können in die entsprechenden Verzeichnisse in der Quell-`man`-Hierarchie übertragen werden oder aber ihre Gültigkeit verlieren; ebenso können formatierte Handbücher in der Quell-`man`-Hierarchie ihre Gültigkeit verlieren, wenn sie über einen bestimmten Zeitraum nicht benutzt wurden.

Vorformatierte Handbücher von einem Nur-Lese-Medium (z.B. CD-ROM) müssen in der Quell-`man`-Hierarchie eingebunden werden (z.B. als `/usr/share/man/cat<Abschnitt>`). Denn `/var/cache/man` ist nur als beschreibbarer Puffer für formatierte Handbücher vorgesehen.

Hintergrund

Im Release 1.2 dieses Standards war hierfür `/var/catman` vorgesehen. Der Pfad wurde zu `/var/cache` geändert, um den dynamischen Charakter der formatierten Handbücher zu betonen. Der Verzeichnisname wurde in `man` abgeändert, um eine Erweiterung der Hierarchie für andere Formatierungen als "cat", wie etwa PostScript, HTML oder DVI, möglich zu machen.

5.6. /var/crash : Systemdumps (optional)

5.6.1. Zweck

Dieses Verzeichnis bietet Platz für Systemdumps. Zum Zeitpunkt der Veröffentlichung dieses Standards wurden solche Dumps von Linux nicht unterstützt, aber von anderen Systemen, die FHS-konform sein können.

5.7. /var/games : Variable Spieledaten (optional)

5.7.1. Zweck

Hier sollen alle variablen Daten von Spielen aus `/usr` abgelegt werden. `/var/games` soll die variablen Daten enthalten, die früher in `/usr` gespeichert wurden; statische Daten wie Hilfetexte, Level-Beschreibungen usw. müssen an anderer Stelle, wie etwa `/usr/share/games`, verbleiben.

Hintergrund

`/var/games` wurde als eigene Hierarchie bestimmt anstatt - wie es noch in Release 1.2 war - mit `/var/lib` zu vermischen. Die Trennung erlaubt lokale Kontrolle über Sicherungsstrategien, Rechte und Plattennutzung, sie erlaubt gemeinsame Nutzung zwischen Rechnern, und sie bereinigt das Durcheinander in `/var/lib`. Zu guter Letzt sei erwähnt, dass `/var/games` der traditionelle Pfad unter BSD ist.

5.8. /var/lib : Variable Statusinformationen

5.8.1. Zweck

Diese Hierarchie enthält Statusinformationen, die Anwendungen oder das System betreffen. Statusinformationen sind solche Daten, die von Programmen während der Laufzeit verändert werden, und die sich auf einen Rechner beziehen. Benutzer sollen Daten in `/var/lib` nicht verändern müssen.

Statusinformationen werden generell dazu genutzt, den Zustand einer Anwendung (oder einer Gruppe zusammengehörender Anwendungen) von einem Aufruf zum nächsten, oder von einer Instanz derselben Anwendung zur nächsten Instanz, zu erhalten. Statusinformationen sollten generell über einen Systemneustart hinaus gültig bleiben. Es soll sich nicht um Logbücher handeln und auch nicht um Spool-Daten.

Eine Anwendung (oder eine Gruppe zusammenhängender Anwendungen) muss für diese Daten ein Unterverzeichnis von `/var/lib` benutzen. Ein Unterverzeichnis, `/var/lib/misc`, ist vorgeschrieben für Statusdateien von solchen Anwendungen, die kein eigenes Unterverzeichnis benötigen, alle anderen Unterverzeichnisse sollen nur angelegt werden, wenn die zugehörige Anwendung auch in der jeweiligen Distribution enthalten ist. [3]

`/var/lib/<Name>` ist auch die Stelle, die die jeweiligen Paketverwaltungen der Distributionen benutzen müssen. Verschiedene Distributionen können, natürlich, verschiedene Namen benutzen.

5.8.2. Anforderungen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen werden in `/var/lib` benötigt:

Verzeichnis	Beschreibung
`misc`	Verschiedene Statusinformationen

5.8.3. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/var/lib` dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
<Editor>	Sicherungen und Status eines Editors (optional)
<Packer>	Daten der Paketverwaltung (optional)
<Paket>	Status eines Pakets oder Subsystems
`hwclock`	Status von **hwclock** (optional)
`xdm`	Status des X-Displaymanagers (optional)

5.8.4. /var/lib/<Editor> : Sicherungen und Status eines Editors (optional)

5.8.4.1. Zweck

Solche Verzeichnisse enthalten Sicherungsdateien, die von unerwarteten Pragrammabbrüchen eines Editors (z.B. **elvis**, **jove**, **nvi**) erzeugt werden.

Andere Editoren benötigen vielleicht kein Verzeichnis für Wiederherstellung nach einem Absturz, aber eventuell brauchen sie eine festgelegte Stelle für andere Laufzeitinformationen. Diese sollten in einem Unterverzeichnis von `/var/lib/<Editor>` abgelegt werden. (Beispielsweise plaziert der GNU-Emacs seine Sperrdateien in `/var/lib/emacs/lock`.)

Zukünftige Editoren brauchen vielleicht noch zusätzliche Statusinformationen, die über Wiederherstellung nach einem Absturz und über Sperrdateien hinaus gehen. Auch diese gehören dann unter `/var/lib/<Editor>`.

Hintergrund

Ältere Linux-Versionen und fast alle kommerziellen Anbieter benutzen `/var/preserve` für den vi und seine Nachahmer. Wie auch immer, jeder Editor benutzt sein eigenes Format für die genannten Dateien, und deshalb wird auch für jeden Editor ein eigenes Verzeichnis gebraucht.

Editorbezogene Sperrdateien unterscheiden sich in der Regel sehr von Geräte-Sperrdateien in `/var/lock`, und werden deshalb in `/var/lib` gespeichert.

5.8.5. /var/lib/hwclock : Status von hwclock (optional)

5.8.5.1. Zweck

Hier ist `/var/lib/hwclock/adjtime` enthalten.

Hintergrund

In der FHS 2.1 hieß diese Datei `/etc/adjtime`. Das war offensichtlich falsch, weil **hwclock** die Datei verändert.

5.8.6. /var/lib/misc : Verschiedene Statusinformationen

5.8.6.1. Zweck

Dieses Verzeichnis enthält variable Statusinformationen, die nicht in einem eigenen Unterverzeichnis von `/var/lib` untergebracht wurden. Man sollte sich um möglichst eindeutige Dateinamen bemühen, um Namenskonflikte zu vermeiden. [4]

5.9. /var/lock : Sperrdateien

5.9.1. Zweck

Sperrdateien sollten im Verzeichnis `/var/lock` gespeichert werden.

Sperrdateien für Geräte und andere Ressourcen, die von mehreren Anwendungen genutzt werden, wie etwa die Sperrdateien der seriellen Anschlüsse, die man ursprünglich in `/usr/spool/locks` oder `/usr/spool/uucp` finden konnte, müssen nun in `/var/lock` gespeichert werden. Die Konvention besagt, dass die Dateinamen mit "LCK.." beginnen worauf dann der Basisname des Gerätes folgt. Will man zum Beispiel /dev/ttyS0 sperren, so erzeugt man eine eine Datei namens "LCK..ttyS0". [5]

Für den Inhalt solcher Sperrdateien gilt das HDB-UUCP-Sperrdateiformat. Der Prozessbezeichner ("process identifier") PID wird in 10 Byte Länge als ASCII-Zeichen gespeichert gefolgt vom Zeichen für Zeilenvorschub. Ein Beispiel: Wenn der Prozess 1230 eine Sperrdatei erzeugt, dann enthält diese elf Zeichen: Leerzeichen, Leerzeichen, Leerzeichen, Leerzeichen, Leerzeichen, Leerzeichen, Eins, Zwei, Drei, Null und Zeilenvorschub.

5.10. /var/log : Logbuchdateien und -verzeichnisse

5.10.1. Zweck

Dieses Verzeichnis enthält verschiedene Logbücher. Die meisten Logbücher gehören hierher oder in ein entsprechendes Unterverzeichnis.

5.10.2. Optionen

Die folgenden Dateien oder symbolischen Links zu Dateien müssen in `/var/log` vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Datei	Beschreibung
`lastlog`	Protokoll der letzten Anmeldung jedes Benutzers
`messages`	Systemmeldungen von **syslogd**
`wtmp`	Protokoll aller An- und Abmeldungen

5.11. /var/mail : Postfächer (optional)

5.11.1. Zweck

Der Spool-Bereich für E-Mails muss unter `/var/mail` zu finden sein, die einzelnen Dateien müssen `<Benutzer>` heißen. [6]

Postfächer hier müssen im UNIX Standard-Postfachformat gespeichert werden.

Hintergrund

Dieses Verzeichnis wurde von `/var/spool/mail` hierher verschoben um den FHS mit nahezu allen UNIXen in Übereinklang zu bringen. Die Änderung ist wichtig fürs Zusammenspiel, weil oft ein einziges Verzeichnis `/var/mail` von mehreren Rechnern und verschiedenen Implementierungen von UNIX gemeinsam genutzt werden (trotz NFS-Sperrmechanismus).

Wichtig in diesem Zusammenhang ist, dass der Spool-Bereich nicht physisch verlagert werden muss. Allerdings müssen Programme und Header-Dateien so geändert werden müssen, dass sie `/var/mail` nutzen.

5.12. /var/opt : Variable Daten für /opt

5.12.1. Zweck

Variable Daten der Pakete in `/opt` müssen unter `/var/opt/<Name>` installiert werden, wobei `<Name>` der Name des Teilbaums in `/opt` ist, in dem sich die statischen Dateien des Pakets befinden. Ausgenommen sind die, die durch Dateien in `/etc` ersetzt wurden. Für `/var/opt/<Name>` ist keine interne Struktur vorgeschrieben.

Hintergrund

Siehe dazu den Hintergrund zu `/opt`.

5.13. /var/run : Daten aktiver Prozesse

5.13.1. Zweck

Dieses Verzeichnis enthält Informationen über das System seit seinem Start. Dateien in diesem Verzeichnis müssen zu Beginn des Systemstarts aufgeräumt werden, gelöscht oder geleert, je nachdem. Für Programme kann ein Unterverzeichnis von `/var/run` vorgesehen sein; dies wird empfohlen für Programme, die mehr als eine Laufzeit-Datei benutzen. [7] PID-Dateien ("process identifier files"), die ursprünglich einmal in `/etc` abgelegt waren, müssen nun in `/var/run` gespeichert werden. Namenskonvention für diese Dateien ist `<Programmname>.pid`. Beispielsweise heißt die PID-Datei des **crond**-Dämons `/var/run/crond.pid`.

5.13.2. Anforderungen

Das interne Format der PID-Dateien bleibt unverändert. Die Datei muss den Prozessbezeichner als Dezimalzahl im ASCII-Format gefolgt von einem Zeilenvorschub enthalten. Hat beispielsweise der **crond**-Dämon den Prozessbezeichner 25, so enthält die Datei `/var/run/crond.pid` drei Zeichen: Zwei, Fünf und das Zeichen für Zeilenvorschub.

Programme, die PID-Dateien zu lesen haben, sollten tolerant sein bei dem, was sie als gültig akzeptieren; z.B. sollten sie Leerstellen jeder Art ("whitespace") ignorieren, ebenso führende Nullen oder das Fehlen des nachfolgenden Zeilenvorschubs, ebenso zusätzliche Zeilen. Programme, die PID-Dateien erzeugen, sollten sich aber an die oben genannte einfache Spezifikation halten.

Hier findet man auch die Datei `utmp`, die Information darüber enthält, wer aktuell das System benutzt.

Systemprogramme, die sich um transiente Sockel von UNIX-Domänen kümmern, müssen diese hier ablegen.

5.14. /var/spool : Spool-Dateien

5.14.1. Zweck

`/var/spool` enthält Daten, die auf Weiterverarbeitung warten. Diese Daten bedeuten "zu erledigende Arbeiten" (durch Programme, Benutzer oder Verwalter); oft werden sie nach der Verarbeitung gelöscht. [8]

5.14.2. Optionen

Die folgenden Verzeichnisse oder symbolischen Links zu Verzeichnissen müssen in `/var/spool` dann vorhanden sein, wenn auch die entsprechenden Subsysteme eingerichtet sind:

Verzeichnis	Beschreibung
`lpd`	Druckerwarteschlange für den **lpd**-Dämon (optional)
`mqueue`	Ausgangswarteschlange für E-Mails (optional)
`news`	Spool-Verzeichnis für Nachrichten (optional)
`rwho`	Dateien des **rhowd**-Dämons (optional)
`uucp`	Spool-Verzeichnis für UUCP (optional)

5.14.3. /var/spool/lpd : Druckerwarteschlange für den lpd-Dämon (optional)

5.14.3.1. Zweck

Die Sperrdatei für **lpd**, `lpd.lock`, muss in `/var/spool/lpd` abgelegt werden. Wir schlagen vor, die Sperrdatei für jeden Drucker ins Spool-Verzeichnis des jeweiligen Druckers zu plazieren und sie `lock` zu nennen.

5.14.3.2. Optionen

Verzeichnis	Beschreibung
`<Drucker> (Im englischen Original steht hier - wohl versehentlich - "printer" ohne spitze Klammern: A.d.Ü.)`	Spool-Verzeichnis eines bestimmten Druckers (optional)

5.14.4. /var/spool/rwho : Dateien des rwhod-Dämons (optional)

5.14.4.1. Zweck

Dieses Verzeichnis enthält die Informationen des **rwhod**-Dämons über andere Systeme im lokalen Netz.


Hintergrund

Einige Auslieferungen ("releases") von BSD benutzen stattdessen `/var/rwho`. Angesichts der langwährenden Praxis anderer Systeme, die Daten in `/var/spool` zu plazieren, und angesichts der guten Übereinstimmung mit der Definition von 'Spool-Daten' wurde diese Stelle hier als angemessener erachtet.

5.15. /var/tmp : Temporäre Dateien, die aber beim Systemneustart erhalten bleiben

5.15.1. Zweck

Das Verzeichnis `/var/tmp` wird zur Verfügung gestellt für Programme, die temporäre Dateien oder Verzeichnisse benötigen, die bei einem Systemneustart erhalten bleiben. Deshalb sind Daten hier dauerhafter als solche in `/tmp`.

Dateien und Verzeichnisse in `/var/tmp` dürfen beim Systemstart nicht gelöscht werden. Auch wenn Daten in `/var/tmp` normalerweise nach Regeln des Betreibers gelöscht werden, empfehlen wir, das Löschen in größeren zeitlichen Abständen durchzuführen als für `/tmp`.

5.16. /var/yp : Datenbank des NIS ("Network Information Service") (optional)

5.16.1. Zweck

Variable Daten des NIS ("Network Information Service"), früher bekannt als YP ("Sun Yellow Pages"), gehören hierher.


Hintergrund

`/var/yp` ist das Standardverzeichnis für NIS (YP). Es wird fast ausschließlich für NIS-Dokumentation von NIS-Systemen genutzt. [9]

Fußnoten

1. Dieser Standard enthält nicht den TeX-Verzeichnisstruktur-Standard ("TeX Directory Structure" - ein Dokument über TeX-Dateien und Verzeichnisse), doch ist es sicher nützlich auch ihn zu lesen. Man findet ihn bei ftp://ctan.tug.org/tex/.
2. Beispielsweise wird `/usr/share/man/man1/ls.1` zu `/var/cache/man/cat1/ls.1` formatiert, und `/usr/X11R6/man/<Lokalanpassung>/man3/XtClass.3x` zu `/var/cache/man/X11R6/<Lokalanpassung>/cat3/XtClass.3x`.
3. Ein wichtiger Unterschied zur vorhergehenden Version dieses Standards ist, dass Anwendungen ein Unterverzeichnis von `/var/lib` benutzen müssen.
4. Hierher sollen Dateien, die zur Zeit noch auf BSD-Systemen unter `/var/db` stehen. Dazu gehören `locate.database`, `mountdtab` und die Symboldatenbank des Kerns.
5. Jede Anwendung, die `/dev/ttyS0` nutzen will, kann jetzt die Sperrdatei lesen und sich entsprechend verhalten. (Alle Sperren in `/var/lock` sollen für alle lesbar sein.)
6. `/var/mail` kann auch ein symbolischer Link auf ein anderes Verzeichnis sein.
7. Niemand außer privilegierten Benutzern (root oder von Benutzern gestartete Dämonen) (Im englischen Original werden - wohl versehentlich - die nicht-privilegierten Benutzer so erläutert: A.d.Ü) soll nach `/var/run` schreiben können. Schreibrechte für Benutzer in diesem Verzeichnis stellen ein gravierendes Sicherheitsproblem dar.
8. UUCP-Sperrdateien gehören nach `/var/lock` (siehe den entsprechenden Abschnitt weiter oben).
9. NIS soll man nicht verwechseln mit Sun NIS+, welches ein anderes Verzeichnis, nämlich `/var/nis` benutzt.

Kapitel 6. Betriebssystemspezifische Ergänzungen

Dieses Kapitel enthält zusätzliche Anforderungen und Empfehlungen, die nur bestimmte Betriebssysteme betreffen. Sie sollten den Basisregeln niemals widersprechen.

6.1. Linux

Dies sind die Ergänzungen für das Betriebsystem Linux,

6.1.1. / : Das Wurzelverzeichnis

Wenn der Systemkern sich in `/` befindet, empfehlen wir für Linux die Namen `vmlinux` oder `vmlinuz`, die auch in aktuellen Quellpaketen von Linux benutzt werden.

6.1.2. /bin : Wesentliche Kommandos (für alle Benutzer)

Wenn Linux-Systeme sie benötigen, plazieren sie folgende zusätzliche Dateien in `/bin`:

- **setserial**

6.1.3. /dev : Gerätedateien

Die folgenden Geräte muss es unter /dev geben:

`/dev/null`

Alle Daten, die hierher geschrieben werden, werden gelöscht. Lesen von hier gibt die EOF-Bedingung ("end of file") zurück.

`/dev/zero`

Dieses Gerät ist Quelle von Nullen. Alle Daten, die hierher geschrieben werden, werden gelöscht. Lesen von hier gibt soviele Bytes mit dem Wert Null zurück, wie angefordert wurden.

`/dev/tty`

Diese Gerätedatei steht für das Endgerät, das einen Prozess kontrolliert. Sobald diese Datei geöffnet wird, verhalten sich alle Lese- und Schreiboperationen, als ob das tatsächliche Endgerät geöffnet worden wäre.

 **Hintergrund**

Vorhergehende Versionen des FHS stellten strengere Anforderungen an `/dev`. Es dürfen jetzt weitere Geräte in `/dev` vorhanden sein. Gerätedateien dürfen auch symbolische Links auf andere Gerätedateien sein, die sich in `/dev` oder Unterverzeichnissen von `/dev` befinden. Es gibt keine Einschränkungen die Haupt- oder Nebengerätenummern betreffend.

6.1.4. /etc : Rechnerspezifische Systemeinstellungen

Wenn Linux-Systeme sie benötigen, plazieren sie folgende zusätzliche Dateien in `/etc`:

- `lilo.conf`

6.1.5. /lib64 und /lib32 : 64/32-Bit-Bibliotheken (architekturspezifisch)

Für die 64-Bit-Architekturene PPC64, s390x, sparc64 und AMD64 müssen die 64-Bit-Bibliotheken in `/lib64` und die 32-Bit-Bibliotheken (bzw. die 31-Bit_bibliothken für s390) in `/lib` plaziert werden.

Für die 64-Bit-Architekture IA64 gehören die 64-Bit-Bibliotheken nach `/lib`.

Hintergrund

Dies ist eine Verfeinerung der generellen Regel für `/lib<Suffix>` und `/usr/lib<Suffix>`. Die Architekturen von PPC64, s390x, sparc64 und AMD64 unterstützen sowohl 32-Bit- (für s390 richtiger: 31-Bit-) als auch 64-Bit-Programme. Dadurch, dass `/lib` für 32-Bit-Programme benutzt wird, funktionieren auch die bereits vorhandenen 32-Bit-Programme - und das sind sicher viele - ohne Änderungen weiter. Beim IA64 wird das anders gehandhabt, weil dort 32-Bit-Programme (und -Bibliotheken) als veraltet gelten.

6.1.6. /proc : Virtuelles Dateisystem für Kern- und Prozessinformationen

Das `proc`-Dateisystem ist der De-fakto-Standard in Linux für die Handhabung von Prozess- und Systeminformationen, also nicht `/dev/kmem` oder andere ähnliche Methoden. Wir empfehlen wärmstens, diese Methode zum Speichern und für den Abruf von Prozess-, Kern-Hauptspeicherinformationen zu nutzen.

6.1.7. /sbin : Wesentliche Systemkommandos

Linux-Systeme plazieren folgende zusätzlichen Dateien in `/sbin`:

- e2fs-Kommandos ("Second Extended Filesystem"):

  - **badblocks**
  - **dumpe2fs**
  - **e2fsck**
  - **mke2fs**
  - **mklost+found**
  - **tune2fs**

- Einrichter ("map installer") des Systemladers (optional):
  - **lilo**

Optionale Dateien für /sbin:

- Statisch lauffähige Programme:
  - **ldconfig**
  - **sln**
  - **ssync**

  Das statische **ln** (**sln**) und das statische **sync** (**ssync**) sind nützlich, wenn mal etwas schiefgeht. Die zunächst wichtigeste Funktion von **sln**, nämlich fehlerhafte symbolische Links unterhalb von `/lib` zu reparieren (etwa nach einem schlecht organisierten Systemupdate), hat nicht mehr diese Bedeutung seitdem es das Programm **ldconfig** (üblicherweise in `/usr/sbin`) gibt, das uns beim Aktualisieren der dynamischen Bibliotheken bei der Hand nimmt. Das statische **sync** ist in mancher Notsituation nützlich. Man beachte, dass die genannten Kommandos statisch gebundene ("linked") Versionen der Standardkommandos **ln** und **sync** sein können, aber nicht müssen.

  ldconfig ist optionaler Bestandteil von `/sbin`, weil ein Betreiber sich dafür entscheiden kann, **ldconfig** beim Systemstart, anstatt nur beim Aktualisieren der Bibliotheken, einzusetzen. (Es ist noch nicht klar, ob es von Vorteil ist, **ldconfig** bei jedem Systemstart einzusetzen, oder nicht.) Jedenfalls möchten es viele für die folgenden (gar so üblichen) Situationen zur Hand haben:

  1. Gerade habe ich `/lib/<Datei>` gelöscht.
  2. Ich kann den Namen der Bibliothek nicht finden, weil **ls** dynamisch gebunden wurde, weil ich eine Shell benutze, die kein **ls** eingebaut hat und weil ich die Alternative "**echo ***" nicht kenne.
  3. Ich habe das statische **sln**-Kommando, aber ich weiß nicht, wie ich den Link nennen soll.

- Verschiedenes:
  - **ctrlaltdel**
  - **kbdrate**

  Um mit einigen Tastaturen zurechtzukommen, die wegen einer zu hohen Wiederholrate unbrauchbar sind, kann auf manchen Systemen **kbdrate** in `/sbin` zur Verfügung gestellt werden.

Weil die Reaktion des Kerns auf die Tastenkombination Strg-Alt-Entf üblicherweise ein sofortiger Systemneustart ist, ist es ratsam, dieses Verhalten zu unterbinden, bevor das Wurzeldateisystem im Schreib-Lese-Modus eingebunden wird. Einige **init**-Verfahren sind dazu in der Lage, andere benötigen das Programm **ctrlaltdel**, das deshalb auf manchen Systemen unter `/sbin` zur Verfügung gestellt wird.

6.1.8. /usr/include : Header-Dateien für C-Programme

Folgende symbolischen Links werden benötigt, wenn ein C- oder ein C++-Compiler installiert ist - aber nur für Systeme, die nicht auf glibc basieren.

```
/usr/include/asm -> /usr/src/linux/include/asm-<Architektur>
/usr/include/linux -> /usr/src/linux/include/linux
```

6.1.9. /usr/src : Quellen

Für Systeme, die auf glibc basieren, gibt es keine weitergehenden Richtlinien zu diesem Verzeichnis. Für Systeme, die auf dem Vorläufer von glibc, der Linux-libc basieren, gelten folgende Regeln und folgender Hintergrund:

Der einzige Quellcode, der an einer bestimmten Stelle gespeichert werden soll, ist der des Linux-Systemkerns. Der wird abgelegt unter `/usr/src/linux`.

Wenn ein C- oder C++-Compiler installiert ist, aber nicht der komplette Quellcode des Linuxkerns, dann müssen die Include-Dateien des Kern-Quellcodes in folgenden Verzeichnissen sein:

```
/usr/src/linux/include/asm-<Architektur>
/usr/src/linux/include/linux
```

`<Architektur>` benennt dabei die Systemarchitektur.

`/usr/src/linux` darf ein symbolischer Link zum Verzeichnis der Kernquellen sein.

Hintergrund

Wichtig ist, dass die Include-Dateien für den Kern sich in `/usr/src/linux` und nicht in `/usr/include` befinden, sodass es keine Probleme gibt, wenn der Systemadministrator zum ersten Mal den Kern aktualisiert ("upgrade").

6.1.10. /var/spool/cron : cron- und at-Jobs

Dieses Verzeichnis enthält die variablen Daten für die Programme **cron** und **at**.

Kapitel 7. Anhang

7.1. Die FHS-Mailingliste

Die FHS-Mailingliste findet man bei <freestandards-fhs-discuss@lists.sourceforge.net>. Sie können die Mailingliste abonnieren über http://sourceforge.net/projects/freestandards/.

Vielen Dank an die Gruppe "Network Operations" der University of California in San Diego. Wir durften dort den exzellenten Mailinglisten-Server nutzen.

Wie auch in der Einleitung erwähnt wird: Bitte senden Sie keine E-Mail an die Mailingliste, ohne zuerst zum FHS-Herausgeber oder einem der aufgeführten Mitarbeiter Kontakt aufgenommen zu haben.

7.2. Was steckt hinter dem FHS ?

Der Entwicklungsprozess für einen Dateisystem-Hierarchiestandard begann im August 1993 mit dem Bemühen, die Datei- und Verzeichnisstruktur von Linux neu zu strukturieren. FSSTND, der Linux-bezogene Dateisystem-Hierarchiestandard, wurde am 14. Februar 1994 veröffentlicht. Folgeversionen kamen am 9. Oktober 1994 und am 28. März 1995 heraus.

Zu Beginn des Jahres 1995 wurde unter Mithilfe von Mitgliedern der BSD-Entwicklergemeinde ein neues Ziel gesteckt: Eine umfassendere Version der FSSTND nicht nur für Linux, sondern auch für andere UNIX-ähnliche Betriebssysteme zu erstellen. Das Ergebnis war eine gemeinsame Anstrengung, die sich auf Aspekte konzentrierte, die allen UNIX-ähnlichen Systemen gemein sind. In Anerkennung des erweiterten Gültigkeitsbereichs wurde der Name dieses Standards abgeändert in "Filesystem Hierarchy Standard" oder kurz FHS.

Am Ende dieses Dokuments sind die Namen der freiwillig Mitwirkenden aufgeführt, die besonders viel zum Standard beigetragen haben. Der Standard beschreibt den Konsens dieser und anderer Mitwirkender.

7.3. Allgemeine Richtlinien

Hier ein paar der Richtlinien, die die Arbeit an diesem Standard bestimmt haben:

- Löse technische Probleme und halte dabei den Änderungsaufwand gering.
- Mache die Spezifikation - in vernünftigem Rahmen - stabil.
- Gewinne die Zustimmung von Distributoren, Entwicklern und anderen Entscheidungsträgern in wichtigen Entwicklungsgruppen und motiviere sie zur Mitarbeit.
- Stelle einen Standard zur Verfügung, der für Entwickler ganz verschiedener UNIX-ähnlicher Systeme attraktiv ist.

7.4. Gültigkeitsbereich

Dieses Dokument legt eine standardisierte Dateisystemhierarchie für FHS-konforme Dateisysteme fest, indem es die Plazierung von Dateien und Verzeichnissen sowie der Inhalt einiger Systemdateien festlegt.

Dieser Standard wurde so gestaltet, dass er genutzt werden kann von Systemintegratoren, Paketentwicklern oder Systemverwaltern sowohl für die Konstruktion als auch für die Wartung von FHS-konformen Dateisystemen. Es ist in erster Linie ein Referenzhandbuch und keine Anleitung für die Handhabung eines solches Dateisystems.

Der FHS erwuchs aus den älteren Arbeiten an dem FSSTND, welches ein Organistionsstandard für das Dateisystem des Betriebssystems Linux war. Aufbauend auf FSSTND behandelt er Zusammenarbeitsaspekte nicht nur innerhalb der Linux-Gemeinde sondern umfasst darüberhinausgehend unter anderem die 4.4BSD-basierten Systeme. Es enthält Erfahrungen aus der BSD-Welt und von anderer Seite, die die Multi-Architektur-Unterstützung und die Erfordernisse beim Betrieb heterogener Netze betreffen.

Auch wenn dieser Standard umfassender ist als die vorhergehenden Versuche einer Standardisierung, so werden doch periodische Updates als Anpassung an die sich weiter entwickende Technik nötig sein. Möglicherweise werden also bessere Lösungen zu den angesprochenen Problemen als die hier vorgeschlagenen entwickelt werden - unsere Lösungen werden dann nicht mehr die besten sein. Zusätzlich zu den periodischen Updates werden eventuell ergänzende Skizzen("supplementary drafts") zu diesem Dokument veröffentlicht. Trotzdem ist Rückwärtskompatibilität von einer Veröffentlichung zur nächsten ein besonders wichtiges Ziel.

Anregungen oder Kommentare zu diesem Standard sind willkommen. Kommentare oder Änderungsvorschläge richten Sie bitte an den FHS-Herausgeber (Daniel Quinlan <quinlan@pathname.com>) oder die FHS-Mailingliste. Kommentare zur Rechtschreibung oder Grammatik richten Sie bitte an den FHS-Herausgeber.

Um ausufernde sich wiederholende Diskussionen zu alten Themen zu vermeiden, nehmen Sie bitte zuerst zum FHS-Herausgeber Kontakt auf, bevor Sie eine E-Mail an die Mailingliste schicken.

Hin und wieder mögen Fragen auftauchen, wie bestimmte Passagen dieses Dokuments zu intepretieren sind. Wenn Sie auf eine Klarstellung angewiesen sind, nehmen Sie bitte zum FHS-Herausgeber Kontakt auf. Da dieser Standard einen Konsens vieler Beteiligter darstellt, ist es auch wichtig sicherzustellen, dass jede Interpretation sich mit diesem Konsens verträgt. Aus diesem Grund kann auch nicht für eine sofortige Antwort garantiert werden, da zu einer Frage zunächst eine Diskussion nötig sein könnte.

7.5. Danksagung

Die Entwickler des FHS möchten sich bedanken bei allen Entwicklern, Systemverwaltern und Benutzern. Deren Beiträge zu diesem Standard waren unverzichtbar. Wir wollen auch allen danken, die dabei halfen, den Standard zu schreiben und zusammenzustellen.

Die "FHS Group" möchte außerdem jenen Linux-Entwicklern danken, die den FSSTND, den Vorläufer dieses Standards, unterstützt haben. Wenn sie nicht gezeigt hätten, dass der FSSTND nützlich war, hätte sich der FHS nicht entwickeln können.

7.6. Mitwirkende

Brandon S. Allbery	<bsa@kf8nh.wariat.org>
Keith Bostic	<bostic@cs.berkeley.edu>
Drew Eckhardt	<drew@colorado.edu>
Rik Faith	<faith@cs.unc.edu>
Stephen Harris	<sweh@spuddy.mew.co.uk>
Ian Jackson	<ijackson@cus.cam.ac.uk>
Andreas Jaeger	<aj@suse.de>
John A. Martin	<jmartin@acm.org>
Ian McCloghrie	<ian@ucsd.edu>
Chris Metcalf	<metcalf@lcs.mit.edu>
Ian Murdock	<imurdock@debian.org>
David C. Niemi	<niemidc@clark.net>
Daniel Quinlan	<quinlan@pathname.com>
Eric S. Raymond	<esr@thyrsus.com>
Rusty Russell	<rusty@rustcorp.com.au>
Mike Sangrey	<mike@sojurn.lns.pa.us>
David H. Silber	<dhs@glowworm.firefly.com>
Thomas Sippel-Dau	<t.sippel-dau@ic.ac.uk>
Theodore Ts’o	<tytso@athena.mit.edu>
Stephen Tweedie	<sct@dcs.ed.ac.uk>
Fred N. van Kempen	<waltje@infomagic.com>
Bernd Warken	<bwarken@mayn.de>
Christopher Yeoh	<cyeoh@samba.org>