Programmierpraktikum

Exercise Sheet 2

SS, 2012

Please bring to the class the code you generated for the last exercise. For the rest of the exercise, you should only show that you have tried the exercises and understood them, as in the first class.

Connect to another computer

- Log into another machine in the lab. Find out the name of your lab computer (use `hostname` and `domainname`). Go to another machine and then log to the first machine using ssh (`ssh username@machine.domain`). You can then work in the other computer.
- Save one step and run a command in the remote computer via `ssh username@machine.domain command`. See what happens if you end the connection while the program is running. Try with a graphical program, for instance firefox (you need an extra option for that, see `man ssh`).

Some useful `GNU/linux` programs and commands (Optional)

Chances are, there are many useful utilities already installed in your linux distribution. Take a look at the following commands:

- `firefox www.uni-frankfurt.de`
- `inkscape` (vector graphics)
- `gimp file.jpg` (download some file from the internet and open it with gimp to modify it)
- `lowriter document.doc` (office)
- Open a graphical interface to your `~/Documents` directory: `nautilus ~/Documents/` (only if you are using gnome, in kde would be `konqueror ~/Documents/`)
- Tired of having this program use your terminal? Press `ctrl+z` to pause it, then send it to background with `bg`. You can also do that by adding `&` at the end of the command when you use launch it.
- Now getting stuff sent from the background program to the terminal? Redirect its output to the "oblivion" (the null device `/dev/null`). `nautilus ~/Documents/ 2>/dev/null &`
- Advanced: Check the command `awk`. This is very useful but a language on itself. See `man awk`.

Making plots with Gnuplot

This is a very useful plotting program, take a look at http://www.gnuplot.info/ or `man gnuplot`. Once you run it, it opens a new command line. You can now plot functions and, more importantly, data. For instance, try:

- `plot [0:10] x`
- `plot [1:10] cos(x), sin(x), log(x)`
- `splot [-10:10][-10:10] exp(-x**2-y**2)`
- Generate a text file (in the same directory where you are run gnuplot) with two columns, both with numbers. Then run `plot 'filename'` to plot this data. Also try `plot 'filename' with lines`. To generate the file, you can start from command line a text editor, via e.g.
  `gedit filename`
  or
  `vim filename`
  and then simply write the numbers as columns (separate the columns simply by spaces or tabs).

Compiling and running a simple java program

Now we are up to compile and run a simple Java program. We will use a program whose only function is to output one sentence. You can find the code in the lessons:
http://itp.uni-frankfurt.de/~gros/Vorlesungen/ProgPrak/Java-basics.html#(6)
You should save the code in a file named "HelloWorld.java". Notice the name of the file must be the same as the name of the class, so you shouldn't use another name.

- To compile, make use of the `javac` command. This command is a *compiler* for java code, that is, it will read your Java file and translate it into machine code (in the case of Java, it is rather bytecode for its virtual machine). The file that it generates says basically the same instructions that you wrote in Java, but in a language that the (virtual) machine can "understand".

  ```
  javac HelloWorld.java
  ```

  If it doesn't output anything, it most probably means the compilation was successful. To check this, a file `HelloWorld.class` should have been created in the same directory.

  Now you can try to run your program. In the directory where the file is try to run the class "HelloWorld" via

  ```
  java HelloWorld
  ```

  Notice you are not calling the file, but the class which you have just compiled. Then java will automatically call the `main` method of the class `HelloWorld`, which will instruct the computer to print the phrase to the console.

- Now that we know how to compile the program, we will make a modification. Modify the code in the file "HelloWorld.java" to print all the odd numbers smaller than 40. Use a `for` loop for this, check the code found in:
  http://itp.uni-frankfurt.de/~gros/Vorlesungen/ProgPrak/Java-basics.html#(7)
  as an example. As in said code, you would like use a loop that will increment the value of a variable (the variable `index`, in that case). This variable should be printed only when it is odd and smaller than 40. You can achieve that by modifying the parameters of the `for` loop.