LINUX USER GROUP AUSTRIA

LUGA

Linux Professional Development

28. Februar 1996

Linux Professional Development

Bestandteile einer Linux GNU Entwicklungsumgebung:

- `gcc`, **der GNU C,C++ Compiler**
- `gdb`, **der Debugger**
- `make`, **das GNU make Programm**
- ***CVS*, die Versionskontrolle**
- `emacs` **, der integrierte GNU Desktop**
- **LaTeX, ein Textsatzprogramm**
- **weitere Public Domain Tools**

©1996 Linux User Group Austria, Alle Rechte vorbehalten

`gcc` – der GNU Compiler

Das Design und die ersten Implementierungen des Compilers wurden von R. M. Stallman gemacht.

unterstützte Sprachen

- **C (ANSI und KNR)**
- **C++ (ANSI)**
- **FORTRAN**
- **Pascal**
- **ADA**
- **Objective C**

©1996 Linux User Group Austria, Alle Rechte vorbehalten

Vorteile von `gcc`

- Hohe Produktreife.
- Leicht an den Prozessor anpassbar.
- Durch Modularisierung ist es relativ leicht Frontends für verschiedene Sprachen zu entwickeln.
- Genaue Fehlerausgaben und Warnings (konfigurierbar).
- Automatische Dependency Generierung aus den `#include` Direktiven der Source Files.
- Trotz Prozessorunabhängigkeit guter Optimizer.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

- Unterstützung von Cross Compilation.
- Keine Lizenzkosten.

Nachteile von `gcc`

- C++ Version relativ unstabil durch ständige Anpassung an die ANSI Standards und Drafts.
- Großer Speicherbedarf bei Optimierung.
- Noch keine automatische Unterstützung von Template Generierung bei C++.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

gdb – der GNU Debugger

Das Design und die ersten Implementierungen des Debuggers wurden von R. M. Stallman gemacht.

Vorteile von gdb

- Portabel auf verschiedene Betriebssysteme und Prozessoren.

- Watch Points unter Verwendung von Hardware Features.

- Conditional Breakpoints.

- Kann auch für Remote Debugging von Kernel und Firmware eingesetzt werden.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

- Ein Debugger für verschiedene Sprachen (C, C++).

- Aufruf von Funktionen im laufenden Programm aus dem Debugger.

Nachteile von `gdb`

- Großer Speicherbedarf.

- Unterstützt noch nicht alle Sprachen, die vom `gcc` verarbeitet werden können.

- Noch keine ausgereifte Unterstützung von multithreaded Programmen.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

`make` – des GNU make Programm

GNU `make` wurde von **Richard Stallman** und Roland McGrath entwickelt und ist ein Kommandogenerator zu Verwaltung und Wartung von Dateien.

Durch sogenannte Makefiles lassen sich Abhängigkeiten zwischen Dateien sowie der automatische Ablauf zur Erzeugung von Executables aus diversen Quellcode Dateien definieren.

`make` ist ein typisches UNIX Werkzeug und exekutiert Shell Kommandos.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

Beispiel eines `Makefile`

```
# Start of Makefile
RM = rm -f
LATEX = latex
DVIPS = dvips
MAKEINDEX = makeindex

.SUFFIXES : .tex .dvi .ps .ind .idx

%.dvi : %.tex
	$(LATEX) $< ; $(LATEX) $<

%.ind : %.idx
	$(MAKEINDEX) $<

%.ps: %.dvi
	$(DVIPS) -o zen-slides.ps $<

all : zen-slides.ps

clean :
	rm -f zen-slides.ps *.dvi *.log
	rm -f *.aux *.toc *.ind *.ilg *.idx

show: all
	ghostview zen-slides.ps

new : clean all
# End of Makefile
```

©1996 Linux User Group Austria, Alle Rechte vorbehalten

CVS – Concurrent Version System

Was ist CVS ?

- ***Source Control* oder *Revision Control* System**

  CVS ist ein Werkzeug zur Sourcecode- und Revisionsverwaltung und zeichnet alle Veränderungen des Sourcecodes auf.

  CVS wurde entwickelt um Gruppen von Programmierern das gleichzeitige Arbeiten auf demselben Sourcestamm zu ermöglichen und synchron zu bleiben.

  CVS unterstützt außerdem die Zusammenfassung von Sourcedateien in Module. Diese Dateigruppen können dann wie einzelne Dateien behandelt werden.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

Was CVS nicht ist

- **ein *Configuration Management System***

  CVS ist kein Configuration Management System, d.h. es ist kein

  – Fehlerverfolgungssystem

  – Installationswerkzeug

  – automatisiertes Testwerkzeug

  – Build System

  CVS verwaltet nur Sourcecode.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

Vorteile von CVS

- **keine Sperr Mechanismem**

  Jeder Entwickler kann jede(s) Datei/Modul bearbeiten bzw. verändern ohne durch andere blockiert zu werden.

- **gute Dokumentation**

  Im CVS Paket ist neben den man pages und den FAQ auch ein 100 seitiges Handbuch in Postscript enthalten.

- **keine Lizenzgebühren**

  CVS ist unter GNU PUBLIC LICENSE und daher gratis.

- **Portabilität**

  CVS ist auf jeder Unix Plattform verfügbar.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

Nachteile von CVS

- **Verwaltung von binären Dateien**

  Binärdateien können mit CVS zwar prinzipiell verwaltet werden, sie werden nur gleich wie Sourcedateien behandelt, d.h. CVS versucht die (byteweisen) Unterschiede zwischen zwei Versionen herauszufinden ...

©1996 Linux User Group Austria, Alle Rechte vorbehalten

Emacs und Xemacs – der integrierte Desktop

Emacs wurde von R. M. Stallman aus dem TECO Editor heraus entwickelt. Er ist eigentlich ein LISP Interpreter mit Editierfunktionen.

Xemacs ist ein Spin-Off Produkt, das andere Features und Eigenschaften als Emacs hat, auf der LISP Ebene aber weitgehendst kompatibel ist.

Vorteile von Emacs

- **Portabel**

  Unix, VMS, Windows NT, DOS, ...

- **Programmierbarkeit des Editors in LISP**

  Da alle Editierfuntionen von LISP Routinen abgearbeitet werden, kann fast jeder Aspekt des Editors an die Wünsche des Benutzers angepaßt werden.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

- **Integrationstool**

  Der Editor kann Programme starten und steuern. Daher kann er als Frontend für die verschiedensten Aufgaben eingesetzt werden (News, Mail, Debugger, LISP, SQL Datenbanken, Shell, . . .).

- Oberflächenunabhängigkeit

  Emacs und Xemacs laufen unter X11, auf ASCII Terminals und Microsoft Windows.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

Nachteile von Emacs

- Speicherbedarf

  Plattenplatz und Hauptspeicherbedarf ist größer als bei vi.

- Lernbedarf

  vi scheint einfacher erlernbar zu sein als Emacs.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

LaTeX – d a s Textsatzprogramm

LaTeX ist eine Generic Markup Language, welche auf TeX aufbaut.

Geschichte von LaTeX bzw. TeX

Jahr	**Version**
1977	Donald Knuth entwickelt TeX
1980	Leslie Lamport entwicket LaTeX
1980 bis 1990	verschiedene LaTeX Dialekte bzw. Zusätze enstehen, `AMSTeX`, `SliTeX`, `BibTeX`, ...
1990	LaTeX2.09
1994	LaTeX2E
199?	LaTeX3 - die Zukunft

©1996 Linux User Group Austria, Alle Rechte vorbehalten

Vorteile von LaTeX

- **LaTeX ist eine Programmiersprache**

  LaTeX ermöglicht strukturierten Text zu erstellen, ohne sich um die Formatierung kümmern zu müssen. LaTeX wird aus diesem Grund auch von Programmierern gut angenommen.

- **”Corporate Look and Feel” durch Styles**

  Durch die Verwendung von `styles` wird ein einheitliches zentral veränderbares Aussehen und Layout ermöglicht (bzw. erzwungen).

©1996 Linux User Group Austria, Alle Rechte vorbehalten

- **Interoperabilität**
  - einfache Einbindung von Graphiken
  - einfache Konvertierung in andere Textformate

    `dvi, PostScript`
  - einfache Konvertierung nach HTML

    durch das `perl` script `latex2html`

- **Portabilität**

  LaTeX ist auf fast jeder Plattform verfügbar (UNIX, Mac, VMS, etc ..)

- **Lizenzkosten**

  LaTeX ist gratis.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

Nachteile von LaTeX

- **LaTeX ist eine Programmiersprache**

  Die Verwendung von LaTeX erfordert gewisse Programmierkenntnisse. LaTeX ist das falsche Werkzeug für Anwender (SekräterInnen).

- **Komplexität**

  Um die volle Mächtigkeit von LaTeX auszunützen bedarf es Ausdauer, Geduld und logischem Verständnis der Abläufe (floats placement).

- **Fehlermeldungen**

  sind gewöhnungsbedürftig.

©1996 Linux User Group Austria, Alle Rechte vorbehalten

weitere Public Domain Tools

- `perl`

  Practical Extraction and Report Language. Interpretierte Sprache. Besonders geeignet für Scannen von Textfiles und Erstellen von Reports (SYSADM). Vereint Funktionalität und Features von `sed`, `C`, `awk` und `sh`.

- `python`

  objektorientierte Interpretersprache. Vereint die Funktionalitäten von `perl` und `tcl/tk`.

- `tcl/tk`

  Interpretierte Sprache.

  - `tcl` die Basissprache
  - `tk` auf `tcl` aufbauend, ein Toolkit für graphische Applikationen (”Look and Feel” ähnlich wie Motif).

©1996 Linux User Group Austria, Alle Rechte vorbehalten