Praktischer Einstieg in Linux und Buildroot

Dieses Aufgabenblatt soll Ihnen helfen, sich 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 Sie das Blatt bearbeitet haben können Sie …

- 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
- Die Grafikausgabe das Raspberry Pi auf Ihren Laptop umleiten
- Selbstgeschriebene Programme in die Firmware integrieren

Benötigte Materialien

Um das Aufgabenblatt durchzuarbeiten benötigen Sie folgende Hardware:

- Einen Raspberry Pi mit SD-Karte
- Zugriff auf einen Bildschirm bzw. den Beamer im Vorlesungsraum
- Einen Laptop mit VirtualBox, SSH und ggf. einem VNC-Viewer
- Ein Netzwerkkabel zur Verbindung der beiden Rechner

Außerdem benötigen Sie folgende Unterlagen:

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

Aufgabe 1: Vorbereitung

Für die Bearbeitung der folgenden Aufgaben benötigen Sie die am Anfang des Projekts zur Verfügung gestellte Linux VM. Falls diese nicht mehr auf Ihrem Laptop eingerichtet ist, folgen Sie der Anleitung aus dem Vorlesungsskript:

- Kapitel 1.1: Einrichten von VirtualBox
- Kapitel 1.2: SSH-Verbindung zur virtuellen Maschine herstellen

Aufgabe 2: Erste Schritte mit Linux

a) Ihre ersten Gehversuche mit Linux können Sie auf einem beliebigen Linux-System machen. Der Einfachheit halber verwenden wir hierfür die Linux VM, mit wir später auch die Firmware Images bauen werden. Starten Sie daher die VM und melden Sie sich mit SSH an. Anschließend führen Sie folgende Aktionen aus. Das Kapitel 1.3 *Erste Schritte mit Linux und der Konsole* hilft Ihnen dabei:

1. Finden Sie den exakten Pfad des aktuellen Arbeitsverzeichnisses heraus.

   Befehl: _____

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

2. Wechseln Sie in das Verzeichnis `~/buildroot/configs`.

   Befehl: _____

3. Lassen Sie sich alle Dateien im aktuellen Verzeichnis anzeigen.

   Befehl: _____

4. Lassen Sie sich alle Dateien, in denen das Wort `raspberrypi` vorkommt, anzeigen.

   Befehl: _____

5. Wechseln Sie zurück in das Home-Verzeichnis.

   Befehl: _____

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

   Befehl: _____

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

   Befehl 1: _____

   Befehl 2: _____

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

1. Legen Sie ein Verzeichnis namens `beispiel1` an:

   Befehl: _____

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

   Befehl: _____

3. Legen Sie eine leere Datei namens `beispiel1/readme.txt` an:

   Befehl: _____

4. Legen Sie eine Datei namens `hallo.txt` und dem Inhalt „Linux ist klasse!“ an:

   Befehl: _____

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

   Befehl: _____

6. Benennen Sie die Datei `hallo.txt` in `linux.txt` um:

   Befehl: _____

7. Benennen Sie das Verzeichnis `beispiel1` und `beispiel2` um:

   Befehl: _____

8. Kopieren Sie alle *.txt-Dateien in das Verzeichnis `beispiel2`:

   Befehl: _____

9. Löschen Sie das Verzeichnis `tutorial` 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 1.4 *Textdateien bearbeiten auf der Konsole* verrät Ihnen, wie das geht.

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

   Befehl: _____

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

   Tastatureingabe: _____

3. Beenden Sie den Editor ohne zu sichern:

   Tastatureingabe: _____

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

   Befehl: _____

5. Schreiben Sie mehrere Zeilen in die Datei und speichern Sie, ohne den Editor zu verlassen:

   Tastatureingabe: _____

6. Schneiden Sie die ersten drei Zielen aus und sichern Sie diese in der Zwischenablage:

   Tastatureingabe: _____

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

   Tastatureingabe: _____

8. Beenden Sie den Editor:

   Tastatureingabe: _____

d) Weiter mit Kapitel 1.3 wollen wir zunächst aber ein paar Programme ausführen und uns anschauen, wie man laufende Programme überwachen kann. Immer noch im Verzeichnis `~/shared` führen Sie folgende Aktionen aus:

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

   Befehl: _____

2. Starten Sie den Befehl `less inhalt.txt` im Hintergrund:

   Befehl: _____

3. Holen Sie `less` wieder in den Vordergrund:

   Befehl: _____

4. Starten Sie `man man` im Hintergrund:

   Befehl: _____

5. Beenden Sie `man`, indem Sie `SIGTERM` an den Prozess schicken:

   Befehl: _____

6. Lassen Sie sich alle vom aktuellen Benutzer gestarteten Programm anzeigen:

   Befehl: _____

7. Lassen Sie sich alle aktuell laufenden Programme anzeigen (zwei Möglichkeiten):

   Befehl 1: _____

   Befehl 2: _____

8. Suchen Sie nach der Prozess ID des ssh-Dienstes:

   Befehl: _____

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

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

[1] Buildroot initialisieren
[2] Ein neues Buildroot-Projekt starten
[4] sdcard.img nach ~/shared kopieren
[E] Ende

Ihre Auswahl: _
```

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

```
# Buildroot initialisieren
cd ~/buildroot
make BR2_EXTERNAL=../custom O=../make help > /dev/null
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 Sie sich soweit mit Linux vertraut gemacht haben, können Sie sich nun an Buildroot heranwagen und ihre 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 2.1 *Wir erkunden Buildroot* und Kapitel 2.4 *Hallo Welt: Unsere erste Firmware auf dem Raspberry Pi*.

a) Führen Sie die am Anfang von Kapitel 2.1 beschriebenen Befehle aus, um Buildroot zu initialisieren. Anschließend lassen Sie sich eine Liste der vorhandenen Standard- bzw. Vorlagekonfigurationen anzeigen.

b) Falls noch nicht geschehen, laden Sie die Vorlagekonfiguration `dhbw_minimal_defconfig`, um ein neues Projekt für Raspberry Pi zu starten und erkunden Sie danach ein wenig das Buildroot-Konfigurationsmenü. Finden Sie heraus, wie Sie nodeJS in die Firmware aufnehmen können?

c) Verlassen Sie das Konfigurationsmenü, ohne Ihre Änderungen zu sichern. Starten Sie danach die Erstellung der Firmware und warten Sie, bis der Vorgang fehlerfrei durchgelaufen ist.

d) Folgen Sie nun der Anleitung aus Kapitel 2.4, um das eben erstelle Firmware Image auf die SD-Karte zu schreiben.

e) Zum Schluss schließen Sie den Raspberry Pi an den Beamer oder einen freien Bildschirm an und fahren ihn hoch. Melden Sie sich mit den in Kapitel 2.4 genannten Zugangsdaten an und erkunden Sie das Dateisystem. Anschließend fahren Sie den Raspberry Pi wieder herunter.

Aufgabe 4: Fernzugriff auf den Raspberry Pi

In der Vorlagekonfiguration ist bereits ein SSH-Server enthalten, so dass Sie sich auch remote auf dem Raspberry Pi einloggen können. Somit können Sie den Raspberry Pi prinzipiell auch ohne Bildschirm, Tastatur und Maus bedienen, was an der DHBW natürlich sehr sinnvoll ist. Allerdings können Sie auf diese Weise nur Kommandozeilenprogramme ausführen. Wenn Ihr Projekt keine grafische Ausgabe auf einem Bildschirm vorsieht, genügt dies bereits. Andernfalls werden wir in der nächsten Aufgabe einen VNC-Server integrieren, so dass Sie die gesamte Grafikausgabe auf Ihren Laptop umleiten können. Im Skript befinden wir uns momentan aber noch im Kapitel 2.5 *SSH-Login am Raspberry Pi.*

a) Verbinden Sie den Raspberry Pi über ein Netzwerkkabel mit Ihrem Laptop. Die beiden Rechner teilen sich dann ein Ad Hoc-Netzwerk, das Sie auf Ihrem Laptop jedoch erst noch konfigurieren müssen. Die IP-Adresse des Raspberry Pi lautet 192.168.99.99. Richten Sie auf Ihrem 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 finden Sie im Internet, wenn Sie nach *How to Assign a Static IP Adress* suchen.

b) Schalten Sie den Raspberry Pi ein und prüfen Sie, ob er auf `ping`-Anfragen reagiert. Hierfür geben Sie auf Ihrem Laptop folgenden Befehl ein[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) Stellen Sie nun eine SSH-Verbindung zum Raspberry Pi her. Hierfür können Sie entweder den SSH-Client auf Ihrem 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, müssen Sie 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. Seit Erstellung der Linux VM hat es wohl eine Änderung in OpenSSH gegeben, die in diesem Fall das Starten des Servers aus Sicherheitsgründen verhindert. In Kapitel *2.11 Zugriffsrechte einzelner Dateien ändern* ist die Lösung des Problems beschrieben. Anschließend bauen Sie ein neues Image und schreiben es auf die SD-Karte.

Aufgabe 5: Umleiten der Grafikausgabe auf Ihren Laptop

Wenn Ihre Firmware später auch eine grafische Bildschirmausgabe erzeugen soll, werden Sie vermutlich keine Lust haben, ständig einen eigenen Bildschirm mit in die DHBW zu schleppen. Zwar gibt es auf eBay schon recht günstig kleine 7“ HDMI-Displays, noch günstiger kommt es aber, wenn Sie die Grafikausgabe des Raspberry Pi stattdessen via VNC auf Ihren Laptop umleiten. Auf diese Weise können Sie den Raspberry Pi (mit einem kleinen Zeitversatz natürlich) vom Laptop aus fast genauso bedienen, als hätten Sie Tastatur, Maus und Bildschirm direkt angeschlossen. Die hierfür benötigten Schritte sind in Kapitel 4.1 *Grafische Ausgaben auf den Entwicklungsrechner umleiten* beschrieben.

Aufgabe 6: Eigene Programme zum Laufen bringen

a) Gehen Sie zunächst wie in Kapitel 5.1 *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 Ihnen stattdessen auch eine fertig aufbereitete ZIP-Datei zukommen lassen.

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

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

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

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

Folgen Sie dann der Anleitung aus Kapitel 2.12 *Automatischer Start von Programmen beim Hochfahren*, um das Skript während dem Bootvorgang auszuführen.

Herzlichen Glückwunsch!

Wenn Sie wirklich bis hierhin alles durchgearbeitet haben, kennen Sie sich inzwischen ganz gut mit Linux aus. Ihrem Projekt steht nun nichts mehr im Weg.