BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

BACHELOR'S THESIS

Brno, 2022 Peter Milan Kluka

BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS

ÚSTAV TELEKOMUNIKACÍ

REDUCTION OF CENTOS OPERATING SYSTEM

REDUKCE OPERAČNÍHO SYSTÉMU CENTOS

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR Peter Milan Kluka

AUTOR PRÁCE

SUPERVISOR prof. Ing. Dan Komosný, Ph.D.

VEDOUCÍ PRÁCE

BRNO 2022

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Bachelor's Thesis

Bachelor's study program **Information Security**

Department of Telecommunications

Student: Peter Milan Kluka ***ID:*** 220893

Year of study: 3 ***Academic year:*** 2021/22

TITLE OF THESIS:

Reduction of CentOS operating system

INSTRUCTION:

Reduce the CentOS Stream operating system from the viewpoint of its size on storage media. The reduced system has to include a graphical web browser, PDF file viewer, and remote access by SSH. Work with the system programs and kernel modules when reducing the system. Create a set of scripts, which will automate the reduction process.

RECOMMENDED LITERATURE:

[1] Linux Dokumentační projekt. 4. vyd. Computer Press, 2008. 1336 s. ISBN: 978-80-251-1525-1.

[2] COOPER, M. Advanced Bash-Scripting Guide. Lulu.com, 2010. ISBN: 978-14-357-5218-4.

Date of project specification: 7.2.2022 ***Deadline for submission:*** 31.5.2022

Supervisor: prof. Ing. Dan Komosný, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
Chair of study program board

ABSTRACT

This thesis serves as a supporting material for the course Networking Operating Systems for the implementation of the course project. This project has an artificial goal which is to reduce the size of the operating system. The specified operating system was the Linux distribution CentOS Stream 8. The reduced system includes a graphical web browser, a PDF document viewer and is accessible via a remote SSH connection. The thesis compares sizes of required graphical frameworks and applications. Based on the analysis a set of scripts was created to automate the minimalization process. The resulting size of the reduced system was 678 MB.

KEYWORDS

Reduction, Linux, CentOS Stream, RHEL, Operating system, Lightweight, Web browser, SSH, Script

ABSTRAKT

Táto práca slúži ako podporný materiál pre predmet Sieťové operačné systémy a ako pomocný materiál pre realizáciu projektu v rámci tohto predmetu. Účelom tejto práce bolo vykonať redukciu operačného systému z hľadiska jeho veľkosti na disku. Zadaným operačným systémom bola Linuxová distribúcia CentOS Stream 8. Redukovaný operačný systém obsahuje grafický web prehliadač, PDF prehliadač dokumentov a je dostupný pomocou vzdialeného pripojenia SSH. Práca porovnáva veľkosti odporúčaných grafických rozhraní a aplikácií. Na základe analýzy bola vytvorená sada skriptov, ktorá zautomatizovala proces minimalizácie. Výsledná veľkosť redukovaného systému bola 678 MB.

KĽÚČOVÉ SLOVÁ

Redukcia, Linux, CentOS Stream, RHEL, Operačný systém, Webový prehliadač, SSH, Skript

KLUKA, Peter Milan. *Reduction of CentOS operating system* [online]. Brno: Brno University of Technology, Faculty of electrical engineering and communications, Dept. of telecommunications, 2022 [cit. 2022-05-26]. 55p. Bachelor's thesis. Accessible from: https://www.vutbr.cz/studenti/zav-prace/detail/141351. Advised by prof. Ing. Dan Komosný, Ph.D.

ROZŠÍRENÝ ABSTRAKT

Táto bakalárska práca sa venuje redukcii operačného systému CentOS Stream. Cieľom bolo zmenšiť jeho veľkosť na disku pomocou skriptov, pričom musel byť stále funkčný a schopný opätovného štartu. Systém musel obsahovať grafický webový prehliadač, PDF prehliadač dokumentov a možnosť vzdialeného pripojenia pomocou SSH.

V teoretickej časti sa popisujú základné časti operačného systému Linux, medzi ktoré patria ovládače, moduly jadra, firmvér, grafické prostredie a aplikácie. Každej z týchto základných častí je venovaná samostatná podkapitola, v ktorej sú definované. Následne sú objasnené dôvody a význam v operačnom systéme. Ďalej sú popísané vlastnosti vybraných priečinkov a podpriečinkov z hľadiska hierarchie a ich význam v rámci operačného systému. Okrem významu sú vysvetlené aj typy súborov, ktoré sa v daných priečinkoch nachádzajú a k čomu sú potrebné. V ďalšej časti práce je popísaný štart operačného systému a predstavené rôzne Linuxové distribúcie. Jednou z predstavených distribúcií je aj Red Hat Enterprise Linux, ktorá je vyvíjaná v zadanom systéme CentOS Stream. CentOS Stream má na rozdiel od Red Hat-u, ktorý je dostupný pre komerčné účely, širokú podporu od komunity a je voľne dostupný.

V praktickej časti je predstavený postup a príprava redukcie operačného systému, rôzne metódy a inštalácia minimalistických aplikácií, ktoré si zadanie tejto práce vyžadovalo. Analýza operačného systému sa zameriava na zistenie veľkosti pôvodných nainštalovaných aplikácii a ich minimalistických náhrad, ktoré mali značný vplyv na výslednú veľkosť redukovaného systému. Okrem aplikácií sa tiež skúmalo, ktoré súbory je možné odstrániť bez toho, aby ovplyvnili beh operačného systému celkovo, alebo len minimálne. Analýza okrem iného zahŕňala aj postup redukcie operačného systému. Bez obmedzenia chodu systému mohli byť odstránené dočasné súbory, manuálové stránky, preklady a rôzne dodatočné aplikácie. Pri redukcii bol zvolený automatizovaný postup podľa zadania práce, ktorý zahŕňa sadu skriptov. Kvôli inštalácií dodatočných aplikácií je nutný prístup na internet. Aby nedošlo k odstráneniu balíčkov, nutných pre chod systému počas vykonávania skriptov, bolo nutné ich pridať do výnimiek. Keďže viaceré dôležité aplikácie zdieľajú rovnaké balíčky, boli tieto aplikácie tak isto uchránené pred nechceným odstránením a znefunkčnením celého systému.

Podkapitola Automated reduction obsahuje ukážky skriptov, ktoré boli vykonávané. Okrem toho je ku každému skriptu popísané, koľko miesta sa pridalo, alebo uvoľnilo. Pre automatizovanú metódu redukcie boli rozdelené skripty na tri základné časti. V prvej časti

sa vykonala dodatočná inštalácia balíčkov a minimalistických aplikácií, v druhej časti nasledovala redukcia systémových balíčkov a v tretej časti redukcia adresárov a podadresárov. Nutnosťou takto redukovaného systému je jeho opätovné spustenie. Minimalistické náhrady boli volené podľa toho, akú veľkosť mali po nainštalovaní a nie, akú veľkosť mala samotná inštalácia. Je to z toho dôvodu, že aj keď samotná inštalácia nezaberá veľa miesta, systémové balíčky môžu túto veľkosť zväčšiť. Kvôli tomu bol zvolený webový prehliadač OtterBrowser, PDF prehliadač XPDF, terminál XTerm a grafické rozhranie Xfce. Okrem OtterBrowseru existujú aj rôzne iné menšie prehliadače, avšak ani jeden z nich neobsahuje JavaScript, ktorý je potrebný pre správne zobrazenie webových stránok. Najzdĺhavejšou časťou tejto práce je redukcia mnohých systémových balíčkov, ktoré boli prechádzané jeden po druhom. Po redukcii adresárov a podadresárov boli odstránené aj knižnice a balíčky vrátane tých, ktoré tvorili výnimky. Toto odstránenie nespôsobilo znefunkčnenie celého operačného systému, ale iba určitých častí. Patrí sem ovládací panel, navigačná lišta, zvuk a prehliadač súborov. Keďže sa dajú všetky potrebné aplikácie spustiť cez terminál, nie je potrebné, aby fungovali priamo z plochy. Pri opätovnom štarte systému je plocha tvorená iba kurzorom a terminálom XTerm, ktorý je používaný na spúšťanie aplikácií a vykonávanie príkazov.

V podkapitole Start of the OS after reduction je popísaný postup po reštartovaní systému, alebo po jeho opätovnom spustení. Keďže boli odstránené viaceré grafické prvky systému, je nutné najskôr sa prihlásiť v konzole a potom manuálne spustiť minimalizované grafické rozhranie Xfce.

Záverečná podkapitola Summary of reduction zhŕňa všetky výsledky tejto bakalárskej práce. Okrem toho informuje, ako sa zmenili veľkosti jednotlivých priečinkov a podpriečinkov po vykonaní jednotlivých skriptov, vrátane toho, akú veľkosť samotný systém postupne nadobúdal.

Výsledná veľkosť redukovaného systému bola 678 MB. Systém bol zbavený softvéru a súborov, ktoré neboli potrebné pre úspešné splnenie zadania. Všetky zadané podmienky práce boli splnené. Systém je schopný zobrazenia webovej stránky v nezmenenej podobe, skrípt v PDF prehliadači a byť vzdialene pripojený pomocou SSH. Možné využitie tohto systému je vhodné pre knižnice, zariadenia s obmedzenou pamäťou alebo výkonom a ako pomôcka pri plnení školského projektu.

Author's Declaration

Author:	*Peter Milan Kluka*
Author's ID:	220893
Paper type:	*Bachelor's Thesis*
Academic year:	*2021/22*
Topic:	*Reduction of CentOS operating system*

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the project and listed in the comprehensive bibliography at the end of the project.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno, May 26, 2022 ----------

author's signature

ACKNOWLEDGEMENT

I would like to thank my supervisor, prof. Ing. Danovi Komosnému, Ph.D. for leading the thesis, for support, help, professional suggestions, and comments. I would also like to thank my parents, brother, and friends for their support.

Contents

List of Figures 10
List of tables 11
List of Listings 12
Introduction 13
1 Linux operating systems 14
2 Parts of Linux operating systems 15
2.1 Kernel 15
2.2 Kernel modules 15
2.3 Firmware 16
2.4 Servers 17
2.5 Graphical environment 18
2.6 Applications 19
2.7 Linux filesystem hierarchy standard 19
3 Start of the OS 22
3.1 Init system 22
3.2 Bootloader 22
4 Distributions 23
4.1 Red Hat Enterprise Linux 23
4.2 CentOS Stream 24
4.3 Fedora 25
4.4 Debian 25
4.5 Ubuntu 26
4.6 Apache 26
5 Manual reduction of CentOS Stream operating system 27
5.1 Analyse of CentOS Stream 27
5.2 Reduction of temporary files 28
5.3 Reduction of utility software 29
5.4 Kernel modules reduction 30

5.5 Reduction of software packages 33

5.6 Selection and installation lightweight graphical environment 36

5.7 Selection and installation lightweight web browser 38

5.8 Automated reduction 43

5.9 Start of the OS after reduction 46

5.10 Summary of reduction 46

6 Conclusion 48

Bibliography 49

List of used abbreviations, quantities, and symbols 52

LIST OF APPENDICES 53

List of Figures

Figure 2.1: Simplified SSH connection 18
Figure 2.2: Linux FHS 20
Figure 5.1: Otter browser 42
Figure 5.2: Opened scripts in XPDF 47
Figure B.1: Currently used boot image 55

List of tables

Table 5.1: Size comparison of default and lightweight software 27
Table 5.2: Comparison of graphical environments 36
Table 5.3: Comparison of lightweight web browsers 39
Table 5.4: Folder comparison after reduction 47

List of Listings

Listing 5.1: Removing log files and log messages 28
Listing 5.2: Removing cache files 29
Listing 5.3: Currently used localization of the system 30
Listing 5.4: Reduction of directory /usr/share 30
Listing 5.5: List of currently established modules 32
Listing 5.6: List of rescue files 32
Listing 5.7: Removing unnecessary firmware 33
Listing 5.8: Deleting graphics drivers. 33
Listing 5.9: List of a few installed packages 34
Listing 5.10: Protecting packages from removal 35
Listing 5.11: Removing unprotected package using yum 36
Listing 5.12: Attempt to delete protected package 36
Listing 5.13: Commands used for installation Xfce 37
Listing 5.14: Installation of Xfce 38
Listing 5.15: Nux dextop and Otter browser installation 42
Listing 5.16: The first section of the script 44
Listing 5.17: The script’s second section 44
Listing 5.18: The script’s final section 46

Introduction

This bachelor's thesis deals with minimization in one of Linux's distributions—CentOS Stream 8. I am doing this thesis as supporting material for a project from the Networking Operating Systems course. This distribution could be used on devices with limited memory. Thanks to minimization, the final size of the operating system was comparable to other Linux minimized distributions like Knoppix, Bodhi, or AntiX. These distributions take up 700 MB–900 MB of space on a storage medium. The main difference is in the minimization methods of these operating systems. Every operating system is built similarly – first the main part is the core (kernel), and then other packages for smooth operation are added. The minimization was executed according to assignment on CentOS Stream 8, which was released in September 2019. The latest version used in minimization was released in April 2022. The reduced operating system had to include a graphical web browser, a PDF file viewer, and remote access by SSH. The minimization was performed on a installation of CentOS Stream with GUI. First, I installed an Otter-Browser, XPDF, XTerm, and Xfce using a set of scripts, and then I started to reduce the operating system with scripts by removing all necessary programs, packages, icons, drivers, fonts, etc. This bachelor's thesis focuses only on Linux as an operating system, which is what chapter 1 deals with. Chapter 2 focuses on parts of the Linux operating system, such as the kernel, firmware, servers, applications, etc. Subchapter 2.4 deals with background services and demons, which monitor and take care of certain subsystems to ensure that the operating system runs properly. The SSH protocol belongs here as well. In Subchapter 2.5, the graphical environment and display server are described, and this chapter explains the most commonly used display server on Linux with other GUI. In chapter 3, the start of the OS and bootloader, as well as GRUB and BIOS, is described. Chapter 4 focuses on Linux distributions like Red Hat Enterprise Linux, CentOS Stream, Fedora, Debian, Apache, and Ubuntu. The practical part begins with chapter 5, where the analysis of CentOS Stream begins. Subchapters 5.6 and 5.7 are focused on the installation of a lightweight web browser and GUI. In these chapters, there is also a comparison of the default and different lightweight web browsers and GUIs. In Subchapter 4.8, are parts of scripts that were used in automated reduction and the results of every part of the script. Subchapter 5.9 describes the start of the operating system after reduction performed by a set of scripts. The results of this performed reduction are in Subchapter 5.10.

1 Linux operating systems

Linux was created in 1991 by Linus Torvalds when he was studying at the University of Helsinki. He built the Linux kernel as a free, open-source operating system (OS) that was used in academic settings. Like other operating systems, we can find here a graphical interface, video editors, audio editors, word processors, and other types of software.

This open-source operating system is suitable for everyone because users can choose from many distributions. Companies choose Linux for servers due to its secure, flexible, and excellent support from the community of users. [2]

Linux is a Unix-like, open-source and community-developed OS for computers, servers, mainframes, mobile devices, and embedded devices. It works on almost every major computer platform, such as x86, ARM, and SPARC. This makes it one of the operating systems with the most support.

Every version of the Linux OS manages hardware resources, launches and handles applications, and provides some form of user interface. Because Linux has a huge community of developers and a wide range of distributions, there is a version for almost any task. Linux is used in many different areas of computing. [35]

2 Parts of Linux operating systems

2.1 Kernel

This is the piece we call Linux. It is the foundation of the Linux operating system. A kernel is the lowest level of software that can interface with computer hardware. All Linux applications and servers also interface with the Linux kernel. All Linux distributions are based on the Linux kernel and use its services to implement various software functions. Unlike the hybrid kernels of Microsoft Windows and Mac OS X, the Linux kernel is monolithic. Monolithic kernels are in charge of the CPU, memory, device drivers, file system management, and system server calls.

The Linux kernel is developed by thousands of open-source contributors worldwide. Each version of the kernel is supported for up to six years. The Linux kernel is released under the GNU General Public License version 2 (GPLv2). The Linux kernel manages OS resources, making sure there is enough memory available for applications to run, optimizing processor usage, and avoiding system deadlocks caused by competing application demands. [4]

2.2 Kernel modules

Kernel modules are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system. A module can be configured as built-in or loadable. In the kernel configuration, a module must be set up as a loadable module for it to be added or taken away on the fly.

There are several advantages that come with using kernel modules. For example, it is easier to diagnose system problems. A bug in a device driver which is bound into the kernel can stop the system from booting at all. It can also be hard to tell which part of the base kernel caused the trouble. If the same device driver is a module, though, the base kernel is up and running before the device driver even gets loaded. If the system dies after the base kernel is up and running, it's an easy matter to track the problem down to the trouble-making device driver and just not load that device driver until the problem is resolved.

Using modules can save memory because they are loaded only when the system is using them. All parts of the base kernel stay loaded, in real storage, not just virtual storage.

Modules are much faster to maintain and troubleshoot. What would normally require a full reboot to do with a filesystem driver built into the kernel can be done with a few quick

commands using modules. It is possible to try out different parameters or even change the code repeatedly in rapid succession without waiting for a boot. [23]

Loadable kernel modules in Linux are loaded (and unloaded) by the modprobe command. They are located in */lib/modules/$(uname -r)/kernel* or */usr/lib/modules/$(uname -r)/kernel* and have had the extension .ko ("kernel object") since version 2.6 (previous versions used the .o extension). The lsmod command lists the loaded kernel modules. In emergency cases, when the system fails to boot due to e.g. broken modules, specific modules can be enabled or disabled by modifying the kernel boot parameters list (for example, if using GRUB, by pressing 'e' in the GRUB start menu, then editing the kernel parameter line). [24]

2.3 Firmware

Firmware is a type of software that is etched directly into a piece of hardware. It operates without going through APIs, the operating system, or device drivers—providing the needed instructions and guidance for the device to communicate with other devices or perform a set of basic tasks and functions as intended. Without firmware, the most basic of devices would not function. That is why it is often stored on a Read-Only Memory (ROM) chip, ensuring it does not get erased accidentally, all while remaining as close as possible to the metal of the device. Firmware varies in complexity and can be found in simple devices such as keyboards and hard drives as well as more complex ones such as graphics cards and Basic Input/Output System (BIOS). In Linux, firmware is located at */lib/firmware/*. [25]

When a device is powered on, firmware is the first part to run and starts sending instructions to the device's processor to execute. If the device is as simple as a keyboard, the firmware does not stop working as there is no software to replace it. However, in more complex devices, such as PCs, laptops, and tablets, there are often multiple firmware sets that interact to achieve a common goal - load the operating system. Regardless of the type of device, firmware can only work with a basic or low-level binary language known as machine language. While the firmware's code could be written in a high-level language for ease and versatility, it needs to be translated into a low-level language before getting etched into the device. [25]

Firmware levels

Depending on where it is stored and the complexity of its functionality, firmware has three levels:

1. Low-Level Firmware: This level of firmware is stored on non-volatile memory chips such as ROM, PROM—also known as One-Time Programmable (OTP) memory—and

Programmable Logic Array (PLA) structures. Because low-level firmware is often stored on read-only chips that cannot be rewritten or updated, it is considered an intrinsic part of the hardware.

2. High-Level Firmware: This firmware is used with flash memory chips to allow for updates. It typically has more complex instructions than low-level firmware, making it closer to software than hardware.

3. Subsystems: A subsystem is a device or unit that is a semi-independent part of a larger system. Because this firmware level has its microcode embedded in flash chips, CPUs, and LCD units, and is similar to high-level firmware, it typically resembles its device. [25]

2.4 Servers

Servers are background services that start up during boot, or after you log into the desktop. They are utility programmes that run silently in the background to monitor and take care of certain subsystems to ensure that the operating system runs properly. A printer daemon monitors and takes care of printing services. A network daemon monitors and maintains network communications, and so on.

Daemons perform certain actions at predefined times or in response to certain events. There are many daemons that run on a Linux system, each specifically designed to watch over its piece of the system, and because they are not under the direct control of a user, they are effectively invisible but essential. [7]

Secure Shell

Secure Shell, or Secure Socket Shell, is a network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network.

SSH also refers to the suite of utilities that implement the SSH protocol. Secure Shell offers strong password authentication and public key authentication, as well as encrypted data communication between two computers connected over an open network, like the internet.

SSH refers both to the cryptographic network protocol and to the suite of utilities that implement that protocol. It uses the client-server model, connecting a Secure Shell client application, which is the end where the session is displayed, with an SSH server, which is the end where the session runs. Its implementations often include support for application protocols used for terminal emulation or file transfers. [26]

The protocol works in the client-server model, which means that the connection is established by the SSH client connecting to the SSH server. The SSH client drives the connection setup process and uses public key cryptography to verify the identity of the SSH server. After the setup phase, the SSH protocol uses strong symmetric encryption and hashing algorithms to ensure the privacy and integrity of the data that is exchanged between the client and server.

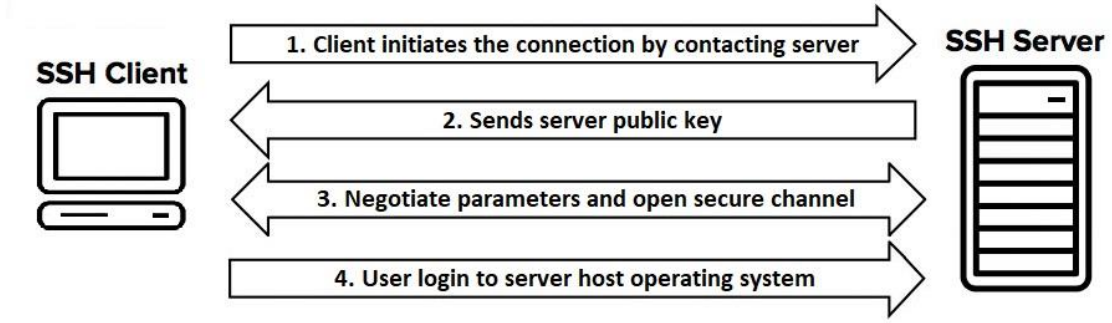

Figure 2.1: Simplified SSH connection

2.5 Graphical environment

A graphical (display) server is a programme which is responsible for the input and output coordination of its clients, to and from the rest of the operating system, and among the hardware and the operating system. Thanks to a display server, you can use your computer graphically (GUI).

The display server provides the framework for a graphical environment so that you can use a mouse and keyboard to interact with applications. This server also communicates with its clients over the display server protocol (e.g. X11 is the most common display server used in Linux distributions). The display server is an important part of any graphical user interface, and the windowing system in particular.

The X11 communication protocol uses the X.org Server display server. It receives input events from device drivers and makes them available to one of its clients. The display server also receives data from its clients. It processes the data and does the compositing, and on Linux it passes the data to one of three kernel components – the DRM, gem, or KMS driver. The X.Org Server is a display server that relies on a second program, the compositing window manager, to do the compositing. GNOME uses Mutter. [8]

The piece that users interact with. It is the bundle of components that provides you with common graphical user interface (GUI) elements such as icons, toolbars, wallpapers,

and desktop widgets. Thanks to the desktop environment, you can use Linux graphically using your mouse and keyboard like you do in other operating systems.

Most of the desktop environments have their set of integrated applications and utilities so that users get a uniform feel while using the OS. Without a desktop environment, your Linux system will just have a terminal-like utility, and you will have to interact with it using commands only.

We choose from many desktop environments (GNOME, Mate, KDE, Xfce, etc.). It also includes built-in applications, such as configuration tools, games, file managers, and others. [9]

2.6 Applications

Like any other OS, Linux offers a lot of software titles that can be easily found and installed. For example, Ubuntu Linux has the „Ubuntu Software Centre", which allows you to search for apps and install them. Most Linux distributions have app stores that help you find and install apps. [1]

2.7 Linux filesystem hierarchy standard

In Unix-like operating systems, the Filesystem Hierarchy Standard (FHS) or the Linux File Hierarchy Structure (FHS) defines the directory structure and contents. It is maintained by the Linux Foundation.

In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices. Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed. Most of these directories exist in all UNIX operating systems and are generally used in much the same way. However, the descriptions here are those used specifically for the FHS and are not considered authoritative for platforms apart from Linux. [14]

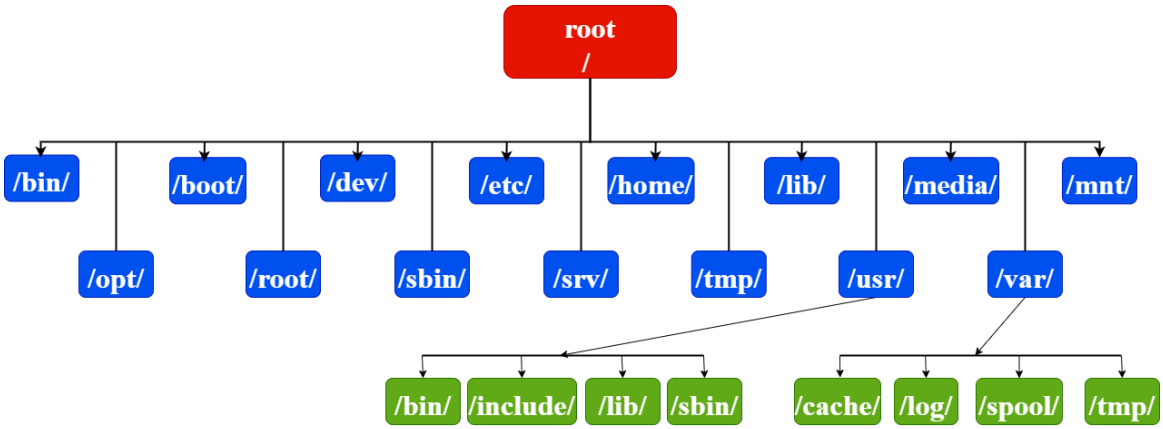

Figure 2.2: Linux FHS

/Root

The */root* directory is the home directory of the root account. This directory contains the user's personal files, directories, and programs. It's not usually accessible to other users on the system. The root account is the username or account that has access to all commands and files on a Unix-like operating system.

/root is a standard first-tier directory in the root directory (as are */bin, /boot, /dev, /etv, /home, /sbin, /mnt and /usr*). The root directory is the top-level directory on any Unix-like operating system – the directory that contains all other directories and their subdirectories. [15]

/bin

This directory contains binaries for use by all users. It also contains executable files, Linux commands that are used in single user mode, and common commands that are used by all the users, like cap, cp, cd, etc. By the FHS, the */bin* directory should contain */bin/cat* and */bin/date*. The *"/bin"* directory doesn't contain directories. [16]

/boot

/boot folder contains the Linux boot configuration files. This is one of the most important folders. Removing anything from this directory or a file getting corrupted will result in an operating system crash after reboot. The system won't be able to boot without files in the */boot* directory. [17]

This directory contains everything required for the boot process, except for the configuration files not needed at boot time and the map installer. The */boot* directory stores data that are used before the kernel begins executing user-mode programs. This may include redundant master boot records, sector/system map files, the kernel, and other critical boot files and data that are not directly edited manually. [18]

/dev

/dev is the location of special or device files. It is a directory that highlights on aspect of the Linux filesystem – everything is a file or a directory. [19]

Device files are employed to provide the operating system and users an interface to the devices that they represent. All Linux device files are in the */dev* directory, which is an integral part of the root filesystem, because these device files must be available to the operating system during the boot process. [20]

/etc

/etc contains all system-related configuration files here or in its subdirectories. A configuration file is defined as a local file used to control the operation of a program. It must be static and cannot be an executable binary. For this reason, it's a good idea to backup this directory regularly. Normally, no binaries should be or are located here. [21]

/home

Linux is a multi-user environment, so each user is also assigned to a specific directory that is accessible only to them, and the system administrator. These are the user home directories, which can be found under *"/home/ $USER"*.

Your home directory contains your personal configuration files, the so-called dot files. Personal configuration files are usually hidden. If you want to see them, you must turn on the appropriate option in your file manager or run ls with the -a switch. If there is a conflict between personal and system-wide configuration files, the settings in the personal file will prevail. [22]

3 Start of the OS

3.1 Init system

A sub-system that bootstraps the user space and is charged with controlling daemons. Also manages the boot process once the initial boot is handed over to the bootloader, for example, Grand Unified Bootloader (GRUB). The init process starts all other processes, that is daemons, services, and other background processes. Therefore, it is the mother of all other processes on the system. [1]

Init is the parent of all processes executed by the kernel during the booting of a system. Its principal role is to create processes from a script stored in the file */etc/inittab*. It usually has entries which cause init to spawn gettys on each line so that users can log in. [5]

The init is a daemon process which starts as soon as the computer starts and continues running until it is shut down. Init is started by the kernel during the booting process. A kernel panic will occur if the kernel is unable to start it. Init is typically assigned a process identifier of 1. [6]

3.2 Bootloader

This is the software that manages the boot process of your computer. A bootloader, or boot manager, is a program that places the operating system into memory. When a computer is powered up or restarted, the basic input/output system (BIOS) performs initial tests, and then transfers control to the Master Boot Record (MBR) where the bootloader resides.

For Linux, the two most common bootloaders are known as LILO (Linux Loader) and LOADLIN (LOAD LINux). An alternative bootloader, called GRUB (Grand Unified Bootloader), is used with Red Hat Linux. LOADLIN is preferred by some users whose computers have multiple operating systems and who spend relatively little time on Linux. GRUB is preferred by many users of Red Hat Linux because it is the default bootloader for that distribution. [3]

4 Distributions

There are many versions that Linux can offer us. These versions can suit any type of user. These different versions are called "distributions". Most distributions can be downloaded and installed for free. Among the most popular distributions are: Linux Mint, Majaro, Debian, Ubuntu, Fedora, Elementary OS etc.

Different people and organizations work on different parts. There's the Linux kernel (the core of operating system), the GNU shell utilities (the terminal interface and other commands you use), the X server (which produces a graphical desktop), the desktop environment (which runs on the X server to provide a graphical desktop), and more. [11]

4.1 Red Hat Enterprise Linux

Red Hat Enterprise Linux is a Linux operating system that has been certified on hundreds of servers and by thousands of vendors. Is mainly used for commercial markets. The first version of Red Hat came onto the market as Red Hat Linux Advanced Server. Later, in 2003, it was rebranded to Red Hat Linux AS. In December 2020, Red Hat announced that he will begin shifting work from CentOS Linux to CentOS Stream. CentOS Stream will become the development platform for the next minor release of Red Hat Enterprise Linux. [12]

Being a Linux distribution, Red Hat Enterprise Linux contains the Linux kernel as well as some applications for performing certain tasks. Like all Linux distributions, RHEL is open-source. Thus, people can view its source code, download it, and make their own customized versions.

Some notable Linux distros that are derived from RHEL include CentOS, Oracle Enterprise Linux, Scientific Linux, and Pie Box Enterprise Linux. In the past, Red Hat gave this enterprise product for free and only charged for support. Later, they decided to create two versions: RHEL, which would have fewer frequent version releases and consequently be more stable, and Fedora, which would undergo relatively more frequent version releases and consequently offer more bleeding-edge technologies.

Fedora, which is given entirely for free, is sponsored by Red Hat (the company) but is actively developed by a community of developers. It is most suitable for Linux enthusiasts. RHEL, on the other hand, takes technologies developed via the Fedora Project and packages them into a more reliable and stable commercial product. Hence, RHEL is best suited for the enterprise. People who subscribe to RHEL can download the installer for free, but have to pay for support. Special editions of RHEL are available for academic institutions who are

willing to pay a smaller fee to use the relatively more stable RHEL rather than Fedora. A typical RHEL distribution would include development tools, applications, services, and utilities such as Compiz, CUPS, DHCP, Firefox, GIMP, MySQL, OpenOffice.org, Samba and Python, to name a few. [29]

4.2 CentOS Stream

CentOS Stream is the new CentOS that will follow in the footsteps of Fedora. It's a developer-forward distribution that aims to help community members, Red Hat partners, and others take full advantage of open-source innovation within a more stable and predictable Linux ecosystem. It's free for anyone to download, use, study, modify, and redistribute. [13]

CentOS began as a community project to pull down all the source RPMs from RHEL and build a usable distro out of them. The only changes made were to remove Red Hat's branding, per the legal requirements. Other than the branding changes, CentOS Linux was as close to a carbon copy of RHEL as possible; it didn't carry fixes, changes, or patches of its own.

The project encountered some challenges familiar to those in open-source: AWOL founders, budget shortfalls, and personal issues. Because the RHEL build system wasn't public, CentOS had to invent their own, which required significant changes to accommodate each major release. Those changes required development time that wasn't always available from volunteers. In the run-up to RHEL4 and RHEL5, there were frequent questions on the mailing list about whether the CentOS versions would be released or not. Since CentOS didn't sell a product or service, it relied on volunteers and donations, like many other community projects. Proclamations of CentOS's death were made by onlookers many times over the years.

Eventually, Red Hat acquired the CentOS project for a variety of reasons, which, among other things, provided critically needed resources. After what was apparently much internal discussion about how best to house CentOS, Red Hat decided to effectively firewall it off so that CentOS had no "special" access to RHEL that other downstreams couldn't get. This often-meant that CentOS was still having to reinvent wheels that RHEL had already invented. Furthermore, it meant CentOS stayed as a "carbon copy" of RHEL. [28]

CentOS's developers use Red Hat's source code to create a final product very similar to RHEL. Red Hat's branding and logos are changed because Red Hat does not allow them to be redistributed. CentOS is available free of charge. Technical support is primarily provided by the community via official mailing lists, web forums, and chat rooms. The project

is affiliated with Red Hat but aspires to be more public, open, and inclusive. Most of the CentOS head developers work for Red Hat, but the CentOS Project itself is supported by donations from users and organizations.

4.3 Fedora

Fedora is a popular open-source Linux-based operating system. Fedora is designed as a secure, general-purpose operating system. The operating system is developed on a six-month release cycle under the auspices of the Fedora Project. Fedora is sponsored by Red Hat.

According to the Fedora Project, Fedora is "always free for anyone to use, modify, and distribute." Fedora is said to be the second-most used Linux distribution, after Ubuntu. There are over a hundred distributions based on Fedora, including Red Hat Enterprise Linux (RHEL) and the One Laptop Per Child (OLPC) project's XO operating system. [30], [31]

The Fedora Project is a group of community developers, volunteers, and Red Hat employees who collaborate and share solutions built on the Fedora platform. It was formed in 2003 as a partnership between Red Hat and volunteer contributors. The Fedora Project is responsible for all innovations and new Fedora releases. It also takes care of other versions of Fedora, like network installers and torrent downloads, that are made for specific needs.

All Fedora editions use Security-Enhanced Linux, a security module that supports access control policies. It reduces the surface of vulnerability through a technique called hardening, which Fedora does for all its packages. Fedora IoT also supports TPM2, Secure Boot, and automated decryption with Clevis. [32]

4.4 Debian

The Debian stable branch is the most popular edition for personal computers and servers. Debian is also the basis for many other distributions, most notably Ubuntu.

Debian is one of the oldest operating systems based on the Linux kernel. The project is coordinated over the Internet by a team of volunteers guided by the Debian Project Leader and three foundational documents: the Debian Social Contract, the Debian Constitution, and the Debian Free Software Guidelines. Updates to new distributions continually happen, and the next candidate is released after a certain amount of time has passed.

Since its founding, Debian has been developed openly and distributed freely according to the principles of the GNU Project. Because of this, the Free Software Foundation sponsored the project from November 1994 to November 1995. When the sponsorship

ended, the Debian Project formed the non-profit organization Software in the Public Interest to continue financially supporting development. [33]

4.5 Ubuntu

Ubuntu Desktop is a Linux distribution developed by Canonical, and it's one of the most popular distributions, thanks to its ease of use. Canonical provides security updates and support for each Ubuntu release, starting from the release date and until the release reaches its designated end-of-life date. Canonical generates revenue through the sale of premium services related to Ubuntu and donations from those who download the Ubuntu software. It's also one of the top choices for people who are getting started with Linux. The server edition is also operating in most internet servers.

Ubuntu is officially released in three editions: Desktop, Server, and Core for Internet of things devices and robots. All editions can be run on a single computer or in a virtual machine. Ubuntu is a popular operating system for cloud computing, with support for OpenStack. [34]

4.6 Apache

Apache is a freely available Web server that is distributed under an "open source" license. Version 2.0 runs on most UNIX-based operating systems (such as Linux, Solaris, Digital UNIX, and AIX), on the other UNIX/POSIX-derived systems (such as Rhapsody, BeOS, and BS2000/OSD), on AmigaOS, and on Windows 2000. According to a Netcraft Web server survey, 60% of all Websites on the Internet are using Apache (62% including Apache derivatives), making Apache more widely used than all other Web servers combined.

According to the Apache Software Foundation, the name 'Apache' was chosen to honour the Native American Indian tribe of Apache, well-known for their superior skills in warfare strategy and their inexhaustible endurance. It also makes a cute pun on "a patchy web server"-a server made from a series of patches—but this was not its origin. The group of developers who released this new software soon started to call themselves the "Apache Group". [36]

5 Manual reduction of CentOS Stream operating system

5.1 Analyse of CentOS Stream

Before reduction, we need to know what we can be removed, while the operating system stays working. Removing the wrong package will mostly lead to a non-working operating system or some of its main functions. The next thing is to check if there are some alternatives for the programmes and environments that we will be using later. Alternatives, which will take up less storage on a hard drive than the default software pre-installed with our operating system.

In my case, I replaced the internet browser, terminal, and graphical environment with much lighter versions of the programs. Then I replaced the default graphical environment GNOME with a lightweight environment called Xfce, the default internet browser with Otter-Browser, the terminal with XTerm, and then I added the XPDF browser because, originally, PDF files were viewed in Firefox. In table 5.1 under this section, you can see the differences in the sizes of default programmes and lightweight installed programs.

Table 5.1: Size comparison of default and lightweight software

Desktop environment	**Installed size (MB)**
GNOME (default)	2487
Xfce	1529
Internet browsers	
Firefox (default)	219
Otter-Browser	112
Others	
Terminal (default)	1,9
XTerm	0,9
XPDF	15,8

As can be seen from the table 5.1, just by changing these two default pre-installed programmes and the environment to a lightweight version, I was able to save up to 1050 MB. Even more space I saved by removing packages that I didn’t use for any other software.

The greatest emphasis was placed on the size of various folders in the filesystem hierarchy structure and the next parts that communicate with the kernel, like modules,

software packages, libraries, and other software. Thanks to this analysis, I was able to create a solution and find the right tools, which I used to perform a reduction. Because every distribution of Linux is different, any other solution would not work correctly and would not be effective enough, which is why my solution will not work with any other distribution. The reduction was done on CentOS Stream 8, which was downloaded in April 2022 and had only a few programmes installed.

5.2 Reduction of temporary files

Every operating system has folders where temporary files are stored. Between these files are files from the internet browser, log files, and cache files, where applications store information about their state. A temporary file is created to hold information while a file is being modified or created, or for a variety of purposes. Such as when a programme cannot allocate enough memory for its tasks, or when the programme is working on data bigger than the architecture's address space. Most temporary files are deleted after the application is closed. On Linux, temporary files are stored in the directory /var. The log files were first, which I removed.

```
rm /var/log/messages
find /var/log -name '*.log*' -delete
```

Listing 5.1: Removing log files and log messages

Log files have a suffix ".log" and, thanks to command **find,** I was able to find them and then remove them. Any user should never modify or remove log files because they are important for the correct operation of the operating system. They contain significant information, thanks to which can help solve problems with the system or restore it to a state when it was operating correctly.

Cache files are not that important, and if some application needs them, it will just generate a new one. Temporary files are meant to store information temporarily and don't rely on the information stored in them. However, deleting temporary files that are in use may cause errors with the program. To help prevent problems, many programmes lock the file while in use to prevent it from being deleted. For our minimization, we can use this folder to delete every file in it. The command **rm**, which is responsible for deleting files and folders, was used in this case. In listing 5.2 is shown the whole command with a path.

```
rm -rfv /var/cache/*
rm -rfv /home/user/.cache/*
```

Listing 5.2: Removing cache files

Removing temporary files, in my case, can save up to 200 MB of storage. It depends on applications installed on the operating system. However, due to generating new temporary files with every start of a different application, the best way is to remove these files after we have done working with them.

5.3 Reduction of utility software

Utility software is software designed to help analyse, configure, optimize, or maintain a computer and enhance its performance. This programme performs a specific task, which is usually related to managing the system resources. Utility software focuses on how the computer's infrastructure operates. That includes computer hardware, application software, an operating system, and data storage programs. These utilities could range from small and simple to large and complex and can perform either a single task or multiple tasks. Some functions performed by these utilities are data compression, disk fragmentation, data recovery, management of computer resources files, system diagnosis, virus detection, and many more. Subdirectories in the directory /usr include manual pages, software documentation, and information. The directory /usr/share contains several subdirectories, including /icons, /themes, /sound, /zoneinfo, /doc, /man, /info and /locale. The largest directory is /locale, which includes a set of parameters that define the user's language, region, and any other preferences that the user wants to see in their user interface. Usually, an identifier locale consists of at least a language code and a country or region code. Number, character classification, date-time, string, currency, paper size, and colour format are examples of locale settings. User chooses localization while installing the operating system or during use. The size of this subdirectory is 280 MB. Localization, which is the operating system currently being used, can be checked with a command locale.

```
[peterk@localhost ~]$ locale
LANG=en_US.UTF-8
LANGUAGE=
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
```

```
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=
```

Listing 5.3: Currently used localization of the system

From listing 5.3, it is obvious that the currently used localization is "en_US". Files that correspond can be left behind, and others can be removed. This leads to removing localizations that are not used. The next step is removing manual pages, software documentation, information files, some icons, themes, sounds, and others.

```
rm -rfv /usr/share/doc/
rm -rfv /usr/share/man/
rm -rfv /usr/share/info/
rm -rfv /usr/share/icons/Adwaita/
rm -rfv /usr/share/icons/hicolor/
rm -rfv /usr/share/sounds/
rm -rfv /usr/share/zoneinfo/
rm -rfv /usr/share/theme/Adwaita/
```

Listing 5.4: Reduction of directory /usr/share

While reducing, some graphical parts of the operating system were also removed. This does not affect the functionality of the operating system because everything is performed via terminal. Although these files weren't that large, I was trying to remove everything I could.

5.4 Kernel modules reduction

The kernel is a computer programme at the core of a computer's operating system and has complete control over everything in the system. It is the part of the operating system code that is always resident in memory and facilitates interactions between hardware and software components. The kernel controls all hardware resources via device drivers, arbitrates conflicts between processes concerning such resources, and optimizes the utilization of common resources like CPU and cache usage, file systems, and network sockets. On most systems,

the kernel is one of the first programmes loaded on startup after the bootloader. It handles the rest of startup as well as memory, peripherals, and input/output requests from software, translating them into data-processing instructions for the CPU. The list of all the modules in the system can be displayed using the command **lsmod**. Every system has a different number of modules. In my case, it was 103. In the listing 5.5, the first 28 modules are shown.

```
[peterk@localhost ~]$ lsmod
Module                  Size  Used by
uinput                 20480  1
nls_utf8               16384  1
isofs                  49152  1
rfcomm                 86016  4
xt_CHECKSUM            16384  1
ipt_MASQUERADE         16384  3
xt_conntrack           16384  1
ipt_REJECT             16384  2
nft_compat             20480  16
nf_nat_tftp            16384  0
nft_objref             16384  1
nf_conntrack_tftp      16384  3 nf_nat_tftp
nft_counter            16384  33
tun                    49152  1
bridge                204800  0
stp                    16384  1 bridge
llc                    16384  2 bridge,stp
nft_fib_inet           16384  1
nft_fib_ipv4           16384  1 nft_fib_inet
nft_fib_ipv6           16384  1 nft_fib_inet
nft_fib                16384  3 nft_fib_ipv6,nft_fib_ipv4,nft_fib_inet
nft_reject_inet        16384  5
nf_reject_ipv4         16384  2 nft_reject_inet,ipt_REJECT
nf_reject_ipv6         16384  1 nft_reject_inet
nft_reject             16384  1 nft_reject_inet
nft_ct                 20480  18
```

```
nf_tables_set          49152
nft_chain_nat          16384
```

Listing 5.5: List of currently established modules

From this listing, two columns are important. The first is a **Module,** while the second is **Used by**. In the Module column are the names of modules, and in the column Used by is the number of kernel or process, which module is currently used. If a module has a value of 1 in the column Used by, then it is currently used and cannot be removed. If a module has a different value than 1 in the column Used by, then it is not currently used and can be removed. However, if another module from the same group has a value of 1, the module cannot be removed, because it is dependent on that module. In a later chapter about removing software packages, some of these modules will be taken out.

The next larger-sized files, which can be removed, and which are not usually used, are the rescue files. Usage of these files is required when the operating system crashes or when a problem with software or hardware occurs. These rescue files are always created by the **dracut-config-rescue** package when the operating system is installed. Rescue files are stored in the ***/boot*** directory and are easily recognizable because they contain "rescue" in their name. The size of rescue files varies for every distribution and for every user. In my case, I can save up to 125 MB, which is the size of half of the folder where they are stored.

```
[root@localhost peterk]# find /boot -name *rescue*
/boot/loader/entries/a15e9d1a5d2c44dba56e987f8d333963-0-rescue.conf
/boot/vmlinuz-0-rescue-a15e9d1a5d2c44dba56e987f8d333963
/boot/initramfs-0-rescue-a15e9d1a5d2c44dba56e987f8d333963.img
```

Listing 5.6: List of rescue files

A computer has software called firmware. The device has software, called a driver, that tells the operating system how to communicate with another device. Firmware packages are loaded every time while booting and are loaded into RAM. Deleting every firmware package would lead to a non-functioning operating system. However, deleting those packages, which are not hidden in other folders, does not affect the proper functioning of the operating system. In hidden folders are packages that the kernel needs to boot and be able to communicate with users. Unprotected packages are located at */usr/lib/firmware* and can be removed with the command **rm** and with the added parameter **-rfv**, which can remove a whole folder.

```
[root@localhost peterk]# rm -rfv /usr/lib/firmware
removed '/usr/lib/firmware/3com/3C359.bin'
removed '/usr/lib/firmware/3com/typhoon.bin'
removed directory '/usr/lib/firmware/3com'
removed directory '/usr/lib/firmware/RTL8192E'
removed '/usr/lib/firmware/dvb-usb-it9135-01.fw'
removed '/usr/lib/firmware/qed/qed_init_values-8.10.9.0.bin'
removed '/usr/lib/firmware/advansys/3550.bin'
removed '/usr/lib/firmware/advansys/38C0800.bin'
```

Listing 5.7: Removing unnecessary firmware

A driver that allows operating system and applications to use a computer's graphics hardware is called a graphics driver. It controls how graphic components work with the computer. This driver communicates with the device, usually through the computer bus to which the hardware connects. A graphics driver is not necessary, but when it's installed, it provides more system stability. On Linux, graphics drivers are in the */lib64/dri* directory. There are drivers for various graphics cards, including those that are not physically present on a computer. These drivers have a size of more than 200 MB and are recognizable by the suffix ".so". Deleting is done by the command **rm**.

```
rm /lib64/dri/i965_dri.so
rm /lib64/dri/nouveau_dri.so
rm /lib64/dri/r600_dri.so
rm /lib64/dri/radeonsi_dri.so
rm /lib64/dri/virtio_gpu_dri.so
rm /lib64/dri/vmwgfx_dri.so
```

Listing 5.8: Deleting graphics drivers

5.5 Reduction of software packages

Each Linux distribution is usually installed as a different software package, each of which contains a specific application, such as a web browser. Each package includes an archive of files and information about the software, such as its name and the specific version. Linux's distributions use package managers that automatically read dependency files and download the packages needed before proceeding with the installation. In CentOS Stream, packages can be installed or removed with the **YUM** package management system. Software packages

take up a lot of disk space because they are necessary for the proper functioning of the system and for installed programs. Deleting all packages with all dependent files and folders would free up a lot of space, but the operating system and other installed programmes would stop working correctly. That's because many packages are dependent on other packages, or they provide stable functioning of the operating system and installed programs. The command **yum list installed** can show all installed packages. The number of packages varies for each distribution of Linux. In my case, it's 1522 packages.

```
[root@localhost peterk]# yum list installed
Installed packages
GConf2.x86_64                          3.2.6-22.el8        @AppStream
ModemManager.x86_64                    1.10.8-4.el8        @anaconda
ModemManager-glib.x86_64               1.10.8-4.el8        @anaconda
NetworkManager.x86_64                  1:1.34.0-0.2.el8    @anaconda
NetworkManager-adsl.x86_64             1:1.34.0-0.2.el8    @anaconda
NetworkManager-bluetooth.x86_64        1:1.34.0-0.2.el8    @anaconda
NetworkManager-config-server.noarch    1:1.34.0-0.2.el8    @anaconda
…
```

Listing 5.9: List of a few installed packages

From listing 5.9, three columns can be seen. In the first column is the name of the package; in the second is the current version; and in the third column is information about which installer and when they were installed. For example, **@anaconda** means, that this package was installed during the installation of the operating system, **@AppStream** means, that packages were installed selectively depending on the purpose for which the system is being configured, and **@baseos** indicates the packages that implement the base core functionality of the operating system.

As I mentioned earlier in this chapter, removing every package would result in a non-functioning operating system or installed software. Therefore, main packages, which are responsible for proper functioning, must be protected. The defined packages, in my case, otter-browser, NetworkManager, openssh, xterm, xfce4-session, and XPDF, must not be removed. The network manager is responsible for configuration and connection functionality on the network. Deleting would result in the loss of internet connection and the inability to load a web page in Otter-Browser or any other web browser. Otter-Browser is a lightweight web browser I replaced the Mozilla with. Without openssh, the connection from another

computer would not work. Because of the removed icons and other packages that were responsible for some functions from GUI, it is not possible to start programs from menu, but directly from the command line using the Xterm terminal. Thanks to XPDF, it is possible to open files with the ".pdf" suffix, which will be needed later. The last one xfce4-session is responsible for installed lightweight GUI. Every single package mentioned will be written to a file with the same name and with **".conf"** suffix added into the */etc/dnf/protected.d/* directory, because every time while removing packages, this folder is reviewed. If a match is found with the package being removed, the package is skipped, and a warning message is displayed.

```
echo “openssh” > /etc/dnf/protected.d/openssh.conf
echo “otter-browser” > /etc/dnf/protected.d/otter-browser.conf
echo “NetworkManager” > /etc/dnf/protected.d/NetworkManager.conf
echo “xterm” > /etc/dnf/protected.d/xterm.conf
echo “xfce4-session” > /etc/dnf/protected.d/xfce4-session.conf
echo “xpdf” > /etc/dnf/protected.d/xpdf.conf
```

Listing 5.10: Protecting packages from removal

After protecting packages, the removal may begin. The CentOS Stream offers the removal of packages and all their dependencies with the command **yum autoremove package_name.** Every time before removing a package, a message shows up to confirm or deny an action. This message can be skipped by adding the parameter **-y**.

```
[root@localhost peterk]# yum autoremove baobab.x86_64
Dependencies resolved.
================================================================
 Package       Architecture   Version              Repository        Size
================================================================
Removing:
 baobab        x86_64         3.28.0-4.el8         @AppStream        1.4 M
Transaction Summary
================================================================
Remove  1 Package
Freed space: 1.4 M
Is this ok [y/N]: y
Running transaction check
```

```
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                        1/1
  Erasing          : baobab-3.28.0-4.el8.x86_64                    1/1
  Running scriptlet: baobab-3.28.0-4.el8.x86_64                    1/1
  Verifying        : baobab-3.28.0-4.el8.x86_64                    1/1
Removed:
  baobab-3.28.0-4.el8.x86_64
Complete!
```

Listing 5.11: Removing unprotected package using yum

```
[root@localhost peterk]# yum autoremove NetworkManager.x86_64
Error:
 Problem: The operation would result in removing the following protected packages:
NetworkManager
(try to add '--skip-broken' to skip uninstallable packages or '--nobest' to use not only
best candidate packages)
```

Listing 5.12: Attempt to delete protected package

5.6 Selection and installation lightweight graphical environment

Choosing a graphical environment isn't as difficult as it may seem. The two main conditions were to keep the size as small as possible and to still be able to show the web browser in full resolution with all colours. The default graphical environment was GNOME, and I had a few other environments to select from. In Table 5.2, different environments are compared by size. Since the main thing is to keep the size as small as possible, I opted for the smallest possible one. The most suitable for fulfilling the assignment was a graphical environment called Xfce.

Table 5.2: Comparison of graphical environments

Desktop environment	Installed size (MB)
GNOME (default)	2487
KDE	2198
Xfce	1529

LXDE	1536
MATE	1631
Cinnamon	2212

Installation of Xfce was easy and took only a few minutes. First, I needed to configure the EPEL repository, as it was where I was installing packages from. EPEL (Extra Packages for Enterprise Linux) is an open-source and free community-based repository project from the Fedora team that provides high-quality software packages for the Linux distributions CentOS Stream, Red Hat Enterprise Linux, and Scientific Linux. Installation of EPEL is provided with the **yum install epel-release -y** command. The next group to be installed before proceeding is "Server with GUI". This group will also install GNOME and around 150 packages, but it is able to change to Xfce. This server is responsible for generating the dynamic web-based user interface. Although "Server with GUI" can be replaced with "X Window System", this group has a lack of packages and is unstable, resulting in crashing applications and a freezing operating system. This installation is provided with **yum groupinstall "Server with GUI" -y**. Installing Xfce is now as simple as installing the "Xfce" package group, which is already configured to install more packages than needed. As a last command to finish the installation of Xfce, is **yum groupinstall "Xfce" -y**. It may take a while to download and install packages.

```
yum install epel-release -y
yum groupinstall “Server with GUI” -y
yum groupinstall “Xfce” -y
```

Listing 5.13: Commands used for installation Xfce

```
root@localhost peterk]# yum groupinstall "Xfce" -y
=================================================================
Package                Arch       Version    Repository     Size
=================================================================
Installing group/module packages:
Thunar                 x86_64     4.16.8-1.el8    epel      01.6  M
mousepad               x86_64     0.5.6-1.el8     epel       339  k
thunar-archive-plugin  x86_64     0.4.0-26.el8    epel        85  k
thunar-volman          x86_64     4.16.0-3.el8    epel       215  k
tumbler                x86_64     0.2.7-1.el8     epel       237  k
xfce-polkit            x86_64     0.3-3.el8       epel        25  k
xfce4-appfinder        x86_64     4.16.1-3.el8    epel       282  k
xfce4-panel            x86_64     4.16.3-1.el8    epel      01.1  M
xfce4-power-manager    x86_64     4.16.0-1.el8    epel       788  k
xfce4-pulseaudio-plugin x86_64    0.4.3-3.el8     epel       123  k
xfce4-screensaver      x86_64     4.16.0-3.el8    epel       300  k
```

```
xfce4-session        x86_64  4.16.0-3.el8   epel        541  k
xfce4-settings       x86_64  4.16.2-1.el8   epel       01.2  M
xfce4-terminal       x86_64  0.8.10-2.el8   epel        670  k
xfconf               x86_64  4.16.0-1.el8   epel        192  k
xfdesktop            x86_64  4.16.0-3.el8   epel       01.6  M
xfwm4                x86_64  4.16.1-1.el8   epel        608  k
Installing dependencies:
exo                  x86_64  4.16.2-1.el8   epel        474  k
garcon               x86_64  4.16.1-1.el8   epel        233  k
libXScrnSaver        x86_64  1.2.3-1.el8    appstream    31  k
libdbusmenu          x86_64  16.04.0-12.el8 appstream   140  k
libdbusmenu-gtk3     x86_64  16.04.0-12.el8 appstream    41  k
libmousepad0         x86_64  0.5.6-1.el8    epel        125  k
libxfce4ui           x86_64  4.16.0-2.el8   epel        280  k
libxfce4util         x86_64  4.16.0-4.el8   epel        186  k
pavucontrol          x86_64  3.0-11.el8     appstream   160  k
Installing Groups:
 Xfce
Transaction Summary
============================================================
Install 26 Packages
Total download size: 11 M
Installed size: 49 M
```

Listing 5.14: Installation of Xfce

After installing these packages, the default target should be automatically updated, meaning that after a reboot, the GUI will automatically be loaded. Finally, after a reboot, on the login screen is the option to choose a GUI. However, to save as much space as possible, this login screen will not be available later, and the only option to start this GUI is through a console. To open this console, three keys need to be pressed – **ctrl+alt+f3**. When the console is displayed, the login name and password are entered. Then the last command is **xinit**. After this, the Xfce GUI should be automatically loaded.

5.7 Selection and installation lightweight web browser

Once the Xfce GUI installation is complete, it's time to select and install a lightweight web browser. In table 5.3 is a list of the top 5 Linux lightweight web browsers, but not all are suitable to show the web pages of VUT University of Brno without a change in look. A Web browser requires JavaScript to display websites correctly. Installation of these web browsers is a bit more complicated and time-consuming because none of the needed software packages are available in the CentOS Stream official repositories. This means that they cannot be installed directly with all their dependencies. There are several solutions to the problem with installation. One of them is to find a third-party repository, which would make the job easier.

If the first solution is not successful, the second is to find open-source codes that are freely available for your own use thanks to a license. These codes can then be compiled on a computer. Some browsers share the same libraries, which must be added before compilation, or will be added with a third-party repository. Without the added libraries, the compilation itself wouldn't even start because the browser constantly requires these libraries. The comparison of individual lightweight browsers with libraries is in the table. Every browser requires a different number of libraries for proper operation. Some browsers share the same libraries. For example, the Otter browser shares a few libraries with Falkon, and Falkon shares the libcrypto library with Links and Netsurf browsers, but these two browsers cannot be used because they do not include JavaScript.

Table 5.3: Comparison of lightweight web browsers

Web browser	Dependencies	Installation size (MB)
Midori	glibc, ld-linux-aarch64.so.1, libarchive.so.1.3, libc.so.6, libcairo-gobject.so.2, libcairo.so.2, libgcr-base-3.so.1, libgcr-ui-3.so.1, libgdk-3.so.0, libgdk_pixbuf-2.0.so.0, libgio-2.0.so.0, libglib-2.0.so.0, libgobject-2.0.so.0, libgtk-3.so.0, libjavascriptcoregtk-4.0.so.1.8, libjson-glib-1.0.so.0, libpeas-1.0.so.0, libpeas-gtk-1.0.so.0, libsoup-2.4.so.1, libsqlite3.so.0, lib-webkit2gtk-4.0.so.37, rtld	4,3
Links	libbz2.so.1, libc.so.6, libcom_err.so.2, lib-crypto.so.1.1, libdl.so.2, libexpat.so.1, lib-gpm.so.2, libgssapi_krb5.so.2, libidn2.so.0, libk5crypto.so.3, libkrb5.so.3, liblua-5.3.so, libm.so.6, libssl.so.1.1, libz.so.1	2,9
Falkon	libQt5Core.so.5, libQt5DBus.so.5, libQt5Gui.so.5, libQt5Network.so.5, libQt5Po-sitioning.so.5, libQt5PrintSupport.so.5, libQt5Qml.so.5, libQt5Quick.so.5, libQt5QuickWidgets.so.5, libQt5Sql.so.5, libQt5WebChannel.so.5,	12,4

	libQt5WebEngineCore.so.5, libQt5WebEngineWidgets.so.5, libQt5Widgets.so.5, libQt5X11Extras.so.5, libc.so.6, libcrypto.so.1.1, libgcc_s.so.1, libm.so.6, libstdc++.so.6, libxcb.so.1, qt5-qtbase, qt5-qtwebengine	
Otter browser	libKF5SonnetCore.so.5, libQt5Core.so.5, libQt5DBus.so.5, libQt5Gui.so.5, libQt5Multimedia.so.5, libQt5Network.so.5, libQt5PrintSupport.so.5, libQt5Script.so.5, libQt5Sql.so.5, libQt5WebKit.so.5, libQt5WebKitWidgets.so.5, libQt5Widgets.so.5, libQt5XmlPatterns.so.5, libc.so.6, libgcc_s.so.1, libm.so.6, libpthread.so.0, libstdc++.so.6	15
Netsurf	libatk-1.0.so.0, libc.so.6, libcairo-gobject.so.2, libcairo.so.2, libcrypto.so.1.1, libcurl.so.4, libexpat.so.1, libgdk-3.so.0, libgdk_pixbuf-2.0.so.0, libgio-2.0.so.0, libglib-2.0.so.0, libgmodule-2.0.so.0, libgobject-2.0.so.0, libgthread-2.0.so.0, libgtk-3.so.0, libjpeg.so.62, libm.so.6, libpango-1.0.so.0, libpangocairo-1.0.so.0, libpng16.so.16, libpthread.so.0, librsvg-2.so.2, libssl.so.1.1, libz.so.1	6,5

The browser is selected by size and by properties. Browsers that are suitable have a graphical environment, so Links doesn't belong here because it is a text-based web browser. Netsurf, Falkon, and Midori have the smallest size after installation, but the final size is much bigger due to all the necessary libraries and dependencies. I have chosen the Otter browser because of its smallest size after installation and for all the necessary properties.

Lightweight web browser installation

As I mentioned in chapter 5.7, there are several solutions to installing software packages that are not available in the CentOS Stream official repositories. I was able to find a third-party repository, which helped me to save some time. This third-party repository is called **Nux**

Dextop. As a first step, the latest release of Nux Dextop must be downloaded. Secondly, this release must be installed via **rpm**. Third, installation of the otter-browser package.

```
[root@localhost Downloads]# rpm -Uvh nux-dextop-release*rpm
warning: nux-dextop-release-0-5.el7.nux.noarch.rpm: Header V4 RSA/SHA1 Signa-
ture, key ID 85c6cd8a: NOKEY
Verifying...                          ################################# [100%]
Preparing...                          ################################# [100%]
Updating / installing...
   1:nux-dextop-release-0-5.el7.nux   ################################
[100%]
[root@localhost Downloads]# yum install otter-browser
Nux.Ro RPMs for general desktop use          1.8 MB/s | 4.2 MB     00:02
Last metadata expiration check: 0:00:02 ago on Mon 06 Dec 2021 10:06:50 AM EST.
Dependencies resolved.
=============================================================
Package          Arch       Version                      Repository        Size
=============================================================
Installing:
otter-browser    x86_64     0.9.09-                      nux-dextop        2,2 M
                            0.2.beta9gitff0bb28.el7.nux
Installing dependencies:
kf5-sonnet-core      x86_64     5.88.0-1.el8          epel          212 k
libatomic            x86_64     8.5.0-4.el8_5         baseos         24 k
openal-soft          x86_64     1.18.2-7.el8          appstream     394 k
pcre2-utf16          x86_64     10.32-2.el8           baseos        229 k
qt5-qtbase           x86_64     5.15.2-4.el8          appstream     3.6 M
qt5-qtbase-common    noarch     5.15.2-4.el8          appstream      41 k
qt5-qtbase-gui       x86_64     5.15.2-4.el8          appstream     6.1 M
qt5-qtdeclarative    x86_64     5.15.2-2.el8          appstream     4.2 M
qt5-qtlocation       x86_64     5.15.2-2.el8          appstream     3.3 M
qt5-qtmultimedia     x86_64     5.15.2-2.el8          appstream     883 k
qt5-qtscript         x86_64     5.15.2-2.el8          appstream     1.1 M
qt5-qtsensors        x86_64     5.15.2-2.el8          appstream     220 k
```

```
qt5-qtwebchannel      x86_64   5.15.2-2.el8              appstream   102 k
qt5-qtwebkit          x86_64   5.212.0-0.60.alpha4.el8   epel        13 M
qt5-qtxmlpatterns     x86_64   5.15.2-2.el8              appstream   1.1 M
xcb-util-image        x86_64   0.4.0-9.el8               appstream   21 k
xcb-util-keysyms      x86_64   0.4.0-7.el8               appstream   16 k
xcb-util-renderutil   x86_64   0.3.9-10.el8              appstream   19 k
xcb-util-wm           x86_64   0.4.1-12.el8              appstream   32 k
Transaction Summary
================================================================
Install 20 Packages
Total download size: 37 M
Installed size: 125 M
Is this ok [y/N]: y
```

Listing 5.15: Nux dextop and Otter browser installation

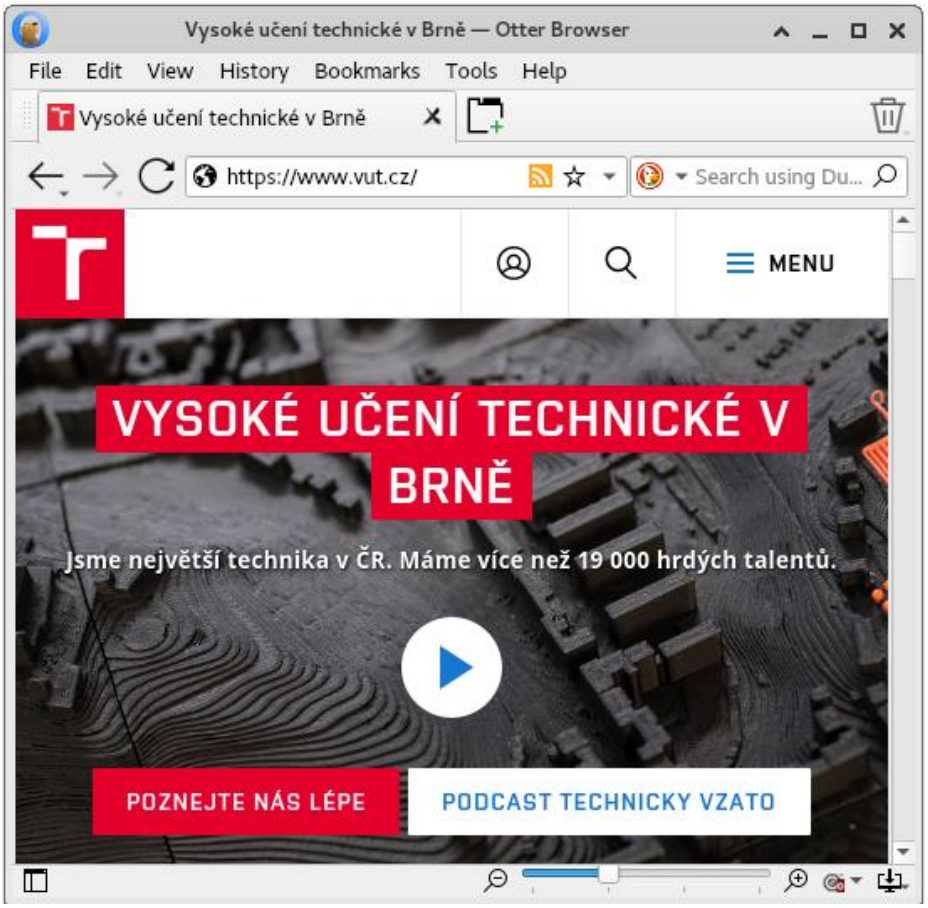

Figure 5.1: Otter browser

5.8 Automated reduction

My script, which is used for reduction, consists of more than 1800 lines, and is divided into 3 different parts. In the first part, I installed libraries for Otter-Browser, GUI Xfce, XTerm, and XPDF and added important libraries and packages into the protected directory. Packages which are added in this directory are protected from being removed, even using the ***yum autoremove -y*** command. In this part, no disk space is freed. Using this part of the script, I added 1400 MB to the disk.

```
#!/bin/sh
#Installation of the libraries needed for Otter-Browser
dnf -y install qt5-qtbase glibc libstdc++ qt5-qtxmlpatterns qt5-qtbase-gui qt5-qtbase-
devel qt5-qtmultimedia-devel qt5-qtsvg-devel qt5-qtwebkit-devel qt5-qtwebengine
qt5-qtwebchannel hunspell qt5-qtmultimedia
#Installation of development tools for Xfce GUI
dnf -y groupinstall "Development Tools"
dnf install epel-release -y
dnf --enablerepo=epel group
dnf config-manager --set-enabled powertools
dnf groupinstall -y "Xfce" "base-x"
#Command to start GUI from the command line
echo "exec /usr/bin/xfce4-session" >> ~/.xinitrc
#Setting GUI login instead of using command line, won't work later
systemctl set-default graphical
#Installation of Xfce GUI
dnf -y install xfce4-session
yum groupinstall -y "Server with GUI"
dnf install -y xorg-x11-server-Xorg
#Installation of third-party repositories needed for installation of Otter-Browser
dnf -y install https://pkgs.dyn.su/el8/base/x86_64/raven-release-1.0-2.el8.noarch.rpm
dnf --enablerepo=raven-multimedia
dnf --enablerepo=raven-extras
#Installation of Otter-Browser
dnf install -y otter-browser
#Installation of necessary libraries for XTerm and its installation
dnf install -y xcb-util xterm
#Installation of PDF file viewer XPDF
dnf install -y https://download-ib01.fedoraproject.org/pub/epel/7/x86_64/Pack-
ages/x/xpdf-3.04-10.el7.x86_64.rpm
dnf install -y motif libpng15 poppler-utils urw-fonts xorg-x11-fonts-ISO8859-1-
100dpi xorg-x11-fonts-ISO8859-1-75dpi
```

```
#Protecting libraries and applications against removing
echo "otter-browser" > /etc/dnf/protected.d/otter-browser.conf
echo "accountsservice" > /etc/dnf/protected.d/accountsservice.conf
echo "xpdf" > /etc/dnf/protected.d/xpdf.conf
echo "libQt5WebEngineWidgets" > /etc/dnf/protected.d/libQt5WebEngineWidgets
echo "libQt5Multimedia" > /etc/dnf/protected.d/libQt5Multimedia.conf
echo "libQt5WebEngineCore" > /etc/dnf/protected.d/libQt5WebEngineCore.conf
echo "xorg-x11-fonts-ISO8859-1-100dpi" > /etc/dnf/protected.d/xorg-x11-fonts-
ISO8859-1-100dpi.conf
#Removing unnecessary software and folders
yum autoremove -y gnome PackageKit.x86_64 chrome-gnome-shell.x86_64 gnome-
control-center-filesysyem.noarch gnome-autoar.x86_64
rm -rfv /mockbuild
rm -rfv /home/peterk/otter-browser-master/build/ …
```

Listing 5.16: The first section of the script

In the second part of the script, I started to remove packages and all their dependencies using the command ***yum autoremove -y***. This means that I don't have to accept every package manually to be removed. This part of the script consists of more than 1400 lines and includes every installed package in the minimal installation of CentOS Stream. A short example is shown in the listing 5.17. Using this part of the script, I freed 4200 MB of disk space.

```
sudo yum autoremove -y GConf2.x86_64 ModemManager.x86_64 ModemManager-
glib.x86_64 NetworkManager.x86_64 NetworkManager-adsl.x86_64 NetworkMana-
ger-bluetooth.x86_64 NetworkManager-config-server.noarch NetworkManager-
libnm.x86_64 NetworkManager-team.x86_64 NetworkManager-tui.x86_64 Ne-
tworkManager-wifi.x86_64 NetworkManager-wwan.x86_64 PackageKit-glib.x86_64
PackageKit-gstreamer-plugin.x86_64 PackageKit-gtk3-module.x86_64
Thunar.x86_64 abattis-cantarell-fonts.noarch accountsservice.x86_64 accountsser-
vice-libs.x86_64 acl.x86_64 adcli.x86_64 adobe-mappings-cmap.noarch adobe-
mappings-cmap-deprecated.noarch adobe-mappings-pdf.noarch adwaita-cursor-
theme.noarch adwaita-gtk2-theme.x86_64 adwaita-icon-theme.noarch . . . xz-
libs.x86_64 yajl.x86_64 yelp.x86_64 yelp-libs.x86_64 yelp-tools.noarch yelp-xsl.no-
arch yum.noarch yum-utils.noarch zenity.x86_64 zip.x86_64 zlib.x86_64 zlib-de-
vel.x86_64 zstd.x86_64 ...
```

Listing 5.17: The script's second section

In the final part of the script, I removed all the remaining and unused software, which wasn't removed during the second part. This happened because some software packages share the same libraries, which I added to the protected directory and thus were protected against being removed using the yum command. At the end, the script removed some un-necessary directories with files and cache. SSH also had to be reinstalled. Using this 160-line part, which is shown in the listing 5.18, I freed another 1920 MB. The final size of the minimized CentOS Stream is 678 MB.

```
#!/bin/bash
#Removing log and rescue files
find /var/log -name '*.log*' -delete
find /boot -name *rescue* -delete
cd /boot
#Removing all system boot images except the currently used ones
shopt -s extglob
rm -rfv !("grub2"|"vmlinuz-4.18.0-383.el8.x86_64"|"initramfs-4.18.0-
383.el8.x86_64.img"|"config-4.18.0-383.el8.x86_64"|"loader")
#Installation and start of SSH
dnf -y install openssh-server openssh-clients
systemctl start sshd
#Removing files and folders
rm -rfv ./.cache
rm -rfv ./.mozilla
rm -rfv /var/cache/*
rm -rfv /etc/firewalld/
rm -rfv /etc/services
rm -rfv /etc/udev
rm -rfv /home/peterk/.cache/*
rm -rfv /home/peterk/.local/
rm -rfv /home/peterk/.mozilla
rm -rfv /home/peterk/Downloads/
rm -rfv /home/peterk/master.zip
rm -rfv /home/peterk/otter-browser-master/
rm -rfv /home/peterk/rpmbuild/sources
```

```
rm -rfv /home/rpmbuild/sources
rm -rfv /lib/modules
rm -rfv /lib64/bluetooth
rm -rfv /lib64/dri/i965_dri.so…
```

Listing 5.18: The script's final section

5.9 Start of the OS after reduction

While performing reduction, the OS must be rebooted several times. Once after the first and a third scripts, and twice during the second script. Every script is performed as a superuser.

After the first script, all the necessary programmes are installed, so there are two ways to restart the system. The first way after the script is to type "reboot" into the terminal, and after reboot, press alt + F3 and then type "xinit". The second, easier way is to manually reboot using the button on the desktop and then manually select on the login screen "Xfce session". After a successful start of the OS, the default Xfce terminal is used to launch the XTerm terminal using the command "xterm". The default terminal will no longer be used.

During the second script, the system automatically reboots once due to removing a system package. The system can be started using the "xinit" command. This time, after the second start of this script, the system is not automatically rebooted, and the script will finish successfully. Then, the system is manually rebooted using the commands "reboot" and "xinit".

The third and final script will run without any problems. The system is rebooted in the same way as after the second script. After a successful reboot, XTerm is automatically launched.

5.10 Summary of reduction

After a reduction performed by script, the size of CentOS Stream is 678 MB. The web browser, with all its needed dependencies, takes up one third of this size. The remaining two thirds are occupied by the reduced operating system. From table 5.4, it is possible to see the size of folders before, while, and after reduction. The least minimized directory is */var/log*. On the other hand, the most minimized directory is */usr* with all its subdirectories. The resulting values were caused by the removal of unnecessary software packages, firmware, and libraries. These values vary in other Linux distributions.

Table 5.4: Folder comparison after reduction

Directory	Before reduction [MB]	After first script [MB]	After second script [MB]	After third script [MB]
/root	5400	6800	2600	678
/etc	31	31	12	2,3
/home	86	123	128	5,2
/tmp	36 kB	12 kB	20 kB	48 kB
/usr	4300	5500	1300	590
/usr/bin	232	320	44	41
/usr/include	72 kB	54	532 kB	0
/usr/lib	1016	1400	367	11
/usr/sbin	73	73	11	9,3
/usr/lib64	1200	1500	488	423
/usr/share	1800	2000	349	100
/var	730	855	830	14
/var/log	7,1	7,2	12	7

Along with the size, the number of software packages has also decreased substantially. From the original, 1522 software packages in this Linux distribution, the number has been reduced to 560, which is almost a third of all the default installed packages.

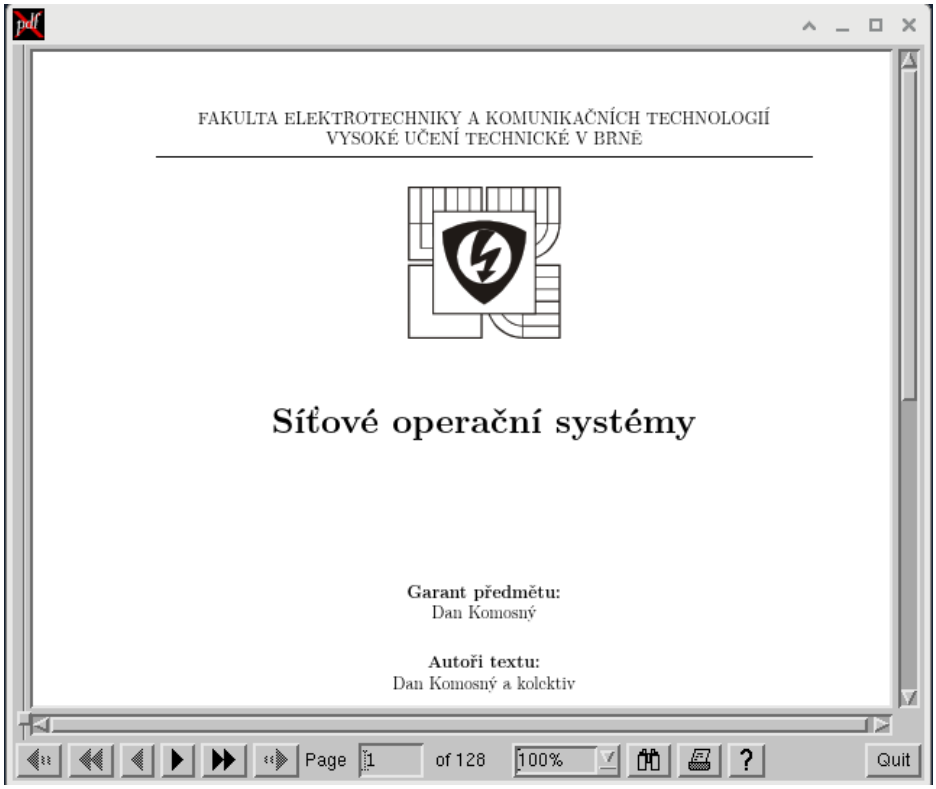

Figure 5.2: Opened scripts in XPDF

6 Conclusion

This was developed to help with the project for the course of Networking Operating Systems. I reduced the size of the CentOS Stream operating system from the viewpoint of its size on storage media using a set of scripts. The reduced system includes a graphical web browser (Otter browser), a PDF file viewer (XPFD) and remote access via SSH. System packages, programs, and kernel modules were mostly removed. The default installation of CentOS Stream with GUI was 5,4 GB, but after reduction, it was only 678 MB. The number of software packages has been reduced by 962. Originally, 1522 packages have been reduced to 560. The most minimized directory is */usr* with all its subdirectories. From the default 4,3 GB, the size was reduced to 590 MB.

Xfce GUI was selected and used. Otter browser was chosen as the most suitable web browser. This browser displays the website in its original form. Because this browser does not support viewing PDF files, an XPDF viewer was additionally installed to display downloaded .pdf scripts. The overall reduction of the system was performed so as not to interfere with functionality.

Bibliography

[1] The Linux Foundation. What is Linux? [online]. 2021 [cit. 2021-10-3]. Accessible from: **https://www.linux.com/what-is-linux/**

[2] Red Hat. What is the difference between Unix and Linux? [online]. 2021 [cit. 2021-10-3]. Accessible from: **https://opensource.com/resources/linux**

[3] TechTarget Contributor. Boot loader [online]. 2021 [cit. 2021-10-3]. Accessible from: **https://searchdatacenter.techtarget.com/definition/boot-loader-boot-manager**

[4] SUSE. Linux Kernel [online]. 2021 [cit. 2021-10-3]. Accessible from: **https://www.suse.com/suse-defines/definition/linux-kernel/**

[5] Arnab Chakraborty. Init process on UNIX and Linux systems [online]. 2019 [cit. 2021-10-5]. Accessible from: **https://www.tutorialspoint.com/init-process-on-unix-and-linux-systems**

[6] Robert Junior. What does init do in Linux? [online]. 2021 [cit. 2021-10-5]. Accessible from: **https://techtrix.co/what-does-init-0-do-in-linux/**

[7] Bill Dyer. What are daemons in Linux? [online]. 2021 [cit. 2021-10-5]. Accessible from: **https://itsfoss.com/linux-daemons/**

[8] Dimitrios. What is display server in Linux? [online]. 2020 [cit. 2021-10-5]. Accessible from: **https://itsfoss.com/display-server/**

[9] Abhishek Prakash. What is Desktop Environment in Linux? [online]. 2020 [cit. 2021-10-5]. Accessible from: **https://itsfoss.com/what-is-desktop-environment/**

[10] Red Hat. What is open-source? [online]. 2019 [cit. 2021-10-5]. Accessible from: **https://www.redhat.com/en/topics/open-source/what-is-open-source**

[11] Chris Hoddman. What Is a Linux Distro, Anyway? [online]. 2016 [cit. 2021-10-12]. Accessible from: **https://www.howtogeek.com/132624/htg-explains-whats-a-linux-distro-and-how-are-they-different/**

[12] Red Hat. What is Red Hat Enterprise Linux? [online]. 2021 [cit. 2021-10-12]. Accessible from: **https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux**

[13] Chris Wright. Transforming the development experience within CentOS [online]. 2019 [cit. 2021-10-12]. Accessible from: **https://www.redhat.com/en/blog/transforming-development-experience-within-centos**

[14] GeeksforGeeks. Linux File Hierarchy Structure [online]. 2021 [cit. 2021-10-12]. Accessible from: **https://www.geeksforgeeks.org/linux-file-hierarchy-structure/**

[15] The Linux Information Project. The /root Directory [online]. 2005 [cit. 2021-10-12]. Accessible from: **http://www.linfo.org/slash_root.html**

[16] Jithin. Linux Binary Directories Explained [online]. 2017 [cit. 2021-10-12]. Accessible from: **https://www.interserver.net/tips/kb/linux-binary-directories-explained/**

[17] Skillset. What is the purpose of /boot folder in Linux? [online]. 2016 [cit. 2021-10-12]. Accessible from: **https://www.skillset.com/questions/what-is-the-purpose-of-boot-folder-in-linux**

[18] Binh Nguyen. Linux Filesystem Hierarchy: /boot [online]. 2004 [cit. 2021-10-15]. Accessible from: **https://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/boot.html**

[19] Binh Nguyen. Linux Filesystem Hierarchy: /dev [online]. 2004 [cit. 2021-10-15]. Accessible from: **https://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/dev.html**

[20] David Both, DataBook, The Linux Documentation Project, Kernel.org. Understanding the /dev directory [online]. 2021 [cit. 2021-10-22]. Accessible from: **http://www.linux-databook.info/?page_id=5108**

[21] Binh Nguyen. Linux Filesystem Hierarchy: /etc [online]. 2004 [cit. 2021-10-22]. Accessible from: **https://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/etc.html**

[22] Binh Nguyen. Linux Filesystem Hierarchy: /home [online]. 2004 [cit. 2021-10-22]. Accessible from: **https://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/home.html**

[23] Unlisted. Linux Kernel Modules [online]. 2021 [cit. 2021-11-3]. Accessible from: **https://eng.libretexts.org/Bookshelves/Computer_Science/Operating_Systems/Linux_-_The_Penguin_Marches_On_(McClanahan)/06%3A_Kernel_Module_Management/2.04%3A_Kernel_Modules**

[24] Peter Jay Salzman, Michael Burian, Ori Pomerantz. The Linux Kernel Module Programming Guide: Compiling Kernel Modules [online]. 2007 [cit. 2021-11-3]. Accessible from: **https://tldp.org/LDP/lkmpg/2.6/html/x181.html**

[25] Kuntal Chakraborty. Firmware [online]. 2021 [cit. 2021-11-3]. Accessible from: **https://www.techopedia.com/definition/2137/firmware**

[26] Peter Loshin. Secure Shell (SSH) [online]. 2021 [cit. 2021-11-3]. Accessible from: **https://www.techtarget.com/searchsecurity/definition/Secure-Shell**

[27] Tatu Ylonen: SSH - Secure Login Connections over the Internet. Proceedings of the 6th USENIX Security Symposium, pp. 37-42, USENIX, [online]. 1996 [cit. 2021-11-10]. Accessible from: **https://www.ssh.com/academy/ssh/protocol**

[28] Phil Dibowitz, CentOS Stream: Why It's Awesome [online]. 2021 [cit. 2021-12-6]. Accessible from: **https://medium.com/swlh/centos-stream-why-its-awesome-c45d944fb22**

[29] Techopedia, Red Hat Enterprise Linux (RHEL) [online]. 2017 [cit. 2021-12-6]. Accessible from: **https://www.techopedia.com/definition/15777/red-hat-enterprise-linux-rhel**

[30] Fedora [online]. 2022 [cit. 2022-4-14]. Accessible from: **https://getfedora.org/**

[31] TechTarget Contributor, What is Fedora? [online]. 2010 [cit. 2022-4-14]. Accessible from: **https://www.techtarget.com/searchdatacenter/definition/Fedora**

[32] Forrest Stroud, Fedora Linux [online]. 2012 [cit. 2022-4-14]. Accessible from: **https://www.webopedia.com/definitions/fedora/**

[33] Artem Ilyin, Debian [online]. 2022 [cit. 2022-4-14]. Accessible from: **https://golden.com/wiki/Debian-5DZ8**

[34] Mohammed Abubakar, What is Ubuntu? [online]. 2021 [cit. 2022-4-15]. Accessible from: **https://www.howtogeek.com/763775/what-is-ubuntu/**

[35] Peter Loshin, Stephen J. Bigelow, Linux Operating System [online]. 2022 [cit. 2022-4-15]. Accessible from: **https://www.techtarget.com/searchdatacenter/definition/Linux-operating-system**

[36] TechTarget Contributor, Apache [online]. 2022 [cit. 2022-4-15]. Accessible from: **https://www.techtarget.com/whatis/definition/Apache**

List of used abbreviations, quantities, and symbols

GUI	Graphical User Interface
kB	kilobyte
MB	Megabyte
GB	Gigabyte
SSH	Secure Shell
API	Application Programming Interface
GRUB	GRand Unified Bootloader
BIOS	Basic Input Output System
RHEL	Red Hat Enterprise Linux

LIST OF APPENDICES

Appendix A – Scripts 54
Appendix B – Running the scripts 55

Appendix A – Scripts

script1.bash

script2.bash

script3.bash

Appendix B – Running the scripts

Archive contains three files, each of which contains commands that have been executed. These scripts can be downloaded from: https://drive.google.com/drive/folders/114QcJS4rajWBAdnhEq5G72YLfB7CJdok?usp=sharing

In file script3.bash is necessary to edit the number of currently used boot image in the command "rm -rfv !("grub2"|"vmlinuz-XXX.el8.x86_64"|"initramfs-XXX.el8.x86_64 .img"|"config-XXX.el8.x86_64"|"loader")", in my case it was **4.18.0-383.el8.x86_64**.

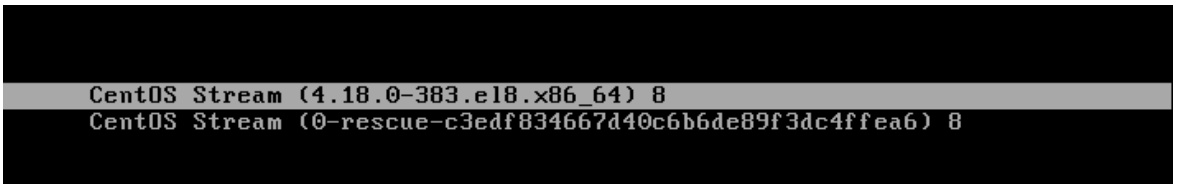

Figure B.1: Currently used boot image

Script execution procedure

1. save downloaded scripts to /root directory and .pdf scripts to /home directory
2. su root, then cd /root
3. sh script1.bash
4. reboot
5. manually choose "Xfce session" at login screen
6. open terminal
7. start XTerm in terminal using command xterm
8. su root, then cd /root
9. sh script2.bash
10. while executing this script, CentOS may restart, in that case it is necessary to login using key combination alt + F3
11. after successful login, use xinit command
12. XTerm should be already running, if not, start it manually and use commands su root, cd /root and sh skript2.bash
13. reboot, login alt + F3, xinit, su root, cd /root
14. sh script3.bash
15. in XTerm use exit command
16. commands otter-browser, xpdf