Universität Göttingen
Department of Computer Science

Julian Kunkel

Exercise Week 1
HPDA / 2022
240 Minutes Total
Discussion in Week 2: 2022-11-07

---

1. The tasks described in this worksheet are part of the formative assessment. They serve the purpose to prepare you for the examination. We will discuss the solutions during the next **interactive session** after they are handed out – while they fit to the lecture of the week they are handed out, they might be discussed in two weeks time due to the bi-weekly exercise schedule.

2. Make sure to plan your time for the whole sheet carefully. The complete exercise should represent approximately three hours of independent study. The time limit indicates how much time you should spend on each task, and not how much time you may actually need; it is important that you engage with the material and not that you complete all tasks perfectly. Feel free to collaborate and team up.

3. The exercises are designed to challenge you and train you further as guided self-study. The time limit might be too ambitious for you; you may team up with colleagues. It is not an issue as long as you manage to at least partially resolve each task within the time budget. If you (and your team) are struggling, reach out for help in Teams! You may also share your thoughts via the Studip Forum.

4. We recommend that you create a (private) Git repository (see `https://gitlab.gwdg.de`) where you store your findings and outcomes while processing the exercises. This portfolio of work can be useful in the future.

Contents

Task 1: Introduction: Understanding HPDA Use Cases (60 min)

Being able to describe typical HPDA use cases is important. In this task, you will research two different use cases.

Tasks

1. Research one recent HPDA use case from industry in the literature, e.g., using Google or Google Scholar. Describe the use case briefly, put an emphasis on performance relevant characteristics, e.g., how much data do they process, how long does the workflow run? Address how the Big Data Challenges (5V) apply. Make sure you cite the used literature properly! Your notes should cover about 1/4-page.

2. Research a scientific HPDA use case following the same instructions as above.

Portfolio (directory: `1/performance`)

`1/performance/hdpa-use-case-industry.txt`	The use case from industry.
`1/performance/hpda-use-case-science.txt`	The use case description from science.

Task 2: Bash Exercises (60 min)

Supercomputers and distributed computers commonly run Linux. This exercise aids to provide an introduction to bash programming, a shell scripting language supported on most Linux distributions. If you are already proficient with bash, please do the (harder) follow-up task instead.

We don't expect for you to be writing long and complex bash scripts straight away. We also respect that moving to a CLI (Command Line Interface), while powerful, can be challenging, especially when you are accustomed to interacting with primarily graphical interfaces on operating systems such as Windows.

Below we have included several Bash commands of varying complexity that also cover more advanced concepts of Bash. We want you to experiment with them, try to run them in a Linux environment and observe the results! Think about what you expect their output to be before you run them, make your own changes to the commands and use this as an opportunity to learn about the shell and to become more comfortable writing your own scripts.

We would like you to produce a text file providing a short descriptions of your findings, i.e. all you need to provide is a brief description of what the command does (if possible write what you expected).

Run the following command **BEFORE** beginning the exercise, this will create a directory and switch into it:
`$ mkdir -p $HOME/exercise1 && cd $HOME/exercise1`
You should complete the steps below in this directory.
To terminate a running command early, you can press **CTRL + c** on your keyboard.

1. `$ echo "Hello World"`
2. `$ echo Hello, World`
3. `$ echo Hello, world; Foo bar`
4. `$ echo Hello, world!`
5. `$ echo "line one";echo "line two"`
6. `$ echo "Hello, world > readme"`
7. `$ echo "Hello, world" > readme`
8. `$ cat readme`
9. `$ example="Hello, World"`
10. `$ echo $example`
11. `$ echo '$example'`
12. `$ echo "$example"`
13. `$ echo "Please enter your name."; read example`
14. `$ echo "Hello $example"`
15. `$ three=1+1+1;echo $three`
16. `$ bc`

17. `$ echo 1+1+1 | bc`
18. `$ let three=1+1+1;echo $three`
19. `$ echo date`
20. `$ cal`
21. `$ which cal`
22. `$ /bin/cal`
23. `$ $(which cal)`
24. `$ ‘which cal‘`
25. `$ echo "The date is $(date)"`
26. `$ seq 0 9`
27. `$ seq 0 9 | wc -l`
28. `$ seq 0 9 > sequence`
29. `$ wc -l < sequence`
30. `$ for I in $(seq 1 9) ; do echo $I ; done`
31. `$ (echo -n 0 ; for I in $(seq 1 9) ; do echo -n +$I ; done ; echo) | bc`

Hints

- Use the up/down arrows to browse the history of the shell to find recently used commands.
- Use the left/right arrows (Pos1/End keys) to change the position of your text cursor.
- Hold CTRL while using the left/right arrows to skip entire words.
- Use the tab key to autocomplete commands when you’re halfway through typing them - it saves time!
- The middle mouse key copies previously marked text into the shell.
- Bash has certain special characters such as `;!<>` that can cause unexpected behaviour when included outside of quotes in a command.
- Use the command `$ help` and `$ man` to find out what a specific command does.

Portfolio (directory: `1/bash`)

`1/bash/findings.txt` An enumerated list that provides for each command a prose sentence what the commands did. Where your expectation differs from observed behaviour, include a sentence describing what surprised you. (At least 200 words).

Further Reading

- Shell scripting (also available as PDF) https://www.shellscript.sh/
- Shebang Wikipedia: https://en.wikipedia.org/wiki/Shebang_(Unix)
- Commands: https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners

- Command line structure: https://www2.cs.duke.edu/csl/docs/unix_course/intro-14.html
- Linux Filesystem Wikipedia: https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

Task 2: Bash Research (60 min)

This is a more difficult **optional** task which can be done instead of Task 2

We would like for you to produce a short (about 1 page, at least 300 words) report on the core concepts behind bash and show them in practice. Your report should give brief explanations and examples of the following concepts:

- Special characters and escaping
- Redirection
- Piping
- Variables
- Arithmetic
- Common Commands that you find useful `beyond our examples (ls, cd, mkdir, ...)`

You should include examples from the alternative (easy) task.

In your report you should consider circumstances in which easily avoidable errors could be encountered and subsequently avoided in your commands (for example, un-escaped special characters such as `;!#&`)

Portfolio (directory: `1/bash`)

`1/bash/report.txt` The 300 words report describing the concepts relevant to Task 1.

Further Reading

- Usage examples of many common bash commands: https://learnxinyminutes.com/docs/bash/
- Nice guide to how quotes work in Bash: https://www.gnu.org/software/bash/manual/html_node/Quoting.html
- Article going over basics and some common quirks in the language which can trip up beginners: https://jvns.ca/blog/2017/03/26/bash-quirks/
- Covers piping and its application: https://www.geeksforgeeks.org/piping-in-unix-or-linux/
- Advanced Bash-Scripting Guide: I/O Redirection: https://www.tldp.org/LDP/abs/html/io-redirection.html

Task 3: Setup of the software environment in a Virtual Machine (60 min)

To get started with a development environment and conduct all (formative) exercises, we suggest installing a Virtual Machine with Ubuntu 20.04 using VirtualBox[1] – if you have VMWare, it works fine too. Details about the usage of a VM are noted in a Google Doc here; several introductory videos are provided. Also, familiarize with the typical remote connection using SSH [2].

[1] https://www.virtualbox.org/

VirtualBox is a free virtualization environment allowing you to run virtual machines on your system (the host system).

Tasks

1. Familiarize yourself with SSH and create a private/public key-pair. You can use SSH on the terminal or via a graphical client, e.g., PuTTY on Windows. PuTTY comes with PUTTYgen for working with private/public key-pairs.
2. Download the Ubuntu 20.04[2] minimal ISO.
3. Install Ubuntu via VirtualBox. You may have to adjust the network settings (see Virtualbox manual).
4. Install Git and openssh-server via: `$ sudo apt install git openssh-server openjdk-11-jdk`
5. Allow SSH to connect from the virtual machine to itself: `$ ssh-keygen`
   Confirm the key creation, pressing enter. Do not enter any keyphrase here. Now, install the new key:
   `$ cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys`
   Test the setup by running: `$ ssh localhost`
   Confirm the SSH Fingerprint with yes. Now you can logout again using CTRL-D or by typing `$ exit`.
6. Connect via SSH from your Hostsystem to the virtual machine. Import the SSH public key in Linux to your `.ssh/authorized_keys` file.
7. Clone the repository for the course: `$ git clone https://github.com/JulianKunkel/hpda-samples.git`
8. Switch into the install-script directory `$ cd hpda-samples/install`
9. Install Hadoop via `$ ./hadoop.sh && source ~/.bashrc`
   You can now start and stop the Hadoop services via:
   `$ start-dfs.sh && start-yarn.sh`
   `$ stop-dfs.sh && stop-yarn.sh`
10. Test that Hadoop is working:
    `$ start-dfs.sh && start-yarn.sh`
    `$ hadoop fs -ls /`
    This should show a single folder `/data`
11. Install Spark via: `$ sudo ./spark.sh`
12. Test that Spark runs: `$ pyspark`. Then insert the following program:

```
rdd = sc.parallelize(["b", "a"])
sorted(rdd.map(lambda x: (x, 1)).collect())
```

    This should output: `[('a', 1), ('b', 1)]`
13. Install MongoDB via `$ sudo ./mongodb.sh`
14. Test that MongoDB [6] works:

```
sudo /etc/init.d/mongodb start
cd ./test/
./mongo.py
```

Congratulations, you have a working standalone Hadoop cluster with support for pyspark and MongoDB!

[2] If you are already using Linux as your main OS, it can still be sensible to set up a VM as you will install software for the exercises that might mess up your system. Having a VM that can be reinstalled without significant effort avoids this.

Portfolio (directory: `1/vm`)

`1/vm/software.txt` Short comments regarding your steps and issues that arose.

Hints

- You can run commands as root (the superuser) using `$ sudo` in the shell.
- The IP Address is shown with the command `$ ip addr show`.
- By default, the SSH port will be forwarded to the Virtual Machine, i.e., you have to use
  `$ ssh -p 2222 USERNAME@localhost`
  Where the username corresponds to the one you used for Ubuntu.

Further Reading

1. Installation for Hadoop in standalone mode https://www.digitalocean.com/community/tutorials/how-to-install-hadoop-in-stand-alone-mode-on-ubuntu-18-04
2. PySpark https://spark.apache.org/docs/latest/api/python/index.html
3. PyMongo https://api.mongodb.com/python/current/

Task 4: Reading CSV Data (Python) (60 min)

A popular, portable, file format for structured data is CSV. In this task, you will read a CSV file and print the mean of one of it's columns. Your program should potentially work with any CSV file (conforming to certain standards). Your program in Python should be run like this:

```
python csv.py [FILE]
```

Create a sample CSV file using Libreoffice (or MS Excel) by filling a table with data and saving it as `.csv`. You can copy and paste your data multiple times inside the file to give it a decent size.

Complete the following tasks:

- Compute and print the mean of each column using your program.
- Use the Unix-tool `time` to evaluate the performance of your program.
- Implement the same functionality in a second programming language of your choice.
- Again use the Unix-tool `time` to evaluate the second program. Which one does perform better?

4.1 Hints and Code-Skeleton

You *may* use the following structure and function (feel free to use any other function from the standard libraries if they help you, but don't use functions that parse the CSV for you).

4.1.1 Python

```
        import sys

        # Commandline arguments are avalible in the sys.argv list
        print(sys.argv)

        # Implement this function, do not use a function to completely parse the CSV file!
        def computeMeanCSV(filename, column_number):
        ...
        return mean
```

Portfolio (directory: `1/csv`)

`1/csv/csv.csv`	Your csv data
`1/csv/csv.py`	Your source code
`1/csv/csv-speed.txt`	Your runtime measurements with a short evaluation.
`1/csv/csv.??`	Your source code