2. LaTeX

In this exercise you will create a small LaTeX document and create a PDF file from it.

a) Open a terminal window and use to create

```
mkdir tex; cd tex
```

a new directory tex, into which you switch directly. Then start by entering

gedit & the text editor; You are editing an empty, unnamed file, and the appended "&" sign keeps the shell usable.

First, familiarize yourself with the keyboard shortcuts you need in the Ulix Devel VM to use the characters \ (backslash), {and} (curly braces) and [and]
(square brackets) - you will need this frequently when editing TeX files.

If you have installed the VM under Windows or Linux, you should be able to access the five special characters using the usual key combinations.

Mac OS users can use the following shortcuts:
[Alt + 8] = [, [Alt + 9] =], [Alt + 7] = {, [Alt + 0] =}, [Alt + ß] = \. (All combinations respectively
with the right (!) Alt key.) Also interesting: [Alt + <] = |, [Alt + Q] = @.

b) Now enter the following four lines in the terminal window:

```
\ documentclass {article}
\ usepackage [utf8] {inputenc}
\ begin {document}
\ end {document}
```

Save the file by pressing [Ctrl + S] and entering the filename e.g. B. practice02.tex
specify. Syntax highlighting should be activated immediately after saving, as the editor now knows that you are editing a TeX file.

Add between \ begin {document} and \ end {document} a little text; use
You also use German umlauts. In the shell you can now (after saving again with [Ctrl + S]) with the command

```
pdflatex exercise02.tex
```

from the tex- a pdf- Make a file and then add it

```
xpdf exercise02.pdf
```

view in PDF viewer. (You exit the viewer with [Ctrl + W].)

Then try out the design options presented in the lecture: In particular, test the section numbering (with \ section, \ subsection and \ sub-
subsection), the text markups in bold and italics (\ textbf {...}, \ emph {...}) as well as numbered lists and bullet lists (via enumerate- and itemize- Environments).

If you start the PDF viewer with the & symbol attached, the terminal can still be used. After "compiling" the tex- File with pdflatex recognizes the PDF viewer
automatically that the PDF file has changed and updates the view. When you are finished,

close the editor and PDF viewer windows.

3. Literate programming: reading and "tangling" code

Call the script update-ulix.sh on - in the folder tex / you will then find a subfolder WC/ with two files (wc.pdf and wc.nw). If the script does not work, you will find the two the files at http://faq.ktug.org/wiki/pds/noweb/wc.pdf and http://ohm.hgesser.de/ulix/wc.nw.

a) Change to the folder tex / wc / in your home directory and read the PDF file:
As a Literate Program, it describes the implementation of the Unix tool toilet (word count), the Returns the number of characters, words and lines for one or more given file (s). The Literate Program uses a *Top down* or one *Bottom up* Approach?

b) Now "tangle" (read: "tangle") the compilable source code: Enter the command

```
notangle wc.nw> wc.c
```

and then open the C source code file extracted from the Literate Program wc.c in the editor - compare the code with the Literate Program. In the above call to notangle you didn't have to enter a code chunk, because there is a chunk in the file called •*• whose definition is on page 2 of the PDF file: This is the "standard chunk" if you do not have any - R. *chunkname* specify.

Can you still tell from the C file whether the program *top-down* or *bottom-up* was developed?

c) Also try out the option to generate line numbers, that goes with

```
notangle -L wc.nw> wc.c
```

Again, look at wc.c in Notepad. You will now find numerous lines of the form # line 111 "wc.nw": These lead to compilation errors with the C compiler (gcc) not the number of the faulty line in the C file, but the one in the nw- File is output.

d) Compile the program by entering gcc wc.c - a warning appears (*wc.nw: 146: warning: incompatible implicit declaration of built-in function 'exit').* Check that in the file wc.nw the command exit (status); is on line 146. You can (despite the Warning) test the generated program by clicking

```
. /a.out wc.nw
```

(with leading point and slash) enter; You should get the following output:

```
    377        1895       12405 wc.nw
```

4. Convert code into a literate program

Now it's time to try out literate programming for yourself. For this you can use a classically programmed and documented C source file shell.c use that automatically in the folder tex / ended up when you did the script in Task 3 update-ulix.sh have called.

a) Read and understand the program that implements a small shell.

b) Convert the program to a Literate Program shell-lp.nw. To do this, break it down into suitable chunks that you document well. Classic C comments shouldn't be left over. A template file shell-lp.nw is already prepared.

c) "Tangle" out (as in exercise 3 b and c) shell-lp.nw the C file shell-lp.c and match this with the original.

d) Create with noweave -index -delay shell-lp.nw> shell-lp.tex the TeX file, then call twice pdflatex shell-lp.tex and watch the result with xpdf shell-lp.pdf.