Introduction to Mathematical Software 1st Exercise Sheet

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Mathematics
PD Dr. Ulf Lorenz
Christian Brandenburg

winter term 2009/2010
26/10/2009

Exercise 1.1 The Linux Terminal

Terminal (or shell) windows allow you to directly work with the file system of your computer. This is done by typing commands to perform specific tasks. An overview of the most important Unix/Linux commands are summerized on the following command reference. A more detailed introduction is given by the UNIX TOOLBOX. These sources should offer you most of what you need to know about Linux commands.

Now, get familiar with the Linux terminal by opening a terminal emulator (e.g. `Konsole`, which can be found in the *K*-menu under `System -> Konsole` and performing the following tasks. Perform all tasks inside the terminal window without using other tools like browsers etc!

a) List the contents of your `home` directory (the directory in which all your personal data is stored). If you are not actually there, change to that directory.

b) Create a subdirectory named `exercise1` .

c) Download the file `textfile`, which is located at `http://www3.mathematik.tu-darmstadt.de/fileadmin/` into your home directory (without using a browser!). Check that the file has correctly been downloaded by listing the contents of your home directory.

d) Find out what the command `less` is good for. (Hint: What does the command `man` do?) Look into the file `textfile` with `less`.

e) Copy the file `textfile` to the directory `exercise1`.

f) Change to the directory `exercise1` and check that the file `textfile` is actually there.

g) Move the file `textfile` to the file `textfile.txt`.

h) Change back to your home directory and delete the file `textfile`.

i) Find out how much disk space you can use at most. How much disk space does your home directory and all its subdirectories use? *Hint:* What does the option `-h` do?

j) Determine the absolute path of your working directory. List the whole contents of your home directory and find out which access rights the files in your home directory have. Does the file `.bash_history` exist? Find out which subdirectories there are in your home directory. List all files in our home directory, sorted by date.

If you have further questions or problems concering the use of Linux you can also ask your tutors in their office hours. Furhtermore, there is help available in the Mathematics department. See the webpage http://www3.mathematik.tu-darmstadt.de/index.php?id=350 for more information.

Exercise 1.2 Text Editors

The task of this exercise is to get familiar with a text editor that is suitable for programming. The editor we recommend for you to use is called *Kate*, which is part of the *KDE desktop environment* (of course, you can use any other text editor you like, but you should make sure that it at least supports *syntax highlighting*).

To start *Kate*, select `Debian -> Applications -> Editors -> Kate` in the *K*-menu.

a) Type the `Hello World` program (Example 2 on page 18 of part 1 of the Lecture Slides) into the editor window and save the program as `hello.c` in the `exercise1` folder that you have created in **G1**. After saving your program observe how different statements in your program are highlighted by color and/or bold-face. Kate recognizes from the ending `.c`, that your file contains a C-program and highlights the syntax accordingly.

b) Kate contains its own terminal window. If it is not open, open it by pressing the `Terminal` button at the bottom of the Kate window or by selecting `Window -> Tool Views -> Show Terminal` from the menu bar. Make sure that the terminal is in the correct directory by listing the contents of the directory and checking for your `hello.c` program.

c) Now, compile your program by typing `gcc hello.c -o hello` in the terminal window. If there don't appear any error messages in the terminal window, run your program by typing `./hello` in the terminal window. The text `Hello World` should appear. If there are error messages, check your program source code in the editor window to make sure you have no errors in your code.

d) Modify your program by replacing the space between `Hello` and `World` in turn with the characters `\n` , `\t` , `\b` , `\f` , `\\` , `\"` , and `\'` .

   In each case, recompile the program. What happens?

Exercise 1.3 Integer Arithmetic

Type in the following program:

```
#include <stdio.h>

int main(void)
{
  short i1 = 11, i2 = 22, i3;
  i3 = i1 + i2;
  printf("i1, i2, i3 = %d  %d  %d \n", i1, i2, i3);
  return(0);
}
```

Save the program into the directory `exercise1`. You can choose a name for the program, but mind the ending `.c`.

a) Compile and run the program. What happens? Is this the result that you expected?

b) Change the line `i3 = i1 + i2;` to `i3 = i1 * i2;` and recompile your program. What happens? Is this the result that you expected?

c) Change the line `i3 = i1 + i2;` to `i3 = i1 / i2;` and recompile your program. What happens? Is this the result that you expected?

d) Change the line `i3 = i1 + i2;` to `i3 = i2 / i1;` and recompile your program. What happens? Is this the result that you expected?

e) Change the line `short i1 = 11, i2 = 22, i3;` to `short i1 = 11111, i2 = 22222, i3;` and repeat the steps a) to d). What happens? Is this the result that you expected?

Exercise 1.4 Floating Point Arithmetic

Type in the following program (you can use your program of **Exercise 1.3** as a starting point):

```
#include <stdio.h>

int main(void)
{
  float x1 = 1.0e5, x2 = 3.1415926, x3;
  x3 = x1 + x2;
  printf("i1, i2, i3 = %f  %f  %f \n", x1, x2, x3);
  return(0);
}
```

Save the program into the directory `exercise1`.

a) Compile and run the program. What happens? Is this the result that you expected? How accurately is `x3` computed?

b) Change the line `x3 = x1 + x2;` to `x3 = x1 * x2;` and recompile your program. What happens? Is this the result that you expected? How accurately is `x3` computed?

c) Change the line `x3 = x1 + x2;` to `x3 = x1 / x2;` and recompile your program. What happens? Is this the result that you expected? How accurately is `x3` computed?

d) Change the line `x3 = x1 + x2;` to `x3 = x2 / x1;` and recompile your program. What happens? Is this the result that you expected? How accurately is `x3` computed?

e) Change the line `x3 = x1 + x2;` to `x3 = x1 % x2;` and recompile your program. What happens? Is this the result that you expected? How accurately is `x3` computed? (`%` is the modulo operator in C).

f) Change the line `float x1 = 1.0e5, x2 = 3.1415926, x3;` to `float x1 = 1.0e25, x2 = 1.0e10, x3;` and repeat the steps a) to e). What happens? Is this the result that you expected?