

C++-Entwicklung mit Linux

Dr. Thomas Wieland

Siemens AG, Corporate Technology

OOP 2002, München

Gliederung

- ◆ Programmentwicklung unter UNIX/Linux
- ◆ Compiler
 - Sprachergänzungen/Abweichungen beim GNU C++-Compiler
 - Alternative Compiler
- ◆ Bibliotheken
- ◆ Werkzeuge
 - Editoren
 - Make
 - Debugger
 - Versionskontrolle
- ◆ IDEs

Programmentwicklung unter Unix und Linux

Besondere Eigenschaften von Unix

◆ Multitasking

- Viele Prozesse laufen parallel
- Müssen sich Systemressourcen (RAM, CPU) teilen
- Beeinflussen sich untereinander i.A. nicht

◆ Multiuser

- Mehrere Benutzer arbeiten gleichzeitig am System
- Prozessdaten müssen benutzerabhängig gespeichert werden
- Ein einzelner User darf das Gesamtsystem nicht in der Stabilität beeinträchtigen

Konsequenzen der Multi-User-Fähigkeit

- ◆ Jede Datei gehört einem Benutzer
 - Zugriffsrechte werden nach Besitzer, dessen Gruppe und andere unterteilt
 - Gilt auch für Verzeichnisse
 - Rechtevergabe auf Dateiebene selbstverständlich
- ◆ Prozesse sind an Benutzer gebunden
 - Temporäre Dateien (in /tmp) müssen eindeutig für einen Prozess sein (z.B. mit Prozessnummer im Dateinamen)
- ◆ Konfigurationsinformationen stets benutzerspezifisch
 - Werden im „Home“-Verzeichnis des Benutzers abgelegt
- ◆ Nur ein „Superuser“ darf das System verändern

Unix und Linux

- ◆ Linux wurde unabhängig von allen anderen Unix-Derivaten entwickelt
- ◆ Verbindet das Beste aus den beiden Linien „System V“ und „BSD“
 - Terminal handling und IPC aus System V
 - Sockets aus BSD
- ◆ Linux ist (weitgehend) POSIX-conform
- ◆ Werkzeuge werden oft erst auf Linux entwickelt und dann auf andere Unix-Varianten portiert

C++-Compiler unter Linux

Die GNU Compiler Collection

- ◆ GCC ist der Referenzcompiler für Linux
 - Kernel wird damit übersetzt
- ◆ Lange Historie
 - Aufspaltung in GCC und EGCS
 - Zusammenführung zu Version 2.95
 - Version 3.0.x ist ANSI-C++-konform
- ◆ Verschiedene Sprachen werden unterstützt
 - C, C++, Objective-C, Java, Fortran 77

Einschränkungen zu ANSI/ISO bei GCC/C++ bis zur Version 2.95

- ◆ Funktionen der Standardbibliothek nicht im Namespace `std`
- ◆ Bei `<new>` keine Exception `bad_alloc`
- ◆ Keine Klasse `stringstream`
- ◆ `ios` Basisklasse aller I/O-Klassen (statt `ios_base`)
- ◆ Kein Import von Methoden aus Basisklassen mittels `using`
- ◆ Weitere kleine Einschränkungen

Was ist bei GCC/C++ 3.0 anders?

- ◆ Einige neue Optimierungen
 - Dafür dauert das Übersetzen z.T. deutlich länger
 - Feedback-Optimierung zur Sprungvorhersage
 - Code-Adaption für Athlon
 - Leichte Geschwindigkeitssteigerungen für die Anwendungen (um 3 %, nach SPEC-CPU2000)
- ◆ Fast vollständige Unterstützung des ANSI/ISO-Standards in der Standardbibliothek
- ◆ Einige GCC-Spracherweiterungen wurden entfernt
 - z.B. Überladen des `?`-Operators, Zuweisungen an `this`

Aber: Mit GCC 3.0 übersetzte Programme lassen sich nicht mit Bibliotheken linkern, die mit einem Vorgänger erzeugt wurden!

Andere Compiler

- ◆ Portland Group C++-Compiler
(www.pgroup.com/prodworkpgcc.htm)
- ◆ Kuck & Associates (www.kai.com/C_plus_plus)
- ◆ Borland (www.borland.com/kylix)
 - Borland C++-Compiler als Teil von Kylix/C++
 - Open Edition frei erhältlich
- ◆ Intel
 - Siehe nächste Folie

Intel C++-Compiler für Linux

- ◆ Für Unternehmen kostenpflichtig, für private Nutzung frei (Aber: 32 MB zum Download)
- ◆ Optimiert für Pentium-Architektur
 - Bei SPEC CPU2000 bis zu drei mal schnellerer Code als mit GCC 3.0 (lt. iX-Messung)
 - Auch auf AMD bessere Optimierung
- ◆ Enthält veränderte Header-Files der glibc, daher abhängig von Linux-Version (Distribution)
- ◆ Teilweise inkompatibel mit GCC
 - Andere Compiler-Schalter
 - Anderes Format für Objekt-Datei
 - Qt, GTK, MySQL u.a. lassen sich nicht mit ICC übersetzen!

C++-Bibliotheken

Qt-Bibliothek

- ◆ Vorbildlich aufgebaute Klassenbibliothek zur GUI-Entwicklung
 - Basis der KDE-Oberfläche und aller KDE-Anwendungen!
 - Enthält auch Container-Klassen und System-Klassen (z.B. für Datum oder Verzeichnisse)
 - Sehr gut dokumentiert, leichter Einstieg
- ◆ Greift direkt auf Xlib zu, daher sehr schnell
- ◆ Auch für Windows erhältlich, daher auch für Cross-Plattform-Entwicklung nützlich
- ◆ Für Open Source-Anwendungen unter GPL-Lizenz, für kommerzielle Applikationen kostenpflichtig
- ◆ Download unter <http://www.trolltech.com/qt>

ACE: Adaptive Computing Environment

- ◆ Klassenbibliothek für systemnahe Programmierung
- ◆ Als Open Source für Linux, Win 32 sowie viele Embedded-Betriebssysteme verfügbar
 - Daher sehr portabel
- ◆ Effiziente Kommunikations- und Netzprogrammierung
 - Multithreading und Synchronisation
 - Service Configurator zur dynamischen Adaption von Anwendungen
 - Interprozess-Kommunikation
 - Basis für CORBA-Implementierung TAO (echtzeitfähiger, sehr performanter ORB, ebenfalls Open Source)
- ◆ Erhältlich unter <http://www.cs.wustl.edu/~schmidt/ACE.html>

XML-Bibliotheken

- ◆ Xerces
 - Parst, erzeugt und validiert XML-Dokumente
 - Unterstützt alle aktuellen Standards: DOM 2.0, SAX 2.0, W3C Schema recommendation etc.)
 - Performant und portabel
- ◆ Xalan
 - XSLT-Prozessor für Transformation von XML-Dokumenten
 - Unterstützt aktuelle Standards von XSLT und Xpath
 - Voll kompatibel zu Xerces
- ◆ Beide unter Apache Lizenz
- ◆ Download von <http://xml.apache.org/>

Werkzeuge

Editoren

- ◆ **vi und -Klone (z.B. <http://www.vim.org>)**
 - Klassisch, gewöhnungsbedürftig, aber sehr effizient
- ◆ **(X)Emacs (<http://www.xemacs.org> oder <http://www.gnu.org/software/emacs>)**
 - Der Alleskönner, fast eine komplette IDE, ebenfalls gewöhnungsbedürftig
 - Durch Lisp-Skripte beliebig erweiterbar
- ◆ **NEdit (<http://www.nedit.org>)**
 - Leicht zu bedienen (Windows-ähnlich), Syntax-Highlighting, über Macros automatisierbar
- ◆ **KWrite (<http://apps.kde.com/na/2/info/id/521>)**
 - Syntax-Highlighting, Mehrfachauswahl, Vertikalauswahl
 - Gut integriert in KDE-Oberfläche
- ◆ **XCoral (<http://xcoral.free.fr>)**
 - Syntax-Highlighting, über C-Skripten erweiterbar (enthält C-Interpreter)
 - Enthält Code-Schablonen für Klassengerüste und einen Klassenbrowser

make

◆ Steuert die Übersetzung

- In Regeln wird formuliert, welcher Befehl nötig ist, um von einer Ausgangsdatei zu einer Zielfile zu kommen, z.B.

```
birthday.o: birthday.cc birthlist.h  
g++ -c birthday.cc
```

- Durch Berücksichtigung der Abhängigkeiten und des bereits vorhandenen wird nur das tatsächliche Nötige gebaut
- Abhängigkeiten auch mit Wildcards beschreibbar
- ◆ Regeln sind in einem „Makefile“ pro Projekt abgelegt
- ◆ Durch Makros automatisierbar
- ◆ Flexibel, auch für Installation und Test anwendbar
- ◆ Unterstützt automatische Bestimmung der Abhängigkeiten

Debugger

◆ Für GCC-Programme: GNU Debugger gdb

- Hat nur eine Kommandozeilen-Schnittstelle
- Aber: verschiedene grafische Frontends verfügbar

◆ DDD (Data Display Debugger)

- Grafischer Debugger auf gdb-Basis
- Kann zusätzlich komplexe Datenstrukturen als Graphen visualisieren

◆ Weitere Debugger

- KDbg: KDE-Frontend zum gdb
- XEmacs: Verbindet sich mit gdb, bietet Schaltflächen und Code-Verfolgung
- xxgdb: X-Windows-Frontend

Versionskontrolle

- ◆ RCS (<http://www.gnu.org/software/rcs>)
 - Dateibasierte Revisionsverwaltung, keine Projektorganisation
 - Speichert jeweils nur Delta zur Vorgängerrevision
 - Zugriff nur über Dateisystem
 - Nur ausschließliche Locks möglich
- ◆ CVS (<http://www.cvshome.org>)
 - Verwaltet ganze Projekte in mehreren Stufen
 - Kann auch gleichzeitige Locks und Merging
 - Eigene Server-Software, gut bei verteilten Projekten
- ◆ Weitere
 - Clearcase: Leistungsstark, aber teuer (<http://www.rational.com>)
 - PVCS: Linux-Portierung (<http://www.pvcs.synergex.com>)

Integrierte Entwicklungsumgebungen

KDevelop

- ◆ Leistungsstarke Entwicklungsumgebung unter GPL
- ◆ Verwaltung komplexer Projekte
 - Unterstützt auch autoconf/automake
 - Enthält Application Wizard zur Generierung von Anwendungsgerüsten, Klassengenerator für einzelne Klassen
- ◆ Integrierter Debugger
- ◆ Graphischer Klassenbrowser
- ◆ Verschiedene Visualisierungsmodi
 - Lässt KDevelop aussehen wie MS VC, Borland C++Builder etc.
- ◆ Umfangreiche Dokumentation
 - Benutzerhandbuch, Programmierhandbuch, Beispielhandbuch, KDE Programmierguides, C/C++ Referenz, Qt- und KDE- Dokumentation

Kylix

- ◆ Das „Delphi für Linux“, jetzt auch für C/C++
 - Ähnliche Oberfläche wie Delphi/C++Builder unter Windows
 - Gleiche Klassenbibliotheken, daher für Cross-Plattform-Entwicklung geeignet
 - CLX-Library baut auf Qt auf
- ◆ Als „Open Edition“ kostenlos
 - Ohne Datenbank- und Server-Komponenten
- ◆ Ab Profi-Version gute Datenbankzugriffsschicht
 - Unterstützt Oracle, Interbase, MySQL, DB/2 u.v.a. nativ
 - Bringt eigene XML in-memory-Datenbank mit
- ◆ Einfache CORBA- und Web-Services-Entwicklung

SNiFF+

- ◆ Umfangreiche IDE, insbesondere zur Source-Code-Analyse und für große Projekte
 - Enthält Symbolbrowser, Klassenbrowser, Include-Browser, Hierarchie-Browser, Cross-Referencer
 - Lässt sich durch Python-Skripts konfigurieren und erweitern
- ◆ Bringt keinen eigene Compiler, Debugger oder Revisionskontrolle mit, sondern bindet vorhandenes ein
 - GCC und GDB
 - RCS, CVS, ClearCase
- ◆ Komplexes Konzept zur Code-Generierung im Team
- ◆ Linux-Version leider nicht mehr kostenfrei!

Weitere IDEs

- ◆ **CodeWarrior** (Metrowerks), kommerziell
 - Bekannte IDE, baut auf GCC 2.95 und GDB/DDD auf
 - Enthält Projektverwaltung, Editor, Debugger, Versionskontrolle, graphischer Dateivergleich, unterstützt paralleles Kompilieren
- ◆ **SourceNavigator** (Red Hat), GPL
 - Dient zur Analyse von Source-Code mit Symbolbrowser, Include-Browser, Cross-Referencer, Hierarchiebrowser, Editor, Debugger-Interface
- ◆ **Code Crusader** (New Planet SW), kommerziell
 - Symbol- und Hierarchiebrowser, Autovervollständigung, Projektverwaltung, Make-Support, Appl.-Vorlagen etc.
- ◆ **XEmacs**
 - unterstützt u.a. make, Versionskontrolle und Debugger

Portierung nach Linux

Portierung von Unix

- ◆ Im Prinzip einfach - hängt davon ab
 - welche Bibliotheken, Middleware, Datenbanken etc. verwendet wurden
 - ob für den CDE-Desktop entwickelt wurde
 - ob plattformspezifische Systemaufrufe vorkommen
 - ob hardware-abhängig programmiert wurde (Wortlänge, Little/Big Endian)
- ◆ Vorgehensweise
 1. Auf Ausgangsplattform (z.B. Solaris) mit GCC übersetzen
 2. Auf Ausgangshardware (z.B. SPARC) unter Linux übersetzen
 3. Auf Zielhardware unter Linux übersetzen

Portierung von Windows

- ◆ Mittels WINE (<http://www.winehq.org>, Open Source)
 - Bietet Windows-API, um Anwendungen unter Linux zu übersetzen oder lediglich laufen zu lassen
 - Wird von einigen SW-Herstellern (z.B. Corel) genutzt
- ◆ Mittels MainWin (<http://www.mainsoft.com>, kommerz.)
 - Vollständige Nachbildung aller wichtigen Windows-Schnittstellen (incl. OLE/COM, Registry, .NET etc.)
 - Wurde schon für Internet Explorer genutzt
- ◆ Durch Verwendung portabler Bibliotheken (Qt, CLX, GSL, ...)
- ◆ Durch Neuschreiben :-)

Zusammenfassung

Fazit

- ◆ Unter Linux sind alle Werkzeuge für eine effiziente C++ Entwicklung verfügbar
- ◆ Die Auswahl erfolgt nach den Projektzielen und persönlichen Vorlieben
- ◆ Bibliotheken unterstützen viele Standardaufgaben
- ◆ IDEs nehmen den „Schrecken“ vor den kleinen Unix-Tools
- ◆ Das Debuggen ist oft einfacher, da es bis auf Systemebene möglich ist (Open Source!)
- ◆ Die Portierung von Unix ist mit überschaubarem Aufwand möglich, von Windows ist oft ein Neuschreiben nötig

Warum Linux?

- ◆ Flexibilität
- ◆ Stabilität
- ◆ Sicherheit
- ◆ Skalierbarkeit
- ◆ Moderate Hardware-Anforderungen
- ◆ Akzeptanz in der Industrie
- ◆ Kosten
- ◆ Viel Know-How und Support verfügbar
- ◆ Verfügbarkeit von Kleingeräten bis zum Mainframe

Literatur

- ◆ T. Wieland: „C++-Entwicklung mit Linux“, dpunkt.verlag, 2001. <http://www.cpp-entwicklung.de> (gesamter Text online)
- ◆ A. Zeller und J. Krinke: „Programmierwerkzeuge“, dpunkt.verlag, 2001. <http://www.programmierwerkzeuge.de>
- ◆ M.K. Dalheimer: „Linux – Wegweiser zur Programmierung & Entwicklung“, O'Reilly Verlag, 1997.
- ◆ N. Matthew und R. Stones: „Beginning Linux Programming“ und „Professional Linux Programming“, Wrox Press, 2000.

Links

- ◆ GNU Compiler
 - <http://gnu.gcc.org>
- ◆ Intel C++-Compiler
 - <http://developer.intel.com/software/products/compiler/s/c50/linux/>
- ◆ GNU Make
 - <http://www.gnu.org/software/make/>
- ◆ GDB
 - <http://www.gnu.org/software/gdb/>
- ◆ DDD
 - <http://www.gnu.org/software/ddd/>

Links (2)

◆ KDevelop

- <http://www.kdevelop.org/>

◆ Kylix

- <http://www.borland.de/kylix>

◆ SNiFF+

- <http://www.windriver.com/products/html/sniff.html>

◆ Code Warrior

- <http://www.metrowerks.com/products/linux/>

◆ Source Navigator

- <http://sources.redhat.com/sourcenav/>

◆ Code Crusader

- <http://www.newplanetsoftware.com/jcc/>

Danke für Ihre Aufmerksamkeit!

Fragen oder Kommentare?

Download der Folien unter <http://www.drwieland.de>