Prof. Dr. Peter Bastian, René Heß

IWR, Universität Heidelberg

Submission date: -

Exercise 1 *Setting up DUNE (practical exercise)*

The goal of this exercise is to set up your DUNE installation and to get familiar with the framework's fundamental structure.

SSH access to the computer pool Since exercises with personal presence are not possible this semester we will use the computer pool in a remote way through ssh. Open a terminal and type

```
ssh username@pool.iwr.uni-heidelberg.de
```

The `username` and `password` will be provided during the first exercise. Next, open a terminal and type the command

```
passwd
```

and follow the instructions to choose a personal password for your account.sting

Downloading DUNE Navigate to your home directory using a terminal. There you find a script named *installer.sh*.

Execute it via the following command:

```
./installer.sh dune
```

This will create a directory named *dune* and download all the software components required for this lecture.

Note: You could pass a different directory name to *installer.sh*. This may be particularly useful if you would like to maintain multiple independent installations of DUNE.

If you would like to work on your own Linux system, you can obtain a copy of the installer script from the lecture's homepage. The steps are identical, however you may have to install certain dependencies (compilers, linear algebra libraries etc.). If that is the case, error messages will point out what's missing.

Compiling DUNE Once the downloads have finished, navigate to the newly created directory *dune*. Inside, you find several DUNE modules and a script named *buildmodules.sh*. In order to compile DUNE, execute the following command:

```
cd dune
./buildmodules.sh
```

This process may take a few minutes. While the console output rushes by, stare at the terminal and pretend to be *Neo* from the movie *The Matrix*. In case you manage to enter *The Matrix*, please defeat the evil machines and save mankind. Otherwise, please proceed with the exercise.

The structure of DUNE DUNE is organized in various modules providing different functionalities. For example, *dune-istl* provides various preconditioners and solvers for linear systems of equations, while *dune-grid* provides grid implementations that we will use for discretizing domains.

In fact, each project using DUNE is itself a DUNE module. During this lecture, you will work on the exercises in the module *dune-parsolve*. Navigate to its directory (*dune/dune-parsolve*). Inside, you find a file named *dune.module*. This file defines the properties of our module.

Open it and look for a line starting with *Depends*. As you can see, in addition to the aforementioned modules, *dune-parsolve* also depends on the DUNE module *dune-pdelab*.This module provides methods for discretizing partial differential equations, and we will use many of its components in later exercises.

Further, the module *dune-parsolve* contains a directory *uebungen*, which contains the source codes you will be working on during the practical exercises.

How to hack As mentioned above, you find the relevant source codes in the *uebungen* directory. The compiled programs, however, are generated by the build script in an entirely separate directory tree. This separation ensures that your sources remain clean and unaffected by the build process.

You can find the build directory (named *release-build*) in your *dune* directory. Inside, you find a directory structure resembling that of your DUNE sources. Only, instead of the source code, you find the compiled binaries ready to run. For example, you can navigate to the *dune/release-build/dune-parsolve/uebungen/uebung01* directory and run the binary *uebung01*.

Whenever you make changes to your source code and want to test the result, you need to recompile before running it. The lazy, but somewhat slower way is to simply run the build script from above again. However, if you only make changes to a specific source directory, e.g. *dune-parsolve/uebungen/uebung01*, it is usually enough to run *make* in the associated build directory. The latter is faster, as it does not check all other modules for changes.

Doing stuff on your own Finally it is worth noting that the two scripts we used above are designed to easily set up everything exactly according to the requirements of this lecture. Later, when working on DUNE on your own, you can create your own DUNE modules using the tool *dune-common/bin/duneproject* and compile arbitrary DUNE modules using *dune-common/bin/dunecontrol*. More details on this can be found on *dune-project.org*.

(0 Points)

Exercise 2 *SSH Tips*

This exercise provides some tips for working with ssh.

- Logging in:

```
ssh parso0XXpool.iwr.uni-heidelberg.de
```

- Copying files from the local machine to remote:

```
scp test.txt parso0XXpool.iwr.uni-heidelberg.de:
```

- Copy files from remote to the local machine:

```
scp parso0XXpool.iwr.uni-heidelberg.de: /installer.sh .
```

- Add the option `-r` to copy folders.

- Instead of always typing the password it might be more comfortable to use a ssh key. If you don't have one you can create it by doing

```
ssh-keygen
```

  on your local machine. You can copy the public key to the remote machine through

```
ssh-copy-id -i ~/.ssh/id_rsa.pub parso0XXpool.iwr.uni-heidelberg.de
```

  Afterwards you only need the password for your ssh key to access the server.

- If you have problems with disconnections or if you want to run something while you disconnect you can start the program GNU screen on the remote server:

```
screen
```

  Search the internet for detaching and reattaching to find out more.

- Many editors can edit files on remote servers, that means you can start the editor on your local machine and open the file through ssh. For compilation and running you still need to access the remote machine.

(0 Points)