Systemprogrammierung - AIN/2

Sommersemester 2022

Übungsaufgabe 6: Bibliotheken, Shell-Scripts, Makefiles

Abgabe bis 23./24.6.2022

Aufteilung in Verzeichnisse

Legen Sie ein neues Verzeichnis `Aufgabe6` mit Unterverzeichnissen `bin` und `lib` an. Teilen Sie darin Ihr in Aufgabe 5 erstelltes Programm wie folgt auf:

- das Unterverzeichnis `bin` soll die Übersetzungseinheit `notenspiegel` mit dem Hauptprogramm `main` enthalten
- das Unterverzeichnis `lib` soll die Übersetzungseinheiten `benotung`, `fachnote` und `fachnoten_liste` enthalten

Statische Bibliothek

Schreiben Sie im Unterverzeichnis `lib` ein Shell-Script `build-static.sh`, das für die drei Übersetzungseinheiten Objketdateien erzeugt und diese drei Objektdateien `benotung.o`, `fachnote.o` und `fachnoten_liste.o` dann in einer statischen Bibliothek `libaufgabe6.a` zusammenfasst. Das Skript soll die verwendeten Befehle auf der Konsole ausgeben und es soll sich beenden, sobald ein Kommando einen Fehlercode zurückgibt.

Verwenden Sie im Skript den `g++` mit den empfohlenen Optionen zur Qualitätssicherung.

Führen Sie das Shell-Script aus und prüfen Sie, ob die erzeugte Bibliothek alle drei Übersetzungseinheiten enthält:

```
./build-static.sh
ar t libaufgabe6.a
```

Lesen Sie nach, was die Option `t` bei `ar` bewirkt: `man ar`

Hinweis: `sh` verlangt Linux-Zeilenwechsel in Skriptdateien. Windows-Zeilenwechsel führen zu Fehlermeldungen.

Dynamische Bibliothek

Schreiben Sie im Unterverzeichnis `lib` ein Shell-Script `build-dynamic.sh`, das für die drei Übersetzungseinheiten wie zuvor mit dem `g++` Objketdateien erzeugt und diese drei Objektdateien `benotung.o`, `fachnote.o` und `fachnoten_liste.o` dann in einer dynamischen Bibliothek

`libaufgabe6.so` zusammenfasst. Das Skript soll die verwendeten Befehle auf der Konsole ausgeben und es soll sich beenden, sobald ein Kommando einen Fehlercode zurückgibt.

Hinweis: Auf den Laborrechnern müssen Sie zum Erzeugen der Objektdateien den g++ mit der Option `-fpic` aufrufen. Sonst lässt sich die dynamische Bibliothek nicht bauen. `pic` steht für position independent code. Achten Sie außerdem wie zuvor auf die empfohlenen Optionen zur Qualitätssicherung.

Führen Sie das Shell-Script aus und prüfen Sie, ob die erzeugte Bibliothek eine dynamische Bibliothek ist und Funktionen ihrer Übersetzungseinheiten enthält:

```
./build-dynamic.sh
file libaufgabe6.so
nm -gC libaufgabe6.so | grep 'benotung\|fachnote\|fachnoten_liste'
```

Lesen Sie nach, was die verwendeten Linux-Kommandos tun: `man file`, `man nm`, `man grep`

Makefiles

Erstellen Sie nacheinander die folgenden Makefiles:

- `Aufgabe6/lib/Makefile`
  soll eine Bibliothek mit den Übersetzungseinheiten `benotung`, `fachnote` und `fachnoten_liste` erstellen, wobei der Bibliothekstyp mittels einer Variablen wählbar ist:

  ```
  make             # statische Bibliothek libaufgabe6.a erstellen
  make LIBTYPE=a   # statische Bibliothek libaufgabe6.a erstellen
  make LIBTYPE=so  # dynamischeBibliothek libaufgabe6.so erstellen
  ```

  Das Makefile soll folgende Regeln enthalten:
  > eine Musterregel für das Übersetzen der C-Quellen
  > Pseudoziele `all` und `clean`
  > eine explizite Regel zum Erstellen der statischen Bibliothek
  > eine explizite Regel zum Erstellen der dynamischen Bibliothek
  > eine explizite Regel zum Erstellen der Datei `depend`

  Das Makefile soll die Abhängigkeitsregeln für die beiden Objektdateien mit dem C++-Compiler automatisch erzeugen (`depend`-Regel) und per `include` einbinden.

  Das Makefile soll überall Variablen gemäß den in der Vorlesung besprochenen Stilregeln verwenden, d.h. alle Kommandos werden über Variablen aufgerufen und für alle Dateinamen gibt es Hilfsvariablen. Dabei sollen alle verwendeten Variablen im Makefile explizit gesetzt werden, selbst wenn sie vordefiniert sind, d.h. `make -R` ohne vordefinierte Variablen soll funktionieren.

  Hinweise:
  > stellen Sie Ihren Editor so ein, dass er Tabulatorzeichen nicht durch Leerzeichen ersetzt
  > bei C++ verwendet man statt der Variablen CC und CFLAGS die Variablen CXX und

CXXFLAGS
> geben Sie bei der `all`-Regel als zu erstellendes Ziel `lib$(LIBNAME).$(LIBTYPE)` an

Testen Sie das Makefile mit den obigen Aufrufen und prüfen Sie die entstandenen Bibliotheken wie bei den Shell-Skripts. Testen Sie insbesondere, ob das Makefile bei Änderung einer Quelldatei genau die erforderlichen Kommandos zur Aktualisierung der Bibliothek ausführen, nicht mehr und nicht weniger. Die Änderung einer Datei können Sie mit dem Kommando `touch` vortäuschen:

```
touch Dateiname
```

Informieren Sie sich mit `man touch` über das Kommando.

- `Aufgabe6/bin/Makefile`
  erstellt ein ausführbares Programm `notenspiegel`, das mit einer der Bibliotheken aus dem Unterverzeichnis `lib` gebunden ist.

  Es sind die gleichen Stilregeln und Anforderungen wie bei `Aufgabe6/lib/Makefile` umzusetzen.

  Verwenden Sie im Makefile die gcc-Optionen `-I`, `-L` und `-l`, damit die Bibliothek und deren Header-Dateien gefunden werden. Die Option `-I` gehört zu den Präprozessor-Optionen, für die die Variable `CPPFLAGS` vorgesehen ist. Die Option `-L` gehört zu den Linker-Optionen, für die die Variable `LDFLAGS` vorgesehen ist.

- `Aufgabe6/Makefile`
  ruft erst das `Makefile` im Unterverzeichnis `lib` und dann das `Makefile` im Unterverzeichnis `bin` rekursiv auf. In den Vorlesungsunterlagen finden Sie eine ausformulierte Vorlage für das rekursive Makefile.

  Rufen Sie zum Testen der Make-Rekursion im Verzeichnis `Aufgabe6` folgende Befehle auf und prüfen Sie die Ergebnisse:

  ```
  make clean
  make -R all
  bin/notenspiegel
  make -R clean
  make -R LIBTYPE=so
  ldd bin/notenspiegel
  LD_LIBRARY_PATH=lib ldd bin/notenspiegel
  LD_LIBRARY_PATH=lib bin/notenspiegel
  ```

  Was hat es mit der Umgebungsvariablen `LD_LIBRARY_PATH` auf sich? Lesen Sie dazu Folie 4-25.

  Was könnte man in Ihren Makefiles alles weglassen, wenn man auf die `make`-Option `-R` verzichten würde?

Abgabe

Führen Sie Ihre Shell-Scripts und Makefiles vor.

Freiwillige Zusatzaufgabe (1 Bonuspunkt)

- Schreiben Sie ein Makefile `aufgabe6/bonus.mak`, das Ihre Lösung von Aufgabe 6 in eine Datei `aufgabe6.tar.gz` verpackt. Halten Sie alle besprochenen Stilregeln für ein gutes Makefile ein. Das Makefile muss mit der Option `-R` ausführbar sein.

Prof. Dr. H. Drachenfels
Hochschule Konstanz - ***Impressum*** - ***Datenschutzerklärung***

Letzte Änderung: 24.2.2022