Praktischer Einstieg in Linux und Buildroot

 Dennis Schulmeister-Zimolong, dhbw@windows3.de

Dieses Aufgabenblatt soll dir helfen, dich im Projekt mit Linux zurechtzufinden. Es enthält daher verschiedene Aufgaben, um Linux kennen zu lernen und eine kleine Firmware für den Raspberry Pi zu bauen. Nachdem du das Blatt bearbeitet hast kannst du …

- Dateien und Ordner mit der Konsole verwalten
- Befehle in der Konsole ausführen
- Einfache Shell-Skripte schreiben
- Mit Buildroot eine Firmware für den Raspberry Pi bauen
- Eine SSH-Verbindung zum Raspberry Pi aufbauen
- Selbstgeschriebene Programme in die Firmware integrieren

Benötigte Materialien

Um das Aufgabenblatt durchzuarbeiten benötigst du folgende Hardware:

- Einen Raspberry Pi mit SD-Karte
- Zugriff auf einen Bildschirm bzw. den Beamer im Vorlesungsraum
- Einen Laptop mit VirtualBox und SSH (z.B. PuTTY)
- Ein Netzwerkkabel zur Verbindung der beiden Rechner

Außerdem benötigst du folgende Unterlagen:

- Das Skript *Linux und Buildroot*[1]
- Linux Command Reference: Learning the Shell

Aufgabe 1: Vorbereitung

Für die Bearbeitung der folgenden Aufgaben benötigst du die am Anfang des Projekts zur Verfügung gestellte Linux VM. Falls diese nicht mehr auf deinem Laptop eingerichtet ist, folge der Anleitung aus dem Vorlesungsskript:

- Kapitel 2.1: Einrichten von VirtualBox
- Kapitel 2.2: SSH-Verbindung zur virtuellen Maschine herstellen

Aufgabe 2: Erste Schritte mit Linux

a) Deine ersten Gehversuche mit Linux kannst duauf einem beliebigen Linux-System machen. Der Einfachheit halber verwenden wir hierfür die Linux VM, mit wir später auch die Firmware Images bauen werden. Starte daher die VM und melde dich sich mit SSH (nicht über das Bildschirm-Fenster von VirtualBox!) an. Anschließend führe folgende Aktionen aus. Das Kapitel 2.3 *Erste Schritte mit Linux und der Konsole* hilft dir dabei:

1. Finde den exakten Pfad des aktuellen Arbeitsverzeichnisses heraus.

   Befehl: _____

1 Auch wenn es immer noch nicht fertig ist :-)

2. Wechsele in das Verzeichnis `~/buildroot/configs`.

   Befehl: _____

3. Lasse dir alle Dateien im aktuellen Verzeichnis anzeigen.

   Befehl: _____

4. Lasse dir alle Dateien, deren Namen das Wort `raspberrypi` enthält, anzeigen.

   Befehl: _____

5. Wechsel zurück in das Home-Verzeichnis.

   Befehl: _____

6. Finde heraus, welchen Dateityp die Datei `install.sh` hat.

   Befehl: _____

7. Gebe den Inhalt der Datei `install.sh` auf der Konsole aus (zwei Möglichkeiten):

   Befehl 1: _____

   Befehl 2: _____

b) Herzlichen Glückwunsch. Du weißt nun, wie man sich in der Konsole umschauen kann. Als nächstes bearbeiten wir ein paar Dateien. Hierfür wechsele in das Verzeichnis `~/shared` und führe darin die folgenden Aktionen aus:

1. Lege ein Verzeichnis namens `tutorial` an:

   Befehl: _____

2. Lege die Verzeichniskette `tutorial/first-steps/directories` an (ein Befehl):

   Befehl: _____

3. Wechsele in das Verzeichnis `tutorial` und lege darin die Datei `readme.txt` an:

   Befehl 1: _____

   Befehl 2: _____

4. Lege eine Datei namens `hallo.txt` und dem Inhalt „Linux ist klasse!“ an (ohne Editor):

   Befehl: _____

5. Hänge eine weitere Textzeile an die Datei `hallo.txt` an (ohne Editor):

   Befehl: _____

6. Benenne die Datei `hallo.txt` in `linux.txt` um:

   Befehl: _____

7. Benenne das Verzeichnis `tutorial` in `beispiel` um:

   Befehl: _____

8. Erzeuge ein Verzeichnis namens `beispiel2` und kopiere alle *.txt-Dateien dorthin:

   Befehl 1: _____

   Befehl 2: _____

9. Lösche das Verzeichnis `beispiel1` mit all seinen Unterverzeichnissen:

   Befehl: _____

c) Als nächstes bearbeiten wir ein paar Dateien. Dies ist unter Linux besonders wichtig, da fast alle Einstellungen in Textdateien hinterlegt sind. Kapitel 2.4 *Textdateien bearbeiten auf der Konsole* verrät dir, wie das geht.

1. Öffne die Datei `~/install.sh` in einem Texteditor.

   Befehl: _____

2. Suche nach allen Vorkommen des Worts „buildroot“ in der Datei:

   Tastatureingabe: _____

3. Beende den Editor ohne zu sichern:

   Tastatureingabe: _____

4. Öffne nun die Datei `~/shared/beispiel2/linux.txt` aus der vorherigen Aufgabe:

   Befehl: _____

5. Schreibe mehrere Zeilen in die Datei und speichere sie, ohne den Editor zu verlassen:

   Tastatureingabe: _____

6. Schneide die ersten drei Zielen aus und sichere diese in der Zwischenablage:

   Tastatureingabe: _____

7. Füge die eben ausgeschnittenen Zeilen am Ende der Datei wieder ein:

   Tastatureingabe: _____

8. Beende den Editor:

   Tastatureingabe: _____

d) Weiter mit Kapitel 2.3 wollen wir zunächst aber ein paar Programme ausführen und uns anschauen, wie man laufende Programme überwachen kann. Immer noch im Verzeichnis `~/shared` führe daher folgende Aktionen aus:

1. Schreibe den Inhalt des aktuellen Verzeichnisses in die Datei `inhalt.txt`:

   Befehl: _____

2. Starte den Befehl `less inhalt.txt` im Hintergrund:

   Befehl: _____

3. Hole `less` wieder in den Vordergrund:

   Befehl: _____

4. Starte `man man` im Hintergrund:

   Befehl: _____

5. Beende `man`, indem du `SIGTERM` an den Prozess schickst:

   Befehl: _____

6. Lasse dir alle vom aktuellen Benutzer gestarteten Programm anzeigen:

   Befehl: _____

7. Lassen dir alle laufenden Programme aller Benutzer anzeigen (zwei Möglichkeiten):

   Befehl 1: _____

   Befehl 2: _____

8. Suche nach der Prozess ID des ssh-Dienstes:

   Befehl: _____

e) Zum Abschluss unserer kleinen Rundumschau schreibe ein Skript, das ein Auswahlmenü zur Automatisierung wichtiger Aufgaben anzeigt.

```
buildroot@debian:~$ ./menu.sh
WICHTIGE AUFGABEN

[1] Buildroot initialisieren (minimale Konfiguration)
[2] Buildroot initialisieren (Konfiguration mit Wayland)
[3] Buildroot initialisieren (Konfiguration mit HTML UI)
[4] sdcard.img nach ~/shared kopieren
[E] Ende
```

```
Deine Auswahl: _
```

Je Menüpunkt soll eine eigene Funktion aufgerufen werden, welche die eigentlich auszuführenden Befehle enthält. Folgende Befehle sollen die Funktionen enthalten:

```
# Buildroot initialisieren (minimale Konfiguration)
cd ~/buildroot
make BR2_EXTERNAL=../custom O=../make dhbw_minimal_defconfig
cd ..

# Buildroot initialisieren (Konfiguration mit Wayland)
cd ~/buildroot
make BR2_EXTERNAL=../custom O=../make dhbw_wayland_defconfig
cd ..

# Buildroot initialisieren (Konfiguration mit HTML UI)
cd ~/buildroot
make BR2_EXTERNAL=../custom O=../make dhbw_html_defconfig
cd ..

# Ein neues Buildroot-Projekt starten
cd ~/make
make dhbw_minimal_defconfig

# sdcard.imag nach ~/shared kopieren
cd ~/make/images/sdcard.img ~/shared
```

Aufgabe 3: Unsere erste eigene Firmware

Nachdem du dich nun mit Linux vertraut gemacht hast, kannst du dich nun an Buildroot heranwagen und deine erste Firmware für den Raspberry Pi bauen. Die hierfür benötigten Schritte sind im Skript in Kapitel 2 ausführlich beschrieben. Die wichtigsten Kapitel für den Einstieg sind dabei Kapitel 3.1 *Wir erkunden Buildroot* und Kapitel 3.4 *Hallo Welt: Unsere erste Firmware auf dem Raspberry Pi*.

a) Führe die am Anfang von Kapitel 3.1 beschriebenen Befehle aus, um Buildroot zu initialisieren. Anschließend lasse dir eine Liste der vorhandenen Vorlagekonfigurationen anzeigen.

b) Falls noch nicht geschehen, lade die Vorlagekonfiguration `dhbw_minimal_defconfig`, um ein neues Projekt für Raspberry Pi zu starten und erkunde danach ein wenig das Buildroot-Konfigurationsmenü. Findest du heraus, wie du nodeJS in die Firmware aufnehmen kannst?

c) Verlasse das Konfigurationsmenü, ohne deine Änderungen zu sichern. Starte danach die Erstellung der Firmware und warte, bis der Vorgang fehlerfrei durchgelaufen ist.

d) Folge nun der Anleitung aus Kapitel 3.4 *Hallo Welt: Unsere erste Firmware auf dem Raspberry Pi*, um das eben erstelle Firmware Image auf die SD-Karte zu schreiben.

e) Zum Schluss schließen den Raspberry Pi an den Beamer oder einen freien Bildschirm an und fahre ihn hoch. Melden dich mit den in Kapitel 3.4 *Hallo Welt: Unsere erste Firmware auf dem Raspberry Pi* genannten Zugangsdaten an und erkunde das Dateisystem. Anschließend fahre den Raspberry Pi wieder herunter.

Aufgabe 4: Fernzugriff auf den Raspberry Pi

In der Vorlagekonfiguration ist bereits ein SSH-Server enthalten, so dass du dich auch remote auf dem Raspberry Pi einloggen kannst. Somit kannst du den Raspberry Pi prinzipiell auch ohne Bildschirm, Tastatur und Maus bedienen, was an der DHBW natürlich sehr sinnvoll ist. Allerdings kannst du auf diese Weise nur Kommandozeilenprogramme ausführen. Wenn euer Projekt keine grafische Ausgabe auf einem Bildschirm vorsieht, genügt dies bereits. Andernfalls müsst ihr euch

später mit Kapitel 5 *Grafische Benutzeroberflächen* auseinandersetzen. Doch alles der Reihe nach. Momentan befinden wir uns noch im Kapitel 3.5 *SSH-Login am Raspberry Pi.*

a) Verbinde den Raspberry Pi über ein Netzwerkkabel mit deinem Laptop. Die beiden Rechner teilen sich dann ein Ad Hoc-Netzwerk, das du auf deinem Laptop jedoch erst noch konfigurieren must. Die IP-Adresse des Raspberry Pi lautet 192.168.99.99. Richte auf deinem Laptop daher ein Kabelnetzwerk mit folgenden Daten ein:

- Netzwerkadresse: 192.168.99.0/24
- Eigene IP-Adresse 192.168.99.xxx (frei wählbar)

Anleitungen hierzu findest du im Internet, wenn du nach *How to Assign a Static IP Adress* suchst.

b) Schalte den Raspberry Pi ein und prüfe, ob er auf `ping`-Anfragen reagiert. Hierfür kannst du auf deinem Laptop folgenden Befehl eingeben[2]:

```
dennis@metropolis:~/make$ ping 192.168.99.99
PING metropolis.lan (192.168.1.100) 56(84) bytes of data.
64 bytes from raspberrypi.lan (192.168.99.99): icmp_seq=1 ttl=63 time=0.180 ms
64 bytes from raspberrypi.lan (192.168.99.99): icmp_seq=2 ttl=63 time=0.334 ms
64 bytes from raspberrypi.lan (192.168.99.99): icmp_seq=3 ttl=63 time=0.272 ms
64 bytes from raspberrypi.lan (192.168.99.99): icmp_seq=4 ttl=63 time=0.292 ms
```

c) Stelle nun eine SSH-Verbindung zum Raspberry Pi her. Hierfür kannst du entweder den SSH-Client auf deinem Laptop oder die Linux VM verwenden. Innerhalb der Linux VM lautet der Befehl wie folgt:

```
$ ssh 192.168.99.99
```

Sollte an dieser Stelle die Fehlermeldung *Connection refused* erscheinen, musst du erst eine kleine Änderung an der Firmware vornehmen. Vermutlich konnte der SSH-Server nicht gestartet werden, da die Berechtigungen der Datei `/etc/sshd/ssh_host_rsa_key` zu offen sind. In Kapitel 3.*11 Zugriffsrechte einzelner Dateien ändern* ist die Lösung des Problems beschrieben. Anschließend baue ein neues Image und schreibe es auf die SD-Karte.

Aufgabe 5: Eigene Programme zum Laufen bringen

a) Gehe zunächst wie in Kapitel 6.1 *Java in die Firmware integrieren* beschrieben vor, um eine Firmware mit Java Runtime Edition zu erstellen. Der Vorgang ist im Prinzip ganz einfach, jedoch ist es etwas umständlich, die richtige Datei auf der Oracle-Downloadseite zu finden. Sollte es beim Entpacken des JDKs zu einem Fehler kommen, kann ich dir stattdessen auch eine fertig aufbereitete ZIP-Datei zukommen lassen.

b) Schreibe ein kleines Java-Programm, dass zum Beispiel „Hallo Welt“ auf der Konsole ausgibt. Kompiliere das Programm und kopiere die kompilierten *.class-Dateien in das Verzeichnis `~/custom/board/overlays/rootfs_overlay_custom/opt/dhbw`.

c) Anschließend schreibe ein kleines Skript, mit dem das Javaprogramm gestartet werden kann und kopiere es ebenfalls nach `~/custom/board/overlays/rootfs_overlay_custom/opt/dhbw`. Das Skript könnte zum Beispiel so aussehen:

```
debian@buildroot:~/custom/overlays/rootfs_oberlay_custom/opt/dhbw$ nano start.sh
#! /bin/sh
cd /opt/dhbw
java HalloWelt
```

2 Ausnahmsweise ist der Befehl für Windows, Mac OS und Linux hier derselbe.

Folge dann der Anleitung aus Kapitel 3.12 *Automatischer Start von Programmen beim Hochfahren*, um das Skript während dem Bootvorgang auszuführen.

Herzlichen Glückwunsch!

Wenn du wirklich bis hierhin alles durchgearbeitet hast, kennst du dich inzwischen ganz gut mit Linux aus. Eurem Projekt steht dann nichts mehr im Weg.

Quelle: Pixabay "Bellinon"
