Einführung in Unix und GNU / Linux

Michael Opdenacker
Free Electrons
http://free-electrons.com
Übersetzung: Hermann J. Beckers

Dank an

- das OpenOffice.org -Projekt, für die Präsentations- und Textprogramme, die alle meine Anforderungen erfüllten.
- die Handhelds.org -Gemeinschaft, die mir soviel Hilfe und auch viele Gelegenheiten gab, selbst zu helfen.
- die Mitglieder der ganzen Gemeinschaft Freier und quelloffener Software, weil sie ihr Bestes teilen: ihre Arbeit, ihr Wissen und ihre Freundschaft.
- Menschen, die Kommentare und Korrekturen schickten: Jeff Ghislain, Leif Thande, Frédéric Desmoulins, Przemysław Ciesielski

Rights to copy

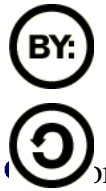
creative commons
COMMONS DEED

Attribution – ShareAlike 2.0

You are free

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions



Attribution. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

License text: http://creativecommons.org/licenses/by-sa/2.0/legalcode

© Copyright 2006-2004
Michael Opdenacker
michael@free-electrons.com

Document sources, updates and translations:
http://free-electrons.com/training/intro_unix_linux

Corrections, suggestions, contributions and translations are welcome!

Dokument-Historie

Soweit nicht anders angegeben, sind dies Beiträge von Michael Opdenacker

- 15. Sep 2009. Letzte Aktualisierung. Korrekturen und kleinere Verbesserungen
- Dec 6, 2004. Neuer Abschnitt Grundlegende Systemverwaltung und verschiedene neue Details
- Sep 28, 2004. Erste Veröffentlichung

Über dieses Dokument

- Dieses Dokument ist in erster Linie als visuelle Hilfe für einen Sprecher oder Dozent gedacht. Daher ist dies nur eine Zusammenfassung oder Ergänzung zum gesprochenen Wort. Daher sind die Erläuterungen nicht erschöpfend.
- Dieses Dokument soll jedoch auch eine Referenz für das Publikum werden. Es zielt auch auf Leser, die an einer Selbstschulung interessiert sind. Daher werden etwas mehr Details geboten, was das Dokument visuell weniger attraktiv macht.

Trainingsinhalte (1)

Einführung

- Unix-Geschichte
- Unix-Philosophie und -Eigenschaften
- Die verschiedenen Schichten in einem Unix-System
- Das GNU-Projekt, die GPL-Lizenz
- Linux-Distributionen
- Andere freie Unix-Systeme

Trainingsinhalte (2)

Shells, Dateisystem und Dateibehandlung

- Befehlszeilen-Interpreter
- Unix Dateisystem-Struktur
- Umgang mit Dateien und Verzeichnissen
- Dateien anzeigen, durchsuchen und sortieren
- Symbolische und Hard-Links
- Dateizugriffsrechte

Trainingsinhalte (3)

Standard-I/O, Redirektion, Pipes

- Standard-Ein- und -Ausgabe
- Standard-Ein- und -Ausgabe zu Dateien umleiten
- Pipes: Standardausgabe an andere Befehle weiterleiten
- Standard-Fehlerausgabe

Trainingsinhalte (4)

Task-Steuerung

- Unix: von Anfang an Multitasking
- Hintergrundausführung, Unterbrechung, Fortsetzung und Abbruch
- Liste aktiver Tasks
- Abbruch 1 oder mehrerer Tasks
- Umgebungsvariablen
- Die Umgebungsvariable `PATH`
- Shell-Aliasnamen, `.bashrc` -Datei

Trainingsinhalte (5)

Verschiedenes

- Texteditoren
- Komprimierung und Archivierung
- Dateien drucken
- Dateien vergleichen
- Dateien suchen
- Anwenderinformationen abrufen

Trainingsinhalte (6)

Grundlagen der Systemverwaltung

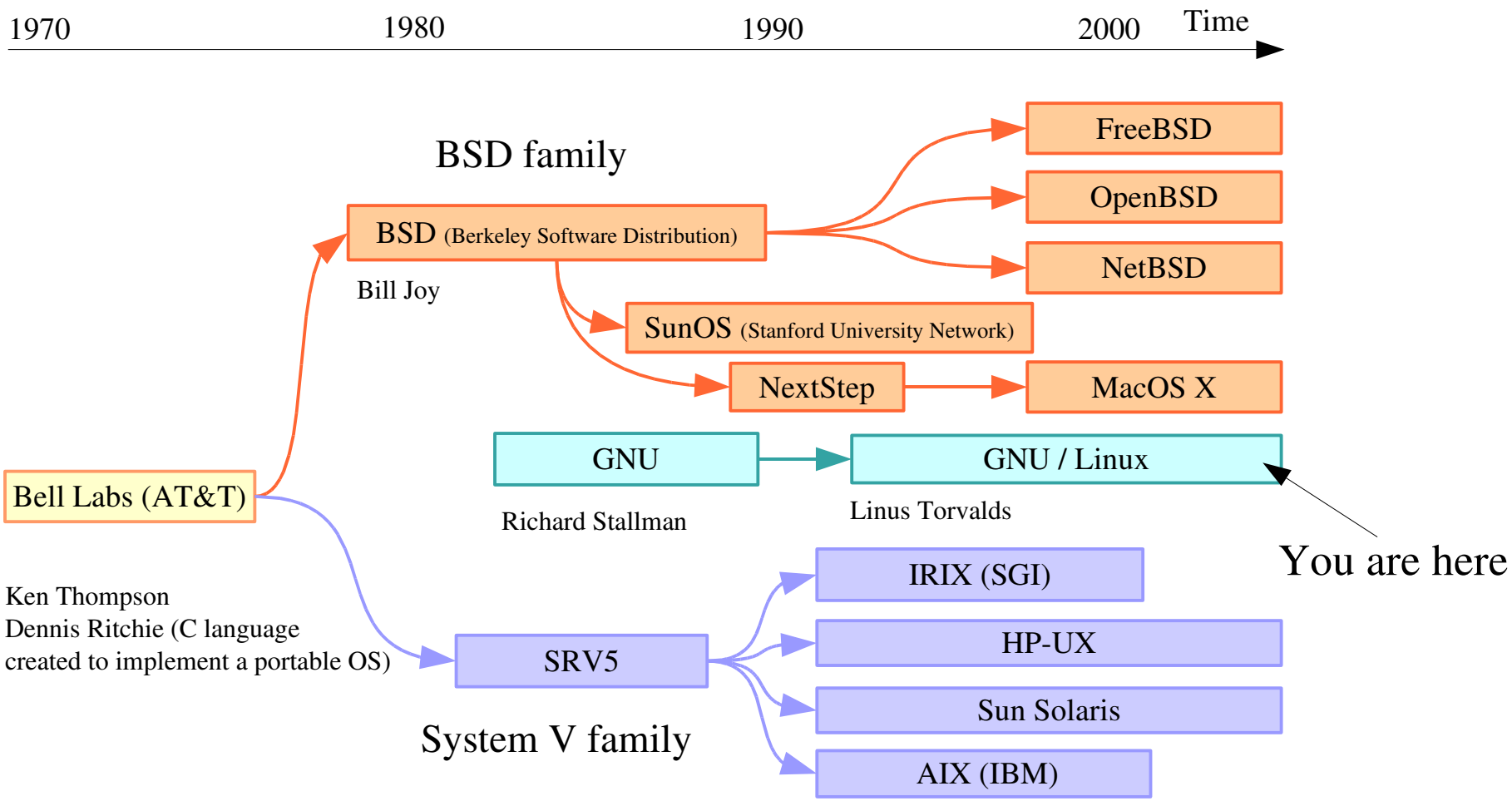
- Verschiedenes: Datei-Eigentümer, Systemstopp ...
- Netzwerk einrichten
- Dateisysteme: erstellen und einhängen

Weitergehendes

- Hilfe erhalten, Zugriff auf Handbuchseiten
- Im Internet nach Ressourcen suchen
- GNU / Linux zuhause benutzen

Einführung

Unix-Stammbaum

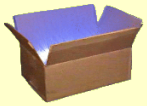


Unix System-Architektur

Grafische Anwendungen


Web-Browser, Büro, Multimedia...

Befehlszeilen-Anwendungen




ls, mkdir, wget, ssh, gcc, busybox...

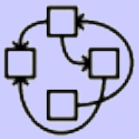
Laufzeit-Bibliotheken

libjpeg, libstdc++, libxml...

C-Bibliothek

GNU C library, uClibc...

Betriebssystem-Kernel

Linux, Hurd...

Hardware und Peripherie



Anwenderbereich

Kernelbereich

Hardware

Die Unix-Philosophie

Die mächtigsten Systeme beruhen auf einem 35 Jahre alten Design!

- Klein ist fein
- Jedes Programm soll eine Sache gut machen
- Bevorzuge Portabilität vor Effizienz
- Vermeide einschränkende Benutzerschnittstellen

System-Abstraktion

- Kernel: Hardware-Schicht
- Shell: Textmodus-Schicht
- X Windows: GUI-Schicht

Unix-Haupteigenschaften

Unix ursprünglich erstellt für große Mehr-Benutzer-Computer

- Mehr-Benutzerfähig und sicher:
  Normale Anwender können nicht mit Dateien anderer Anwender arbeiten (Standard)
  Insbesondere können sie keine Systemeinstellungen ändern, Programme entfernen usw.
- `root`: Verwaltungsanwender mit allen Privilegien
- Preemptives Multi-Tasking
- Mehrprozessor-Unterstützung
- Extrem flexibel
- Netzwerk-Unterstützung
- Portabilität
- Skalierbarkeit

Das GNU-Projekt

GNU = GNU ist Nicht Unix (ein rekursives Acronym!)

- Projekt zur Implementierung eines vollständig freien Un ähnlichen Betriebssystems
- Gestartet 1984 von Richard Stallman, einem MIT-Forscher, zu einer Zeit, als die Unix-Quellen nicht mehr frei waren.
- Anfängliche Komponenten: C compiler (gcc), make (GNU make), Emacs, C library (glibc), coreutils (ls, cp ...)
- In 1991 hatte das GNU-Projekt immer noch keinen Kernel und lief nur auf proprietären Unix-Systemen.

Freie Software

Freie Software garantiert dem Anwender die folgenden 4 Freiheiten:

- Die Freiheit, das Programm für jeden Zweck zu benutzen
- Die Freiheit zu untersuchen, wie das Programm arbeitet, und es an die eigenen Bedürfnisse anzupassen
- Die Freiheit, Kopien zu verteilen, um anderen zu helfen
- Die Freiheit, das Programm zu verbessern, und die eigenen Verbesserungen zu veröffentlichen

Siehe http://www.gnu.org/philosophy/free-sw.html

BSD-ähnliche Freie Software-Lizenzen

- Garantieren dem Anwender natürlich die 4 Freiheiten
- Erlauben es aber, davon proprietäre Software zu erstellen
- Beispiel-Lizenzen: BSD, Apache

GNU General Public License (GPL)

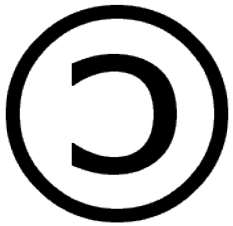
Der Hauptbeitrag des GNU-Projekts!

- *Copyleft* -Lizenzen benutzen das Copyright-Recht, damit der Autor bestimmen kann, das modifizierte Versionen auch wieder freie Software sind.
  http://www.gnu.org/copyleft/copyleft.html
- Die GNU GPL verlangt, das Veränderungen und abgeleitete Arbeiten auch der GPL unterliegen:
  - Gilt nur für veröffentliche Software
  - Jedes Programm, das GePeLlten Code nutzt (durch statisches oder sogar dynamisches Linken) wird als Erweiterung dieses Codes angesehen

GPL FAQ: http://www.gnu.org/licenses/gpl-faq.html

GNU Lesser General Public License

http://www.gnu.org/copyleft/lesser.html



- Copyleft-Lizenz ähnlich der GNU GPL:
  Veränderungen müssen zu den gleichen Bedingungen bereitgestellt werden
- Erlaubt jedoch das Linken mit nicht-freien Modulen
- Wird von verschiedenen freien Softwarebibliotheken genutzt. Beispiele:
  glibc, GTK, Wine, SDL

Freie Software und quelloffene Software

Die Freie Software-Bewegung

- Auf Prinzipien ausgerichteter Ansatz
- Fokussiert auf individueller Freiheit und der sozialen Nützlichkeit von Kooperation. Siehe http://www.gnu.org/philosophy/free-software-for-freedom.html

Die Open Source-Bewegung

- Pragmatischer Ansatz
- Hebt hauptsächlich die Vorteile der gemeinsamen Nutzung der Quellen hervor und trifft Entscheidungen basierend auf technischer Überlegenheit.

Obwohl die grundlegenden Motive verschieden sind, arbeiten beide Bewegungen zusammen und kommen gut miteinander aus!

Linux

- Freier Unix-ähnlicher Kernel, 1991 von Linus Torvalds begründet
- Das ganze System benutzt GNU-Programme:
  C library, gcc, binutils, fileutils, make, emacs...

- Daher wird das gesamte System als "GNU / Linux" bezeichnet
- Schon sehr früh als Freie Software (GPL Lizenz) bereitgestellt, wodurch mehr und mehr Mitarbeiter und Benutzer angezogen wurden.
- Wächst seit 1991 schneller als jedes andere Betriebssystem (nicht nur Unix).

GNU / Linux-Distributionen

- Kümmern sich um eine kompatible Zusammenstellung von Kernel, C-Bibliothek, Compilern und Hilfsprogrammen ... Tatsächlich eine Menge Arbeit!
- Programme sind in *Paketen* verfügbar, die leicht installiert, entfernt oder aktualisiert werden können. Versionsabhängigkeiten werden automatisch verwaltet.
- Kommerzielle Distributionen: enthalten Unterstützung. Quellen sind frei, aber normalerweise nicht die Binärprogramme.
- Gemeinschafts-Distributionen: Quellen und Binärprogrammen sind frei. Standardmäßig keine Unterstützung.
- Verwechseln Sie nicht die Kernel- mit der Distributions-Version!

Kommerzielle Distributionen

- Red Hat: http://www.redhat.com/
  Die populärste. Verläßlich, sicher, leicht zu installieren, benutzerfreundlich, unterstützt von allen Hardware- und Softwareherstellern.

- Suse (Novell): http://www.suse.com/
  Die Haupt-Alternative. Leicht zu installieren, benutzerfreundlich, stabil. Wird von mehr und mehr Hardware- und Software-Herstellern unterstützt.

- Mandriva (ehemals Mandrake): http://mandrivalinux.com/
  Anwenderfreundlich, leicht zu installieren, innovativer, aber weniger stabil. Mehr auf individuelle Anwender ausgerichtet. Wenig Hersteller-Unterstützung.

Gemeinschafts-Distributionen

- Fedora Core: http://fedora.redhat.com/
  Stabil, sicher, benutzerfreundlich, leicht zu installieren. Häufig vollständige Veröffentlichungen.

- Ubuntu Linux: http://ubuntu-linux.org/
  Die wachsende Gemeinschafts-Distribution. Debian-basierend, aber alle 6 Monate stabile Veröffentlichungen. Anwenderfreundlich. Gut für Anfänger.

- Debian: http://debian.org/
  Sehr stabil und sicher, aber schwieriger zu konfigurieren und zu installieren. Entwickler- aber noch nicht benutzerfreundlich. Stabile Veröffentlichungen zu selten (alle 2 - 3 Jahre). Gut für Server, aber nicht für Anfänger!

- Mandriva Community: http://mandrivalinux.com/
  Leicht zu installieren, sicher, benutzerfreundlich, häufig vollständige Veröffentlichungen, aber weniger stabil (nicht ausreichende Tests und Berücksichtigung von Anwendermeldungen).

Andere freie Unix-Systeme (1)

GNU / Hurd: http://www.gnu.org/software/hurd/hurd.html

- GNU-Programme mit Hurd, dem GNU-Kernel (microkernel)
- Reift heran, aber noch nicht für allgemeine Verwendung geeignet.
  Bis jetzt vor allem von Hurd-Entwicklern genutzt (2005).

BSD Familie

- FreeBSD: http://www.freebsd.org/
  Mächtiges, Mehr-Plattformfähiges, sicheres und populäres BSD-System.
- OpenBSD: http://openbsd.org/
  Auf extreme Sicherheit und Verlässlichkeit ausgelegt. Für Internet-Server populär.

- NetBSD: http://netbsd.org/
  BSD-Distribution, die auf Portabilität hin entwickelt wurde
  (verfügbar auf ARM und anderen)

Andere freie Unix-Systeme (2)

- ECOS: http://ecos.sourceware.org/
  Sehr leichtgewichtiges eingebettetes Echtzeit-System von Red Hat / Cygnus solutions.
  POSIX-konformes API.

Unix-Dateisystem

Alles ist eine Datei

Fast alles in Unix ist eine Datei!

- Reguläre Dateien
- Verzeichnisse
  Verzeichnisse sind nur Dateien, die eine Sammlung von Dateien enthalten
- Symbolische Links
  Dateien, die sich auf den Namen einer anderen Datei beziehen
- Geräte-/Peripheriedateien
  Lesen/Schreiben von/zu Geräten wie mit regulären Dateien
- Pipes
  Zur Programmkaskadierung genutzt
  ```
  cat *.log | grep error
  ```
- Sockets
  Inter-Prozess-Kommunikation

Dateinamen

Dateinamen-Eigenschaften seit den Unix-Anfängen

- Unterscheidung Groß-/Kleinschreibung
- Keine offensichtliche Längenbegrenzung
- Kann jedes Zeichen enthalten (einschließlich nicht druckbaren, Ausnahme `/`).
  Dateityp in der Datei gespeichert (“magic numbers”).
  Dateinamen-Erweiterungen sind nicht erforderlich und werden nicht interpretiert. Sie dienen nur der Benutzerbequemlichkeit.
- Beispiele für Dateinamen:

```
README          .bashrc          Windows Buglist
index.htm       index.html       index.html.old
```

Dateipfade

Ein *Pfad* ist eine Sequenz von verschachtelten Verzeichnissen, mit einer Datei oder einem Verzeichnis am Ende, getrennt durch das `/`-Zeichen

- Relativer Pfad: `documents/fun/microsoft_jokes.html`
  Relativ zum aktuellen Verzeichnis
- Absoluter Pfad:
  `/home/bill/bugs/crash9402031614568`
- `/` : *root directory*.
  Start des absoluten Pfads für alle Dateien auf dem System (selbst für Dateien auf entfernbaren Geräten oder Netzlaufwerken).

GNU / Linux Dateisystem-Struktur (1)

Nicht vom System erzwungen. Kann von einem zum anderen System variieren, selbst zwischen zwei GNU/Linux-Installationen!

`/`	Root-Verzeichnis
`/bin/`	Grundlegende, essentielle Systembefehle
`/boot/`	Kernel-Abbilder, initrd und Konfigurationsdateien
`/dev/`	Dateien, die Geräte repräsentieren
	`/dev/hda`: erste IDE-Festplatte
`/etc/`	System-Konfigurationsdateien
`/home/`	Benutzerverzeichnisse
`/lib/`	wesentliche Laufzeit-Bibliotheken

GNU / Linux Dateisystem- Struktur (2)

`/lost+found`	Defekte Dateien die das System retten wollte
`/mnt/`	eingehängte Dateisysteme `/mnt/usbdisk/`, `/mnt/windows/` ...
`/opt/`	Spezielle, vom Administrator installierte Programme
	`/usr/local/` oft stattdessen benutzt
`/proc/`	Zugriff auf System- Information `/proc/cpuinfo`, `/proc/version` ...
`/root/`	root-Startverzeichnis
`/sbin/`	Nur Administrator-Befehle
`/sys/`	System- und Geräte-Steuerungen (cpu frequency, device power, etc.)

GNU / Linux Dateisystem- Struktur (3)

`/tmp/`	Temporäre Dateien
`/usr/`	normale Anwenderprogramme (nicht essentiell für das System) `/usr/bin/`, `/usr/lib/`, `/usr/sbin`...
`/usr/local/`	Spezifische, vom Administrator installierte Software (oft `/opt/` vorgezogen)
`/var/`	vom System oder Systemservern benutzte Daten `/var/log/`, `/var/spool/mail` (eingehende Post), `/var/spool/lpd` (Druckaufträge)...

Shells und Dateibehandlung

Befehlszeilen-Interpreter

- Shells: Programme zum Ausführen von Benutzerbefehlen
- “shells” genannt, weil sie die Einzelheiten des unterliegenden Betriebssystems unter der Shell-Oberfläche verstecken.
- Befehle werden in einem Textterminal eingegeben, entweder ein Fenster in einer grafischen Umgebung oder einer Nur-Text-Konsole.
- Ergebnisse werden auch im Terminal dargestellt. Es wird keine Grafik benötigt.
- Shells sind skriptfähig: sie bieten Ressourcen zum Schreiben komplexer Programme (Variablen, Bedingungen, Schleifen ...)

Sehr bekannte Shells

Berühmte und populäre Shells

- sh: Bourne shell (obsolet)
  Traditionelle, einfache Shell auf Unix-Systemen, von Steve Bourne.
- csh: C shell (obsolet)
  Einst populäre Shell mit C-ähnlicher Syntax
- tcsh: TC Shell (immer noch populär)
  Eine C-Shell-kompatible Implementation mit entwickelten Fähigkeiten (Befehlsvervollständigung, History-Editierung und mehr ...)
- bash: Bourne Again shell (populärste)
  Eine verbesserte Implementation der sh mit einer Unmenge hinzugefügter Eigenschaften.

ls-Befehl

Zeigt die Dateien im aktuellen Verzeichnis in alfanumerischer Sortierung ohne Dateien die mit dem Zeichen "`.`" beginnen.

- `ls -a` (all)
  Zeigt alle Dateien
  (einschliesslich `.*` -Dateien)
- `ls -l` (long)
  Lange Anzeige (Typ, Datum,
  Grösse, Eigner, Berechtigungen)
- `ls -t (time)`
  Zeigt die neuesten Dateien zuerst
- `ls -S` (size)
  Zeigt die größten Dateien zuerst
- `ls -r` (reverse)
  Kehrt die Sortierung um
- `ls -ltr` (Optionen können kombiniert werden)
  Lange Anzeige, neue Dateien am Ende

Musterersetzung für Dateinamen

Besser durch Beispiele vorgestellt!

- `ls *txt`
  Die Shell ersetzt zuerst `*txt` durch alle Datei- und Verzeichnisnamen, die auf `txt` enden (einschließlich `.txt`), mit Ausnahme der mit `.` beginnenden, und führt dann den Befehl `ls` aus.

- `ls -d .*`
  Zeigt alle mit `.` beginnenden Dateien und Verzeichnisse an.
  `-d` weist `ls` an, den Inhalt von `.*` Verzeichnissen nicht anzuzeigen

- `cat ?.log`
  Zeigt alle Dateien an, deren Namen mit einem Zeichen beginnt und auf `.log` endet.

Besondere Verzeichnisse (1)

`./`

- Das aktuelle Verzeichnis. Nützlich für Befehle, die ein Verzeichnis als Argument verwenden. Auch nützlich, um Befehle im aktuellen Verzeichnis auszuführen (s. weiter unten).
- `./readme.txt` und `readme.txt` sind gleichbedeutend

`../`

- Das Eltern-(übergeordnete) Verzeichnis. Gehört immer zum `.` -Verzeichnis (siehe `ls -a`). Einzige Referenz zum Eltern-Verzeichnis.
- Typische Verwendung:
  `cd ..`

Besondere Verzeichnisse (2)

`~/`

- Tatsächlich kein besonderes Verzeichnis. Shells ersetzen das nur durch das Startverzeichnis des aktuellen Anwenders
- Kann in den meisten Programmen genutzt werden, da es kein echtes Verzeichnis ist.

`~sydney/`

- Ähnlich, von Shells durch das Startverzeichnis des Anwenders `sydney` ersetzt.

Die Befehle cd und pwd

- `cd <dir>`
  Wechselt das aktuelle Verzeichnis zu `<dir>`
- `pwd`
  Zeigt das aktuelle Verzeichnis ("working directory")

Der Befehl cp

- `cp <source_file> <target_file>`
  Kopiert die Quelldatei zum Ziel
- `cp file1 file2 file3 ... dir`
  Kopiert die Dateien ins Zielverzeichnis (letztes Argument)
- `cp -i` (interactive)
  Fragt nach Bestätigung, wenn die Zieldatei bereits existiert
- `cp -r <source_dir> <target_dir>` (rekursiv)
  Kopiert das gesamte Verzeichnis

Elegante Verzeichnis-Kopien mit rsync

rsync (remote sync) wurde entwickelt, um Verzeichnisse auf 2 Rechnern mit einer langsamen Verbindung zu synchronisieren.

- Kopiert nur veränderte Dateien, Dateien mit gleicher Größe werden mit Prüfsummen verglichen.
- Kopiert nur Blöcke, die in der Datei verändert wurden!
- Kann die übertragenen Blöcke komprimieren
- Erhält symbolische Links und Dateirechte: auch nützlich für Kopien auf der gleichen Maschine.
- Kann über ssh (secure remote shell) arbeiten. Sehr nützlich, um z. B. die Inhalte einer Webpräsenz zu aktualisieren.

rsync-Beispiele (1)

- `rsync -a /home/arvin/sd6_agents/ /home/sydney/misc/`

  `-a`: Archive-Modus. Entspricht `-rlptgoD`... leichter Weg, wenn Sie rekursiv arbeiten und fast alles erhalten wollen.

- `rsync -Pav --delete /home/steve/ideas/ /home/bill/my_ideas/`

  `-P`: `--partial` (behalte teilweise übertragene Dateien) und `--progress` (Fortschrittsanzeige bei der Übertragung)

  `--delete`: lösche Dateien im Ziel, die in der Quelle nicht vorhanden sind.

  Vorsicht: Verzeichnisnamen müssen mit `/` enden. Sonst erhalten Sie im Ziel ein Verzeichnis `my_ideas/ideas/`

rsync-Beispiele (2)

- Zu einer entfernten Maschine kopieren

```
rsync -Pav /home/bill/legal/arguments/ \
bill@www.sco.com:/home/legal/arguments/
```

Benutzer `bill` wird nach dem Passwort gefragt

- Von einer entfernten Maschine per ssh kopieren

```
rsync -Pav -e ssh
homer@tank.duff.com/prod/beer/ \
fridge/homer/beer/
```

Benutzer `homer` wird nach seinem ssh key password gefragt.

Befehle mv und rm

- `mv <old_name> <new_name>` (move)
  Benennt Datei/Verzeichnis um
- `mv -i` (interactive)
  Wenn die neue Datei bereits vorhanden ist, wird der Anwender zur Bestätigung aufgefordert
- `rm file1 file2 file3 ...` (remove)
  Entfernt die angebenen Dateien
- `rm -i` (interactive)
  Fragt den Anwender immer nach Bestätigung
- `rm -r dir1 dir2 dir3` (recursive)
  Entfernt die angegebenen Verzeichnisse mit allen Inhalten

Verzeichnisse erstellen und entfernen

- `mkdir dir1 dir2 dir3 ...` (make dir)
  Erstellt Verzeichnisse mit den angegebenen Namen
- `rmdir dir1 dir2 dir3 ...` (remove dir)
  Entfernt die angegebenen Verzeichnisse
  Sicher: klappt nur, wenn die Verzeichnisse leer sind
  Alternative: `rm -r`

Dateiinhalte anzeigen

Mehrere Möglichkeiten, den Inhalt von Dateien anzuzeigen

- `cat file1 file2 file3 ...` (concatenate)
  Verknüpft und gibt den Inhalt der angegebenen Dateien aus
- `more file1 file2 file3 ...`
  Bittet den Anwender, nach jeder Seite eine Taste zu drücken.
  Kann auch zum ersten Auftreten eines Schlüsselwortes springen (`/` command)
- `less file1 file2 file3 ...`
  Macht mehr als `more` mit less
  Liest beim Start nicht die komplette Datei
  Unterstützt Rückwärtsbewegung in der Datei (`?` command)

Die Befehle head und tail

- `head [-<n>] <file>`
  Zeigt die ersten <n> Zeilen (Fehlwert 10) der angegebenen Datei an.
  Muss dazu nicht die gesamte Datei öffnen!
- `tail [-<n>] <file>`
  Zeigt die letzten <n> Zeilen (Fehlwert 10) der genannten Datei an.
  Muss nicht die gesamte Datei ins RAM laden! Sehr nützlich für große Dateien.
- `tail -f <file>` (follow)
  Zeigt die letzten 10 Zeilen der Datei und zeigt neue Zeilen an, sobald diese an die Datei angefügt werden.
  Sehr nützlich, um z.B. Änderungen in einer Logdatei zu verfolgen.
- Beispiele

```
head windows_bugs.txt
tail -f outlook_vulnerabilities.txt
```

Der Befehl grep

- `grep <pattern> <files>`
  Durchsucht die Dateien und zeigt die Zeilen an, die dem Muster entsprechen.
- `grep error *.log`
  Zeigt alle Zeilen der `*.log` -Dateien, die `error` enthalten.
- `grep -i error *.log`
  Das gleiche, aber keine Groß-/Klein-Unterscheidung
- `grep -ri error .`
  Das gleiche, aber rekursiv in allen Dateien in `.` und seinen Unterverzeichnissen
- `grep -v info *.log`
  Gibt alle Zeilen der Dateien aus, die nicht `info` enthalten.

Der Befehl sort

- `sort <file>`
  Sortiert die Zeilen der Datei in Zeichenfolge und gibt sie aus
- `sort -r <file>`
  Das gleiche, aber in umgekehrter Reihenfolge
- `sort -ru <file>`
  `u`: unique. Das gleiche, gibt identische Zeilen aber nur einmal aus.
- Weitere Möglichkeiten werden später beschrieben!

Symbolische Links

Ein symbolischer Link ist eine Spezialdatei, die nur eine Referenz auf einen anderen Namen (Datei oder Verzeichnis) ist:

- Nützlich, um Plattenplatz und Komplexität zu verringern, wenn 2 Dateien den gleichen Inhalt haben
- Beispiel:
  `anakin_skywalker_biography -> darth_vador_biography`
- Wie man symbolische Links erkennt:
  - `ls -l` zeigt `->` und den Dateinamen der gelinkten Datei
  - GNU `ls` zeigt Links in einer anderen Farbe

Erstellung symbolischer Links

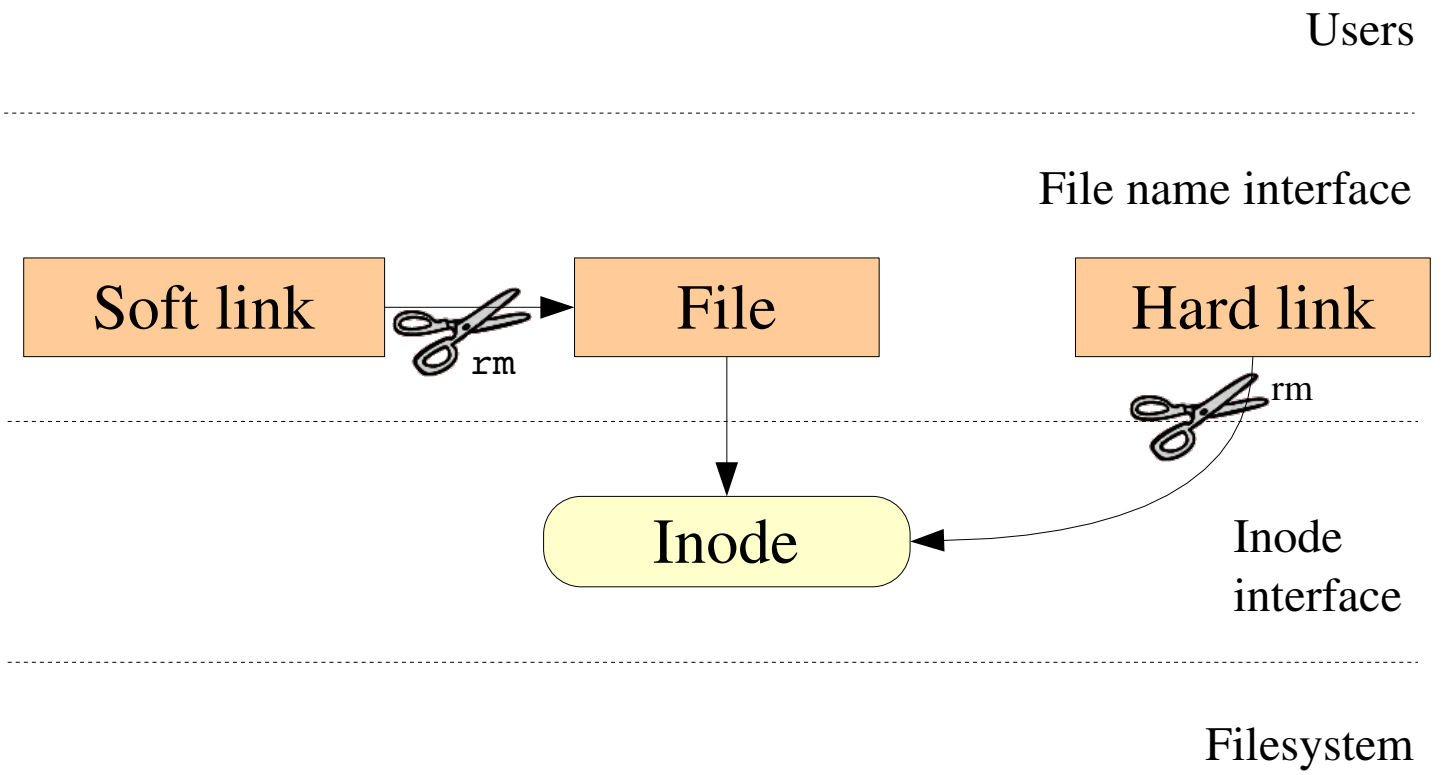
- Erstellen eines symbolischen Links (gleiche Ordnung wie bei `cp`):
  ```
  ln -s file_name link_name
  ```
- Erstellen eines Links zu einer Datei in einem anderen Verzeichnis mit dem gleichen Namen:
  ```
  ln -s ../README.txt
  ```
- Erstellen mehrerer Links in einem angegebenen Verzeichnis:
  ```
  ln -s file1 file2 file3 ... dir
  ```
- Entfernen eines Links:
  ```
  rm link_name
  ```
  Das entfernt natürlich nicht die verlinkte Datei!

Hard links

- Fehlwert für `ln` ist die Erstellung von *hard links*
- Ein *hard link* auf eine Datei ist eine reguläre Datei mit genau den gleichen Inhalten
- Obwohl sie auch Platz sparen, können Hard-Links nicht von der Originaldatei unterschieden werden.
- Wenn Sie die Originaldatei entfernen, hat dies keinen Einfluss auf den Inhalt des Hard-Links.
- Die Inhalte werden entfernt, wenn es keine Dateiverweise (Hard-Links) mehr zu ihnen gibt.

Dateinamen und Inodes

Erleichtert das Verstehen von Hard-Links und symbolischen (Soft-) Links.



Dateizugriffsrechte

Benutzen Sie ls -l zum Prüfen der Zugriffsrechte

3 Arten von Zugriffsrechten

- Lesezugriff (`r`)
- Schreibzugriff (`w`)
- Ausführungsrechte (`x`)

3 Arten von Zugriffsebenen

- Eigner (`u`): für den Dateibesitzer
- Gruppe (`g`): jede Datei hat auch ein “Gruppen”-Atribut entsprechend einer bestimmten Anwenderliste
- SOnstige (`o`): für alle anderen Anwender

Beschränkungen der Zugriffsrechte

- `x` ohne `r` ist zulässig, aber nutzlos
  Zum Ausführen einer Datei muss sie gelesen werden können
- `r` und `x` -Berechtigung sind für Verzeichnisse erforderlich: `x` zum Betreten, `r` zum Anzeigen der Inhalte.
- Sie können Dateien in einem Verzeichnis nicht umbenennen, entfernen oder kopieren, wenn Sie keinen `w` -Zugriff auf das Verzeichnis haben.
- Wenn Sie `w` -Zugriff auf ein Verzeichnis haben, KÖNNEN Sie Dateien entfernen, obwohl Sie keinen Schreibzugriff auf die Datei haben (denken Sie daran, das ein Verzeichnis nur eine Datei ist, die eine Liste von Dateien beschreibt). Damit können Sie sogar ohne w-Zugriff Dateien verändern (löschen und neu erstellen).

Beispiele für Zugriffsrechte

- `-rw-r--r--`
  Les- und schreibbar für den Eigner, nur lesbar für andere
- `-rw-r-----`
  Les- und schreibbar für den Eigner, nur lesbar für Gruppenmitglieder.
- `drwx------`
  Nur für den Eigner zugängliches Verzeichnis
- `-------r-x`
  Datei ausführbar für andere, aber nicht für Sie und Ihre Freunde. Netter Schutz für eine Falle ...

chmod: Rechte ändern

- `chmod <permissions> <files>`
  2 Formate für Rechte:
- Octal-Format (abc):
  `a,b,c = r*4+w*2+x` (`r`, `w`, `x`: boolsche Werte)
  Beispiel: `chmod 644 <file>`
  (`rw` für `u`, `r` für `g` und `o`)
- Oder symbolisches Format. Durch Beispiele leicht zu verstehen:
  `chmod go+r`: fügt Leserecht für Gruppe und andere hinzu
  `chmod u-w`: entfernt Schreibrecht für Eigner
  `chmod a-x`: (a: all) entfernt Ausführungsrecht für alle

Mehr chmod (1)

```
chmod -R a+rX linux/
```

Macht `linux` und alles darin für jedermann verfügbar!

- `R`: führt Änderungen rekursiv durch
- `X`: `x`, aber nur für bereits ausführbare Dateien und Verzeichnisse
  Sehr nützlich für rekursiven Zugriff auf Verzeichnisse, ohne Ausführungsrechte für alle Dateien zu vergeben.

Mehr chmod (2)

```
chmod a+t /tmp
```

- `t`: (sticky). Besonderes Recht für Dateien, ermöglicht nur dem Verzeichnis- und Dateieigner das Löschen einer Datei in einem Verzeichnnis.
- Nützlich für Verzeichnisse mit Schreibrecht für alle wie `/tmp`.
- Durch `ls -l` mit dem `t`-Zeichen angezeigt

Standard-I/O, Redirektion, Pipes

Standard-Ausgabe

Mehr über Befehlsausgaben

- Alle Befehle, die Text auf Ihrem Terminal ausgeben, tun dies durch Schreiben auf *standard output*.
- Standard-Ausgabe kann in eine Datei umgeleitet werden durch das > Symbol
- Standard-Ausgabe kann an eine bestehende Datei angehängt werden durch das >> Symbol

Beispiele für Umleitung der Standard-Ausgabe

- `ls ~saddam/* > ~gwb/weapons_mass_destruction.txt`
- `cat obiwan_kenobi.txt > starwars_biographies.txt`
  `cat han_solo.txt >> starwars_biographies.txt`
- `echo "README: No such file or directory" > README`
  Nützlicher Weg zum Erstellen einer Datei ohne einen Texteditor.
  In diesem Fall auch ein netter Unix-Scherz.

Standard-Eingabe

Mehr über Befehlseingabe

- Eine Menge von Befehlen lesen ihre Eingabe von *standard input*, wenn keine Eingabeargumente angegeben werden
- `sort`
  `windows`
  `linux`
  `[Ctrl][D]`
  `linux`
  `windows`

  sort nimmt die Eingabe über Standard-Eingabe: in diesem Fall Ihre Terminaleingabe (beendet durch `[Ctrl][D]`)
- `sort < participants.txt`
  Die Standard-Eingabe für sort wird aus der angegebenen Datei gelesen.

Pipes

- Unix-Pipes sind sehr nützlich, um die Standard-Ausgabe eines Befehls auf die Standard-Eingabe eines anderen umzuleiten.
- Examples
  - `cat *.log | grep -i error | sort`
  - `grep -ri error . | grep -v “ignored” | sort -u \`
    `> serious_errors.log`
  - `cat /home/*/homework.txt | grep mark | more`
- Dies ist eine der mächtigsten Eigenschaften in Unix-Shells!

Der Befehl tee

```
tee [-a] file
```

- Der `tee` -Befehl kann benutzt werden, um die Standardausgabe gleichzeitg auf den Bildschirm und in eine Datei zu senden
- `make | tee build.log`
  Startet den `make` Befehl und speichert die Ausgabe in `build.log`
- `make install | tee -a build.log`
  Startet `make install` und fügt die Ausgabe an `build.log` an.

Standard-Error

- Fehlermeldungen werden meist (wenn das Programm gut geschrieben ist) statt auf die Standardausgabe auf *standard error* geschrieben.
- Die Fehler-Ausgabe kann umgeleitet werden mittels `2>` or `2>>`
- Beispiel:

```
cat f1 f2 nofile > newfile 2> errfile
```

- Hinweis: `1` ist der Deskriptor für Standard-Ausgabe, daher ist `1>` equivalent zu `>`
- Standard- und Fehler-Ausgabe können zur gleichen Datei geleitet werden mittels `&>`

```
cat f1 f2 nofile &> wholefile
```

Der Befehl yes

Nützlich, um die Standard-Eingabe immer mit der gleichen Zeichenkette zu füllen

- `yes <string> | <command>`
  Füllt die Standard-Eingabe von `<command>` mit `<string>` (Fehlwert `y`)
- Beispiele
  ```
  yes | rm -r dir/
  bank> yes no | credit_applicant
  yes "" | make oldconfig
  ```
  (entspricht dem Drücken von `Enter` zum Akzeptieren aller Fehlwerte)

Spezial-Dateien

Sehen aus wie richtige Dateien, aber

- `/dev/null`
  Der Datenabfluß! Verwirft alle Daten, die darauf geschrieben werden.
  Nützlich, um unerwünschte Ausgaben zu entfernen, typischerweise Log-Informationen:
  ```
  mplayer black_adder_4th.avi &> /dev/null
  ```

- `/dev/zero`
  Lesen aus dieser Datei gibt immer `\0` Zeichen zurück
  Nützlich zum Füllen einer Datei mit Nullen:
  ```
  dd if=/dev/zero of=disk.img bs=1k count=2048
  ```

Task-Steuerung

Vollständige Task-Kontrolle

- Seit dem Anfang unterstützt Unix wirklich preemptives Multitasking.
- Fähigkeit, viele Tasks parallel auszuführen, und die Möglichkeit des Abbruchs, wenn sie ihren eigenen Zustand und Daten gefährden.
- Fähigkeit der Auswahl, welche Programme Sie starten.
- Fähigkeit der Bestimmung der Eingabe Ihres Programmes und wohin die Ausgabe geht.

Prozesse

“Alles in Unix ist eine Datei
Alles, was keine Datei ist, ist ein Prozess”

Prozesse

- Instanzen laufender Programme
- Mehrere Instanzen des gleichen Programms können zur gleichen Zeit laufen
- Mit Prozessen verbundene Daten:
  Offene Dateien, zugeordneter Speicher, Stack, Prozess-ID, Parent, Priorität, Zustand ...

Jobs im Hintergrund ausführen

Gleiche Verwendung in allen Shells

- Nützlich
  - Für Befehlszeilenprogramme, derer Ausgabe später untersucht werden kann, besonders für zeitintensive.
  - Zum Start grafischer Programme von der Befehlszeile und Fortsetzung per Maus-Bedienung.
- Start einer Task: fügen Sie `&` am Ende der Zeile hinzu:

```
find_prince_charming --cute --clever --rich &
```


Steuerung der Hintergrund-Jobs

- `jobs`
  Zeigt die Liste der Hintergrund-Jobs aus der gleichen Shell

```
[1]-  Running ~/bin/find_meaning_of_life --without-god &
[2]+  Running make mistakes &
```

- `fg`
  `fg %<n>`
  Holt den letzten / n.ten Hintergrund-Job in den Vordergrund
- Aktuelle Task in den Hintergrund stellen:
  `[Ctrl] Z`
  `bg`
- `kill %<n>`
  Bricht den n.ten Job ab.

Beispiel Job-Steuerung

```
> jobs
[1]-  Running ~/bin/find_meaning_of_life --without-god &
[2]+  Running make mistakes &

> fg
make mistakes

> [Ctrl] Z
[2]+  Stopped make mistakes

> bg
[2]+ make mistakes &

> kill %1
[1]+  Terminated ~/bin/find_meaning_of_life --without-god
```


Auflistung aller Prozesse

... gleich, von welcher Shell, Skript oder Prozess sie gestartet wurden

- `ps -ux`
  Zeigt alle dem aktuellen Benutzer gehörenden Prozesse
- `ps -aux` (Hinweis: `ps -edf` auf System V Systemen)
  Zeigt alle auf dem System laufenden Prozesse
-
```
ps -aux | grep bart | grep bash
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
bart      3039  0.0  0.2   5916  1380 pts/2    S    14:35   0:00 /bin/bash
bart      3134  0.0  0.2   5388  1380 pts/3    S    14:36   0:00 /bin/bash
bart      3190  0.0  0.2   6368  1360 pts/4    S    14:37   0:00 /bin/bash
bart      3416  0.0  0.0      0     0 pts/2    RW   15:07   0:00 [bash]
```
- PID: Process id
  VSZ: Virtual process size (code + data + stack)
  RSS: Process resident size: number of KB currently in RAM
  TTY: Terminal
  STAT: Status: R (Runnable), S (Sleep), W (paging), Z (Zombie)...

Prozesse löschen (1)

- `kill <pids>`
  Sendet ein Abbruchsignal an die benannten Prozesse. Ermöglicht ihnen das Speichern von Daten und eigenes Beenden. Sollte zuerst benutzt werden. Beispiel:
  `kill 3039 3134 3190 3416`
- `kill -9 <pids>`
  Sendet ein unmittelbares Abbruchsignal. Das System selbst beendet die Prozesse. Nützlich, wenn ein Prozess wirklich hängt (reagiert nicht auf `kill -1`).
- `kill -9 -1`
  Löscht alle Prozesse des aktuellen Benutzers. `-1`: bedeutet alle Prozesse.

Prozesse löschen (2)

- `killall [-<signal>] <command>`
  Löscht alle Jobs des Befehls `<command>`. Beispiel:
  `killall bash`
- `xkill`
  Ermöglicht das Löschen einer grafischen Anwendung durch Klick darauf!
  Sehr schnell! Bequem, wenn Sie nicht den Namen der Anwendung wissen.

Aktuelle Prozess-Aktivität

- `top` – Zeigt die wichtigsten Prozesse an, sortiert nach CPU-Prozentnutzung

```
top - 15:44:33 up  1:11,  5 users,  load average: 0.98, 0.61, 0.59
Tasks:  81 total,   5 running,  76 sleeping,   0 stopped,   0 zombie
Cpu(s): 92.7% us,  5.3% sy,  0.0% ni,  0.0% id,  1.7% wa,  0.3% hi,  0.0% si
Mem:    515344k total,   512384k used,     2960k free,    20464k buffers
Swap:  1044184k total,        0k used,  1044184k free,   277660k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
3809 jdoe      25   0  6256 3932 1312 R 93.8  0.8   0:21.49 bunzip2
2769 root      16   0  157m  80m  90m R  2.7 16.0   5:21.01 X
3006 jdoe      15   0 30928  15m  27m S  0.3  3.0   0:22.40 kdeinit
3008 jdoe      16   0  5624  892 4468 S  0.3  0.2   0:06.59 autorun
3034 jdoe      15   0 26764  12m  24m S  0.3  2.5   0:12.68 kscd
3810 jdoe      16   0  2892  916 1620 R  0.3  0.2   0:00.06 top
```

- Sie könen die Sortier-Ordnung ändern durch
  `M`: Speichernutzung, `P`: %CPU, `T`: Zeit.

- Sie können eine Task durch Eingabe von `k` und der Prozess-ID löschen.

Hängende Grafik-Anwendung wiederbeleben

- Wenn Ihre grafische Sitzung hängt und Sie keine Eingaben mehr machen können, rebooten Sie nicht!
- Es ist sehr wahrscheinlich, das Ihr System noch OK ist. Versuchen Sie mittels der Tasten `[Ctrl][Alt][F1]` (oder `[F2]`,`[F3]` für weitere Textkonsolen) eine Textkonsole zu öffnen.
- In der Textkonsole können Sie versuchen, die fehlerhafte Anwendung zu löschen.
- Danach können Sie zurück in Ihre grafische Sitzung mittels `[Ctrl][Alt][F5]` oder `[Ctrl][Alt][F7]` (abhängig von Ihrer Distribution)
- Wenn Sie das hängende Programm nicht erkennen, können Sie auch alle Ihre Prozesse löschen mittels: `kill -9 -1`
  Sie kommen dann zum Anmelde-Bildschirm zurück.

Sequentielle Befehle

- Der nächste Befehl kann schon eingegeben werden, selbst wenn der aktuelle noch nicht beendet ist.
- Befehle können separiert werden mit dem `;` Symbol:
  ```
  echo "I love thee"; sleep 10;  echo " not"
  ```
- Bedingungen: use `||` (or) or `&&` (and):
  ```
  more God || echo "Sorry, God doesn't exist"
  ```
  Startet `echo` nur, wenn der erste Befehl fehlerhaft endet

  ```
  ls ~sd6 && cat ~sd6/* > ~sydney/recipes.txt
  ```
  Verknüpft die Verzeichnisinhalte nur, wenn der `ls` Befehl erfolgreich ist (bedeutet Leserecht).

Maskieren (1)

Doppelte (") Anführungszeichen können benutzt werden, um eine Interpretation von Leerzeichen als Argument-Trenner oder die Dateinamen-Expansion durch die Shell zu unterdrücken.

```
> echo "Hello World"
  Hello World

> echo "You are logged as $USER"
  You are logged as bgates

> echo *.log
  find_prince_charming.log cosmetic_buys.log

> echo "*.log"
  *.log
```

Maskieren (2)

Einzelne Anführungszeichen haben eine ähnliche Funktionalität, aber alles zwischen den Zeichen wird nie ersetzt.

```
> echo 'You are logged as $USER'
  You are logged as $USER
```

Umgekehrte Anführungsz. (`) können benutzt werden, um einen Befehl innerhalb eines anderen aufzurufen

```
> cd /lib/modules/`uname -r`; pwd
  /lib/modules/2.6.9-1.6_FC2
```

Diese können auch innerhalb doppelter Anführungszeichen benutzt werden

```
> echo "You are using Linux `uname -r`"
  You are using Linux 2.6.9-1.6_FC2
```


Zeitbedarf messen

- `time find_expensive_housing --near <...command output...>`

```
real    0m2.304s  (actual elapsed time)
user    0m0.449s  (CPU time running program code)
sys     0m0.106s  (CPU time running system calls)
```

real = user + sys + *waiting*

waiting = I/O waiting time + idle time (Ablauf anderer Tasks)

Umgebungsvariablen

- Shells ermöglichen dem Benutzer die Definition von *Variablen*.
  Sie können in Shells benutzt werden.
  Konvention: klein geschrieben
- Sie können auch *Umgebungs-Variablen* definieren: sie sind auch sichtbar innerhalb von Skripten oder Programmen, die von der Shell aufgerufen werden.
  Konvention: groß geschrieben
- `env`
  Zeigt alle definierten Umgebungs-Variablen und deren Werte

Shell-Variablen: Beispiele

Shell-Variablen (bash)

- `projdir=/home/marshall/coolstuff`
  `ls -la $projdir; cd $projdir`

Umgebungs-Variablen (bash)

- `cd $HOME`
- `export DEBUG=1`
  `./find_extraterrestrial_life`
  (zeigt Debug-Informationen, wenn `DEBUG` gesetzt ist)

Standard-Umgebungs- Variablen

Von vielen Anwendungen genutzt!

- `LD_LIBRARY_PATH`
  Suchpfad für Laufzeitbibliotheken
- `DISPLAY`
  Screen-ID für X-(grafische) Anwendungen.
- `EDITOR`
  Standard-Editor (vi, emacs...)
- `HOME`
  Benutzer-Startverzeichnis
- `HOSTNAME`
  Name des lokalen Rechners
- `MANPATH`
  Suchpfad Handbuchseiten
- `PATH`
  Suchpfad für Befehle
- `PRINTER`
  Standarddrucker
- `SHELL`
  Name der aktuellen Shell
- `TERM`
  aktueller Terminalname
- `USER`
  aktueller Benutzername

PATH Umgebungs-Variablen

- `PATH`
  Spezifiziert die Suchreihenfolge für Befehle

```
/
home/acox/bin:/usr/local/bin:/usr/kerberos/bin:/usr
/bin:/bin:/usr/X11R6/bin:/bin:/usr/bin
```

- `LD_LIBRARY_PATH`
  Spezifiert die Suchordnung für Laufzeit-Bibliotheken (von Anwendungen gemeinsam genutzte Binärcode-Bibliotheken wie die C-Bibliothek) für `ld`

```
/usr/local/lib:/usr/lib:/lib:/usr/X11R6/lib
```

- `MANPATH`
  Spezifiert die Suchordnung für Handbuchseiten

```
/usr/local/man:/usr/share/man
```

Warnung: PATH Benutzung

Es wird sehr empfohlen, das “.”-Verzeichnis nicht in die Variable `PATH` aufzunehmen, vor allem nicht an den Anfang:

- Ein Cracker könnte ein bösartiges `ls` in Ihre Verzeichnisse stellen. Es würde ausgeführt werden, wenn Sie in dem Verzeichnis `ls` starten und könnte Ihren Daten Böses antun.
- Wenn Sie ein Programm namens `test` in einem Verzeichnis haben, setzt dies das Standard `test` Programm außer Kraft und einige Skripte arbeiten nicht mehr ordnungsgemäß.
- Wann immer Sie in ein neues Verzeichnis wechseln, verschwendet die Shell Zeit, um die Liste verfügbarer Befehle zu aktualisieren.

Starten Sie Ihre lokalen Befehle wie folgt: `./test`

Alias

Shells ermöglichen die Definition von Befehls *aliasen*: Abkürzungen für häufig benutzte Befehle

Beispiele

- `alias ls='ls -la'`
  Nützlich, um Befehle immer mit gleichen Argumenten aufzurufen
- `alias rm='rm -i'`
  Nützlich, damit `rm` immer nach Bestätigung fragt
- `alias frd='find_rambaldi_device --asap --risky'`
  Nützlich, um sehr lange und häufige Befehle zu ersetzen
- `alias cia='. /home/sydney/env/cia.sh'`
  Nützlich, um schnell eine Umgebung zu setzen
  (`.` ist ein Shell-Befehl, um den Inhalt eines Shell-Skripts auszuführen)

Der Befehl which

Bevor Sie einen Befehl ausführen, zeigt `which` Ihnen, wo er gefunden wird

- ```
  bash> which ls
  alias ls='ls --color=tty'
          /bin/ls
  ```
- ```
  tcsh> which ls
  ls:       aliased to ls --color=tty
  ```
- ```
  bash> which alias
  /usr/bin/which: no alias in
  (/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin)
  ```
- ```
  tcsh> which alias
  alias: shell built-in command.
  ```


~/.bashrc-Datei

- `~/.bashrc`
  Dieses Shell-Skript wird gelesen, wenn eine `bash` Shell gestartet wird
- Dies kann benutzt werden zur Definition von
  - Ihren Standard-Umgebungsvariablen (`PATH`, `EDITOR`...)
  - Aliasnamen
  - Ihrem Prompt (siehe das `bash` Handbuch zu Einzelheiten)
  - einer Begrüßungsmeldung

verschiedene Hilfsprogramme

Befehlszeileneditierung

- Sie können die Links- und Rechtspfeil-Tasten zum Bewegen des Cursors benutzen.
- Mit `[Ctrl][a]` kommen Sie zum Anfang und mit `[Ctrl][e]` zum Ende der Zeile.
- Mit den Rauf- und Runter-Pfeilen selektieren Sie frühere Befehle.

Befehlsverlauf (1)

- `history`
  Zeigt die zuletzt ausgeführten Befehle und ihre Nummer. Sie können Befehle kopieren und einfügen.
- Sie können den letzten Befehl zurückholen:
  `!!`
- Sie können einen Befehl über seine Nummer zurückholen
  `!1003`
- Sie können den letzten Befehl zurückholen, der einem Start-Muster entspricht:
  `!cat`

Befehlsverlauf (2)

- Sie können Ersetzungen für den letzten Befehl durchführen:
  `^more^less`
- Sie können einen anderen Befehl mit den gleichen Argumenten ausführen:
  `more !*`

Text-Editoren

Grafische Text-Editoren

Ausreichend für die meisten Bedürfnisse

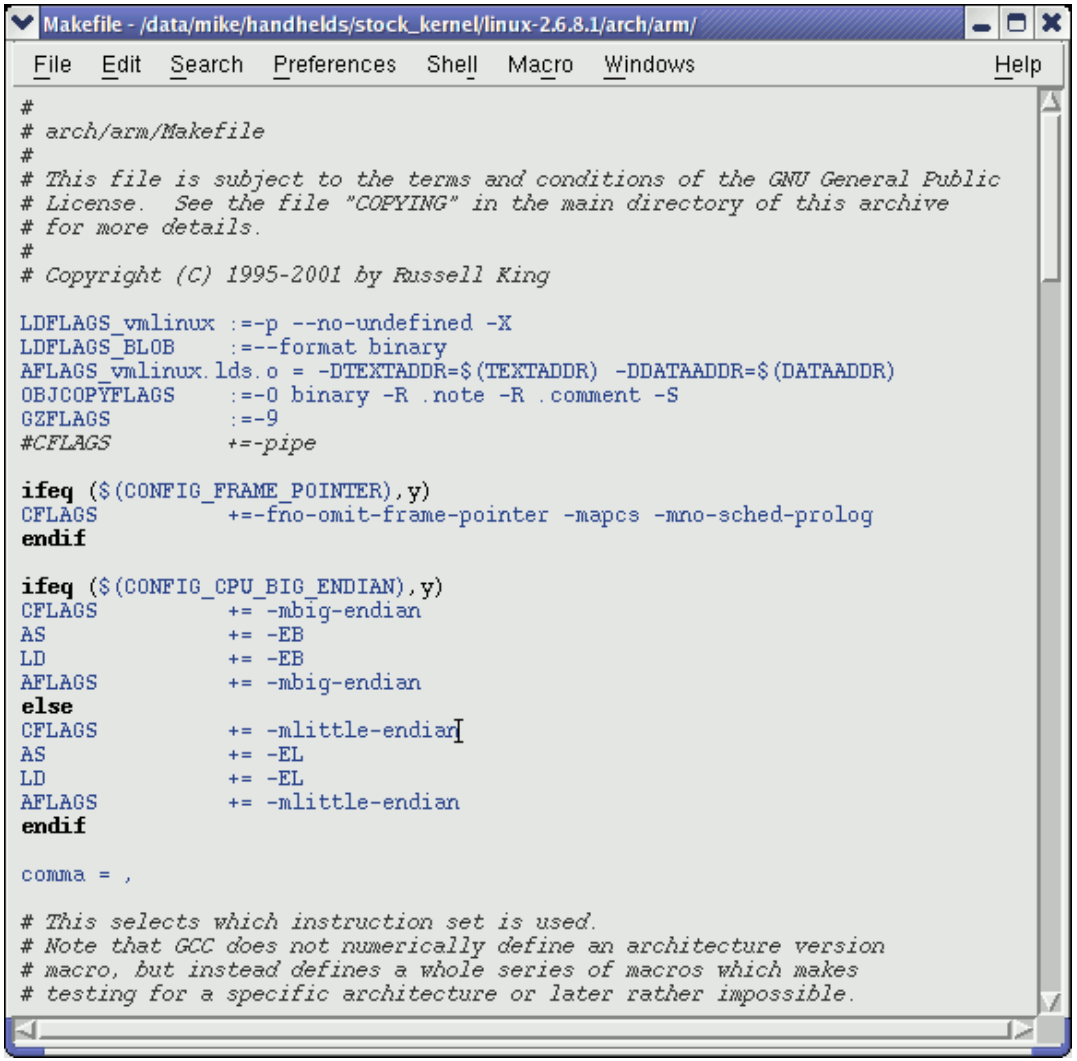
- nedit
- Emacs, Xemacs

Nur-Text Text-Editoren

Oft für Systemadministratoren benötigt und wichtig für "power user"

- vi
- nano

nedit

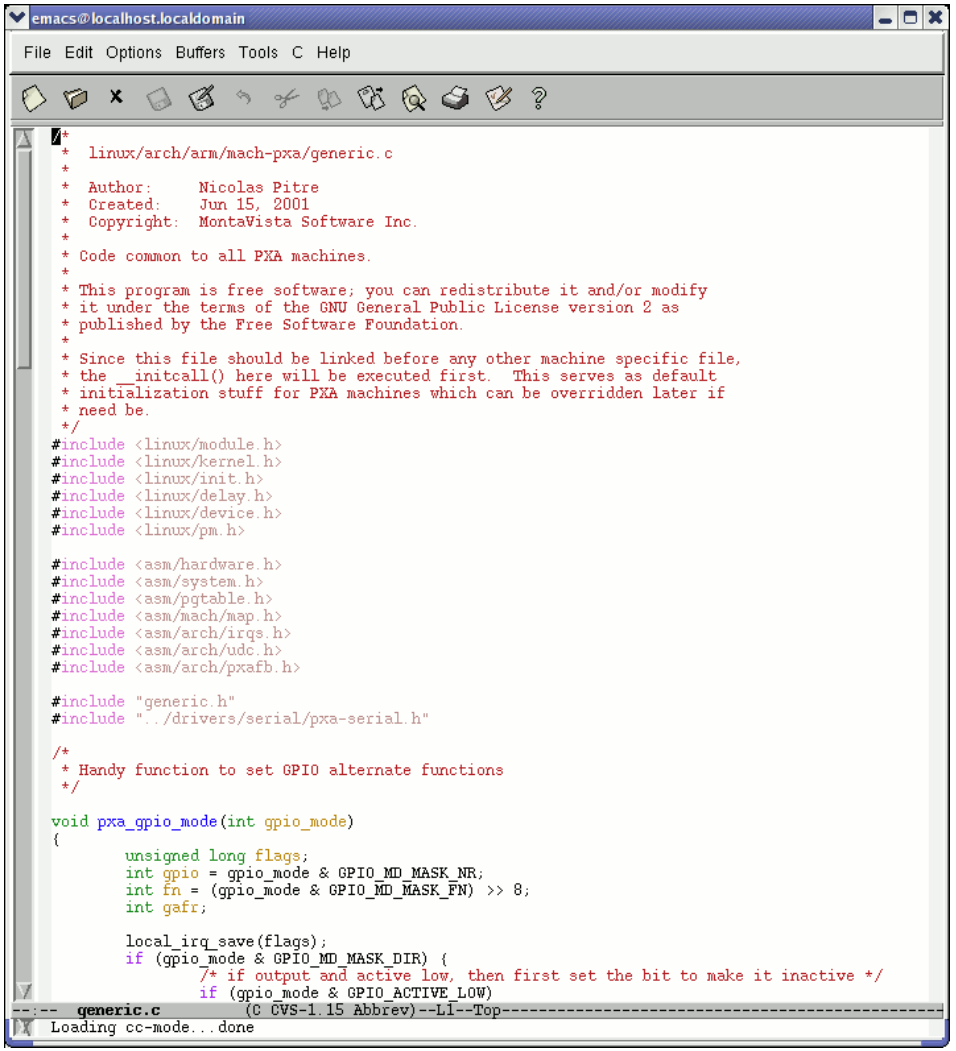


http://www.nedit.org/

nedit (2)

- Bester Text-Editor für Menschen, die keine `vi` oder `emacs` Experten sind
- Besondere Eigenschaften:
  - Sehr einfache Textauswahl und -Verschiebung
  - Syntax-Hervorhebung für die meisten Sprachen und Formate. Kann für Ihre eigenen Logdateien angepasst werden, um besondere Fehler und Warnungen hervorzuheben.
  - Leicht über Menues anzupassen.
- Standardmäßig nicht von allen Distributionen installiert

Emacs / Xemacs



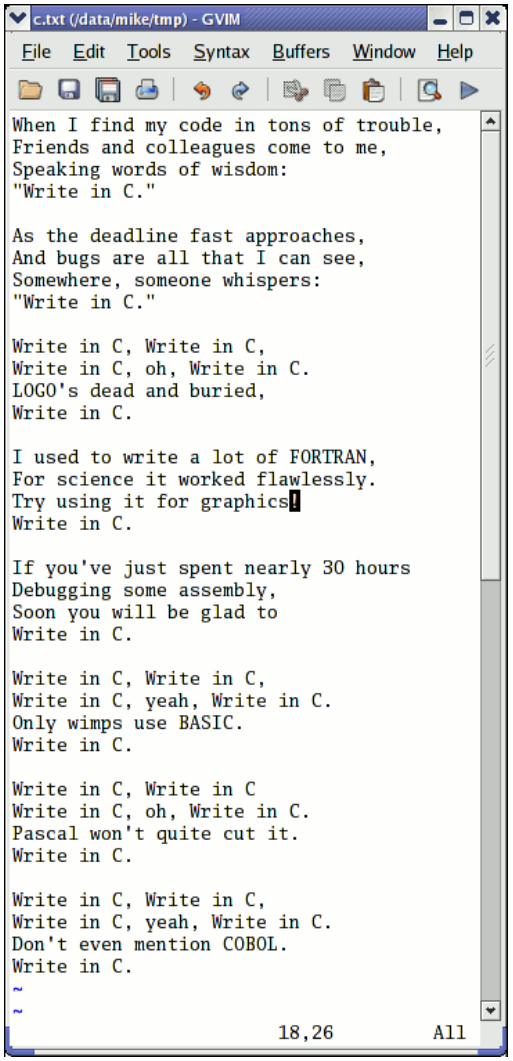
- Emacs und Xemacs sind sehr ähnlich (von Ihren Einstellungen abhängig)
- Extrem mächtige Text-Editor-Eigenschaften
- Großartig für “power user”
- Weniger ergonomisch als `nedit`
- Keine Standard-Abkürzungstasten
- Viel mehr als ein Texteditor (Spiele, Email, Shell, Browser)
- Einige mächtige Befehle müssen erlernt werden

vi

Textmodus-Texteditor, der in allen Unix-Systemen verfügbar ist. Erschaffen, bevor es Computer mit Mäusen gab.

- Schwierig zu erlernen für Anfänger, die grafische Texteditoren gewöhnt sind.
- Sehr produktiv für “power user”.
- Kann oft nicht ersetzt werden für das Editieren von Dateien der Systemadministration oder in eingebetteten Systemen, wenn nur eine Textkonsole zur Verfügung steht.

vim - vi improved



- `vi` -Implementierung, die sich in den meisten GNU / Linux-Systemen findet
- Implementiert viele der Fähigkeiten moderner Editoren: Syntaxhervorhebung, Hilfe, beschränktes Rückgängigmachen und vieles mehr.
- Coole Eigenschaft: kann direkt komprimierte Textdateien öffnen.
- Kommt mit einer grafischen GTK-Oberfläche (`gvim`)
- Leider keine freie Software (wegen einer kleinen Beschränkung hinsichtlich der Freiheit bei Änderungen)

vi Zusammenfassung einfacher Befehle

[Esc]	Switch to command mode (when in edit mode)
i	Insert text
a	Identical to insert command, except starts inserting at character after cursor
r	Replace character under cursor
R	Replace multiple characters
J	Join lines (default 2)
ndd	Delete n lines. Deleted lines are copied to buffer
nyy	Yank n lines. n lines starting at cursor line are copied to buffer
p	Paste contents of buffer after the line that contains the cursor
D	Delete from cursor position to end of line
nG	Goto line n. If n is not specified, goes to the last line
H	Goto first line of file
/string	Find next occurrence of string
:1,$s/str1/str2/g	Replaces every occurrence of str1 with str2, starting from line 1 to the end of text
n	Find next occurrence of the last search string
?string	Find previous occurrence of string
^f	Go forward a page
^b	Go backward a page
h	Move cursor left
l	Move cursor right
j	Move cursor down
k	Move cursor up
^L	Redraw screen
u	Undo the latest change
U	Undo all changes on a line, while not having moved off of it
:wq	Save file and exit vi
:w name	Save to file “name”
:x,y w name	Writes lines x through y to file “name”
:q!	Exit vi without saving changes
:f	Displays file information on bottom on screen

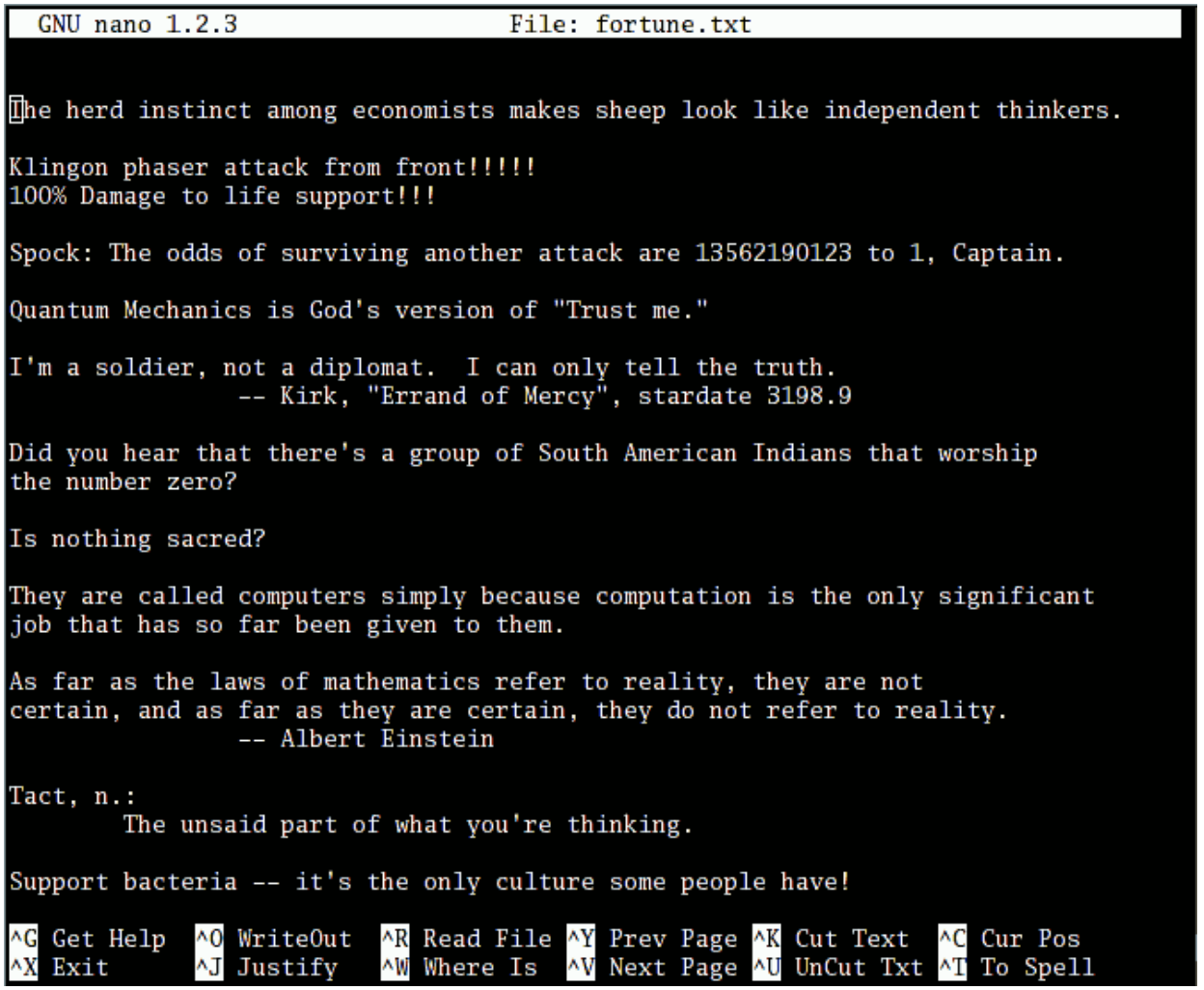
Starten Sie `vimtutor` um mehr zu lernen!
Nur 30 Minuten zum Bearbeien des Handbuches!

GNU nano

http://www.nano-editor.org/

- Ein weiterer Nur-Text-, Maus-freier Editor.
- Ein erweiteter Pico -Nachbau-Clone (nicht freier Editor in Pine)
- Freundlich und für Anfänger leicht zu lernen dank Befehlszusammenfassungen auf dem Bildschirm.
- Verfügbar in Binärpaketen für mehrere Plattformen.
- Eine Alternative zu vi in eingebetteten Systemen. Jedoch nicht in busybox enthalten.

GNU nano Bildschirmfoto



Verschiedenes
Komprimierung und Archivierung

Dateigrössen berechnen

- `du -h <file>` (Disk Benutzung)
  `-h`: gibt die Größe der Datei zurück, in menschlich lesbarer Form: K (Kilobyte), M (Megabyte) oder G (Gigabyte) Sonst gibt `du` die Anzahl der von der Datei belegten Plattenblöcke zurück (schwer zu lesen).

  Hinweis: Die `-h` -Option existiert nur in GNU `du`. Z. B. nicht verfügbar im Sun Solaris `du`.

- `du -sh <dir>`
  `-s`: gibt die Summer der Grösse aller Dateien im Verzeichnis zurück.

Plattenplatz berechnen

- `df -h <dir>`
  Gibt Plattenbelegung und freien Platz des Dateisystems zurück, welches das Verzeichnis enthält.
  Die `-h` -Option gibt es auch nur in GNU `df`.

- Beispiel:

```
> df -h .
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda5             9.2G  7.1G  1.8G  81% /
```

- `df -h`
  Zeigt Platteninformationen über alle Dateisysteme im System.
  Nützlich, um bei Fehlern nach vollen Dateisystemen zu suchen.

Komprimierung

Sehr nützlich zum Verkleinern großer Dateien und für Platzeinsparungen

- `[un]compress <file>`
  Traditionelles Unix-Komprimierungsprogramm. Erstellt `.Z` -Dateien.
  Nur noch für Kompabilitätszwecke. Durchschnittliches Performanz.
- `g[un]zip <file>`
  GNU zip-Komprimierungsprogramm. Erstelle `.gz` -Dateien.
  Recht gute Performanz (ähnlich Zip).
- `b[un]zip2 <file>`
  Das neueste und effektivste Komprimierungs-Programm. Erstellt `.bz2` -Dateien. Allgemein 20-25 % besser als `gzip`.
  Benutzen Sie es! Nun auf allen Unix-Systemen vorhanden.

Archivierung (1)

Nützlich zum Sichern oder Veröffentlichen einer Dateimenge in einer Datei

- `tar`: ursprünglich “tape archive”
- Erstellung eines Archives:
  `tar cvf <archive> <files or directories>`
  `c`: create
  `v`: verbose. Nützlich zum Verfolgen des Archivierungsvorgangs
  `f`: file. Archiv wird in Datei erstellt (sonst wird ein Band benutzt)
- Beispiel:
  ```
  tar cvf /backup/home.tar /home
  bzip2 /backup/home.tar
  ```


Archivierung (2)

- Inhaltsanzeige oder Integritätsprüfung eines Archives:
  `tar tvf <archive>`
  `t`: test
- Extrahieren aller Dateien aus einem Archiv:
  `tar xvf <archive>`
- Extrahierung von einigen Dateien aus einem Archiv:
  `tar xvf <archive> <files or directories>`
  Dateien/Verzeichnisse werden mit relativen Pfaden hinsichtlich des Archiv-Start-Verzeichnisses angegeben.

Zusätzliche Optionen in GNU tar

`tar` = `gtar` = GNU `tar` on GNU / Linux

Kann Archive während des (Ent-)Packens (de)komprimieren.
Nützlich, um riesige Zwischendateien zu vermeiden.
Einfacher als mit `tar` und `bzip2`!

- `j` Option: [de]komprimiere mit `bzip2`
- `z` Option: [de]komprimiere mit `gzip`
- Beispiele (welches merken Sie sich?)
  - `gtar jcvf bills_bugs.tar.bz2 bills_bugs`
  - `tar cvf - bills_bugs | bzip2 > bills_bugs.tar.bz2`

Der wget-Befehl

Anstatt Dateien mit Ihrem Browser abzurufen, kopieren Sie einfach die URL und rufen Sie die Datei mit `wget` ab!

`wget` Eigenschaften

- http und ftp-Unterstützung
- Kann abgebrochene Übertragung fortsetzen
- Kann ganze Webseiten abrufen oder zumindest auf defekte Links testen
- Sehr nützlich in Skripten oder bei grafiklosen Systemen (System-Administration, eingebettete Systeme)
- Proxy-Unterstützung (`http_proxy` und `ftp_proxy` Umgebungsvariablen)

wget Beispiele

- `wget -c \`
  `http://microsoft.com/customers/dogs/winxp4dogs.zip`
  Setzt einen abgebrochenen Abruf fort
- `wget -m http://lwn.net/`
  Spiegelt eine Seite
- `wget -r -np http://www.xml.com/ldd/chapter/book/`
  Rekursiver Abruf eines Online-Buches für lokalen Zugriff.
  `-np`: "no-parent". Folgt nur Links im aktuellen Verzeichnis.

Datei-Integrität testen

Sehr kostengünstige Lösung zum Testen der Datei-Integrität

- `md5sum FC3-i386-disk*.iso > MD5SUM`
  Berechnet eine MD5 (Message Digest Algorithm 5) 128-Bit-Prüfsumme der Dateien. Ausgabe wird üblicherweise in eine Datei geleitet.

- Beispielausgabe:
```
db8c7254beeb4f6b891d1ed3f689b412 FC3-i386-disc1.iso
2c11674cf429fe570445afd9d5ff564e FC3-i386-disc2.iso
f88f6ab5947ca41f3cf31db04487279b FC3-i386-disc3.iso
6331c00aa3e8c088cc365eeb7ef230ea FC3-i386-disc4.iso
```

- `md5sum -c MD5SUM`
  Prüft die Integrität der Dateien in `MD5SUM` durch Vergleich ihrer aktuellen MD5 -Prüfsumme mit dem Original.

Verschiedenes

Drucken

Unix-Drucken

- Multi-user, multi-job, multi-client, multi-printer
  In Unix / Linux drucken Druckbefehle nicht wirklich. Sie senden Aufträge an Druckschlangen, evtl. auf der lokalen Maschine, an Netzwerkdruckserver oder Netzwerkdrucker

- Drucker-unabhängiges System:
  Druckserver akzeptieren Aufträge nur in Textformat oder PostScript. Druckertreiber auf dem Server kümmern sich um die Konvertierung in das Format des Druckers.
- Robustes System:
  Nach einem Reboot werden anhängige Aufträge weiter gedruckt.

Druckbefehle

- Nützliche Umgebungsvariable: `PRINTER`
  Setzt den Standarddrucker auf dem System. Beispiel:
  `export PRINTER=lp`
- `lpr [-P<queue>] <files>`
  Sendet die Dateien an die angegebene Druckschlange.
  Die Dateien müssen im Text- oder PostScript- Format vorliegen.
  Sonst drucken Sie nur Müll.
- `a2ps [-P<queue>] <files>`
  “Any to PostScript” konvertiert viele Formate in PostScript und schickt die Ausgabe an die angegebene Warteschlange. Nützliche Eigenschaften: mehrere Seiten/Blatt, Seitennummerierung, Informationsrahmen ...

Druckjob-Steuerung

- `lpq [-P<queue>]`
  Zeigt alle Druckaufträge in der angegebenen oder der Standard-Warteschlange.

```
lp is not ready
Rank    Owner   Job     File(s)                         Total Size
1st     asloane 84      nsa_windows_backdoors.ps        60416 bytes
2nd     amoore  85      gw_bush_iraq_mistakes.ps        65024000 bytes
```

- `cancel <job#> [<queue>]`
  Entfernt den angegebenen Job aus der (Standard-)Warteschlange

Benutzung von PostScript- und PDF-Dateien

Betrachten einer PostScript-Datei

- Es gibt PostScript-Betrachter, aber deren Qualität ist sehr schlecht.
- Besser in PDF konvertieren mit `ps2pdf`:

```
ps2pdf decss_algorithm.ps
xpdf decss_algorithm.pdf &
```

Druck einer PDF-Datei

- Sie müssen keinen PDF-Betrachter öffnen!
- Besser in PostScript konvertieren mit `pdf2ps`:

```
pdf2ps rambaldi_artifacts_for_dummies.pdf
lpr rambaldi_artifacts_for_dummies.ps
```


Verschiedenes

Dateien und Verzeichnisse vergleichen

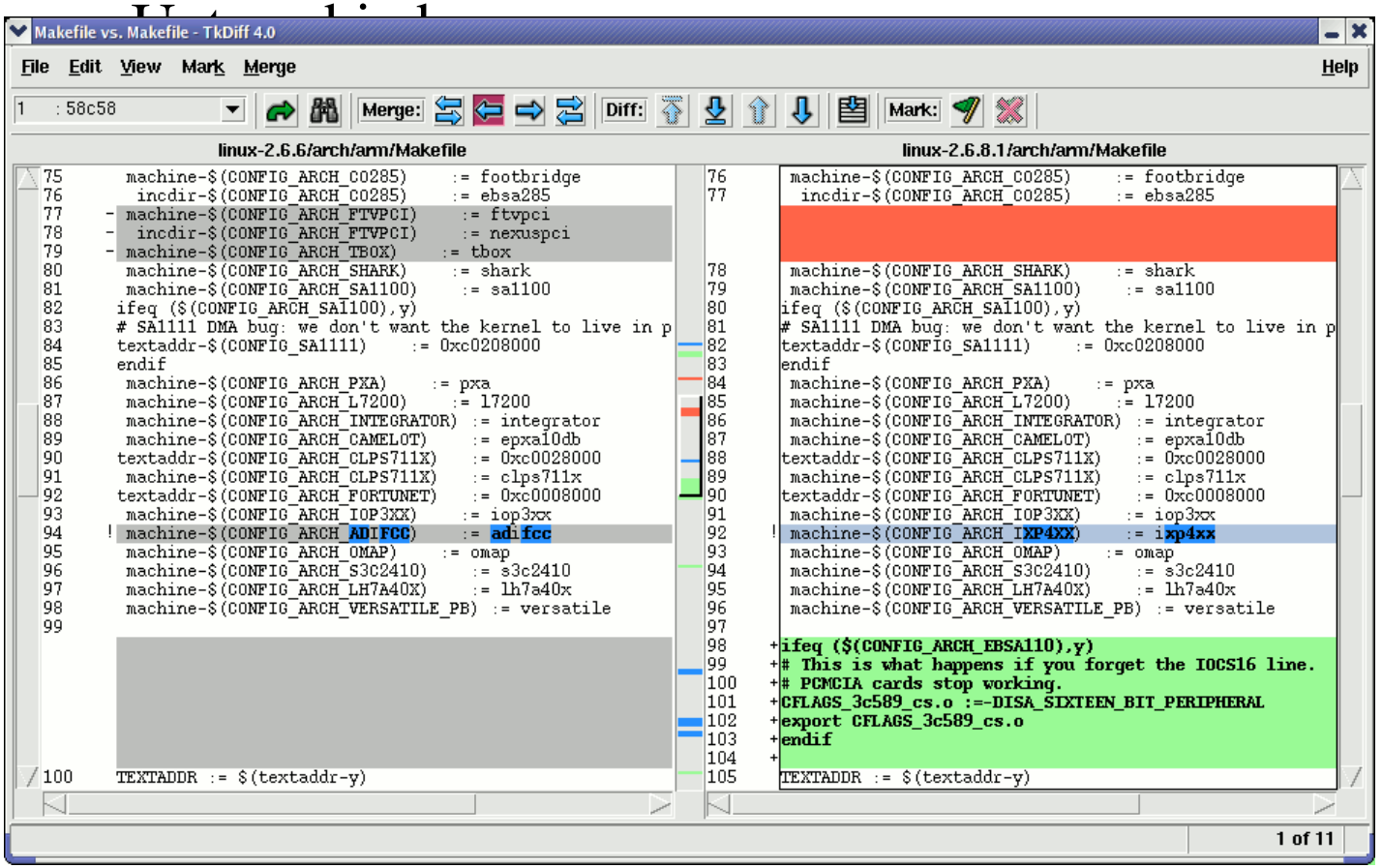
Dateien und Verzeichnisse vergleichen

- `diff file1 file2`
  Gibt die Unterschiede zwischen 2 Dateien aus, oder nichts, wenn die Dateien gleich sind.
- `diff -r dir1/ dir2/`
  Zeigt alle Unterschiede in Dateien mit gleichen Namen in den 2 Verzeichnissen.
- Um die Unterschiede im Detail zu untersuchen, nutzen Sie besser grafische Programme!

tkdiff

http://tkdiff.sourceforge.net/

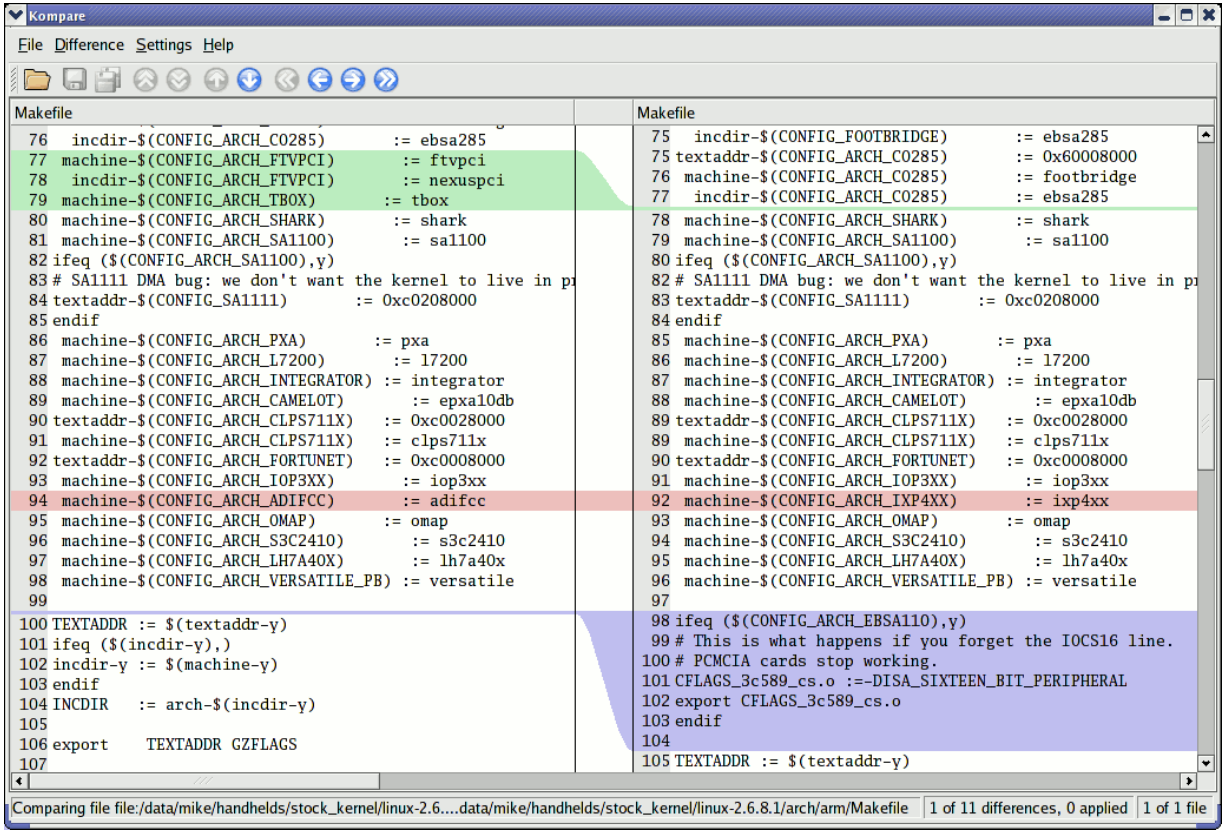
Nützliches Programm zum Datei-Vergleich und Zusammenführen



kompare

Datei-Vergleich und Zusammenführen von Unterschieden

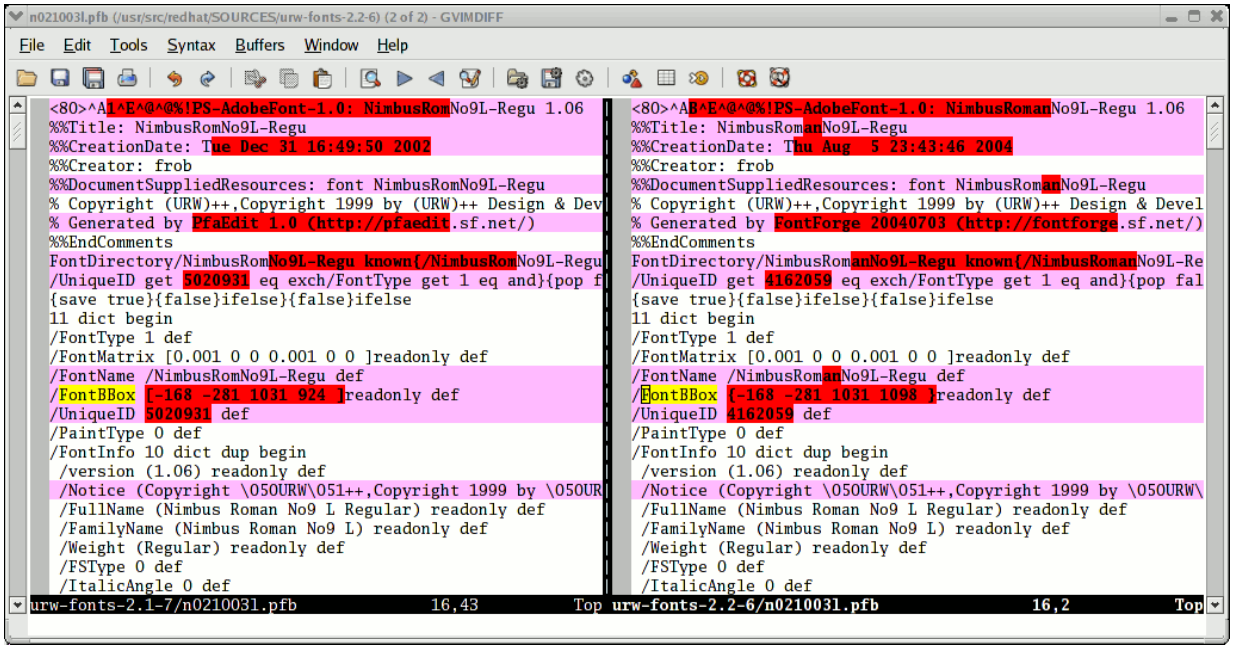
Teil des `kdesdk` -Paketes (Fedora Core)



gvimdiff

Noch ein Programm zum Anzeigen von Datei-Differenzen

In vielen Distributionen mit `gvim` verfügbar, verwendet nicht `diff`. Kein Problem mit Binärbereichen in Dateien!



Verschiedenes

Suchen nach Dateien

Der find-Befehl

Besser durch Beispiele erklärt!

- `find . -name "*.pdf"`
  Zeigt alle `*.pdf` -Dateien im aktuellen (`.`) Verzeichnis oder Unterverzeichnissen. Sie benötigen die doppelten Anführungszeichen, damit die Shell nicht das Zeichen `*` expandiert.
- `find docs -name "*.pdf" -exec xpdf {} ';'`
  Findet alle `*.pdf` -Dateien im Verzeichnis `docs` und zeigt sie nacheinander an.
- Viel mehr Möglichkeiten vorhanden! Die obigen 2 Beispiele decken aber die meisten Bedürfnisse ab.

Der locate-Befehl

Viel schnellere Suchalternative mit regulären Ausdrücken zu `find`

- `locate keys`
  Zeigt alle Dateien im System mit `keys` in ihren Namen.
- `locate "*.pdf"`
  Zeigt alle `*.pdf`-Dateien auf dem gesamten Rechner
- `locate "/home/fridge/*beer*"`
  Zeigt alle `*beer*` -Dateien im angegebenen Verzeichnis (absoluter Pfad)
- `locate` ist viel schneller, weil es alle Dateien in einer speziellen Datenbank indexiert, die regelmäßig aktualisiert wird.
- `find` ist besser geeignet zum Suchen frisch erstellter Dateien.

Verschiedenes

Verschiedene Befehle

Information über Anwender abrufen

- `who`
  Zeigt alle angemeldeten Anwender
- `whoami`
  Zeigt an, als welcher Anwender ich angemeldet bin
- `groups`
  Zeigt, zu welchen Gruppen ich gehöre
- `groups <user>`
  Zeigt, zu welchen Gruppen <user> gehört
- `finger <user>`
  Zeigt weitere Einzelheiten (richtiger Name, etc) über <user>
  In einigen Systemen aus Sicherheitsgründen deaktiviert

Anwender wechseln

Sie müssen sich zum Anwenderwechsel nicht ab- und anmelden!

- `su hyde`
  (Selten) Wechsel auf das `hyde` -Konto, die Umgebung des ursprünglichen Anwenders wird aber beibehalten.
- `su - jekyll`
  (Häufiger) Anmeldung in das `jekyll`-Konto, mit den gleichen Einstellungen wie der neue Anwender.
- `su -`
  Ohne Argument wird zum `root` -Anwender gewechselt.

Verschiedene Befehle (1)

- `sleep 60`
  Wartet 60 Sekunden (verbraucht keine System-Ressourcen)
- `wc report.txt` (word count)
  `438  2115 18302 report.txt`
  Zählt die Anzahl Zeilen, Worte und Zeichen in einer Datei oder der Standardeingabe.

Verschiedene Befehle (2)

- `bc` ("basic calculator?")
  `bc` ist ein praktischer, aber vollständiger Rechner.
  Enthält sogar eine Programmiersprache! Verwenden Sie die Option `-l` für die Mathematik-Bibliothek.
- `date`
  Gibt das aktuelle Datum zurück. Nützlich in Skripten, um den Start und das Ende von Befehlen zu verzeichnen.

Grundlagen der System-Administration

File ownership

- `chown -R sco /home/linux/src` (`-R`: recursive)
  Macht `sco` zum neuen Eigner aller Dateien in `/home/linux/src`
- `chgrp -R empire /home/askywalker`
  Macht `empire` zur neuen Gruppe für alles in `/home/askywalker`
- `chown -R borg:aliens usss_entreprise/`
  `chown` kann benutzt werden, um Eigner und Gruppe gleichzeitig zu ändern.

System-Stopp

- `shutdown -h +5` (`-h`: halt)
  Stoppt das System in 5 Minuten. Die Anwender erhalten eine Warnmeldung in ihrer Konsole.
- `shutdown -r now` (`-r`: reboot)
- `init 0`
  Eine andere Möglichkeit zum Stopp (benutzt von `shutdown`)
- `init 6`
  Andere Neustartmöglichkeit (benutzt von `shutdown`)
- `[Ctrl][Alt][Del]`
  Funktioniert auch unter GNU/Linux (zumindest bei PCs!)

Netzwerk-Einrichtung (1)

- `ifconfig -a`
  Zeigt Einzelheiten über alle im System verfügbaren Netzwerkschnittstellen an.
- `ifconfig eth0`
  Zeigt Einzelheiten über die Schnittstelle `eth0`
- `ifconfig eth0 192.168.0.100`
  Weist die IP-Adresse `192.168.0.100` IP an `eth0` zu (1 IP -Adresse pro Schnittstelle)
- `ifconfig eth0 down`
  Stoppt die Schnittstelle `eth0`
  (gibt die IP-Adresse frei)

Netzwerk-Einrichtung (2)

- `route add default gw 192.168.0.1`
  Setzt die Default-Route für Pakete außerhalb des lokalen Netzwerks. Das Gateway (hier `192.168.0.1`) ist für das Weiterleiten zum nächsten Gateway verantwortlich bis zur entgültigen Zielstation.
- `route`
  Zeigt die bestehenden Routen
- `route del default`
  `route del <IP>`
  Löscht die angegebene Route
  Nützlich für die Neudefinition einer Route.

Netzwerk-Tests

- `ping freshmeat.net`
  `ping 192.168.1.1`
  Versucht, Pakete an die angegebene Maschine zu senden und Bestätigungspakete zu erhalten.

```
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=150 time=2.51 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=150 time=3.16 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=150 time=2.71 ms
```

- Wenn Sie Ihr Gateway anpingen können, arbeitet Ihre Netzwerkschnittstelle ordnungsgemäß.
- Wenn Sie eine externe IP-Adresse anpingen können, sind Ihre Netzwerkeinstellungen korrekt!

Zusammenfassung Netzwerk-Einrichtung

Nur für einfache Fälle mit einer Schnittstelle, keinem DHCP-Server...

- Verbindung zum Netzwerk (Kabel, Funkkarte oder -Gerät ...)
- Netzwerk-Schnittstelle identifizieren:
  `ifconfig -a`
- Der Schnittstelle eine IP-Adresse zuweisen (angenommen `eth0`)
  `ifconfig eth0 192.168.0.100` (Beispiel)
- Dem Gateway eine Route hinzufügen (Annahme `192.168.0.1`) für Pakete außerhalb des Netzwerks:
  `route add default gw 192.168.0.1`

Namens-Auflösung

- Ihre Programme müssen wissen, welche IP-Adresse mit einem bestimmten Host korrespondiert (wie `kernel.org`)
- Domain Name Server (DNS) kümmern sich darum.
- Sie müssen nur die IP-Adresse eines oder mehrerer DNS-Server in die Datei `/etc/resolv.conf` eintragen:

```
nameserver 217.19.192.132
nameserver 212.27.32.177
```

- Die Änderungen wirken sich sofort aus!

Dateisysteme erstellen

Beispiele

- `mkfs.ext2 /dev/sda1`
  Formatiert Ihren USB-Stick (`/dev/sda1`: 1st Partition, Rohformat) im Format ext2
- `mkfs.ext2 -F disk.img`
  Formatiert ein Plattenabbild im Format ext2
- `mkfs.vfat -v -F 32 /dev/sda1` (`-v`: verbose)
  Formatiert Ihren USB-Stick zurück in das Format FAT32
- `mkfs.vfat -v -F 32 disk.img`
  Formatiert ein Plattenabbild im Format FAT32

Leere Plattenabbilder können wie im folgenden Beispiel erstellt werden:

```
dd if=/dev/zero of=disk.img bs=1024 count=65536
```


Geräte einhängen (1)

- Um Dateisysteme auf einem Gerät (interner oder externer Speicher) in Ihrem System sichtbar zu machen, müssen diese *gemount-ed* werden.
- Beim ersten Mal erstellen Sie einen Einhängepunkt in Ihrem System: `mkdir /mnt/usbdisk` (Beispiel)
- Dann hängen Sie es ein:
  `mount -t vfat /dev/sda1 /mnt/usbdisk`
  `/dev/sda1`: physisches Gerät
  `-t:` spezifiert den Dateisystemtyp (Format)
  (`ext2`, `ext3`, `vfat`, `reiserfs`, `iso9660`...)

Geräte einhängen (2)

- Unmenge von Mount-Optionen sind verfügbar, besonders zum Setzen der Rechte oder Datei-Eigner und -Gruppe. Zu Einelheiten siehe das Handbuch zu `mount`.
- Mount-Optionen für jedes Gerät können in der Datei `/etc/fstab` gespeichert werden.
- Sie können auch ein Dateisystem-Abbild einhängen, das in einer normalen Datei gespeichert ist (*loopback-Geräte*)
  - Nützlich zum Zugriff auf ein ISO-CD-Rom-Abbild, ohne dieses brennen zu müssen.
  - Nützlich zum Erstellen einer Linux-Partition auf einer Festplatte, die nur Windows-Partitionen enthält

```
cp /dev/sda1 usbkey.img
mount -o loop -t vfat usbkey.img /mnt/usbdisk
```


Anzeige eingehängter Dateisysteme

- Benutzen Sie einfach den Befehl `mount` ohne Argumente:

```
/dev/hda6 on / type ext3 (rw,noatime)
none on /proc type proc (rw,noatime)
none on /sys type sysfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/hda4 on /data type ext3 (rw,noatime)
none on /dev/shm type tmpfs (rw)
/dev/hda1 on /win type vfat (rw,uid=501,gid=501)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

- Oder zeigen Sie die Datei `/etc/mtab` an
  (gleiches Ergebnis, bei jedem Aufruf von `mount` und `umount` aktualisiert)

Geräte aushängen

- `umount /mnt/usbdisk`
  Führt alle ausstehenden Schreibvorgänge durch und hängt das benannte Gerät aus, das dann ohne Gefahr entfernt werden kann.
- Um ein Gerät aushängen zu können, müssen Sie alle offenen Dateien darauf schliessen:
  - Schliessen Sie Anwendungen, die Dateien in der eingehängten Partition geöffnet haben
  - Stellen Sie sicher, das keine Ihrer Shells ein Arbeitsverzeichnis in diesem Einhängepunkt hat.
  - Sie können den Befehl `lsof` (list open files) ausführen, um zu sehen, welche Prozesse immer noch offene Dateien in der eingehängten Partionen haben.

Tiefer einsteigen

Befehlshilfe

Einige Unix-Befehle und die meisten GNU/Linux-Befehle bieten zumindest ein Hilfe-Argument:

- `-h`
  (`-` wird zumeist für 1-buchstabige Optionen verwendet)
- `--help`
  (`--` wird immer benutzt um die entsprechende “lange” Option zu kennzeichnen, die das Verstehen von Skripten erleichtert)

Sie erhalten oft auch eine kurze Options-Zusammenfassung, wenn Sie ein ungültiges Argument benutzen.

Handbuch-Seiten

`man <keyword>`

Zeigt eine oder mehrere Handbuchseiten zu `<keyword>` an.

- `man man`

Die verfügbaren Handbuchseiten beziehen sich vor allem auf Unix-Befehle, einige aber auch auf C-Funktionen, Header-Dateien und Datenstrukturen oder sogar System-Konfigurationsdateien!

- `man stdio.h`
- `man fstab` (für `/etc/fstab`)

Nach Handbuchseiten wird in den Verzeichnissen gesucht, die in der Umgebungs-Variablen `MANPATH` angegeben sind.

Info-Seiten

- In GNU werden Handbuch-Seiten durch info-Seiten ersetzt. Einige Handbuchseiten verweisen sogar ausdrücklich darauf, stattdessen die info-Seiten zu verwenden.

```
info <command>
```

- `info` -Eigenschaften:
  - Dokumentation ist in Sektionen (“nodes”) und Untersektionen (“subnodes”) gegliedert
  - Navigationsmöglichkeit in dieser Struktur: top, next, prev, up
  - Info-Seiten werden aus der gleichen texinfo-Quelle generiert wie die HTML-Dokumentations-Seiten

Ressourcen-Suche im Internet (1)

Problem untersuchen

- Viele Foren und Mailinglisten-Archive sind öffentlich, und werden häufig durch Google indexiert.
- Wenn Sie eine Fehlermeldung suchen, kopieren Sie diese wörtlich in das Suchformular, eingeschlossen in doppelten Anführungszeichen (“error message”). Die Chance ist groß, das bereits jemand das gleiche Problem hatte
- Vergessen Sie die Google Gruppen nicht: http://groups.google.com/ Diese Seite indexiert mehr als 20 Jahre Newsgroups-Nachrichten.

Ressourcen-Suche im Internet (2)

Suche nach Dokumentation

- Suchen Sie nach `<tool>` oder `<tool>Seite`, um die Tool- oder Projekt-Webseite und dort die aktuellsten Dokumentationsquellen zu finden.
- Suchen Sie nach `<tool> Dokumentation` oder `<tool> Manual` in Ihrer Lieblings-Suchmaschine.

Suche nach allgemeiner technischer Information

- WikiPedia: http://wikipedia.org
  Menge nützlicher Definitionen der Computer-Wissenschaft. Eine echte Enzyklopädie! Für jedermanns Beiträge offen.

Tiefer einsteigen

GNU / Linux zuhause benutzen

Überblick über Desktop-Anwendungen

Auf Bildschirm mit einem Projektor anzeigen!

- Mozilla: Web-Browser, Mail-Client und HTML-Editor
- Firefox: leichtgewichtiger, Mozilla-basierter Web-Browser
- OpenOffice: vollständige, MS Office-kompatible Bürosammlung: Textverarbeitung, Tabellenkalkulation, Präsentationen, Grafikprogramm ...
- GIMP: sehr mächtiger Graphik-Editor
- Gqview: Betrachter für Foto-Galerien
- Evolution: Email-Client/Kalendar-Programm ala Outlook

GNU/Linux Alternativen für Windows-Programme

Internet Explorer	Mozilla Firefox
IIS	Apache
Money	GNU Cash
MS Office	OpenOffice
MS Outlook	Evolution
MS Project	Mr Project (Planner)
Nero	k3b
Photoshop	The GIMP
WinAmp	xmms
W. Media Player	xine mplayer

Ich kenne nicht genug Windows-Programme.

Senden Sie uns mehr Beispiele!

GNU / Linux zuhause (1)

GNU / Linux ist auch eine gute Alternative für private Windows-Nutzer

Sicherheit

- Virenfrei
  Viele Viren werden geschrieben zur Ausnutzung von Windows-Sicherheitslücken und haben keine Auswirkung unter GNU / Linux
- Virengeschützt
  Selbst wenn Sie einen Linux-kompatiblen Virus ausführen, hat er keine Rechte, das System zu modifizieren.
- Fehlergeschützt
  Familienmitglieder können keine Systemdateien oder von anderen ändern. Sie können nur ihre eigenen Dateien beschädigen.
- Cracker-abweisend
  Selbst bei ständiger Internet-Anbindung ist das System für Cracker nicht attraktiv.

GNU / Linux zuhause (2)

Privatsphäre

- Ihr System sammelt und überträgt nicht heimlich Informationen über Ihre Film- oder Webvorlieben.

Anwenderfreundlich

- Ihre Programme werden von Anwendern für Anwender gemacht. Diese werden eher Ihren Bedürfnissen entsprechen.
- Sie können den Entwicklern leicht neue Eigenschaften vorschlagen.

Freiheit

- Ihre Daten gehören Ihnen. Sie sind nicht an proprietäre Anwendungen und (evtl. patentierte) Formate gebunden.
- Sie können Ihren Nachbarn helfen, indem Sie diese Programme mit ihnen teilen.
- Sie dürfen Ihre zuhause genutzten Programme auch im Büro verwenden!

GNU / Linux zuhause (3)

Eine Migration zu GNU / Linux ist möglich für:

- Büroarbeit: Textverarbeitung, Tabellenkalkulation, Präsentationen
- Internet: Web-Browsen und Email
- Multimedia: Video, Sound und Grafik (einschließlich Ddigitalkameras)
- Lernen über Computer und -Programmierung

Wenn Sie noch eine Windows-Kopie haben, können Sie diese behalten (Doppelboot) für:

- Spiele: Viele Spiele unterstützen immer noch nur Windows oder Mac.
- Benutzung spezieller proprietärer Programm oder Bildungs-CD-Roms
- Benutzung von durch GNU/Linux bisher nicht unterstützter Hardware

Versuchen Sie GNU / Linux ohne Risiko

Knoppix ist eine Live-GNU / Linux CD-Rom
http://knoppix.net

- Lädt GNU / Linux ins RAM, auf der Festplatte wird nichts installiert
- Erstaunliche Fähigkeiten zur Hardware-Erkennung.
- Mehr als 2 GB an Anwendungen verfügbar!
- Sie können auf Ihre Windows-Dateien zugreifen, sie öffnen oder sogar editieren mit GNU / Linux-Anwendungen.
- Eine großartige Möglichkeit, GNU / Linux zu probieren und zu demonstrieren!
- Bietet die Möglichkeit einer permanenten Installation auf Ihrer Festplatte

Benutzung von GNU / Linux-Distributionen

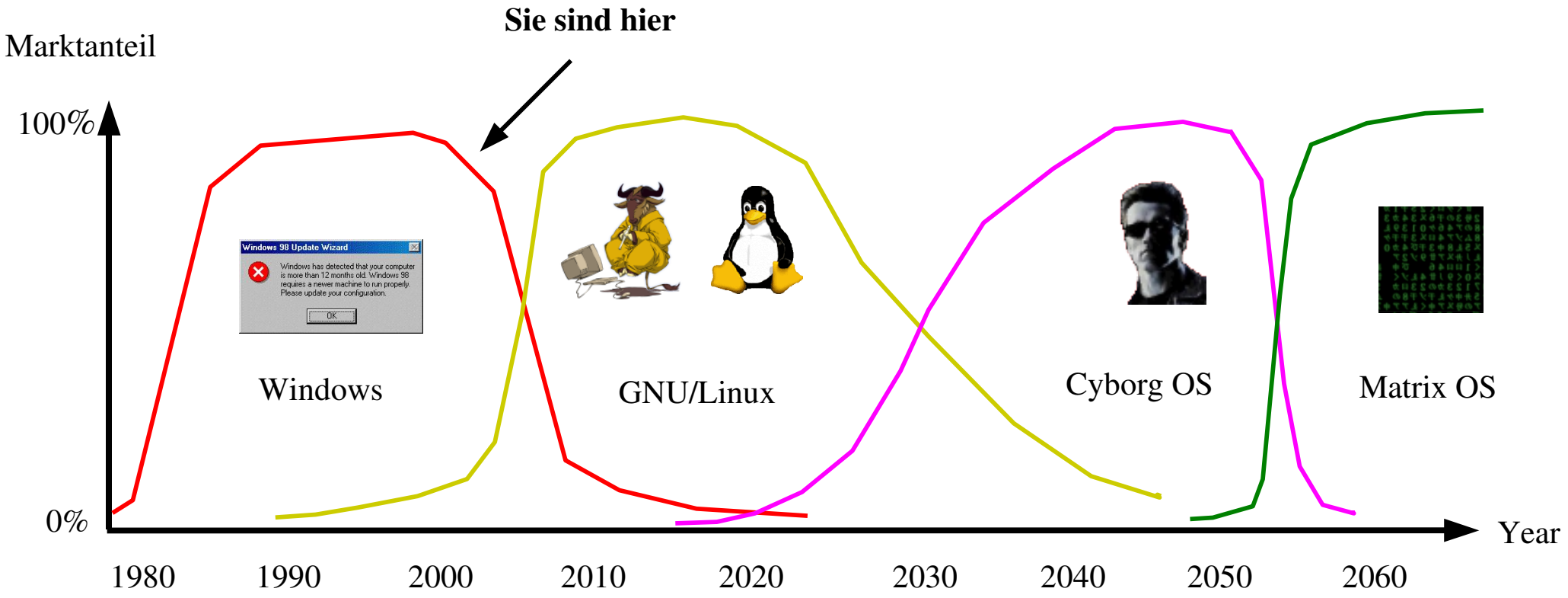
GNU / Linux-Distributionen

- Ermöglichen das Installieren von GNU / Linux im freien Bereich Ihrer Festplatte und das Beibehalten von Windows (“Doppelboot”)
- Verfügen über sehr anwenderfreundliche Installations-Oberflächen, welche fast alle Hardware automatisch erkennen. Sie müssen keine Treiber installieren!
- Sie können die zu installierenden Anwendungen auswählen
- Bieten anwenderfreundliche Konfigurations-Programme
- Für Anfänger empfohlene Distributionen:
  Fedora Core oder Mandriva (siehe die beiden nächsten Seitent)

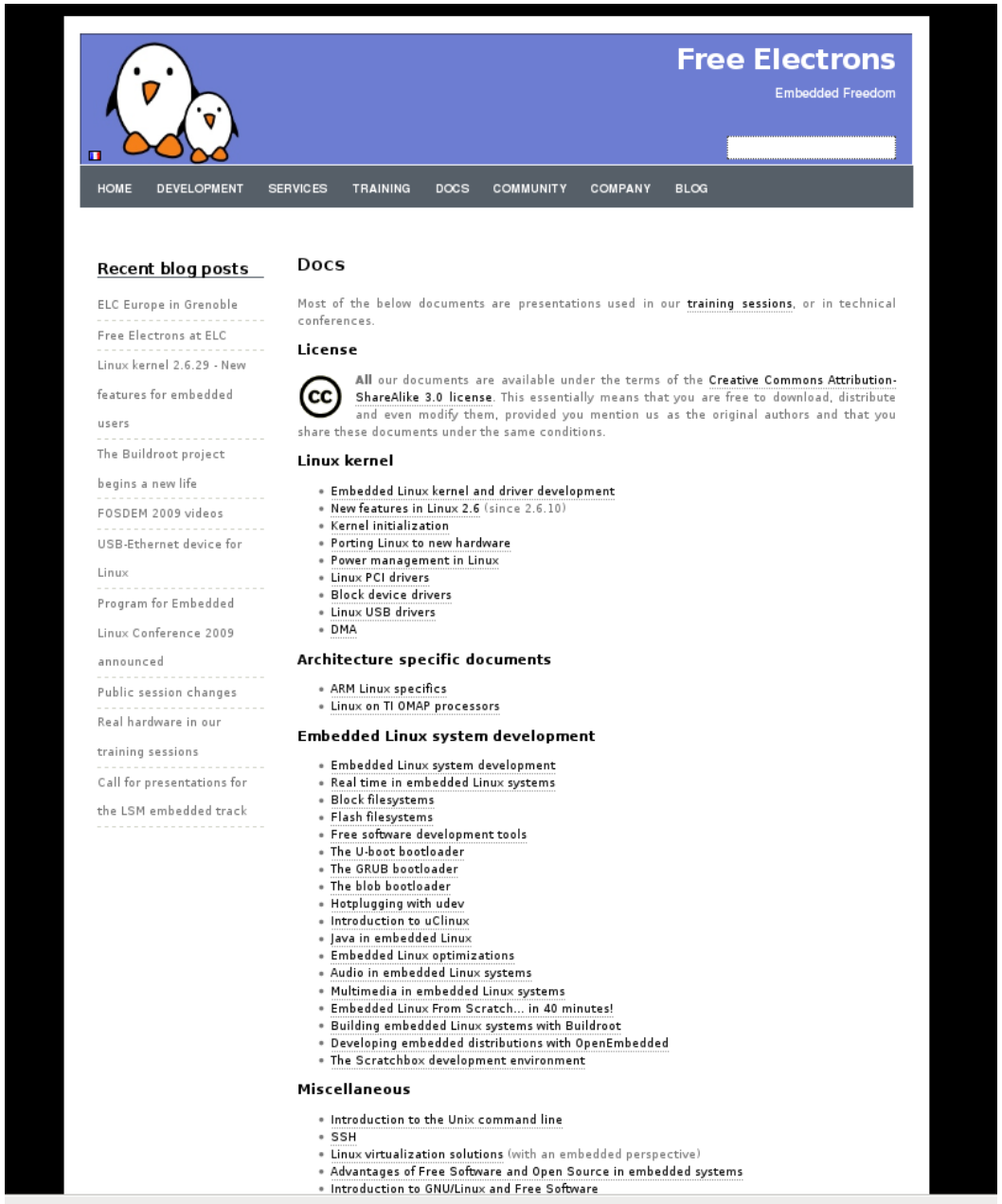
Zusammenfassung

Zeit, um auf den Zug zu springen!

OS roadmap

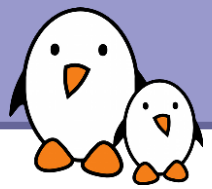


Related documents



All our technical presentations on http://free-electrons.com/docs

- Linux kernel
- Device drivers
- Architecture specifics
- Embedded Linux system development



How to help

You can help us to improve and maintain this document...

- By sending corrections, suggestions, contributions and translations
- By asking your organization to order development, consulting and training services performed by the authors of these documents (see http://free-electrons.com/).
- By sharing this document with your friends, colleagues and with the local Free Software community.
- By adding links on your website to our on-line materials, to increase their visibility in search engine results.

Linux kernel

Linux device drivers
Board support code
Mainstreaming kernel code
Kernel debugging

Embedded Linux Training

All materials released with a free license!

Unix and GNU/Linux basics
Linux kernel and drivers development
Real-time Linux, uClinux
Development and profiling tools
Lightweight tools for embedded systems
Root filesystem creation
Audio and multimedia
System optimization

Free Electrons

Our services

Custom Development

System integration
Embedded Linux demos and prototypes
System optimization
Application and interface development

Consulting and technical support

Help in decision making
System architecture
System design and performance review
Development tool and application support
Investigating issues and fixing tool bugs