1. Experimente mit der Ulix-Devel-VM

In der ersten Übung installieren Sie die virtuelle Maschine und lernen die Verzeichnisstruktur für die Ulix-Entwicklung kennen.

a) Installieren Sie die für Ihr Notebook passende VirtualBox-Version, spielen Sie dann die VirtualBox-Erweiterung *Oracle_VM_VirtualBox_Extension_Pack-4.2.18-88780.vbox-extpack* ein und installieren Sie anschließend die Ulix-Devel-VM aus der Datei *Ulix-Devel.ova*. Im letzten Schritt öffnet sich ein Dialog, der über den geplanten Import der VM informiert. Setzen Sie hier nicht (!) das Häkchen bei der Option *Zuweisen neuer MAC-Adressen für alle Netzwerkkarten*.

b) Wenn der Import abgeschlossen ist, starten Sie die VM per Doppelklick auf den neuen Eintrag *Ulix-Devel* in der linken Leiste des VirtualBox-Fensters. Es erscheinen verschiedene Hinweisfenster, in denen Sie jeweils die Option markieren, dass diese Hinweise nicht erneut erscheinen sollen; dann schließen Sie die Hinweisfenster.

c) Das Booten der VM dauert je nach Rechner zwischen einer halben Minute und mehreren Minuten; der Vorgang ist abgeschlossen, wenn Sie einen Login-Dialog im Grafikmodus sehen. Geben Sie hier als Benutzername `ulix` und als Passwort auch `ulix` (jeweils komplett in Kleinbuchstaben) ein. Der Bootvorgang ist nur beim allerersten Start erforderlich; um die VM auszuschalten, klicken Sie auf die Schließen-Schaltfläche des Fensters und wählen die Option *Save the Machine State*. Später wird dann beim Reaktivieren der VM der letzte Zustand wiederhergestellt, vergleichbar mit Suspend-to-disk bei einem PC.

d) Die Statusleiste am unteren Rand des Desktops bietet links ein Startmenü, vergleichbar Windows 95 oder XP, das Sie aber in der Regel nicht brauchen werden. Klicken Sie auf das direkt rechts davon sichtbare Konsolen-Icon: Damit öffnen Sie ein Terminalfenster, in dem Sie Befehle eingeben können.

e) Testen Sie zunächst mit `ping`, ob die Internet-Verbindung funktioniert (was voraussetzt, dass Ihr Notebook online ist): Geben Sie dafür in der Shell z. B.

```
ping ohm.hgesser.de
```

ein. Sie sollten Meldungen der folgenden Form erhalten:

```
ulix@ulixdevel:~$ ping ohm.hgesser.de
PING ohm.hgesser.de (217.160.135.96) 56(84) bytes of data.
64 bytes from hgesser.com (217.160.135.96): icmp_req=1 ttl=63 time=26.5 ms
64 bytes from hgesser.com (217.160.135.96): icmp_req=2 ttl=63 time=28.0 ms
…
```

Mit [Strg-C] brechen Sie die Ausgabe ab. Erscheint stattdessen nur eine Fehlermeldung, geben Sie die folgenden beiden Kommandos ein:

```
ulix@ulixdevel:~$ sudo /etc/init.d/networking stop
Deconfiguring network interfaces...done.
ulix@ulixdevel:~$ sudo /etc/init.d/networking start
Configuring network interfaces...done.
```

und probieren es dann erneut.

f) Die VM enthält ein Programm, das über meinen Webserver Ulix-Quellcode-Dateien aktualisieren kann. Testen Sie, ob es funktioniert, indem Sie das Kommando `update-ulix.sh` eingeben:

```
ulix@ulixdevel:~$ update-ulix.sh
Ulix-Update
Keine Updates verfuegbar.
```

Sie sollten die oben dargestellte Ausgabe erhalten, weil es (heute) noch keine Updates gibt.

g) Im nächsten Schritt testen Sie, ob Sie die vorinstallierten Ulix-Quelldateien kompilieren und die virtuelle Maschine starten können. Das geht mit den folgenden Kommandos:

```
ulix@ulixdevel:~$ cd ulix
ulix@ulixdevel:~/ulix$ make
[….]
ulix@ulixdevel:~/ulix$ make run
```

Das letzte der drei Kommandos startet den PC-Emulator `qemu` und bootet nach kurzer Wartezeit (oder wenn Sie [Eingabe] drücken) die Ulix-VM. Hier erscheinen in einem separaten Fenster wenige Bootmeldungen, danach sehen Sie einen Shell-Prompt.

h) Jetzt können Sie Ulix ausprobieren; testen Sie z. B. die Kommandos `ls`, `ps`, `hexdump sh` und `fork`. Mit [Alt-1] bis [Alt-5] wechseln Sie zwischen fünf virtuellen Konsolen (tty0 bis tty4) hin und her, in denen jeweils ein Shell-Prozess läuft.

i) Werfen Sie auch einen Blick in das Terminalfenster, aus dem heraus Sie die VM gestartet haben: Es zeigt diverse Debug-Ausgaben an, wenn Sie im Ulix-Fenster etwas eingeben oder einen Befehl ausführen. Diese Debug-Ausgaben landen auch in der Datei `ulix.output` im Unterverzeichnis `bin-build/`. Um die Ulix-Maschine (nicht die VirtualBox-VM) auszuschalten, drücken Sie im Terminalfenster [Strg-C], dann erscheint wieder der Shell-Prompt.

j) Schalten Sie jetzt die Entwicklungs-VM Ulix-Devel aus, indem Sie ihren Zustand sichern (siehe Aufbabe **c)**). Starten Sie dann die VM erneut – Sie sollten an der ursprünglichen Stelle landen. Das funktioniert auch denn, wenn innerhalb der VM gerade die Ulix-Maschine aktiv ist.

k) Werfen Sie einen Blick in die Ulix-Quellcodedateien: Es handelt sich um `ulix-book.nw`, `ulix-lib.nw` und `student.nw` im Verzeichnis `/home/ulix/ulix/` (in dem Sie sich noch befinden sollten). Sie können diese z. B. mit `gedit` öffnen. Probieren Sie das mit der Datei `student.nw`:

```
ulix@ulixdevel:~/ulix$ gedit student.nw
```

Der Editor zeigt den LaTeX-Quelltext mit Syntax Highlighting ein. Suchen Sie nach der Zeile

```
printf ("Initializing student's module.\n");
```

und ergänzen Sie als Test einen zusätzlichen Text. Speichern Sie die Datei mit [Strg-S] und beenden Sie den Editor mit [Strg-Q]. In der Shell übersetzen und starten Sie Ulix erneut mit `make` und `make run`. (Bei Änderungen sind immer beide Kommandos nötig; `make run` alleine löst kein Neukompilieren aus.) Die Änderung sollte nun in den Ulix-Bootmeldungen sichtbar sein.

l) Werfen Sie abschließend einen Blick in die Unterordner von `/home/ulix/ulix/`: Hier gibt es

– `bin-build/`: Hier landen die `*.c`- und `*.asm`-Dateien mit dem eigentlichen Ulix-Sourcecode, und hier läuft auch der Compiler.

– `lib-build/`: Die Bibliothek für die User-Mode-Programme wird hier entpackt und gebaut.

– `tex-build/`, `libtex-build/` und `student-build/`: In diese Ordner extrahiert das `make`-Kommando (wenn Sie es mit `make pdf` aufrufen) die LaTeX-Dateien aus den Noweb-Dateien (*.nw) und erzeugt die PDF-Dateien, die anschließend als `ulix-book.pdf`, `ulixlib.pdf` und `student.pdf` im „Hauptverzeichnis“ landen. Die Datei `student.pdf` wird später das Ergebnis Ihrer Arbeit enthalten.

– `mountpoint/`: Hier mountet das Makefile das Disketten-Image `bin-build/minixdata.img`.

Glückwunsch: Ihr Rechner ist nun bereit für die Arbeit mit dem Ulix-Quellcode!