UNIVERSITÄT BASEL

Lecturer: Thorsten Möller - **thorsten.moeller@unibas.ch**
Tutors: Andreas Wassmer - **andreas.wassmer@unibas.ch**
Maximilian Barth - **m.barth@unibas.ch**

Programming Paradigms – Setup FS 2022

Tutorial: Setting up the Environment Session: 01.04.2022

Note: Attendance at the exercise session on the 1st of April is not mandatory and you will not have to hand in or show any part of this tutorial. However, you will need a working C++ compiler and working interpreters for Haskell, Prolog, and Python to be able to successfully complete the exercises. If you have any questions or need help with installation you are recommended to take part in the tutorial session or otherwise send an email to **andreas.wassmer@unibas.ch** or **m.barth@unibas.ch**.

Question 1: Team (0 points)

Please define your working groups on Adam to ensure group-uploads of the exercise solutions. Please make sure that you hand in the file to Adam as a team.
All future hand-ins must have the name of both contributors on it.

Question 2: Introduction (0 points)

This is a quick guide for installing the tools required in this course. You are free to use the environment you prefer most. This introduction further explains what is needed to install the tools on your system.

a) VM: optional isolated development environment (Note: Despite advantages, this is purely optional, the necessary tools work on all tested systems: Win x86-32/64, Mac x86-64, Mac ARM 64, Linux 32/64.)

   Contrary to the following options that install the development environment into the live OS, we present here an option to install the environment into an isolated system, such that it is free of conflicts with its external environment or complex situation-specific requirements. Furthermore, this environment can be backed up or (re)moved easily.

   For this solution, we recommend Oracle Virtual Box (https://www.virtualbox.org/) for all users with an x86 CPU. For Windows und Mac ARM (M1 chip) users, we recommend QEMU to emulate a x86 VM (https://www.qemu.org/).

   Alternatively, there are also pre-built variants (with certain limitations) found with

tools like Vagrant VMs (https://www.vagrantup.com/downloads), Windows Subsystem for Linux or Cygwin that could work as well (not recommended).

To setup your own VM, a mini Linux setup is sufficient; e.g., a Fedora/Debian Core (terminal only) or AntiX, Rufus, Etcher (Mini-Linux), or even smaller with Alpine-Virtual.

For a VM that includes a desktop (graphic user interface), the virtual disk should have a size of about 10GB or more, and at least 2GB RAM. For a terminal-only VM, a 5GB disk with <1GB RAM should suffice.

To simplify the data exchange, we also recommend shared folders or bidirectional drag and drop.

Examples of detailed tutorials can be found here:

https://www.howtoforge.com/tutorial/debian-minimal-server/
or
https://www.howtoforge.com/tutorial/debian-minimal-server/
or
https://pqsec.org/2021/01/05/intall-debian-ubuntu-in-qemu.html.

b) Linux

For Linux the installed package manager should provide everything needed.
Packages: `gcc`, `g++`, `python3` (version ≥3.9), `ghc`, `swi-prolog`

c) macOS

For macOS we suggest that you install *Homebrew*. In a nutshell, Homebrew is an easy-to-use package manager, similar to package managers known on Linux platforms. To install it, execute the following command in the Terminal:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

You can also visit the official Web site `https://brew.sh` to copy this command or to learn more if you want to install Homebrew manually.

d) Windows

For Windows we recommend the *Cygwin* environment. Cygwin provides a UNIX-like environment for Windows. Download and launch the installer `https://www.cygwin.com`. During the installation process you will be presented a list with all available packages to install. After the installation you can rerun the installer to add additional packages.

e) IDEs

We recommend using the compilers and interpreters mentioned in this document without the use of a real IDE. However, you may write your code using whichever editor or IDE you prefer, as long as the final code compiles/runs with the required compilers/interpreters.

Question 3: C++ (0 points)

This section explains how you can install the *GNU Compiler Collection* (GCC, `https://gcc.gnu.org`).

a) Linux

Run the following command in a terminal:

```
sudo apt-get install build-essential
```

b) macOS

```
brew install gcc
```

Alternatively, you might want to use *XCode* instead (which is a professional IDE that can be used for much more than only C/C++ programming). It is technically possible to install the command line tools without XCode, but an XCode installation is advised. After installation is complete, open XCode at least once to complete first time setup. Once initial setup has finished, open a terminal window and run:

```
xcode-select --install
```

Follow the instructions in the window that pops up to install the command line tools, which include the `g++`/`gcc` compiler.

c) Windows

Find the `gcc-g++` package in the cygwin installation and install it.

To test your installation, you can use the following hello world program:

```
// hello.cpp
#include <iostream>
int main(int argc, const char** argv)
{
    std::cout << "Hello C++ World!" << std::endl;
}
```

On a terminal shell, compile it using `g++ hello.cpp -o hello` (where `hello.cpp` is the source file and `-o hello` specifies the name of executable to be generated).

Then, run it using `./hello` .

Question 4: Haskell (0 points)

For the Haskell exercises, we will use the *Glasgow Haskell Compiler* (GHC, `https://www.haskell.org/ghc/`). It is sufficient to only install the compiler `ghc` itself.

If you like, you can also install the entire *Haskell Platform*. It contains additional tools, such as *Cabal* and *Scion-Browser*.

a) Linux

```
sudo apt-get install ghc
or
sudo apt-get install haskell-platform
```

b) macOS

```
brew install ghc
or
brew install haskell-platform
```

c) Windows

The *Haskell Platform* can be downloaded from the Haskell homepage (`https://www.haskell.org`). The binaries should be automatically linked to your Cygwin environment.

To test your installation, you can use the following hello world program:

```
-- hellohaskell.hs
module Main where

main::IO()
main = putStrLn "Hello, Haskell World!"
```

Start it directly using the Haskell interpreter: `ghci hellohaskell.hs`. This gives you an interactive Haskell command prompt, where you would type `main` to launch the hello world program. The command prompt can be closed using CTRL-Z.

Alternatively, you can also compile it into a regular executable using `ghc hellohaskell.hs -o hellohaskell`. In that case, launch it directly using `./hellohaskell`.

Question 5: Prolog (0 points)

For Prolog, we use the *SWI Prolog* (`http://www.swi-prolog.org`) implementation.

a) Linux

```
sudo apt-get install swi-prolog
```

b) macOS

```
brew install swi-prolog
```

c) Windows

Find the `swi-prolog` package in the Cygwin installation.

To test your installation, use the following hello world program (note that the file name must start with a lowercase letter):

```
% helloprolog.pl
hello :- write("Hello Prolog World!").
```

Depending on your system, you can now use the `swipl` or the `pl` command to open an interactive Prolog interpreter (console). Inside the console, run the hello world program using the steps below. Note that the periods (`.`) are also part of the commands.

1. Type `consult(helloprolog).` to load the `helloprolog.pl` application.
2. Type `hello.` to call the hello procedure.
3. Finally, close the interpreter using `halt.` or by pressing CTRL-Z.

Question 6: Python (0 points)

For Python, we use Python 3 (`https://www.python.org`). You should use version 3.9 or newer.

a) Linux

Python 3 should be preinstalled.

b) macOS

```
brew install python@3.9
```

c) Windows

Find the `python` package in the Cygwin installation, or download the installer from `https://www.python.org`.

To test your Python installation, start the interpreter from the terminal using the command `python3` and enter:

```
print("Hello World!")
```

To exit the Python interpreter enter `quit()` or press CTRL-Z.

Congratulations, you are now ready to start with the exercises!