|  |  |  |  |
| --- | --- | --- | --- |
| **Übungsprotokoll**  **SYTB – Systemtechnik Betriebssysteme** | | | |
|  | **Übungsdatum:**  11. September 2023 | **Klasse:**  3BHIT | **Name:**  Gabriel Martin |
| **Abgabedatum:**  25. September 2023 | **Gruppe:**  SYTB2 | **Note:** |
| **Leitung:**  Gerald Brandstetter | **Mitübende:**  None | | |
| **Übungsbezeichnung**:  Aufsetzen Debian / Knottenbelt Lecture 2 | | | |

Contents

[1 Aufgabenstellung 2](#_Toc145918163)

[2 Abstract 2](#_Toc145918164)

[3 Theoretische Grundlagen 2](#_Toc145918165)

[4 Übungsdurchführung 2](#_Toc145918166)

[4.1 Installation 2](#_Toc145918167)

[4.2 Intro to Linux 5](#_Toc145918168)

[4.3 Unix Filesystem: 6](#_Toc145918169)

[4.4 Symbolic Links: 8](#_Toc145918170)

[4.4.1 Soft Links: 8](#_Toc145918171)

[4.4.2 Hard Links: 8](#_Toc145918172)

[4.4.3 Inodes: 8](#_Toc145918173)

[4.4.4 Viewing Inodes: 9](#_Toc145918174)

[4.4.5 Hard Links Limitations: 9](#_Toc145918175)

[4.4.6 Soft Links Limitations: 9](#_Toc145918176)

[4.5 Running multiple commands: 9](#_Toc145918177)

[5 Findings 9](#_Toc145918178)

[6 Code 9](#_Toc145918179)

[7 Kommentar 9](#_Toc145918180)

# Aufgabenstellung

Installation of Debian Linux as a virtual machine.

Getting to know the Linux operating system.

# Abstract

Installation of Debian Linux using the unattended install in VirtualBox. Then a brief overview of the core functionality of Debian Linux.

# Theoretische Grundlagen

Virtulisation Plattform: VirtualBox

* Ease of use

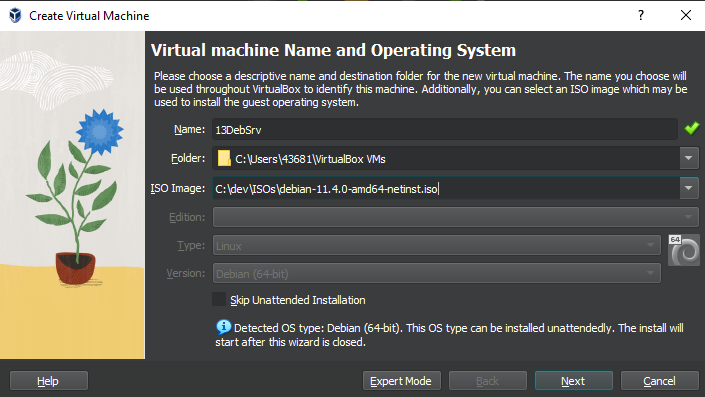
Installation Media (Debian 11.4.0)

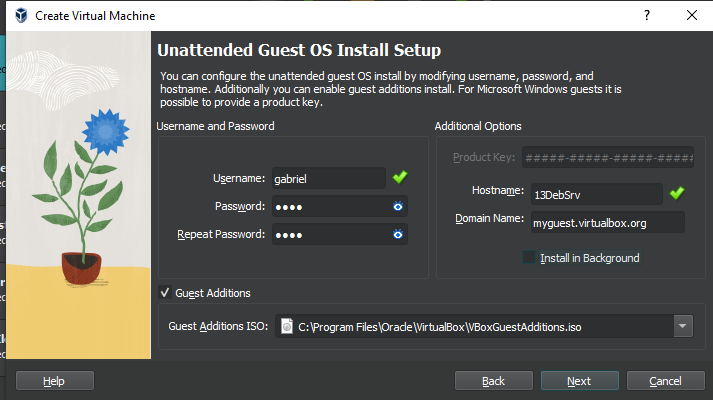
* Free on debian.org
* Choosing an ISO
  + Plattform (amd64)
  + Installationtype (networkinstaller)

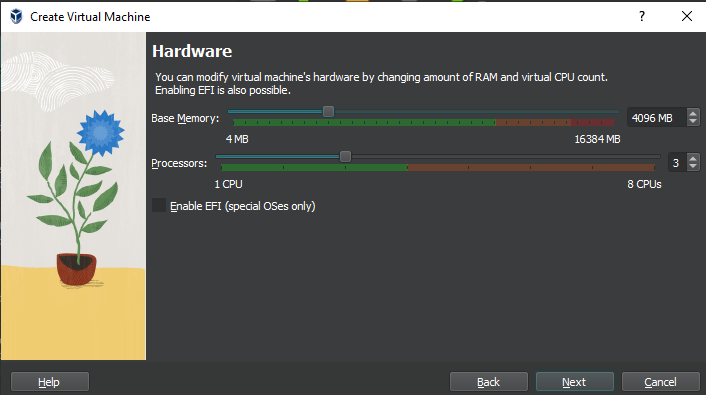
# Übungsdurchführung

## Installation

Erstellen einer neuen Virtuellen Maschine (Debian 11.4.0). Eintragen von VM-Name und Hostname(13DebSrv) und festlegen der Systemresourcen.



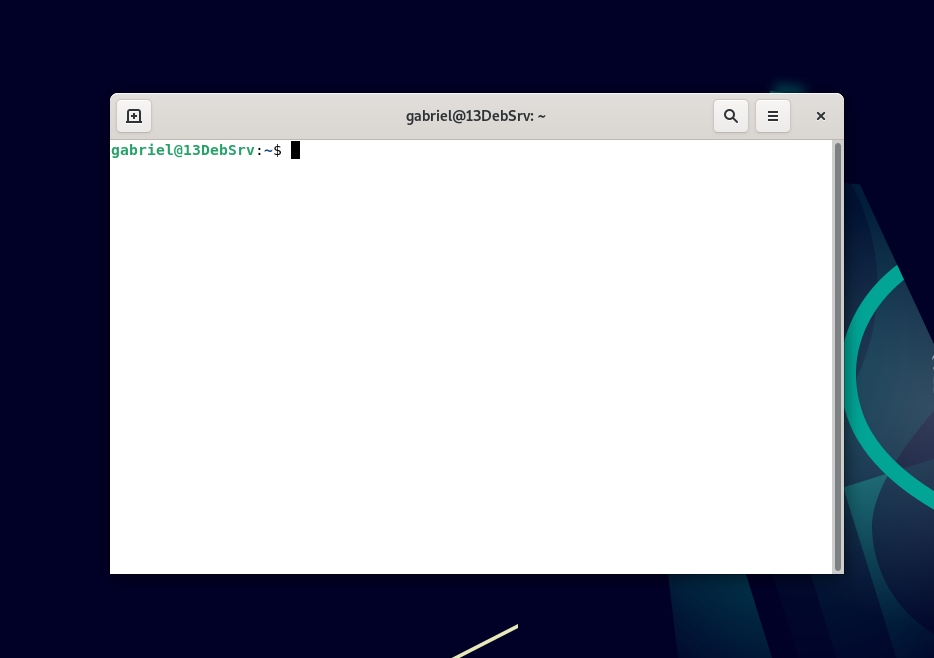




After the unattended installation is completed the VM will restart and begin installing the VirtualBox Guest Additions.

Next, we will log in with the credentials specified in the Username and Password fields of the Setup.

Now we can open a terminal by simply searching for the terminal application.

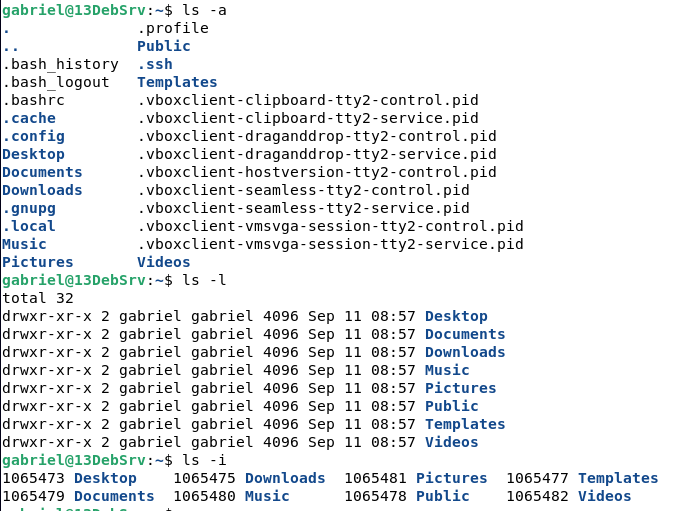


## Intro to Linux

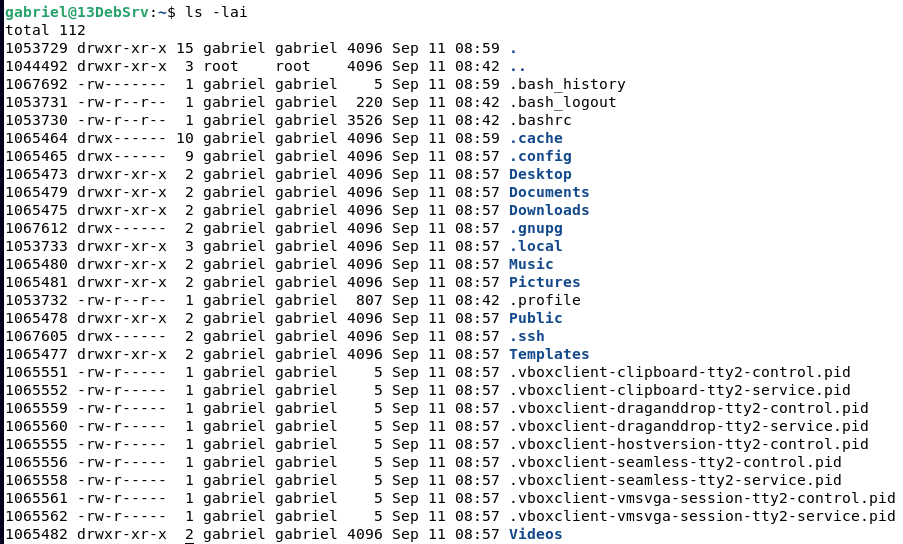
We can use the ls command to view the contents of a directory.



We can also call ls with parameters. For example: “ls -a” to show hidden files and directories, “ls -I” to show contents in a long list format and “ls -I” to display the Inode(see section “Symbolic Links”) id for each element.



We can also combine these options however we like.



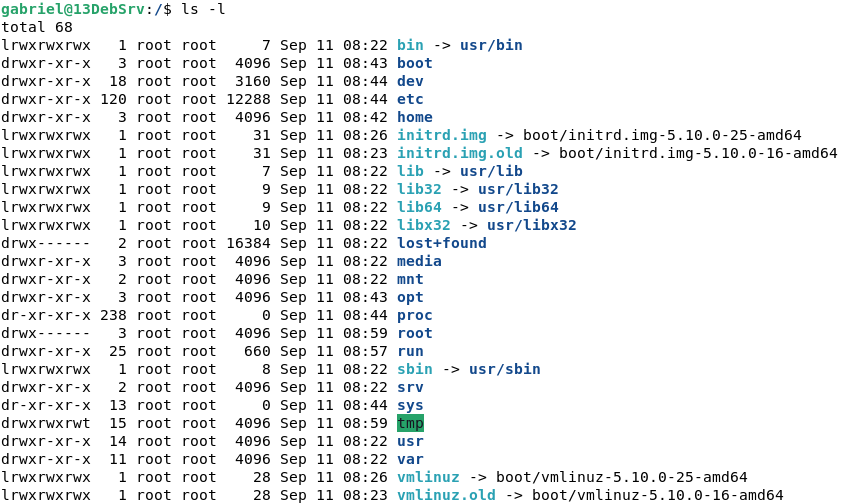
Now let’s take a closer look to the left of our cursor.

username@hostname:workingdirectory$

The username shows the user you are currently logged in with. The hostname shows what machine you are logged into. The working directory is the Directory you are currently in. All commands you run will execute from this directory. “~” is your users home directory. “/” is your systems root directory (see “Unix filesystem”). The “$” symbol represents the shell you are currently using (bash). There are different types of shells you can choose from with bash being the default one preinstalled.

## Unix Filesystem:

Unlike Windows Unix has a root path (“/”) that contains everything. We can use the command cd (change directory) to navigate to the root out our system. Then we can run “ls -l” again to get a list of all files and directories present at the root of our system.



Now we can see a lot of data. In the first column we can see the type and permissions (see “Users and Groups”). The first character specifies the type of item, “-“ being a regular file, “d” being a directory, “c” being devices and “l” being a symbolic link (also called Shortcut on Windows) (see “Symbolic Links”).

The 3rd and 4th column specify the user and group an item belongs to.

Let’s quickly go over some of the directories we see in this list.

/boot: contains data that is used before the Linux kernel begins execution

/dev: stores all hardware devices. In Linux every device connected to your machine is represented by a file path. We will find things like “bus”, “cpu”, “cdrom”, dvd”, “stdin” “stdout” and “stderr”, sda being our primary hard drive and sda1,2,5 (see “Hard drives and partitions”), “input” and many more.



/etc: contains files for system and application configuration. The “settings” app of the Linux terminal.

/home: contains the home directories for every user (“~”)

/bin, /sbin and /usr/bin: contain system binaries (for example “ls”)

/lib: contains libraries that can be used by a compiler to link to

/mnt: this is where your hard drives will be mounted (see “Hard drives and partitions”)

/media: is where your system can automatically mount drives

/proc: pseudo filesystem for kernel communication. Contains directories for every process.

In Linux, there are a few characters you should avoid using in filenames due to their reserved or special meanings. These characters include the forward slash (/), null character (null byte), and the ASCII control characters, such as newline, tab, and carriage return. Additionally, using leading dots (.) in filenames can make them hidden, and while not strictly forbidden, filenames with leading or trailing spaces are best avoided to prevent potential issues in command-line operations. It's generally recommended to stick to alphanumeric characters, hyphens, underscores, and periods for safe and compatible filenames.

Now we will look at common commands that help us navigate the filesystem.

ls: List all files and directories in a specified directory (if none the current working directory)

cd: Change the current working directory to the directory specified.

pwd: “Print working directory” print the directory you are currently in

mkdir: “Make directory” followed by the name of the directory to create.

touch: Creates a file with the name specified.

cp: “Copy” followed by the source file and the target file path including file name.

mv: “Move” followed by the source path and the target path.

rm: “Remove” followed by the target file (for directories use the “r” flag to recursively delete all files in it and the “f” flag standing for force (“rm -rf [directory name]”)

cat: Print out the contents of a file to the terminal.

echo: Print text to the console (“echo \*” print directory content like ls but without formatting, “echo hello world” writes text to console)

ln: To create soft and hard links (see below)

## Symbolic Links:

file management goes beyond simply creating, editing, and deleting files. One advanced feature that comes in handy for various purposes is the use of links, specifically symbolic (soft) links and hard links.

### Soft Links:

Symbolic links, often referred to as "symlinks" or "soft links," are essentially pointers or shortcuts to other files or directories. They act as references to the target file or directory and are particularly useful when you need to create cross-directory references or easily replace files without changing their location. Symlinks are created using the ln -s command.

### Hard Links:

Hard links are a bit different from symbolic links. They are direct references to the same inode (more on that shortly) as the target file or directory. Hard links are created using the ln command without the -s option.

### Inodes:

To understand links fully, you need to know about inodes. Inodes are data structures used by the Linux file system to store metadata about files and directories. Each file or directory in a Linux file system is associated with an inode, which contains information like file permissions, ownership, timestamps, and the actual data block pointers.

When you create a file, an inode is created alongside it to store this metadata. When you create a hard link to that file, both the original file and the hard link reference the same inode, making them essentially identical. This is why changes made to one are immediately reflected in the other.

### Viewing Inodes:

To view the inode of a file or directory, you can use the ls -i command.

### Hard Links Limitations:

Hard links can only be created for files, not directories.

Hard links cannot reference files on different file systems or partitions.

Removing the original file doesn't delete the hard link, but it makes it an orphaned inode, which can lead to data loss if not handled carefully.

### Soft Links Limitations:

Soft links can reference directories but may lead to issues if circular references are created.

Symbolic links can point to non-existent locations, potentially causing "broken links" if the target is moved or deleted.

They require additional system resources to follow, as the system needs to resolve the link to the target.

## Running multiple commands:

The structures with multiple commands in one, such as brace expansion and command substitution, work by allowing you to combine and execute multiple commands or generate a list of items based on a defined pattern.

Brace Expansion ({}): Brace expansion allows you to generate a list of items by specifying a pattern inside curly braces. For example, {a,b} would expand to "a" and "b," and {1..3} would expand to "1," "2," and "3." This feature is commonly used for tasks like creating sequences, generating file lists, or performing batch operations.

\*\*Command Substitution ( or $()):\*\* Command substitution allows you to embed the output of one command into another command. You can use backticks () or the $() syntax to achieve this. For instance, if you want to store the result of a command in a variable, you can use var=$(command).

# Findings

Linux can do a lot of things and it can do them quickly and easily.

# Code

None.

# Kommentar

I am certainly not smarter now than I was before, and I also haven’t learned anything new. I still hope this was a good overview of the Linux operating system.