Betriebssysteme

Paketverwaltung

Autor: Ronny Bodach

Stand: 24.03.2022

Agenda

1. Notwendigkeit
2. Bestandteile
3. Umsetzungen
4. App-Stores
5. Debian Package Manager
6. Redhat Package Manager
7. Advance Packaging Tool
8. Aufbau Paket-Container
9. Build-System

(C) 02.05.2023 Hochschule Mittweida

Notwendigkeit Paketverwaltung

- Zentrale Verwaltung von Software
- Bereitstellung in Programmpaketen
- Vorkonfiguriert für spez. Betriebssysteme / OS-Versionen / Distributionen
- Zwei Arten der Umsetzung:
  - Programme, die Pakete direkt installieren / löschen, ohne Abhängigkeiten & Konfliktprüfung
  - Programme, die aus anderen Quellen Daten nachladen, Abhängigkeiten & Konflikte prüfen

(C) 02.05.2023 Hochschule Mittweida

Bestandteile einer Paketverwaltung

- Installation
- Konfiguration
- Aktualisierung
- Deinstallation

(C) 02.05.2023 Hochschule Mittweida

Umsetzungen

- Windows:
  - .msi-Pakete (MicroSoft Installer)
  - Laufzeitumgebung für Installationsroutinen unter Microsoft-Windows-Betriebssystemen
  - Windows-Systemdienst + Paketdateien + Patchdateien
  - Externe Abhängigkeiten oft nicht beachtet
- macOS:
  - .dmg Pakete (Apple Disk Image)
- iOS:
  - .ipa Pakete (Apple iPhone Application)
- Android:
  - .apk Pakete (Android Package Kit)

- Was repräsentieren moderne App-Stores?

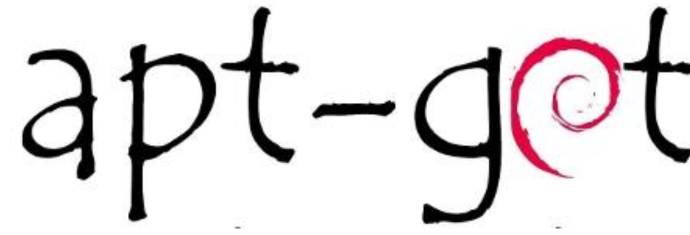
(C) 02.05.2023 Hochschule Mittweida

Diskussion App-Stores

(C) 02.05.2023 Hochschule Mittweida

Paketverwaltung unter Unix-Betriebssystemen

Zitat Ian Murdock:
„the single biggest advancement Linux has brought to the industry“



(C) 02.05.2023 Hochschule Mittweida

Debian Package Manager (dpkg)

- dpkg –i <Paket>.deb
  - Paket installieren
- dpkg –r <Paketname>
  - Paket deinstallieren („remove“)
- dpkg –l
  - Pakete auflisten („list“)
- dpkg –L <Paketname>
  - Paketinhalt auflisten

(C) 02.05.2023 Hochschule Mittweida

RedHat Package Manager (rpm)

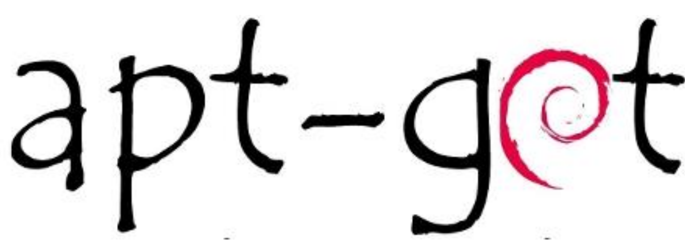
- rpm –i <Paket>.rpm
  - Paket installieren (install)
- rpm –e <Paketname>
  - Paket deinstallieren (erase)
- rpm –qa
  - Pakete auflisten (query all)
- rpm –l <Paketname>
  - Paketinhalt auflisten (list)

(C) 02.05.2023 Hochschule Mittweida

Advanced Packaging Tool (APT)

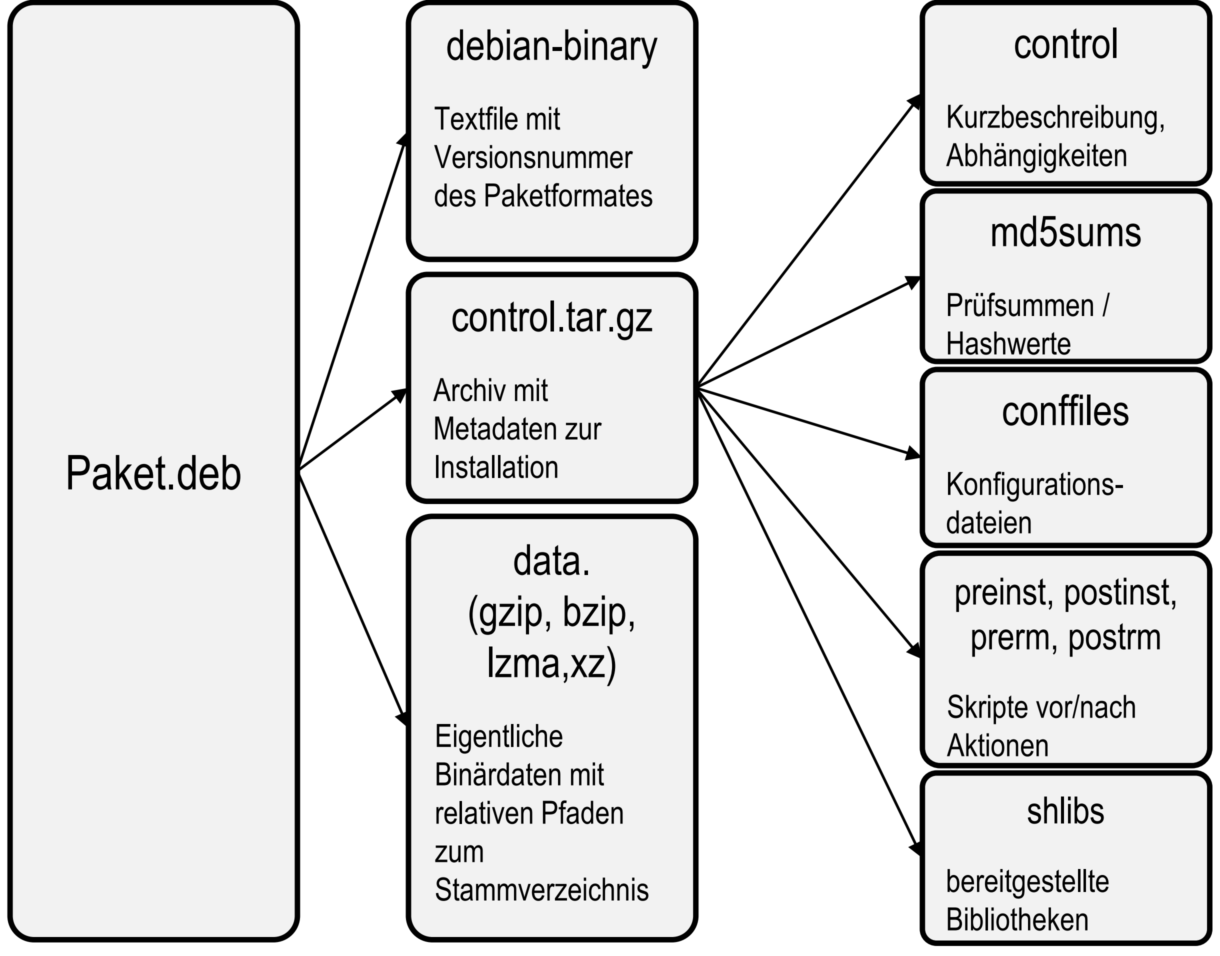
- Erweiterung für dpkg, löst Abhängigkeiten auf
- apt-get update
  - Paketliste aktualisieren
- apt-get upgrade
  - Update aller apt-get veralteten / installierten Pakete
- apt-get dist-upgrade
  - Upgrade der gesamten Distribution, inkl. zurückgehaltener Pakete
- apt-get install <Paketname>
  - Paket installieren
- apt-get remove <Paketname>
  - Paket deinstallieren
- apt-get purge <Paketname>
  - Paket und Konfiguration deinstallieren
- apt-cache search <Paketname>
  - Paket suchen

- Nachfolger apt
- apt = apt-get + apt-cache + Zusatzfunktionen
- gleicher Befehlsaufbau
- verbesserte Visualisierung:
  - Farben
  - Fortschrittsbalken



(C) 02.05.2023 Hochschule Mittweida

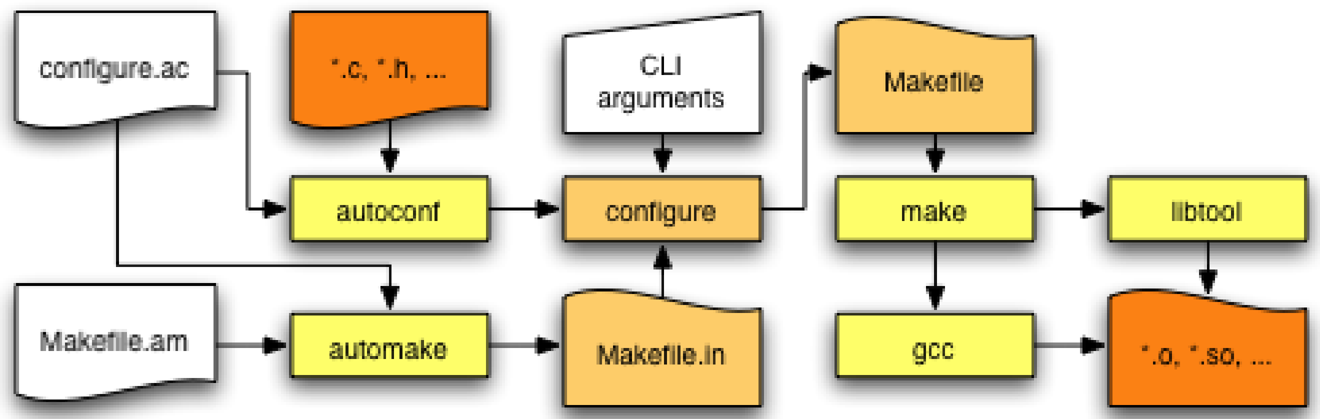
Aufbau Paket-Container .deb Paket



(C) 02.05.2023 Hochschule Mittweida

Build-System

- „Baue die Software spezifisch für mein Ziel-System“
- Quelltextdateien, die durch den Compiler (maschinenspezifisch) verarbeitet werden
- Objektdateien, die vom Linker verbunden / eingebunden werden
- zusätzliche Skripte für Pre- / Post-Prozesse
- make Befehl nutzt Makefile als Bauanleitung
- Makefiles automatisiert erstellen mit GNU Build System
  - autoscan, alocal, autoheader, autoconf, automake



(C) 02.05.2023 Hochschule Mittweida

Build-System

- Nutzer verwendet zur Installation von Software Quellcode anstatt bereits compilierte Binärdaten
  - configure (Skript)
  - Makefile.in
  - config.h.in
- Bei der Ausführung von configure entstehen
  - config.status
  - Makefile
  - config.h
- Standart-Ablauf für Installation
  - ./configure
  - make
  - make install

(C) 02.05.2023 Hochschule Mittweida

Makefile

1. `CC = cc`
`LD = ld` → 1. Festlegen des Compiler & Festlegen des Linker

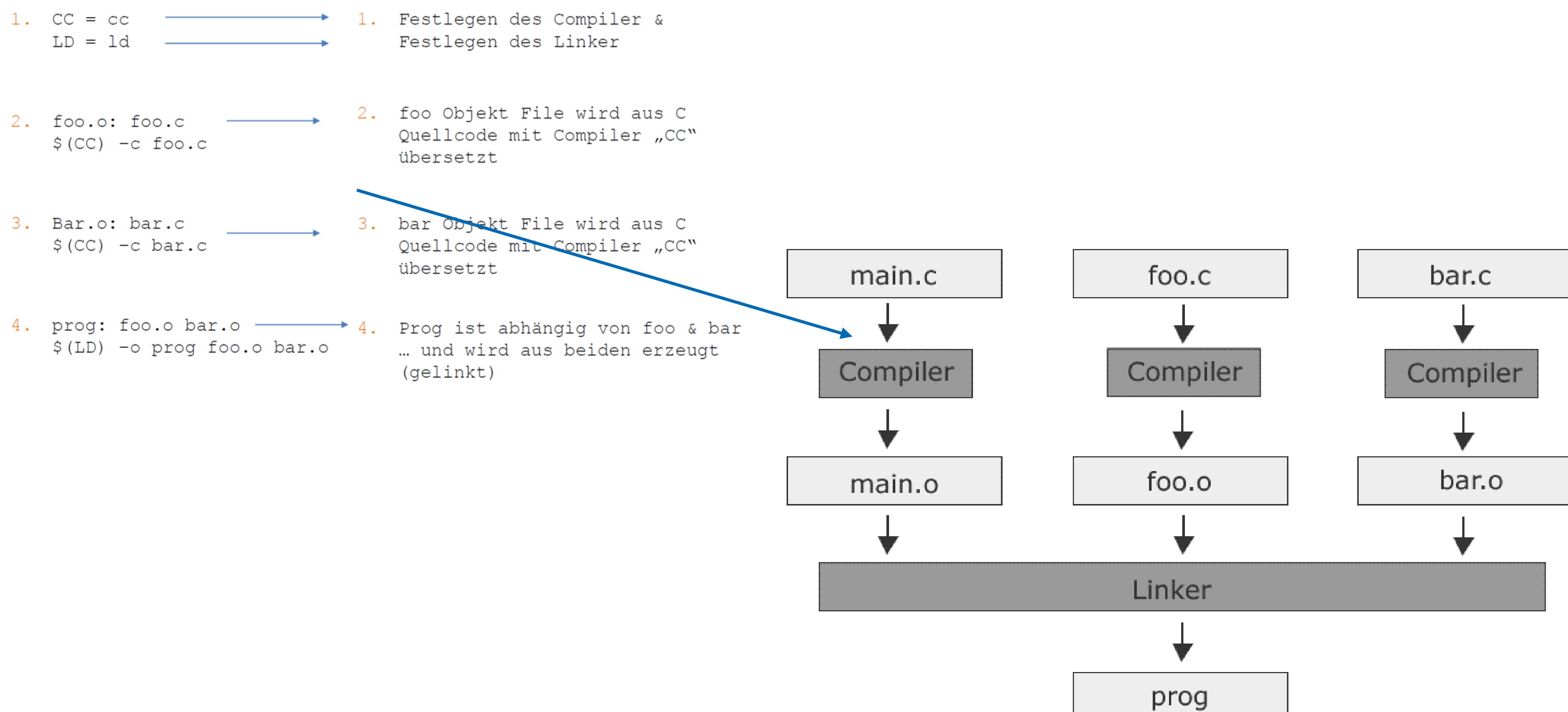
2. `foo.o: foo.c`
`$(CC) -c foo.c` → 2. foo Objekt File wird aus C Quellcode mit Compiler „CC“ übersetzt

3. `Bar.o: bar.c`
`$(CC) -c bar.c` → 3. bar Objekt File wird aus C Quellcode mit Compiler „CC“ übersetzt

4. `prog: foo.o bar.o`
`$(LD) -o prog foo.o bar.o` → 4. Prog ist abhängig von foo & bar … und wird aus beiden erzeugt (gelinkt)

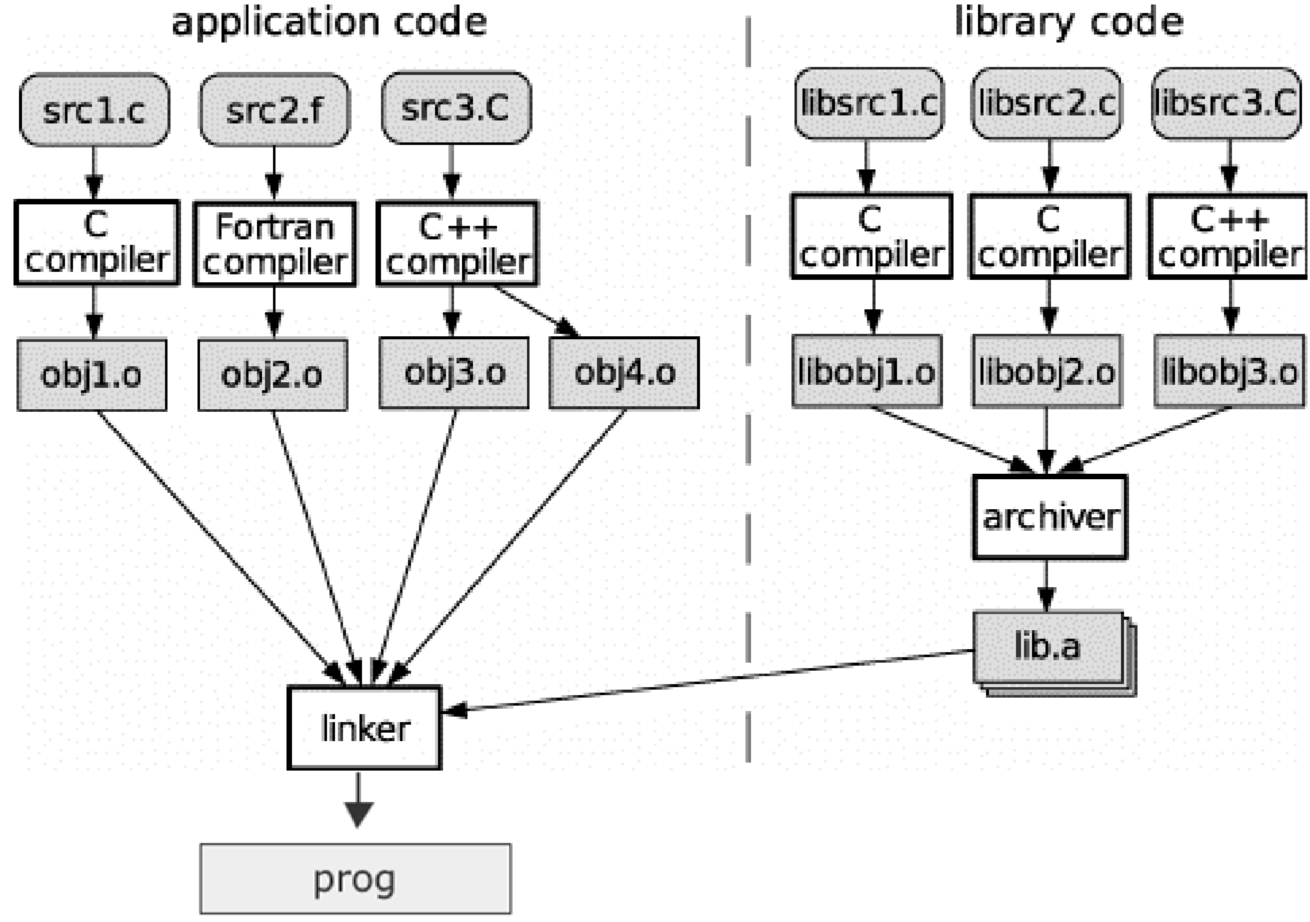
(C) 02.05.2023 Hochschule Mittweida

Makefile



(C) 02.05.2023 Hochschule Mittweida
Fraunhofer Lernlabor Bundeskriminalamt
HOCHSCHULE MITTWEIDA

Makefile externe Libaries



(C) 02.05.2023 Hochschule Mittweida

Paketverwaltung vs Build-System

Paketverwaltung Vorteile	Build-System Vorteile
zeitsparend	Sicherheit, das Executable auf Basis des vorhandenen Quellcodes erstellt wurde
fehlerresistenter	Sourcecode anpassbar
Abhängigkeiten vollständig automatisiert	Abhängigkeiten automatisiert geprüft, jedoch manuell aufgelöst
automatische Updates	Angepasst auf eigene Hardwarekonfiguration

(C) 02.05.2023 Hochschule Mittweida

Vielen Dank

Prof. Ronny Bodach

Hochschule Mittweida | University of Applied Sciences
Technikumplatz 17 | 09648 Mittweida
Fakultät Angewandte Computer- und Biowissenschaften

T +49 (0) 3727 58-1011
F +49 (0) 3727 58-21011
bodach@hs-mittweida.de
www.cb.hs-mittweida.de

Haus 8 | Richard-Stücklen Bau| Raum 8-205
Am Schwanenteich 6b| 09648 Mittweida

Felix Fischer

Hochschule Mittweida | University of Applied Sciences
Technikumplatz 17 | 09648 Mittweida
Fakultät Angewandte Computer- und Biowissenschaften

fische11@hs-mittweida.de
www.cb.hs-mittweida.de