Übersicht: Wie funktioniert UNIX / Linux?

15. Juni 2008

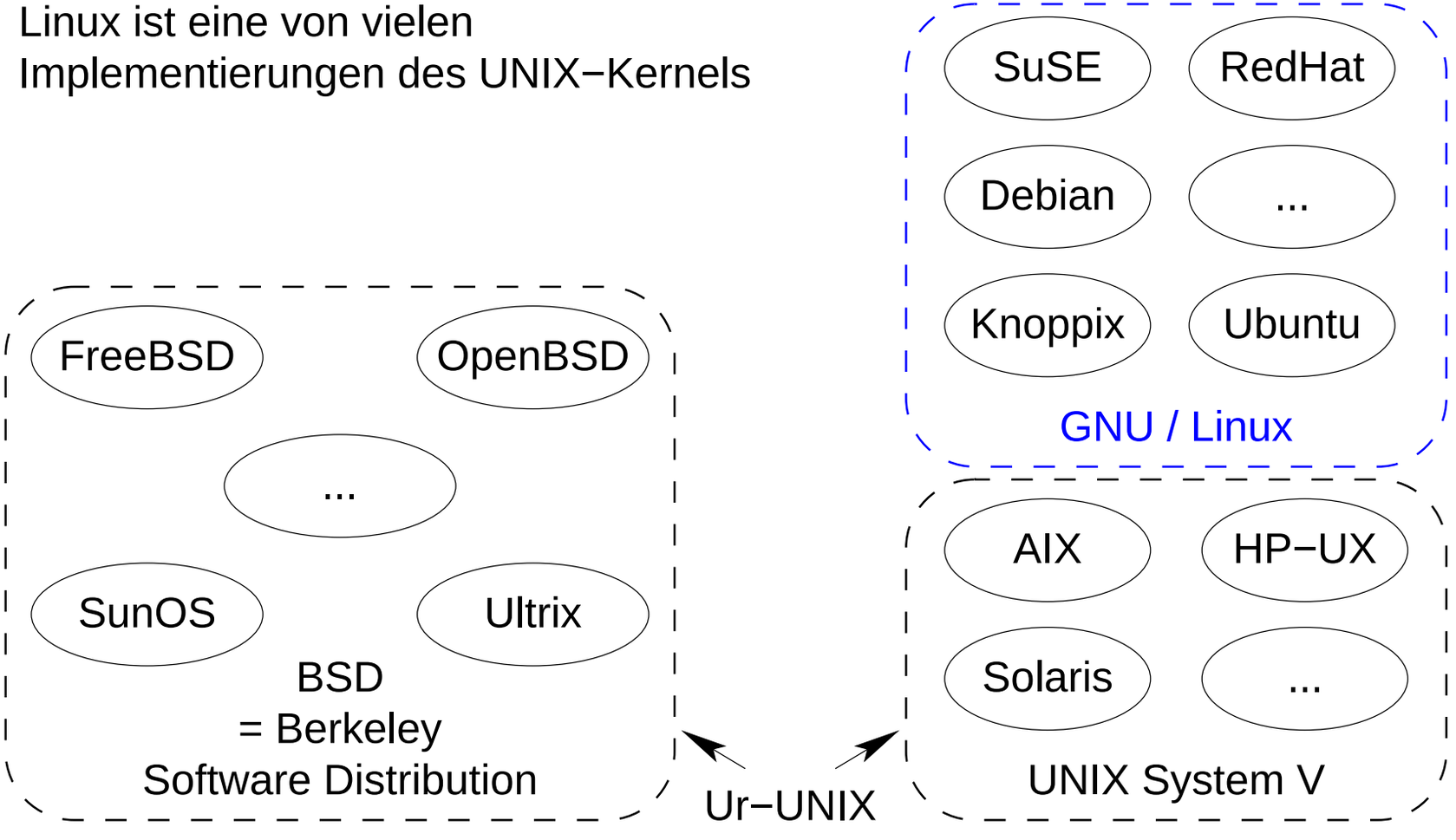
http://home.ntz.de/phleps/unix/linux_folien.pdf

Inhalt

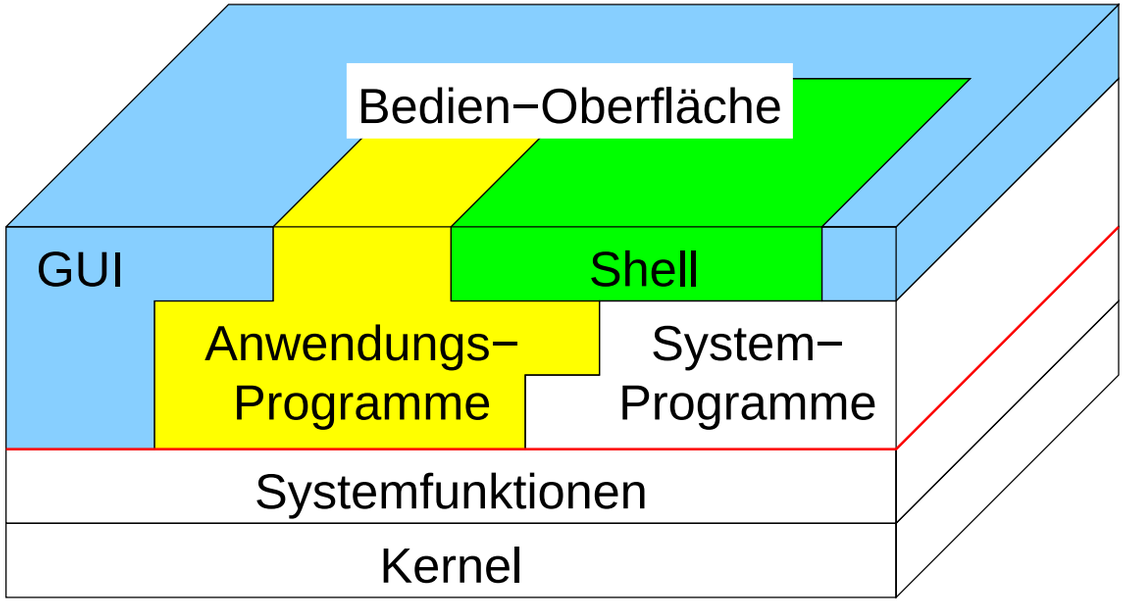
- Unterschied UNIX / Linux
- UNIX Systemarchitektur
- Wesentliche UNIX-Eigenschaften
- Arbeiten mit der UNIX Befehlszeile
- Was ist eine Shell und was tut sie?
- Baukastenprinzip von UNIX-Anwendungen
- Prinzip der UNIX Systemkonfiguration
- Installieren von Programmen und Treibern
- SW-Entwicklung für UNIX
- Literatur

Unterschied UNIX / Linux

Linux ist eine von vielen
Implementierungen des UNIX−Kernels



UNIX Systemarchitektur



Wesentliche UNIX-Eigenschaften (1)

- Multi-User und Multi-Tasking System
- Der Benutzer mit UID = 0 hat Administratorrechte.
  Benutzername: *root*
- Hierarchische Verzeichnisstruktur
- Es gibt keine Laufwerks-Buchstaben.
  Zusätzliche Platten werden als Teil der Verzeichnisstruktur eingehängt (“gemountet”).

Wesentliche UNIX-Eigenschaften (2)

- UNIX unterscheidet zwischen Groß- und Kleinbuchstaben
- Datei-Endungen haben für das System keine Bedeutung.
  Sie sind nur Orientierungshilfe für Anwender, Dateimanager und Anwendungen.
- Trenner in Pfad-Angaben ist / (d. h. nicht \ wie bei MS-DOS)
- Die wichtigsten Zeichen mit besonderer Bedeutung für die Befehlszeile:
  Leerzeichen - * ? [ ] ( ) { } $ ~ " ' ` / \ ; & | < > #

Wichtige Verzeichnisse (1)

/	*root*-Verzeichnis = *“Wurzel”* des Dateibaumes
/bin	wichtige, elementare UNIX-Programme
/dev	Gerätedateien
/etc	Rechner-spezifische Konfigurationsdateien
/home	Basis für benutzerspezifische Verzeichnisse: Jeder Benutzer hat hier ein eigenes *HOME*-Verzeichnis, z. B. */home/dagobert* für Benutzer *dagobert*.
/lib	wichtige Systembibliotheken
/opt	Programme und statische Dateien von Anwendungen. Jede Anwendung hat ein eigenes Unterverzeichnis.
/root	*HOME*-Verzeichnis für Systemverwalter *root*
/sbin	Programme für den Systemverwalter

Wichtige Verzeichnisse (2)

/tmp	temporäre Dateien
/usr/bin	allgemeine UNIX-Programme
/usr/include	Header-Dateien für C-Compiler, z. B. stdio.h
/usr/lib	allgemeine Bibliotheken
/var	Dateien, deren Größe sich im Lauf der Zeit ändert
/var/opt	Daten zu Anwendungen in /opt: /var/opt/*applikation* gehört zu /opt/*applikation*

Weitere Informationen finden Sie in [16]
http://www.freestandards.org/en/Specifications

Arbeiten mit der UNIX Befehlszeile (1)

- Hilfe (Alternativen):
  `man -k` *Schlüsselwort*
  `man` *Befehl*
  `info` *Befehl*
- Abmelden, eigenen Prozess beenden: **`exit`**
- Aktuelles Verzeichnis anzeigen: **`pwd`**
- Liste der Dateien: **`ls`** oder **`ls`** *Verzeichnisname*
- Verzeichnis wechseln: **`cd`** *Zielverzeichnis*
- Verzeichnis anlegen: **`mkdir`** *Verzeichnisname*
- leeres Verzeichnis löschen: **`rmdir`** *Verzeichnisname*
- rekursiv löschen: **`rm -r`** *Verzeichnisname*
  oder **`rm -ri`** *Verzeichnisname*

Arbeiten mit der UNIX Befehlszeile (2)

- Datei löschen: **`rm`** *Dateiname*
  oder **`rm -i`** *Dateiname*
- Datei kopieren: **`cp`** *Quelle Ziel*
- Datei umbenennen: **`mv`** *Quelle Ziel*
- Datei-Inhalt auf Bildschirm ausgeben: **`cat`** *Dateiname*
- Seitenweise Ausgabe auf den Bildschirm: **`more`** *Dateiname*
  oder **`less`** *Dateiname*

Weitere UNIX-Befehle finden Sie in “Hinweise zum Arbeiten mit UNIX”
http://home.ntz.de/phleps/unix/ux_bedien.pdf

Datei-Zugriffsrechte (1)

Ausgabe des Befehls **`ls -l`** in einem Beispielverzeichnis:

```
-rw-r--r--   1 mueller  users          4 Feb  6  1999 datei1.txt
-rwxr-xr-x   1 mueller  users          9 Jul 24 19:47 datei2.sh
-rw-r--r--   1 schulze  guest         35 Jul 24 19:48 datei3.txt
-rw-rw-r--   1 maier    users         22 Jul 24 19:48 datei4.txt
-rw-rw-rw-   1 mueller  users         27 Mar 28  2000 file
lrwxrwxrwx   1 maier    users         10 Dec 29 11:53 symlink -> datei4.txt
drwxr-xr-x   2 schulze  guest       1024 Dec 29 11:50 verzeichnis
```

Bedeutung:

```
|uuugggooo   |   Eigentümer:       Datei-  Zeitstempel   Name
|            | User     Gruppe     größe
|Zugriffs-   |
| rechte     +- Link-Count
|
+------------- Dateityp
```

Datei-Zugriffsrechte (2)

Für jede Berechtigungsklasse

u = *User*, d. h. Eigentümer der Datei

g = *Group*, d. h. Gruppe, der die Datei gehört

o = *Others*, d. h. der “Rest der Welt”

werden folgende Zugriffsrechte unterschieden:

`r--`	*read*	Lesezugriff erlaubt
`-w-`	*write*	Schreibzugriff erlaubt Bei Verzeichnis: Dateien anlegen und löschen erlaubt.
`--x`	*execute*	Ausführen der Datei erlaubt Bei Verzeichnis: Durchsuchen erlaubt

Zugrigffsrechte werden mit **`chmod`** geändert.

Was ist eine Shell und was tut sie?

Eine “*Shell*” ist ein UNIX Kommando-Prozessor. Sie

- liest Befehle für das Betriebssystem von der Tastatur oder aus einer Datei (“Shell-Skript”),
- interpretiert die Eingaben und
- ruft die entsprechenden Systemfunktionen oder Programme auf.

Oft kann zwischen verschiedenen Shells gewählt werden, z. B.

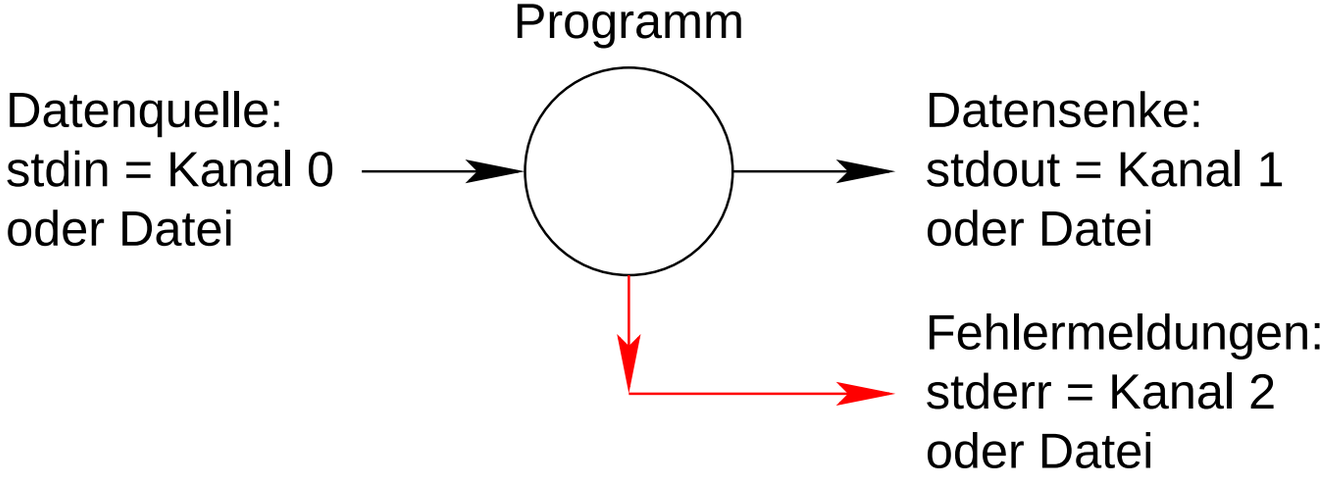
- POSIX-Shell
- bash = Bourne again Shell
- ksh = Korn Shell
- csh = C-Shell

Typische UNIX Programm-Architektur

Philosophie einfacher und kleiner Programme, die ihre Aufgabe optimal erfüllen:

"*KISS*-Prinzip": "Keep it small and simple"

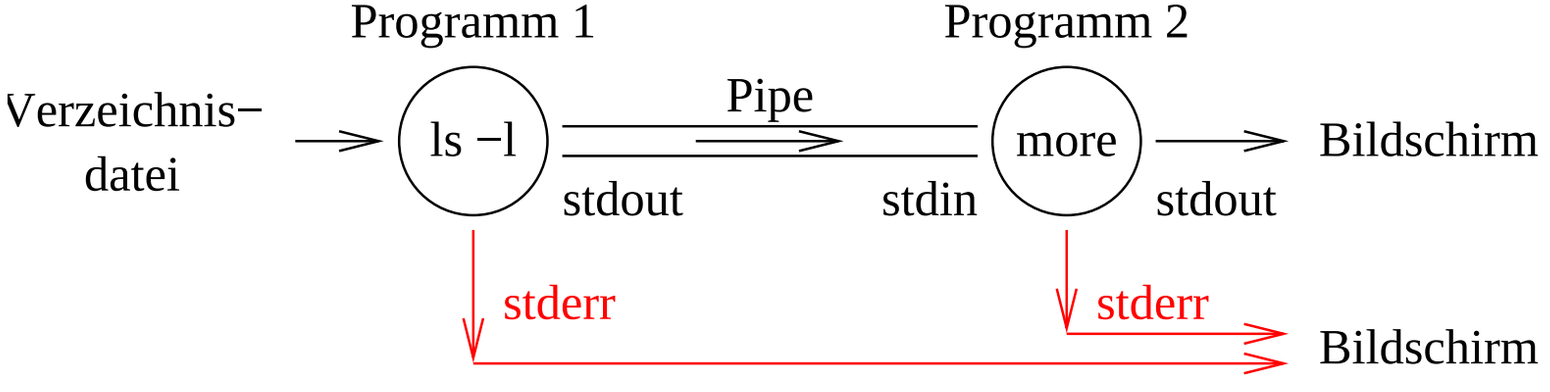
Arbeitsweise als Filter:



Baukasten-Prinzip

Die Arbeitsweise als Filter ermöglicht das Baukasten-Prinzip.

Beispiel: **`ls -l | more`**



Durch Verbinden von Programmen mit Pipes können sehr komplexe Aufgaben gelöst werden.

Prinzip der UNIX Systemkonfiguration

- Die UNIX-Systemkonfiguration ist in normalen Text-Dateien gespeichert.
- Die System-Konfigurationsdateien sind unterhalb des Verzeichnisses **`/etc`**
- Einstellungen, die die Benutzer-Umgebung betreffen, sind auf mehreren Ebenen möglich:
  - Systemweite Voreinstellung erfolgt in Datei(en) unterhalb von **`/etc`**

    Beispiel: Datei **`/etc/bash.bashrc`**
  - Jeder Benutzer kann die Voreinstellung für sein individuelles Arbeitsumfeld ergänzen oder ändern.

    Dauerhafte Änderungen erfolgen über Dateien im HOME-Verzeichnis des betreffenden Benutzers.

    Beispiel: Datei **`$HOME/.bashrc`**

Installieren von Programmen unter Linux (1)

Empfehlung: Wenn möglich, Package-Manager verwenden.

Unterschiedliche Package-Manager je nach Distribution:

rpm: z. B. bei SuSE, RedHat, Fedora, Mandriva
Befehl: **`rpm`**

Anleitung: Siehe `http://linuxwiki.de/RPM`

dpkg und apt: bei Debian und Debian-basierenden Distributionen, z. B. Ubuntu
Befehl: **`dpkg`** bzw. **`apt-get`** / **`apt-cache`**

Anleitung: Siehe `http://linuxwiki.de/apt/MiniHowTo`

Installieren von Programmen (2)

Alternativen:

- Installieren eines tar- oder zip-Pakets mit Binär-Dateien:
  Paket in passendes Verzeichnis entpacken
- Übersetzen und installieren eines Quellcode-Pakets
  Standard-Ablauf:
  - Paket entpacken
  - Dateien README und INSTALL lesen
  - **`./configure`**
  - **`make`**
  - **`make install`**
    Je nach Zugriffsrechten des Zielverzeichnisses der Installation muss **`make install`** als root ausgeführt werden.

Linux: Treiber

Hardware-Treiber werden bevorzugt als ladbare Kernelmodule bereitgestellt.

Treiber, die fest in den Kernel eingebaut sind, sollten auf das notwendige Minimum beschränkt werden.

Treiber sind meistens als Quellcode vorhanden. Zum übersetzen sind die Kernel-Quellen notwendig.

Kernel-Quellen: meistens in **`/usr/src/linux-`***Version*

Symbolischer Link **`/usr/src/linux`** -> **`/usr/src/linux-`***Version* zeigt auf Verzeichnis mit der zu verwendenden Version

Zugehörige Module sind in **`/lib/modules/`***Version*

Linux: Installieren von Treibern (1)

Prinzipieller Ablauf (kann je nach System in Details abweichen):

- Wenn auch der Kernel neu kompiliert werden soll (empfohlen):
  - Wechsel in das Verzeichnis mit den Kernel-Quellen
  - Kernel konfigurieren: **`make menuconfig`**
    oder **`make xconfig`**
    oder **`make config`**
    Die Kernel-Konfiguration ist in der Datei **`.config`** gespeichert.
  - Dependencies-Dateien erstellen: **`make dep`**
  - Neues Kernel-Image bauen: **`make bzimage`**

Linux: Installieren von Treibern (2)

- Erstelltes Kernel-Image **`arch/i386/boot/bzimage`** nach **`/boot/vmlinuz-`***Kernel-Versionsnr* kopieren.

  ACHTUNG: Kernel des laufenden Systems nicht überschreiben!
  Version des laufenden Kernels anzeigen: **`uname -a`**

- Kernel in Konfigurationsdatei des Boot-Managers eintragen.
  Details dazu: siehe Beschreibung des Boot-Managers.

- Module übersetzen: **`make modules`**
- Module installieren: **`make modules_install`**

Linux: Befehle zum Umgang mit Kernel-Modulen

Befehle zum Umgang mit Kernel-Modulen:

- Modul manuell laden: **`modprobe`** *Modulname*
- Anzeige der geladenen Module: **`lsmod`**
- Abhängigkeiten zwischen Modulen ermitteln: **`depmod -a`**

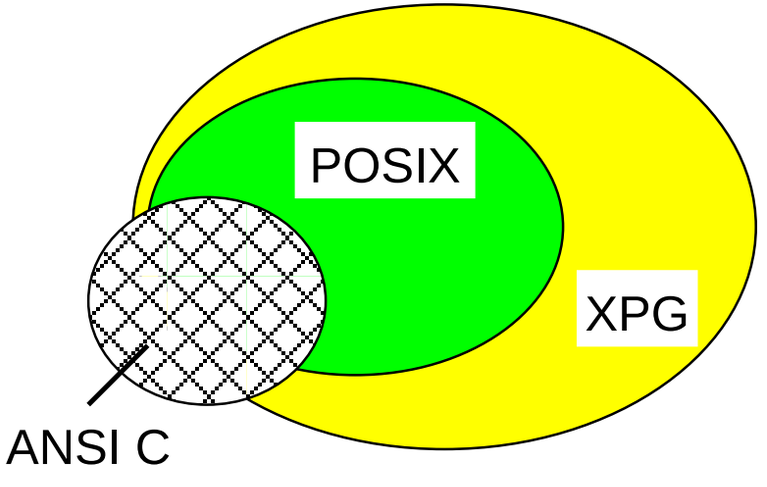
Literaturhinweis:

Susanne Schmidt: Treibereien. Neue Hardware unter Linux einrichten.
c't 13/2003, S. 220 bis 225

Software-Entwicklung für UNIX (1)

APIs für Systemfunktionen:

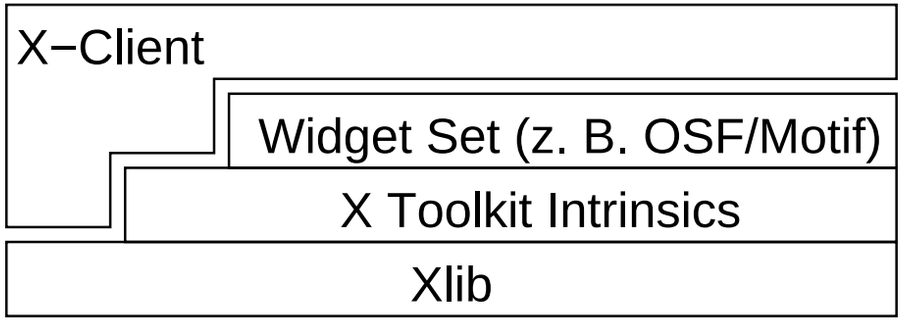
- ANSI-C
- POSIX
- XPG-Standards (X/Open Group)
- Standard-Funktionen von System V bzw. BSD



Software-Entwicklung für UNIX (2)

APIs für Graphische Bedienoberfläche:

- X11: Basisbibliothek *Xlib*
- X Toolkit Intrinsics = Xt Intrinsics: Abstrahierender Aufsatz auf Xlib
- OSF/Motif: Widget-Set (mit Festlegung des “Look and Feel“) basierend auf Xt Intrinsics



Quelle: [30]

Literatur

[1] Michael Kofler: Linux. Installation, Konfiguration, Anwendung
Addison Wesley, 8. Auflage 2007, ISBN 3–8273–2478–5

[2] Oliver Böhm: Spaß mit UNIX / Linux
http://www.aosd.de/Unix/Unterlagen/Unix.pdf

[3] Prof. Jürgen Plate: Betriebssystem UNIX / Linux
http://www.netzmafia.de/skripten/unix/

[4] Dr. Oliver Diedrich, Bernd Butscheidt: Linux. Antworten auf die häufigsten Fragen
c't 22/2005, S. 206
http://www.heise.de/ct/05/22/206/default.shtml

[5] Michael Riepe: Err Tee Eff Emm. Dokumentation unter Unix/Linux
iX 6/2003, S. 138 bis 139

[6] LinuxWiki
http://linuxwiki.de/

[7] Prof. Jürgen Plate: Betriebssystem UNIX / Linux: UNIX-Kurzreferenz
http://www.netzmafia.de/skripten/unix/unix-tabelle.html

[8] Æleen Frisch: Unix Systemadministration kurz & gut
O'Reillys Taschenbibliothek 2003, ISBN 3–89721–250–1

[9] Ingo Phleps: Hinweise zum Arbeiten mit UNIX
http://home.ntz.de/phleps/unix/ux_bedien.pdf

[10] Helmut Herold: UNIX und seine Werkzeuge. UNIX-Grundlagen, Kommandos und Konzepte
Addison Wesley, 3. Auflage 1994, ISBN 3–89319–734–6

[11] Maurice J. Bach: UNIX – Wie funktioniert das Betriebssystem?
Deutsche Übersetzung von “The Design of the UNIX Operating System”
Hanser Verlag 1991, ISBN 3–446–15693–3
Prentice Hall 1991, ISBN 0–13–201740–7

[12] Erik Heim: A long way $HOME. Wie Init-Skripte das System konfigurieren
c’t 12/1999, S. 174 bis 176

[13] Konrad Heuer: Reichliche Auswahl. Shells im Überblick
iX 2/1996, S. 170 bis 174

[14] Dr. Oliver Diedrich: Mit flinken Fingern. Die Power der Kommandozeile
c’t 12/1999, S. 170 bis 172

[15] Susanne Schmidt: On a highway to shell . . . Die Linux-Shell richtig ausnutzen
c’t 23/1999, S. 278 bis 283

[16] Free Standards Group: Linux Standard Base Core Specification 3.1
http://www.freestandards.org/en/Specifications

[17] Susanne Schmidt: Die X-Files. X Window konfigurieren
c't 12/1999, S. 178 bis 180

[18] Susanne Schmidt: Treibereien. Neue Hardware unter Linux einrichten.
c't 13/2003, S. 220 bis 225

[19] Susanne Schmidt: Des Pudels Kern. Umgang mit Kernelmodulen
c't 5/2000, S. 230 bis 234

[20] Oliver Diedrich: Die freie Wahl. Software installieren unter Linux
c't 14/2003, S. 208 bis 213

[21] Oliver Diedrich: Hotline: Software unter Linux installieren
c't 22/2001, S. 212
http://www.heise.de/ct/01/22/212/default.shtml

[22] Hajo Schulz: Marke Eigenbau. Einstieg ins Programmieren unter Linux
c't 22/2003, S. 238 bis 241

[23] Unix Programming Frequently Asked Questions
http://www.erlenstar.demon.co.uk/unix/faq_toc.html

[24] W. Richard Stevens: Advanced Programming in the UNIX Environment
Addison-Wesley 1993, ISBN 0–201–56317–7

[25] Donald Lewine: POSIX Programmer's Guide: Writing Portable UNIX Programs
O'Reilly & Associates 1994, ISBN 0–937175–73–0

[26] Oliver Lau: Tauziehen. Programmieren mit POSIX-Threads
c't 16/2006, S. 212 bis 215

[27] Blaise M. Barney: Introduction to POSIX Threads Programming
http://www.llnl.gov/computing/tutorials/pthreads/html/

[28] Peter Wächtler: Fadenscheinig. Next Generation POSIX Threads für Linux.
iX 12/2002, S. 110 bis 114

[29] Martin Weitzel: Nur für Gurus? Treiber unter PC-Unix programmieren
iX 4/1996, S. 178 bis 183

[30] Hans-Joachim Brede, Nicolai Josuttis, Sabine Lemberg, Achim Lörke: Programmieren mit OSF/Motif Version 2.
2. Auflage Verlag Addison Wesley 1995, ISBN 3-89319-727-3