Hochschule Furtwangen
Fakultät Informatik
Prof. Dr. Lothar Piepmeyer

Plattformen für eingebettete Systeme

Aufgabenblatt 4

Im Rahmen dieser Aufgaben installieren und testen Sie die Toolchain, mit der Sie Software für das Target entwickeln können. Abschließend übersetzen und installieren Sie das bekannte Datenbanksystem SQLite.

Aufgabe 1:
Im Rahmen des GNU-Projektes wurde der bekannte gcc-Compiler entwickelt. Von GNU gibt es aber auch eine Toolchain mit Cross-Compiler. Diese wurde an die Anforderungen des Raspberry Pi angepasst und wird in einem Git-Repository zur Verfügung gestellt. Kopieren Sie es mit

```
git clone https://github.com/raspberrypi/tools
```

auf Ihre lokale Festplatte. Sie finden ein Unterverzeichnis namens `tools`, das Sie am Besten in das Verzeichnis `/opt` Ihres Hosts verschieben. Grundsätzlich gibt es auch ein Ubuntu-Paket, doch ist dies nur begrenzt für unsere Zwecke einsetzbar.

Aufgabe 2:
Im Verzeichnis `/opt/tools/arm-bcm2708` finden Sie mehrere Unterverzeichnisse, die Toolchains enthalten. Wir arbeiten mit der Toolchain aus dem Verzeichnis `gcc-linaro-arm-linux-gnueabihf-raspbian-x64`.
a. Setzen Sie die Umgebungsvariablen `PATH, ARCH, CROSS_COMPILE, CC` korrekt.
b. Sorgen Sie mit Hilfe der Datei `.bashrc` dafür, dass die Umgebungsvariablen gleich richtig gesetzt werden, wenn Sie eine neue Shell öffnen.
c. Prüfen Sie mit dem Befehl `env` *unbedingt* Ihre Umgebungsvariablen auf Vollständigkeit!

Aufgabe 3:
Wechseln Sie in Ihr Projektverzeichnis auf dem Host und erzeugen Sie ein neues Verzeichnis namens `c`. Wechseln Sie in das neue Verzeichnis. Kopieren und übersetzen Sie die erste Datei `hello.c` aus dem Vorlesungsskript, so dass eine Datei `hello` erzeugt wird, die auf dem Target ausgeführt werden kann. Führen Sie das übersetzte Programm auf dem Target aus.

Aufgabe 4:
Sie werden jetzt mit Shared Libraries arbeiten. Kopieren und übersetzen Sie dazu die Dateien `hello.c, add.c, multiply.c` aus dem Vorlesungsskript.

Binden Sie `add.c` und `multiply.c`, wie im Vorlesungsskript beschrieben, zu einer Shared Library. Binden Sie diese Bibliothek zu `hello.c` hinzu. Die Schritte werden im Skript beschrieben.

Ergreifen Sie auf dem Target geeignete Maßnahmen, damit die Binärdatei `hello` ausgeführt wird und führen Sie `hello` aus.

Aufgabe 5 (Hausaufgabe):
Entwickeln Sie auf dem *Host* ein Makefile, das die Schritte aus der vorgehenden Aufgabe automatisiert.

Aufgabe 6:
Legen Sie auf dem *Host* in Ihrem Projektverzeichnis ein Verzeichnis **`sqlite`** an und wechseln Sie in dieses Verzeichnis. Auf der Webseite www.sqlite.org finden Sie in der Rubrik ‚Downloads' zwei Archive mit Quellcode. Laden Sie das Archiv, das das Wort ‚autoconfig' enthält mit **`wget`** herunter und packen Sie die Daten aus. Es sollte ein neues Verzeichnis vorhanden sein, in das Sie jetzt wechseln.

Aufgabe 7:
Softwarepakete können für verschiedene Plattformen übersetzt werden. Spezielle Komponenten wie eine GUI sollen vorhanden oder nicht vorhanden sein. Entwickler könnten für jeden dieser Fälle ein passendes Makefile zur Verfügung stellen. Dies ist aber aufwändig und fehleranfällig.
Mit Hilfe der Werkzeuge `autoconfig` und `automake` lassen sie sich Shell-Skripte generieren, die man über Parameter steuern kann. Das Ergebnis des Aufrufs des Shell-Skripts ist dann ein Makefile, das die Software, wie über die Parameter eingestellt, übersetzt. Diese Shell-Skripte tragen standardmäßig den Namen `configure`. Die Handhabung beider Werkzeuge ist aufwändig du soll nicht in diesem Workshop diskutiert werden.

Bei SQLite wollen wir nicht viel konfigurieren. Wir teilen nur mit, dass für ARM-Prozessoren auf Linux-System übersetzt werden soll:

```
./configure --host=arm-linux
```

Schauen Sie sich mit

```
ls -l Makefile
```

den Zeitstempel des Makefiles an. Es wurde tatsächlich gerade erzeugt.

Aufgabe 8:
Starten Sie die Übersetzung wie gewohnt mit

```
make
```

Aufgabe 9:
Der Aufruf von `make` hat Bibliotheken und Binärdateien erzeugt, die wir jetzt suchen und in die passenden Unterverzeichnisse von `/usr/local` auf dem Target kopieren *könnten*. Wir lassen uns aber – viel einfacher - von `make` ein Verzeichnis erstellen, in dem die übersetzen Dateien bereits passend abgelegt sind

```
make DESTDIR=$PWD/install install
```

Die Umgebungsvariable `PWD` wird vom Betriebssystem verwaltet und repräsentiert das aktuelle Verzeichnis. Wechseln Sie in das neue Unterverzeichnis `install` und informieren Sie sich, indem Sie die Verzeichnisstruktur durchlaufen.

Aufgabe 10:
Wechseln auf dem *Target* über NFS in das `install`-Verzeichnis auf dem Host. Kopieren Sie die Verzeichnisstruktur des dortigen `usr`-Verzeichnisses mit `cp -R` in das Root-Verzeichnis Ihres Targets. Wechseln Sie anschließend nach `/usr/local/bin`. Dort sollten Sie jetzt eine Datei namens `sqlite3` sehen.

Aufgabe 11:
Legen Sie in der Datenbank eine Tabelle an, fügen Sie einen Datensatz ein und lassen Sie ihn sich anzeigen.