Hochschule Furtwangen
Fakultät Informatik
Prof. Dr. Lothar Piepmeyer

Workshop , Plattformen für mobile Systeme'

Aufgabenblatt 9

Im Rahmen dieser Aufgaben erstellen Sie ein vollwertiges Root-Dateisystem

Aufgabe 1:
Laden Sie auf dem Host unter
http://www.busybox.net/downloads
die Version 1.23 von Busybox mit `wget` in Ihr Verzeichnis `mobile-project` und packen Sie das Archiv aus. Sie können es auch mit neueren Versioen versuchen. Ich bin mir aber nicht sicher, ob diese auch in unserer Umgebung fehlerfrei gebaut werden.

Aufgabe 2:
a. Die Umgebung zum Erstellen der Busybox ist ganz ähnlich wie die des Kernels. So liefert auch hier

```
make help
```

eine Übersicht zu den möglichen `make`-Targets

b. Sie erzeugen zunächst die Standardkonfiguration

```
make defconfig
```

c. Wie beim Kernel können Sie auch Busybox menügesteuert konfigurieren:

```
make menuconfig
```

Schauen Sie sich einige der Konfigurationsmöglichkeiten an.

d. Standardmäßig wird Busybox mit Shared-Libraries gebaut. Da die Bibliothek nur genau einmal benötigt wird, ist das nicht nötig. Wir bauen Busybox als statische Binärdatei und entziehen uns so auch dem Problem, dass Bibliotheken gefunden werden müssen. Gehen Sie in das Konfigurationsmenu und dort zu
Busybox Settings->Build Options
und aktivieren Sie
Build BusyBox as a static binary

e. Starten die Erstellung mit

```
make
```

Aufgabe 3:

a. Schließen Sie wie in Aufgabenblatt 3 beschreiben Ihr Target mit Hilfe des seriellen Kabels an den Host an. Beachten Sie, dass das Kabel das Target auch mit Strom versorgt. Verwenden Sie also nicht das Netzteil! Eine Stromversorgung reicht! Ziehen Sie das rote Kabel ab, um die Stromversorgung zu unterbrechen. Starten Sie wie bereits früher `screen` mit geeigneten Parametern. Schließen Sie das rote Kabel wieder an.

b. Am Ende des Boot-Vorgangs wird gemeldet, dass das Target über das Gerät `ttyAMA0` mit Ihnen kommuniziert. Das können Sie auch ausprobieren:

```
echo Hello > /dev/ttyAMA0
```

c. Rufen Sie mit `ifconfig` Ihre IP-Adresse ab und notieren Sie diese unbedingt.

Aufgabe 4:

a. Hängen Sie das Root-Dateisystem Ihrer SD-Karte in den Host etwa unter `/mnt` ein und löschen Sie alle Dateien und Verzeichnisse.

b. Gehen Sie in das Verzeichnis `/mnt` und schauen Sie sich mit der Anweisung `ls –l` die Rechte des eingehängten Root-Dateisystems an. Geben Sie mit der Anweisung

```
sudo chmod –R o+w <verzeichnis>
```

allen Anwendern Rechte in dieses und alle untergeordneten Verzeichnissen zu *schreiben*.

c. Jetzt lassen Sie sich von `make` noch alle erforderlichen Links auf die ELF-Datei `busybox` erzeugen:

```
make install CONFIG_PREFIX=<Pfad zum Root-Dateisystem auf SD>
```

d. Erzeugen Sie - wie in der Vorlesung beschrieben – auf der der SD-Karte eine Verzeichnisstruktur, die dem LHS entspricht.

e. Da es nach dem Hochfahren des Kernels möglich sein soll Ausgaben zu sehen, muss das der Konsole entsprechende Gerät im `/dev`-Verzeichnis des Targets existieren. Hierzu reicht ein einfacher `mkdir` oder `touch` nicht. Jedes Gerät wird durch eine Gerätedatei repräsentiert. Gerätedateien werden wie folgt erzeugt:

```
mknod ttyAMA0 c 204 64
```

Die Rechte werden auch für Gerätedateien wie unter Unix üblich vergeben:

```
chmod 666 ttyAMA0
```

Den Namen `ttyAMA0` haben Sie ja bereits in Aufgabe 1 ermittelt.

f. Nachdem die Rechte Ihres Root-Dateisystems angepasst und `make install` ausgeführt wurde, sollte das Root-Dateisystem jetzt die benötigten Dateien enthalten. Machen Sie etwa im `bin`-Verzeichnis auf dem Root-Dateisystem Ihrer SD-Karte eine Stichprobe. Die Datei `busybox` muss auf *ARM*-Prozessoren ausführbar sein.

g. Hängen Sie die SD-Karte wieder aus.

Aufgabe 5:

a. Versuchen Sie das Target von der SD-Karte zu booten. Wenn Sie alles richtig gemacht haben, gibt es keine Kernel-Panic. Der `init`-Prozess startet und macht dann nichts Sichtbares mehr. Er bietet Ihnen auch nicht –etwa in Form einer Shell – eine Unterhaltung an. Um das zu ändern, hängen wir zunächst die SD-Karte am Target aus und am Host weder ein.

b. Der `init`-Prozess unter Busybox sucht in der Datei `/etc/inittab` nach Arbeit. Die wollen wir ihm jetzt geben:

```
::sysinit:/etc/rcS
::askfirst:/bin/ash
ttyAMA0::askfirst:/bin/ash
```

die referenzierte Datei `/etc/rcS` gibt es noch nicht. Das ändern wir später. Die Anweisungen sorgen dafür, dass das Terminal für eine Shell genutzt wird. Schließen Sie die Datei mit einer Lerrzeile ab.

c. Wechseln Sie die SD-Karte und starten Sie das Target neu. Sie sollten jetzt eine Shell erhalten. Möglicherweise müssen Sie dazu nach dem Boot-Vorgang noch Enter drücken.

Aufgabe 6:

a. Noch besser wird Ihr System, wenn Sie es netzwerkfähig machen. Dazu hängen Sie noch zwei Pseudo-Dateisysteme ein:

```
mount –t proc none /proc
mount –t sysfs none /sys
```

und geben unserem Target noch die IP-Adresse, die Sie sich aufgeschrieben haben:

```
ifconfig eth0 <IP-address>
```

b. Prüfen Sie auf dem Host, ob Sie Ihr Target mit dem Befehl `ping` erreichen können.

c. Die drei Befehle aus a. kommen jetzt in die Datei `/etc/rcS` auf der SD-Karte. Die Karte muss dazu aus- und am Host wieder eingehängt werden. Die Befehle werden gleich beim Neustart ausgeführt. Da die Initialisierung des Targets etwas dauert, empfiehlt es sich, vor der Zuweisung der IP-Adresse die folgende Zeile einzufügen:

```
sleep 3
```

d. Machen Sie die Datei ausführbar:

```
chmod +x /etc/rcS
```

e. Starten Sie das Target und verifizieren Sie die Erreichbarkeit vom Host aus.

f. Standardmäßig hängt Busybox das Root-Dateisystem nur für lesende Zugriff eine. Sie können das ändern:

```
mount –o remount,rw /dev/root  /
```

g. (Bonusaufgabe) Was muss getan werden, damit Busybox bereits beim Start das Root-Dateisystem zum lesen `und` schreiben einhängt?

h. Legen Sie unter `/var` das Verzeichnis `www` an und fügen Sie dort eine ganz einfache HTML-Datei `index.html` ein.

i. Starten Sie Busybox als Webserver:

```
busybox httpd -f  -h /var/www &
```

Prüfen Sie die Erreichbarkeit vom Browser Ihres Host-Systemes.

j. (Bonusaufgabe) Statische IP-Adressen sind unpraktisch. Tatsächlich bringt Busybox den DHCP-Client `udhcpc` mit. Er wird zusammen mit einem Skript aufgerufen:

```
udhcpc –s /etc/simple.script
```

Ersetzen Sie die `ifconfig` Zeile in `/etc/rcS` durch diese Anweisung. Die Datei `simple.script` finden Sie im Unterverzeichnis `examples/udhcp` Ihres Busybox-Quellcodes. Kopieren Sie die Datei geeignet auf Ihre SD-Karte. Das Target sollte jetzt starten. Das macht es aber nicht. Warum nicht? Was muss getan werden, damit `udhcpc` gleich beim Start von Busybox ausgeführt wird?

k. (Bonusaufgabe) Wenn man `udhcpc` wie beschrieben von der Kommandozeile ausführt, erhält das Target eine dynamische IP-Adresse. Das funktioniert so weit. Wenn das Root-Dateisystem wie gezeigt auch beschrieben werden kann, wird auch DNS konfiguriert. Dennoch schlägt

```
ping www.google.de
```

fehl. Warum? Was muss getan werden, damit der Befehl fehlerfrei ausgeführt wird?