

Proseminar: Linux für Umsteiger

Universität Heidelberg

Sourcen

Wie baue ich Programme von ihren
Sourcen, wie installiere ich diese,
Probleme und deren Lösungen

13.Mai, Sommersemester 2008
Oleksandr Maykivets

Motivation

- Oft liegen keine RPM- oder DEB-Pakete vor, nur Source-Code
- Wenn man ein Programm mitentwickeln möchte, muss man selbst den Source-Code verändern und kompilieren

Aufbau von C-Programmen

- Die meisten Programme für Linux sind in C oder C++ geschrieben
- Source-Code-Dateien mit der Endung `.c` müssen einzeln kompiliert werden
- Dann entstehen Objekt-Dateien mit der Endung `.o`
- Danach werden diese Objekt-Dateien zusammen gelinkt

Beispiel: HelloWorld in Programmiersprache C

- start.c erstellen (eine Datei muss main()-Methode enthalten):

```
int main() {  
    printHallo();  
} // Funktion printHallo wird aufgerufen
```

- hallo.c erstellen:

```
#include <stdio.h>  
  
void printHallo() {  
    printf("Hallo!\nEs hat geklappt\n");  
}
```

HelloWorld kompilieren

- `gcc -c start.c`

(der Schalter `-c` bewirkt Erstellung von Objekt-Dateien)

- `gcc -c hallo.c`

- Zusammenlinken und ein Programm namens `gruss` erstellen:

- `gcc -o gruss start.o hallo.o`

- Programm `gruss` ausführen:

`./gruss`

Das Programm make & Makefile-Datei (1)

- Im Beispiel (oben) sind es nur zwei Dateien
- Meistens besteht ein Programm aus hunderten oder tausenden Source-Dateien, die einzeln kompiliert und zusammen gelinkt werden müssen
- Diese Aufgaben erledigt das Programm make

Das Programm make & Makefile-Datei (2)

- Vorteil von make:

make vergleicht Änderungsdatum von Quellcode- und Objektdateien:

wenn Objektdatei jünger ist, wird Quellcodedatei nicht nochmal kompiliert.

Das spart Zeit, da bei kleinen Änderungen keine komplette Neukompilierung notwendig ist.

Das Programm make & Makefile-Datei (3)

- make sucht nach Makefile
- Aufbau von Makefile für HelloWorld:

```
comp = gcc
```

```
gruss: start.o hallo.o
```

```
    $(comp) -o gruss start.o hallo.o
```

```
start.o: start.c
```

```
    $(comp) -c start.c
```

```
hallo.o: hallo.c
```

```
    $(comp) -c hallo.c
```

Softwareinstallation ohne Pakete. Standardvorgehen

- Tarball entpacken
- README- und INSTALL-Datei lesen
- `configure` ausführen: erstellt Makefile passend zum System und meldet, welche Bibliotheken noch fehlen

(`./configure --prefix=/usr/lokal/xyz` oder `/home/xyz`;
bei default `/usr/lokal/bin`)

- `make`
- `make install`

Beispiel 1: ntfs-3g installieren

- download
- `tar xvf ntfs-3g-1.2412.tar`
- `cd ntfs-3g-1.2412`
- README und INSTALL lesen
- `./configure`
- `make`
- `make install`
- `umount „Pfad zu Windows“, falls mounted`
- `ntfs-3g /dev/hda1 /mnt/Windows/`

Beispiel 2: VLC (Dependency Hell)

- .System Suse10.2
- configure meldet fehlende Bibliotheken: mad, ffmpeg, mpeg2dec, wxWidgets
- Bibliotheken nachinstalliert. wxWidgets-Installation klappt nicht.
- ffmpeg wird von configure nicht erkannt
- ffmpeg und wxWidgets kann man beim konfigurieren ausschalten (--disable-...)
- Fehler beim kompilieren
- ältere VLC-Version kompiliert, aber ... es gibt dann nur Ton aber kein Bild beim Abspielen von Videodateien

Diskussion

- Was macht man mit Code, der nicht in C oder C++ geschrieben ist?
- Java: Apache Ant
- SCons, an Open Source software construction tool
- GNU Build System (autoconf, automake)

Literatur

- galileocomputing.de/openbook/linux
- fibel.org/linux/lfo-0.6.0-1/lfo.html
- Wikipedia
- mitlinux.de/webservices/art.htm
- <http://www.scons.org/>
- <http://de.wikipedia.org/wiki/Autoconf>
- http://de.wikipedia.org/wiki/C_%28Programmiersprache%29
- de.wikipedia.org/wiki/GNU_Compiler_Collection
- <http://de.wikipedia.org/wiki/Makefile>

Proseminar: Linux für Umsteiger

Universität Heidelberg

Sourcen

Wie baue ich Programme von ihren
Sourcen, wie installiere ich diese,
Probleme und deren Lösungen

13.Mai, Sommersemester 2008
Oleksandr Maykivets

Motivation

- Oft liegen keine RPM- oder DEB-Pakete vor, nur Source-Code
- Wenn man ein Programm mitentwickeln möchte, muss man selbst den Source-Code verändern und kompilieren

Alle Open-Source-Programme sind als Quellcode verfügbar. Für die meistbenutzten Programme werden RPM- und DEB-Pakete erstellt. Jedoch gibt es viele nützliche Programme, für die man keine Pakete im Internet findet.

Aufbau von C-Programmen

- Die meisten Programme für Linux sind in C oder C++ geschrieben
- Source-Code-Dateien mit der Endung .c müssen einzeln kompiliert werden
- Dann entstehen Objekt-Dateien mit der Endung .o
- Danach werden diese Objekt-Dateien zusammen gelinkt

Ein Compiler ist ein Computerprogramm, das ein in einer Quellsprache (z.B. C, C++, Java) geschriebenes Programm – genannt Quellprogramm – in ein semantisch äquivalentes Programm einer Zielsprache (üblicherweise Assembler, Bytecode oder Maschinensprache) umwandelt.

Beispiel: HelloWorld in Programmiersprache C

- start.c erstellen (eine Datei muss main()-Methode enthalten):

```
int main() {  
    printHallo();  
} // Funktion printHallo wird aufgerufen
```

- hallo.c erstellen:

```
#include <stdio.h>  
void printHallo() {  
    printf("Hallo!\nEs hat geklappt\n");  
}
```

Ein Hallo-Welt-Programm ist ein kleines Computerprogramm und soll auf möglichst einfache Weise zeigen, welche Anweisungen oder Bestandteile für ein vollständiges Programm in einer Programmiersprache benötigt werden und somit einen ersten Einblick in die Syntax geben.

`#include <stdio.h>` ermöglicht Verwendung der Funktionen aus „standard-input/output“-Bibliothek, in diesem Fall die Funktion `printf()`.

`main()` ist die Einstiegsfunktion eines C-Programms. `main()` wird automatisch als erste Funktion aufgerufen. Anfang und Ende der Funktion `main()` werden durch die beiden geschweiften Klammern markiert.

HelloWorld kompilieren

- `gcc -c start.c`
(der Schalter `-c` bewirkt Erstellung von Objekt-Dateien)
- `gcc -c hallo.c`

- Zusammenlinken und ein Programm namens `gruss` erstellen:
- `gcc -o gruss start.o hallo.o`

- Programm `gruss` ausführen:
`./gruss`

GNU Compiler Collection (GCC) ist der Name der Compiler-Suite des GNU-Projekts. GCC steht ursprünglich für GNU C Compiler (analog zu dem UNIX-Kommando `cc` für C Compiler). Da GCC heute aber außer C noch einige andere Programmiersprachen übersetzen kann, hat GCC inzwischen die Bedeutung GNU Compiler Collection erhalten (engl. für GNU-Compilersammlung).

Das Kommando `gcc` (in Kleinbuchstaben) steht weiterhin für den C-Compiler.

Das Programm make & Makefile-Datei (1)

- Im Beispiel (oben) sind es nur zwei Dateien
- Meistens besteht ein Programm aus hunderten oder tausenden Source-Dateien, die einzeln kompiliert und zusammen gelinkt werden müssen
- Diese Aufgaben erledigt das Programm make

Das Programm make & Makefile-Datei (2)

- Vorteil von make:

make vergleicht Änderungsdatum von Quellcode- und Objektdateien:

wenn Objektdatei jünger ist, wird Quellcodedatei nicht nochmal kompiliert.

Das spart Zeit, da bei kleinen Änderungen keine komplette Neukompilierung notwendig ist.

Das Programm make & Makefile-Datei (3)

- make sucht nach Makefile
- Aufbau von Makefile für HelloWorld:

```
comp = gcc
gruss: start.o hallo.o
    $(comp) -o gruss start.o hallo.o
start.o: start.c
    $(comp) -c start.c
hallo.o: hallo.c
    $(comp) -c hallo.c
```

Das Prinzip:

Das Erstellen einer Datei wird im Makefile als ein Ziel (Target) bezeichnet. Die Randbedingungen dazu werden in einem Eintrag beschrieben.

Beispiel:

```
A: B C
    D
```

Diese Zeile bedeutet: Ziel A hängt von den Zielen B und C ab. („Hängt ab von“ bedeutet meistens „wird erstellt aus“.) Wenn A erstellt werden soll, werden B und C betrachtet. Ist eins von beiden jünger als A, wird D ausgeführt, um A neu zu erstellen. Ist A jünger als B und C, wird A als aktuell betrachtet.

Achtung! Zeilen, die Programme aufrufen, müssen mit einem Tabulatorzeichen beginnen.

Softwareinstallation ohne Pakete. Standardvorgehen

- Tarball entpacken
 - README- und INSTALL-Datei lesen
 - configure ausführen: erstellt Makefile passend zum System und meldet, welche Bibliotheken noch fehlen
- (./configure --prefix=/usr/lokal/xyz oder /home/xyz;
bei default /usr/lokal/bin)
- make
 - make install

Die meisten größeren Quellcodepakete enthalten ein „configure“-Skript. Dieses Skript muss vom Benutzer weder bearbeitet oder mit Schaltern beim Starten konfiguriert werden. Seine Aufgabe ist es, die Systemkonfiguration auf Kompiler, Bibliotheken und andere wichtige Elemente für die Kompilierung zu testen. Auf der Basis der ermittelten Informationen schreibt das Skript eine individuelle Installationskonfigurationsdatei (Makefile) passend für das System. Sollte configure Fehler wie fehlende Bibliotheken finden, so meldet das Skript das mit einer Fehlermeldung.

Beispiel 1: ntfs-3g installieren

- download
- tar xvf ntfs-3g-1.2412.tar
- cd ntfs-3g-1.2412
- README und INSTALL lesen
- ./configure
- make
- make install
- umount „Pfad zu Windows“, falls mounted
- ntfs-3g /dev/hda1 /mnt/Windows/

Proseminar Linux für Umsteiger

10

Tar ist der Name eines im Unix-Umfeld sehr geläufigen Archivierungsprogramms. Außerdem wird so auch das Dateiformat bezeichnet, das von diesem Programm verwendet wird.

Für tar.gz-Archive kann sollte man vorher gunzip < dateiname.tar.gz ausführen.

In README- und INSTALL-Dateien findet man ausführliche Informationen über Installation mit allen möglichen Optionen.

Das Kommando „make install“ installiert das kompilierte Programm auf dem Betriebssystem.

Beispiel 2: VLC (Dependency Hell)

- .System Suse10.2
- configure meldet fehlende Bibliotheken: mad, ffmpeg, mpeg2dec, wxWidgets
- Bibliotheken nachinstalliert. wxWidgets-Installation klappt nicht.
- ffmpeg wird von configure nicht erkannt
- ffmpeg und wxWidgets kann man beim konfigurieren ausschalten (--disable-...)
- Fehler beim kompilieren
- ältere VLC-Version kompiliert, aber ... es gibt dann nur Ton aber kein Bild beim Abspielen von Videodateien

Diskussion

- Was macht man mit Code, der nicht in C oder C++ geschrieben ist?
- Java: Apache Ant
- SCons, an Open Source software construction tool
- GNU Build System (autoconf, automake)

Literatur

- galileocomputing.de/openbook/linux
- fibel.org/linux/lfo-0.6.0-1/lfo.html
- Wikipedia
- mitlinux.de/webservices/art.htm
- <http://www.scons.org/>
- <http://de.wikipedia.org/wiki/Autoconf>
- http://de.wikipedia.org/wiki/C_%28Programmiersprache%29
- de.wikipedia.org/wiki/GNU_Compiler_Collection
- <http://de.wikipedia.org/wiki/Makefile>