1. Experiments with the Ulix Devel VM

In the first exercise, you will install the virtual machine and learn about the directory structure for Ulix development.

a) Install a suitable version of VirtualBox, then play the VirtualBox expansion *Oracle_VM_VirtualBox_Extension_Pack-4.3.24-98716.vbox-extpack* and then install the Ulix Devel VM from the file *FOM-Debian-Ulix-2015.ova*. In the last step, a dialog opens that provides information about the planned import of the VM. Do not (!) Check the option *Assigning new MAC addresses for all network cards.*

b) When the import is complete, start the VM by double-clicking the new entry *FOM-Debian-Ulix* in the left bar of the VirtualBox window. Various message windows appear in which you mark the option that these messages should not appear again; then close the notification window.

c) Booting the VM takes between half a minute and several minutes, depending on the computer; the process is complete when you see a login dialog in text mode. Enter here as a username ulix and as a password too ulix (each completely in lowercase

ben). The boot process is only required for the very first start; to turn off the VM, click the window's close button and select the option *Save the machine state.* Later, when the VM is reactivated, the last status is restored, comparable to suspend-to-disk on a PC.

d) Give startx to activate the graphical user interface. The status bar at the bottom

The edge of the desktop offers a start menu on the left, comparable to Windows 95 or XP, but which you will usually not need. Click on the console icon visible directly to the right of it: This opens a terminal window in which you can enter commands.

e) First test with ping, whether the Internet connection works (which requires that your notebook is online): Enter e.g. B.

ping ohm.hgesser.de

one. You should receive messages in the following form:

ulix @ fomdebian : ~ $ **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
...

You can cancel the output with [Ctrl-C]. If only an error message appears instead, enter the following two commands:

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

and then try again.

f) The VM contains a program that updates Ulix source code files via my web server.
can ren. Test if it works by running the command update-ulix.sh enter:

ulix @ fomdebian : ~ $ **update-ulix.sh**
Ulix update
Script download ok, no updates available.

You should get the output shown above because there are no updates (today).

G) The next step is to test whether you are compiling the preinstalled Ulix source files and using the can start the generated kernel in the PC emulator. This can be done with the following commands:

ulix @ fomdebian : ~ $ **cd ulix**
ulix @ fomdebian : ~ / ulix $ **make**
[….]
ulix @ fomdebian : ~ / ulix $ **make run**

The last of the three commands starts the PC emulator qemu and boots after a short waiting time (or if you press Enter) the Ulix VM. A few boot messages appear in a separate window, after which you will see a shell prompt.

H) Now you can try Ulix; test e.g. B. the commands ls, ps, hexdump, sh and fork. Use [Alt-1] to [Alt-5] to switch between five virtual consoles (*tty0* to *tty4)* back and forth, each with a shell process running.

i) Also take a look at the terminal window from which you started the VM: It shows various debug output when you enter something in the Ulix window or execute a command. This debug output also ends up in the file ulix.output in sub-drawing bin-build /. To switch off the Ulix machine running in Qemu (not the VirtualBox VM), press [Ctrl-C] in the terminal window, the shell prompt then appears again.

j) Now turn on the development VM *FOM-Debian-Ulix* by saving their status (see **c)).** Then restart the VM - you should end up in the original location. This also works if the Ulix machine is currently active within the VM.

k) Take a look at the Ulix source code files: they are the files ulix-book.nw, ulixlib.nw and student.nw in the register / home / ulix / ulix / (in which you yourself should still be located). You can use this e.g. B. with gedit to open. Try that with the file student.nw:

ulix @ fomdebian : ~ / ulix $ **gedit student.nw**

The editor shows the LaTeX source text with syntax highlighting. Look for the line

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

and add an additional text as a test. Save the file with [Ctrl-S] and exit the editor with [Ctrl-Q]. Compile in the shell and start Ulix again with make and make run. (Both commands are always required for changes; make run alone does not trigger a recompilation.) The change should now be visible in the Ulix boot messages.

l) Finally take a look at the subfolders of / home / ulix / ulix /: Here there are

- bin-build /: This is where the *. c- and *. asm- Files with the actual Ulix source code, and this is where the compiler runs.
- lib-build /: The library for the user mode programs is unpacked and built here.
- tex-build /, libtex-build / and student-build /: The make-Command (if you use it with make pdf call) the LaTeX files from the Noweb files (*. nw) and creates the PDF files, which are then saved as ulix-book.pdf, ulixlib.pdf and student.pdf land in the "main directory". The file student.pdf will later contain the result of your work.
- mountpoint /: This is where the Makefile mounts the floppy image bin-build / minixdata.img.

Congratulations: Your computer is now ready to work with the Ulix source code!