VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

LINUXOVÁ DISTRIBUCE PRO SENIORY

LINUX DISTRIBUTION FOR SENIORS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE Tarik Alkanan

AUTHOR

VEDOUCÍ PRÁCE prof. Ing. Dan Komosný, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Tarik Alkanan ***ID:*** 221267

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Linuxová distribuce pro seniory

POKYNY PRO VYPRACOVÁNÍ:

Upravte distribuci operačního systému s jádrem Linux. Distribuce bude mít jednoduché grafické rozhraní, které bude vhodné pro seniory ve věkové skupině 90 let a více. Grafické rozhraní bude umožňovat spuštění zadaných aplikací pro seniory. Grafické rozhraní vytvořte v programovacím jazyce Python. Upravená distribuce bude provozována na USB disku. Výsledky práce publikujte na repozitáři GitHub pod licencí MIT.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: prof. Ing. Dan Komosný, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce se zaměřuje na modifikaci operačního systému s linuxovým jádrem tak, aby byl spustitelný z USB disku. Výsledný systém umožňuje uživatelům snadný přístup k modifikovanému operačnímu systému na libovolném počítači. Kromě toho byl vyvinut spouštěč aplikací pro seniory napsané v jazyce Python a speciálně navržený pro seniory ve skupině 90 let. Aplikace poskytuje jednoduché a uživatelsky přívětivé rozhraní, které seniorům umožňuje spouštět běžně používané aplikace. Výsledky této práce jsou k dispozici na repositáři GitHub.

KLÍČOVÁ SLOVA

Linuxová distribuce pro seniory, Operační systém, Seniory, Fedora, USB, Tkinter, Python

ABSTRACT

This Bachelor thesis focuses on modifying the operating system with a Linux kernel so that it can be run from a USB drive. The resulting system allows users to easily access the modified operating system on any computer. In addition, an app launcher was developed for seniors using the Python language and specifically designed for seniors over the age of 90. The app provides a simple and user-friendly interface that allows seniors to run commonly used applications. The results of this work are available in the Github repository.

KEYWORDS

Linux distribution for seniors, Operating system, Seniors, Fedora, USB, Tkinter, Python

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

ALKANAN, Tarik. *Linuxová distribuce pro seniory*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 76 s. Bakalářská práce. Vedoucí práce: prof. Ing. Dan Komosný, Ph.D

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Tarik Alkanan
VUT ID autora:	221267
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Linuxová distribuce pro seniory

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu prof.Ing. Danovi Komosnému, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

S rostoucím používání počítačů a internetu se senioři potýkají s rostoucím počtem problémů při navigaci ve složitých systémech. Pro řešení těchto problémů byl vytvořen projekt Operační systém pro seniory, který má seniorům poskytnout jednodušší a uživatelsky přívětivější prostředí. Cílem této bakalářské práce je dosáhnout tohoto cíle úpravou operačního systému linuxového jádra tak, aby byl spustitelný z USB disku a poskytoval seniorům přístup ke stejnému operačnímu systému na jakémkoliv počítači. Dále byla vytvořena jednoduchá grafická aplikace uživatelského rozhraní (GUI) v Pythonu s názvem Spouštěč aplikací pro seniory (*srun*), která nahrazuje tradiční desktopové okno zjednodušeným rozhraním, což umožňuje seniorům snadný přístup a spouštění jejich aplikací.

V první teoretické části je uveden přehled operačních systémů a jejich součástí se zaměřením na operační systémy s linuxovým jádrem, včetně jeho architektury a hlavních vlastností. V druhé části je představen přehled systému X Window, který je zodpovědný za správu grafických uživatelských rozhraní (GUI) v operačních systémech založených na Linuxu. Dále jsou popsání některé populární desktopové manažery jako Gnome a KDE, které poskytují uživatelsky rozhraní pro uživatele. Třetí část pokrývá spouštěcí proces operačního systému. Je vysvětleno, jak je hardware inicializován systémem BIOS/UEFI a jak je jádro operačního systému zavaděčem načteno do paměti. V poslední části jsou popsané moduly Pythonu potřebné k vytvoření aplikace (Spouštěč aplikací pro seniory). To zahrnuje moduly pro programování GUI, zpracování souborů a systémovou interakci.

Praktická část této bakalářské práce je rozdělena do tří částí. První část zahrnuje kroky k vytvoření aplikace (Spouštěč aplikací pro seniory), včetně její struktury a způsobu spuštění programu ve virtuálním prostředí Pythonu. Druhá část obsahuje podrobné vysvětlení postupu instalace operačního systému na USB disk. A konečně třetí část se týká instalace všech aplikací pro seniory, které byly vytvořeny v rámci projektu (Operační systém pro seniory).

Bakalářská práce je zakončena shrnutím jejích výsledků, které obsahuje jasný popis vyřešeného problému a přehled navrhovaných cílů pro další zlepšení aplikace.

1 Operační systémy

Popis operačního systému

Operační systém (OS) je kolekce softwaru, která spravuje a řídí hardwarové a softwarové prostředky počítače. Působí jako prostředník mezi uživatelem a počítačovým hardwarem, což uživateli umožňuje interakci s hardwarem počítače. Operační systém odpovídá za správu paměti počítače, plánování úkolů a procesů, řízení vstupních a výstupních operací, zabezpečení a řízení přístupu a poskytování uživatelského rozhraní [1].

Operační systém se skládá z několika komponent včetně jádra, ovladačů zařízení, systémových knihoven a systémových utilit. Jádro je základní složkou operačního systému a spravuje systémové prostředky, jako jsou paměť, procesor a vstupní/výstupní zařízení. Ovladače zařízení umožňují operačnímu systému interakci s hardwarovými zařízeními připojenými k počítači. Systémové knihovny jsou sbírky předem napsaného kódu, které aplikacím poskytují běžně používané funkce [2].
Lze počítačový systém rozdělit na čtyři komponenty, hardware, operační systém, aplikační programy a uživatele [1].

- **Aplikační program**, je to softwarový balíček, který vykonává specifickou funkci přímo pro koncového uživatele nebo pro jinou aplikaci. Aplikace může být samostatná nebo skupina programů. Program je sada operací, která spouští aplikaci pro uživatele.
- **Operační systém**, je zásadní softwarová součást, která spravuje a řídí hardwarové a softwarové prostředky počítače a poskytuje rozhraní pro interakci uživatele s počítačem.
- **Hardware**, poskytuje základní výpočetní prostředky pro systém, jako centrální procesorová jednotka (CPU), paměť (RAM), pevný disk (HDD) vstupně/výstupní zařízení (I/O).

2 Operační systémy s jádrem Linux

Operační systém s linuxovým jádrem je typ počítačového operačního systému, který je vybudován pomocí linuxového jádra, open-source monolitického unixového operačního systému [3]. Linus Torvalds poprvé zveřejnil linuxové jádro v roce 1991 a od té doby se stalo jedním z nejpoužívanějších jader na světě, které pohání celou řadu zařízení od osobních počítačů až po superpočítače, servery a vestavěná zařízení.

Linuxové jádro je známé svou stabilitou, bezpečností a flexibilitou. Poskytuje základní funkce operačního systému, včetně ovladačů zařízení, správy procesů, správy paměti a podpory systému souborů. Operační systém sestavený s linuxovým jádrem je obvykle distribuován jako linuxová distribuce, která zahrnuje nejen jádro, ale také aplikací a knihoven. Příklady populárních linuxových distribucí jsou Fedora, CentOS, Debian a Ubuntu [3].
Vzhledem k tomu, že linuxové jádro je open-source, mohou jej vývojáři upravovat a přizpůsobovat podle svých specifických potřeb. To vedlo k vývoji široké škály specializovaných operačních systémů založených na Linuxu, včetně těch, které jsou určeny pro specifická odvětví, jako je zdravotnictví nebo finance, a těch, které jsou optimalizovány pro určité typy zařízení, jako jsou smartphony nebo routery.
Je nezbytně tedy rozlišovat mezi linuxovým jádrem (kernel) a kompletním linuxovým systémem (Distribuce) [1].

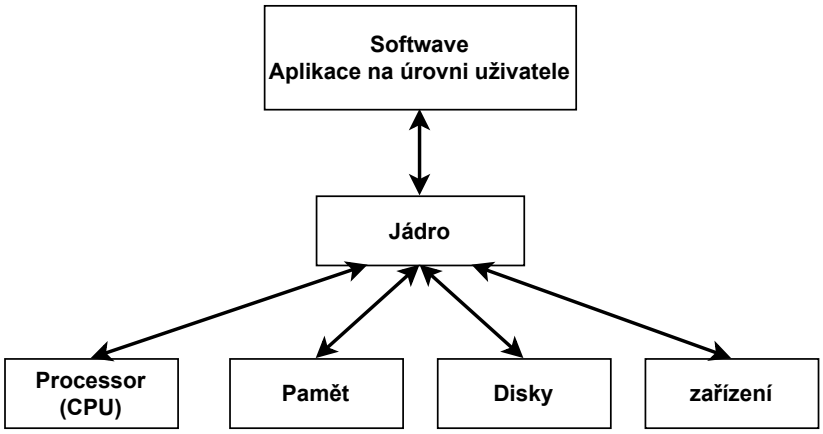
- **Linuxové jádro (kernel)**, je originální kus softwaru vyvinutý od nuly komunitou Linuxu.
- **Linuxový systém (Distribuce)**, obsahuje velké množství komponent, některé jsou napsané od nuly, nebo jsou vypůjčené z jiných vývojových projektů, které spoléhají na jádro a poskytují funkce, s nimiž uživatelé komunikují.

2.1 Jádro Linux

Linuxové jádro je základní součástí linuxového operačního systému. Odpovídá za správu systémových prostředků, jako jsou procesor, paměť a vstupní/výstupní zařízení, a za poskytování služeb aplikacím, které běží nad ním.
Mezi klíčové funkce linuxového jádra patří[1]:

1. **Správa procesů:** Jádro spravuje procesy, což jsou spuštěné instance programů. Plánuje procesy tak, aby běžely na CPU, přiděluje procesům paměť a spravuje meziprocesovou komunikaci.
2. **Správa paměti:** Jádro zodpovídá za správu paměti systému, včetně přidělování paměti procesům, mapování paměti na disk a případné výměny paměti na disk.

3. **Správa ovladačů zařízení:** Jádro poskytuje vrstvu abstrakce mezi hardwarovými zařízeními a aplikacemi. Spravuje ovladače zařízení, což jsou softwarové komponenty, které se propojují s hardwarovými zařízeními a poskytují standardizované rozhraní pro aplikace.
4. **Správa souborového systému:** Jádro spravuje souborové systémy, které jsou způsobem ukládání a organizování dat na discích. Poskytuje standardizované rozhraní pro aplikace pro přístup k souborům a adresářům.
5. **Správa sítě:** Jádro poskytuje síťové možnosti, včetně správy síťových rozhraní, směrování paketů a poskytování protokolů pro komunikaci mezi systémy.



Obr. 2.1: Linuxové jádro

2.2 Projeckt GNU

Project GNU je svobodný a open-source softwarový projekt, který spustil Richard Stallman v roce 1983. Jeho cílem je vytvořit zcela svobodný a uživatelsky přívětivý operační systém, který je kompatibilní s Unixem. Název "GNU" znamená "GNU's Not Unix", což odráží záměr projektu vytvořit Unix-like operační systém, který je prostý proprietárního softwaru, který byl v té době typický pro komerční Unix systémy.

Projekt byl zahájen vytvořením kolekce GNU Compiler (GCC), která je dodnes široce používán pro kompilaci softwaru. Mezi další důležité součásti projektu patří GNU Debugger (GDB), textový editor GNU Emacs a knihovna GNU C (glibc), která poskytuje základní funkce pro C programy.

Linuxové jádro není součástí projektu GNU, ale používá stejnou licenci jako GNU software. Většina utilit a vývojových nástrojů, které nejsou specifické pro Linux, jsou převzaty z projektu GNU. Protože každý použitelný systém musí obsahovat

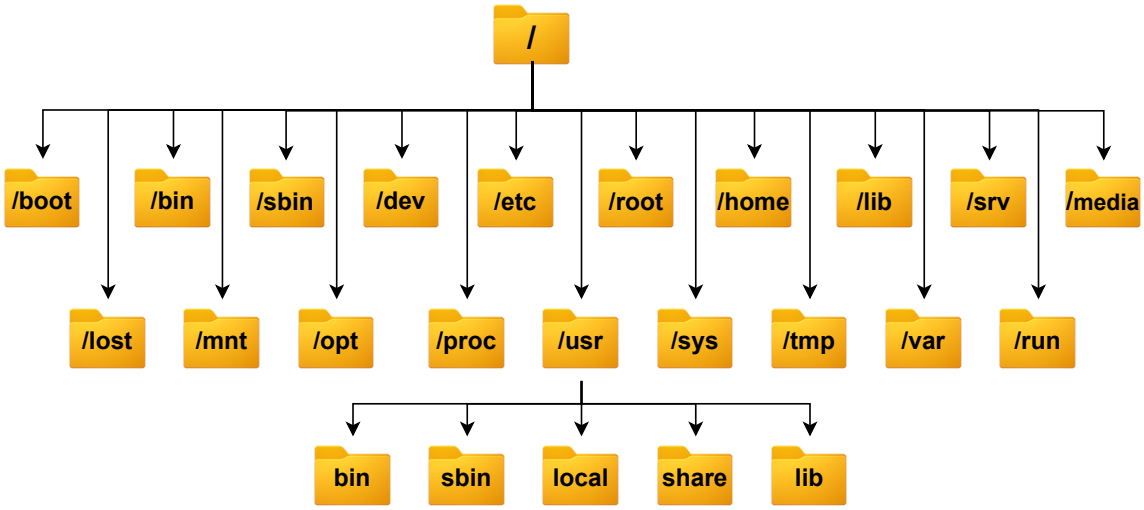
jak jádro, tak alespoň minimální sadu utilit. Z tohoto důvodu takový systém by se měl nazývat GNU/Linux systém [5].

Seznam běžného GNU softwaru [6]:

- Gnome: Desktopové prostředí GNU.
- GCC: C GNU Kompilátor.
- GDB: GNU Debugger.
- Bash: GNU shell.
- Coreutils: Jako ls, cat a chmod.
- GNU Photo: Software pro interakci s digitálními fotoaparáty.
- GNU SQL: Relační databázový systém.

2.3 Organizace souborů

The directory structure in Linux-based operating systems follows the Linux file-system hierarchy standard (Filesystem Hierarchy Standard FHS) definovaným a udržovaným Linuxovou nadací (Linux Foundation). Díky dobře definovanému standardu mohou uživatelé a vývojáři softwaru snadněji znát umístění nainstalovaných binárních souborů, systémových souborů atd. Tímto způsobem nemusí být linuxové aplikace šité na míru konkrétní distribuci a mohou být používány univerzálně [7].



Obr. 2.2: Standardní hierarchie souborového systému v Linuxu

Poslední verzi FHS vydala nadace Linux v roce 2015, viz obrázek 2.2.
Nadcházející sekce bude obsahovat popis několika adresářů Linuxu v souladu s FHS.

- Adresář **/root**: hlavní uživatelský adresář (root). Je přístupný pouze tomuto uživateli, který může použít každý linuxový příkaz a každý soubor.
- Adresář **/bin**: ukládá základní příkazové binárky, které může používat jak systémový administrátor, tak uživatel, jako cat, ls, mv atd.
- Adresář **/sbin**: stejně jako /bin. /sbin obsahuje také základní systémové binárky. Tyto binárky však má používat pouze správce systému (root) než běžný uživatel. Například fdisk, ifconfig, reboot.
- Adresář **/boot**: tento adresář obsahuje všechny soubory potřebné pro spuštění systému.
- Adresář **/dev**: obsahuje soubory zařízení pro všechna fyzická a virtuální zařízení připojená v systému. Obvykle se primární úložiště nazývá sda (/dev/sda).
- Adresář **/proc**: jedná se o informace týkající se paměti, oddílů, hardwaru (baterie, teplota atd.), všech načtených modulů jádra atd.
- Adresář **/etc**: tento adresář obsahuje konfigurační soubory systému. Zde je v konfiguračních souborech uložen název zařízení, hesla, síťová konfigurace, DNS, datum a čas atd. Tyto konfigurační soubory ovlivňují všechny uživatele systému.Pokud mají být provedeny změny konfigurace pro konkrétního uživatele, měl by být namísto "/etc/" použit "∼.conf/ ".
- Adresář **/home**: tento adresář obsahuje všechny osobní soubory specifické pro uživatele. Obsahuje samostatné adresáře pro každého uživatele, ke kterým lze přistupovat pomocí "cd /home/username". Konfigurační soubor specifický pro uživatele pro každou aplikaci lze nalézt v adresáři "/home/[uživatelské jméno]/.conf".
- Adresář **/lib**: tento adresář obsahuje všechny knihovny potřebné k zavedení systému a ke spuštění příkazů v /bin a /sbin. Obsahuje také moduly jádra, které ovládají většinu hardwaru a funkčnosti zařízení. Často jsou různé 32bitové a 64bitové knihovny se stejným názvem.
- Adresář **/opt**: tento adresář obsahuje knihovny a binárky vztahující se k balíčkům, které nejsou instalovány správci balíčků systému, ale jsou instalovány pomocí prostředků třetích stran.
- Adresář **/run**: tento adresář obsahuje data zařízení od posledního spuštění systému. Zahrnuje data všech systémových procesů a démonů, kteří byli spuštěni v aktuální relaci.
- Adresář **/tmp**: obsahuje dočasné soubory aktuálně spuštěných procesů. Tato data jsou také smazána po každém spuštění.

2.4 Distribuce systému

Linuxová distribuce obsahuje linuxové jádro, nástroje projektu GNU a libovolný počet open-source softwarových projektů, které systému poskytují rozmanité funkce. Existuje spousta různých linuxových distribucí, které splňují prakticky všechny výpočetní požadavky. Většina distribucí je přizpůsobena pro konkrétní skupiny uživatelů, jako jsou vývojáři softwaru, firemní uživatelé nebo běžní uživatelé. Každá přizpůsobená distribuce obsahuje softwarové balíčky potřebné pro podporu specializovaných funkcí [8].

Linuxové distribuce jsou rozděleny do tří kategorií:

- **Linuxová distribuce:** obsahuje operační systémy s jádrem Linux, GNU a jedna nebo více grafických desktopových prostředí a některé linuxové aplikace který je k dispozici, připraven k instalaci a spuštění.
- **Testovací distribuce Live-CD:** umožňuje načíst a spustit linuxovou distribuci z USB disku bez nutnosti instalace na pevný disk
- **Specializované distribuce:** jsou linuxové distribuce, které obsahují pouze aplikací, které by dávaly smysl pro konkrétní oblast použití [8].

Tabulka 2.1 zobrazuje některé z populárních linuxových distribucí, které jsou k dispozici.

Jádrové linuxové distribuce		
Distribuce	**Internetová stránka**	**Popis**
Red Hat	www.redhat.com	Obchodní distribuce využívaná hlavně pro internetové servery
Fedora	www.getfedora.org	Od RedHat, určená pro domácí použití.
Debian	www.debian.org	Populární u odborníků na Linux
Testovací distribuce Live-CD		
Distribuce	**Internetová stránka**	**Popis**
Ubuntu	www.ubuntu.com	Celosvětový linuxový projekt, určený pro mnoho jazyků
Knoppix	www.knoppix.net	Německá linuxová distribuce, první vyvíjené linuxové LiveCD
SimplyMEPIS	www.mepis.org	Desktopová distribuce pro domácí uživatele.

Tab. 2.1: Tabulka populárních linuxových distribucí

2.5 Operační systém Fedora

Operační systém, na kterém bude spuštěna vytvořená aplikace (Spouštěč aplikací pro seniora), je Fedora. Fedora je linuxový operační systém, který je sponzorován společností RedHat. Jeho kořeny pocházejí z RedHat Linuxu, který pod tímto názvem ukončil svůj vývoj v roce 2003. V té době společnost RedHat převedla svou jedinou distribuci RedHat Linux na Fedora Core (nyní nazývanou jednoduše Fedora) a Red Hat Enterprise Linux.
Jednou z klíčových vlastností Fedory je její rychlý cyklus vydávání, kdy nové verze vycházejí každých šest měsíců. Cílem bylo zůstat na špičce open source technologie a zároveň poskytovat vývojovou platformu pro software podnikové kvality, který by se mohl stát součástí RedHat Enterprise Linuxu.
Jelikož Fedora a RHEL jsou open source operační systémy, postavené na licenci GNU public, mohou si vývojáři vzít zdrojový kód z těchto linuxových systémů a vytvořit si vlastní linuxové distribuce. A to je právě to, co udělali. Příkladem jsou následující operační systémy:

- **CentOS:** mnoho linuxových konzultantů, kteří nepotřebují podporu RedHatu a nechtějí platit poplatky za předplatné RedHatu, se přeneslo na CentOS. CentOS je přestavbou zdrojového kódu RHEL.
- **Yellow Dog Linux:** založený původně na RedHat Linuxu, Yellow Dog Linux běží na různých Apple hardwarech (iBook, iMac, G4, G5 a tak dále) a také na PlayStation 3.
- **Další přestavby RHEL a Fedory:** jsou to další linuxové systémy čerpají z technologií vyvinutých částečně společností RedHat. Například distribuce "Mandriva", "PCLinuxOS a Linspire" používají systém správy balíčků RPM.

2.6 Srovnání linuxových distribucí

Ve srovnání se stabilnějším a méně aktualizovaným RedHat Enterprise Linux. Rychlost, s jakou je Fedora vyvíjena (nové vydání přibližně každých šest měsíců), je dokonalá pro uživatele Linuxu, kteří chtějí nejnovější verze softwaru a dokáží se vypořádat s určitou mírou nestability.
Existuje několik důvodů, proč byl operační systém Fedora vybrán pro instalaci vytvořené aplikace (Spouštěč aplikací pro seniory). Zde jsou některé z hlavních výhod:[10]:

1. **Grafické prostředí Gnome:** nadace Gnome a vývojáři (Fedora project) spolu velmi úzce spolupracují. Společně poskytují nejnovější vydání Gnome Shell, a to tak, že poskytují nové funkce, které žádný jiný linuxový operační systém nemůže poskytnout[12].
2. **Nejnovější verze softwaru**: Fedora je známá tím, že nabízí nejnovější softwarové verze, což znamená přístup k nejnovějším funkcím a opravám chyb.
3. **Má řadu různých verzí**: jednou z největších předností Fedory a hlavním důvodem pro její použití je skutečnost, že má pro uživatele k dispozici velké množství verzí. Tyto verze jsou alternativními verzemi Linuxu Fedora s různými desktopy a výchozími programy.
4. **Podporováno RedHatem**: Fedora má silnou komunitu uživatelů a vývojářů, kteří neustále pracují na zlepšení operačního systému a poskytují podporu ostatním uživatelům.
5. **Bezpečnost**: Fedora je známá svými silnými bezpečnostními prvky, včetně SELinuxu (Security-Enhanced Linux) a dalších bezpečnostních nástrojů.
6. **Zdarma a open source**: podobně jako mnoho dalších linuxových distribucí, Fedora je volně dostupná a open source, což uživatelům umožňuje využívat, upravovat a distribuovat jej podle potřeby.

Níže uvedená tabulka popisuje výhody a nevýhody některých linuxových distribucí jako Fedora, Debian a Ubuntu.

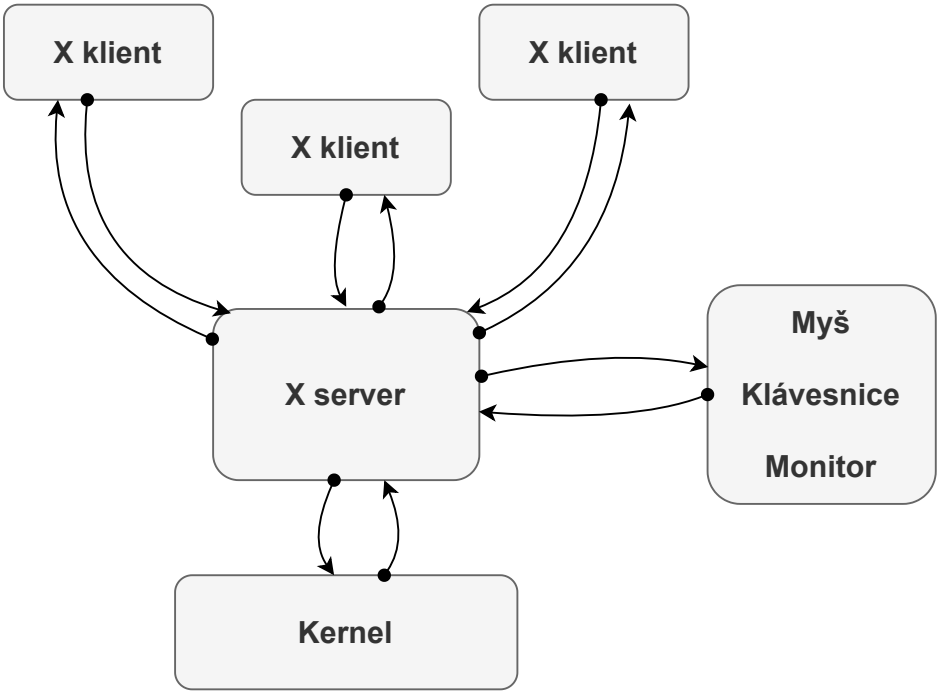
Distribuce	Vlastnosti
Fedora	+ Nabídne nejnovější verze softwarových balíčků. + Zaměřuje se na open source software. + Má relativně krátký uvolňovací cyklus, což znamená, že nové funkce jsou přidávány častěji. + Zahrnuje širokou škálu softwaru a nástrojů pro vývojáře. + Má vysokou podporu pro nejnovější hardware. - Má kratší cyklus podpory ve srovnání s ostatními linuxovými distribucemi. - Může být někdy méně stabilní kvůli častým aktualizacím. - Může vyžadovat dodatečnou konfiguraci pro přístup k neotevřenému softwaru.
Debian	+ Má velké úložiště softwarových balíčků. + Má k dispozici verzi s dlouhodobou podporou (LTS). + Je vysoce stabilní a spolehlivý díky přísnému testovacímu procesu. + Má velkou a vstřícnou komunitu. - Nemusí obsahovat nejnovější verze softwarových balíčků. - Může být obtížnější instalovat a konfigurovat než ostatní linuxové distribuce. - Pravděpodobně nemá podporu pro nejnovější hardware.
Ubuntu	+ Nabídne uživatelsky přívětivé rozhraní. + Má velké úložiště softwarových balíčků. + Má k dispozici verzi s dlouhodobou podporou. + Má vysokou podporu pro nejnovější hardware. - Je méně stabilní než některé jiné distribuce Linuxu. - Ve výchozím nastavení může obsahovat software, který není open source. - Může vyžadovat dodatečnou konfiguraci pro přístup k neotevřenému softwaru.

Tab. 2.2: Výhody a nevýhody Distribucí Fedora, Debian, Ubuntu.

3 Grafické prostředí

Spousta operačních systémů zahrnuje zobrazovací server (display server), což umožňuje rychlejší a snazší manipulaci se systémy a nabízí grafické uživatelské rozhraní (GUI), a reaguje na manipulaci s myší a klávesnicí, viz Obrázek 3.1.

Pro komunikaci mezi hardwarem a GUI, obsahuje zobrazovací server protokol. Různé operační systémy používají různé protokoly zobrazovacích serverů, linuxové distribuce používají většinou protokol X11. Zobrazovací server, který protokol X11 používá, se nazývá X.org a stará se o klientské vstupy a výstupy [8].



Obr. 3.1: Architektura systému X window

3.1 Systém X Window

X Window System (X nebo X11) je distribuované grafické uživatelské prostředí, a byl částečně vyvinut výzkumníky na MIT (Massachusetts Institute of Technology). Unix a Linux nezahrnuje uživatelská rozhraní do jádra systému a místo toho používá k implementaci programů na úrovni uživatele. To platí pro obě textová režim a grafické uživatelské prostředí. Takže systém X window není závislý na operačním systému, ve kterém byl implementován. A díky distribuovanosti lze zobrazit aplikace, které běží v jiném počítači [9]. Systém X Window však neimplementuje uživatelské rozhraní přímo, implementuje pouze systém okna a sada nástrojů, pomocí

kterých může být grafické uživatelské rozhraní provádění. Systém X Window vykresluje prvky GUI (grafické uživatelské rozhraní) na obrazovce a obsahuje metody pro zpracování interakcí mezi uživatelem a aplikací. To usnadňuje implementaci pro každý program ruzná uživatelská rozhraní. Uživatel může snadno změnit uživatelské rozhraní ukončí aktuálně používaný správce oken (window manager) a spustí jiného. V současné době existuje kompletní desktopové prostředí, které kombinuje správce oken s dalšími nástroji GUI. Mezi dvě populární desktopová prostředí patří Gnome, které dává na výběr několik různých správců oken, a KDE se správci oken KWin [12].

Zpravidla běží na vrcholu X dvě vrstvy, správce plochy (desktop manager) a správce oken (window manager).

1. **Správce plochy (desktop manager):** Je uživatelské rozhraní orientované na obrázky, které umožňuje interakci se systémovými programy pomocí manipulace s ikonami [13]. Mezi nejpopulárnějších správců ploch patří:
   (a) GNOME: Je nejpoužívanější linuxové desktopové prostředí.
   (b) KDE (Plasma): Je nejrozšířenější desktopové prostředí Linuxu po GNOME.
   (c) XFCE: Je kompatibilní s celou řadou linuxových distros.
   (d) MATE: Je rozšíření GNOME 2.

2. **Správce oken (window manager):** Je program, který běží pod správcem plochy a umožňuje uživatelům otevírat a zavírat okna, spouštět programy a nastavit myš [13]. Správce oken také dává obrazovce její osobnost. Mezi nejpopulárnějších správců oken patří
   (a) Mutter: Je výchozí pod GNOME 3.
   (b) kwin: Je výchozí pod KDE.
   (c) Openbox: Je vysoce konfigurovatelný správce oken.

3.2 Grafické prostředí Gnome

GNU Network Object Model Environment, známé také jako Gnome, je volné a open-source desktopové prostředí vydáno pod GNU Public License. je výkonné a snadno použitelné prostředí skládající se především z pracovní plochy, panelu a sady GUI nástrojů, díky nim lze vytvářet programová rozhraní. Gnome je navrženo tak, aby poskytovalo flexibilní platformu pro vývoj výkonných aplikací. Mezi nejznámější linuxové distribuce, které využívají grafické prostředí Gnome patří Ubuntu, Fedora a Debian [11].

Historie Gnome sahá do roku 1997, kdy se skupina vývojářů vedená Miguelem de Icazou vytvořili svobodné softwarové desktopové prostředí, které používalo GTK+ knihovny. Gnome byl přímý konkurent desktopového prostředí KDE, které v té

době byl velmi populární. První verze Gnome měla velký úspěch, protože projekt brzy porazil desktopové prostředí KDE. Druhá iterace Gnome 2 byla vydána v roce 2002. Vydání přineslo obrovskou sadu úprav, funkcí a zlepšení kvality desktopového prostředí [8]. Gnome verze 42, vydané v roce 2021 a později, zjednodušily vzhled uživatelského rozhraní a bylo přidano mnoho užitečných funkcí.

3.3 Grafické prostředí KDE

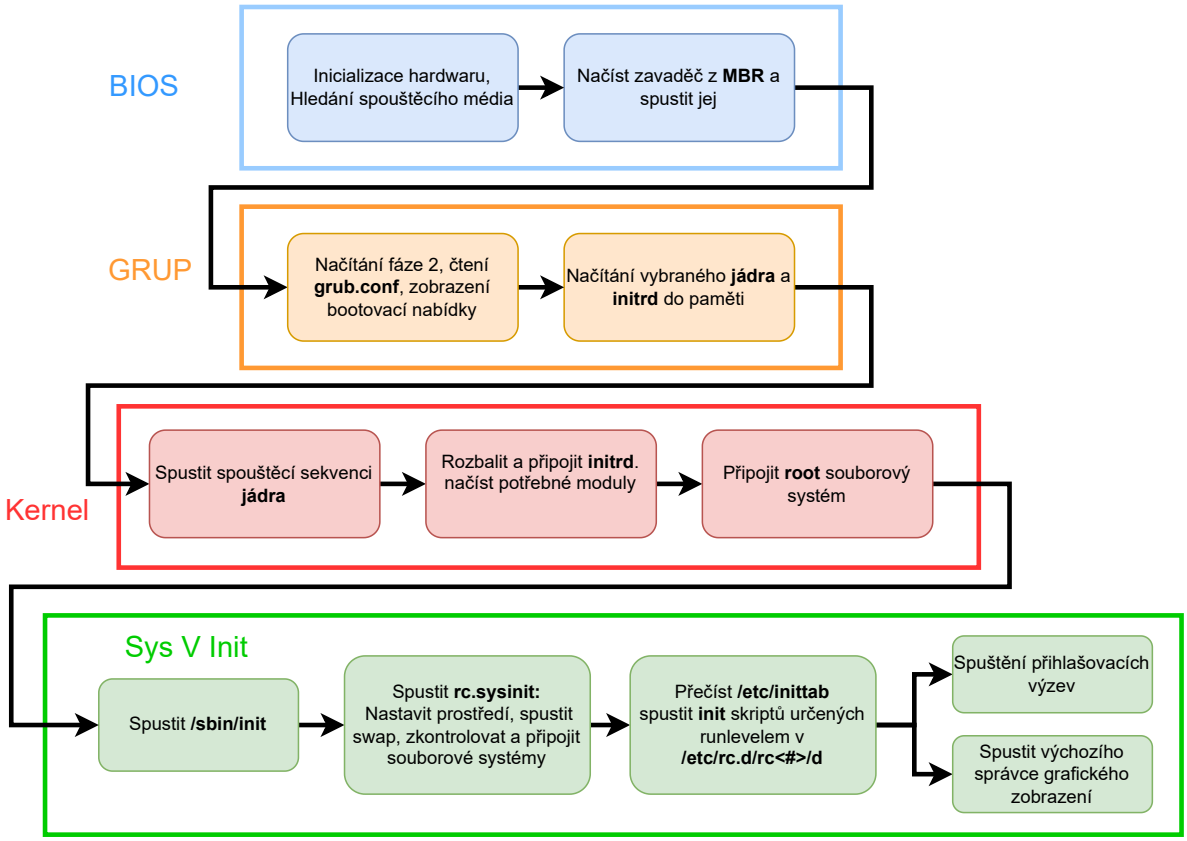
K Desktop Environment (KDE) je síťově průhledná plocha, která obsahuje standardní funkce plochy, jako je správce oken a správce souborů, a také rozsáhlou sadu aplikací, která pokrývá většinu úloh Linuxu. Plocha KDE je vyvíjena a distribuována projektem KDE. KDE je zcela svobodný a otevřený software poskytovaný pod GNU Public License a je dostupný zdarma spolu s jeho zdrojovým kódem [11].
Softwarové balíčky KDE lze stáhnout ze stránky www.kde.org. V současné době je pro KDE k dispozici velké množství softwarových aplikací, Takové aplikace mají obvykle písmeno K jako součást svého názvu například KWord nebo KMail.

KDE, které inicioval Matthias Ettrich v říjnu 1996, v současné době má rozsáhlý seznam sponzorů, včetně SuSE, Red Hat, Fedora, a dalších. KDE je navrženo tak, aby běželo na jakékoliv implementaci Unixu, včetně Linuxu, Solarisu, HP-UX a FreeBSD.

KDE používá jako svou knihovnu Qt, vyvinutou a podporovanou **Trolltech**. Qt je považována za jednu z nejlepších GUI knihoven dostupných pro Unix/Linux systémy. Použití Qt má tu výhodu, že se spoléhá na komerčně vyvinutou a podporovanou GUI knihovnu. Použití knihoven Qt také drasticky zkracuje dobu vývoje pro KDE. Knihovna Qt je zdarma pro svobodné a open source aplikace, ale ne pro komerční aplikace.

4 Start operačního systému GNU/Linux

Inicializace systému načtením jádra do paměti se běžně nazývá “bootování” počítače. (booting the system). Jeden z nejsilnějších aspektů Linuxu se týká jeho otevřeného způsobu spouštění a zastavování operačního systému, kdy načítá zadané programy pomocí jejich konkrétních konfigurací, umožňuje uživatelům měnit tyto konfigurace pro ovládání zaváděcího procesu [6].



Obr. 4.1: Zaváděcí proces Linuxu

4.1 Základní vstupní a výstupní systém

První krok spouštění systému je univerzální pro různé operační systémy. Při prvním stisknutí tlačítka napájení na počítači se spustí POST (Power On Self Test), jedná se o otestování provozuschopnosti systému, který je součástí systému BIOS (Basic I/O System). BIOS byl navržen od IBM v roce 1981 tak, aby inicializoval hardwarové komponenty. Úkolem POST je zajistit správné fungování hardwaru počítače. POST má definovaný seznam komponent, které musí zkontrolovat, a v případě že se funkce

POST nezdaří, počítač nemusí být použitelný, a spouštěcí proces proto nepokračuje. Mezi důležité kontroly patří: test procesoru, operační paměť a grafická karta. Následuje testování pevných disků a dalších zařízení SATA nebo USB. Pokud jsou hlavní komponenty funkční, je zaváděcí program načten z paměti, ve které je BIOS umístěn, také známé jako zavaděč bootstrapu (bootstrap loader). Cílem programu je načtení a spuštění zavaděče, v závislosti na nastavené vyhledávací sekvenci. Ta je umístěna ve spouštěcím sektoru pevného disku, USB diskety nebo jiného média. Konkrétně se jedná o první 512 bajtový sektor disku, ve kterém je umístěn zaváděcí program (zavaděč / bootloader). Tento sektor se nazývá Master Boot Record, zkráceně MBR. Pokud je zavaděč detekován, je načten do paměti a poté BIOS předá řízení nad systémem zavaděči [14].

4.2 Zavaděč systému

Druhým krokem startu je řízen zavaděčem (bootloader). který se nachází v prvním sektoru zaváděcího disku. Konkrétně je v adresáři /dev/sda.Toto umístění je známé jako Master Boot Record (MBR). Obsah MBR je 512 bajtů. Prvních 446 bajtů přenáší informace o primárním zavaděči, následuje 64 bajtů reprodukující informace o tabulce oddílů a nakonec jsou přidány 2 bajty pro kontrolu ověření. Jeho povinností je nahrát zavaděč, jako je GRUB. Výhodou zavaděče GRUB je možností pro snadnou ruční konfiguraci a interaktivní konfigurace při startu. Mezi výhody lze zařadit také konfiguraci jádra OS Linux. Mezi další zavaděče patří LILO , který se primárně používá pro serverové instance. LILO nabízí snadnou instalaci RAID nebo jednorázovou možnost příkazu pro nadcházející spuštění jádra OS. Používá se ve starších systémech a novější používají převážně GRUB, konkrétně GRUB 2 [14].

Po spuštění sofistikovanějšího zavaděče, který získal kontrolu nad systémem ze systému BIOS, začnou dvě po sobě jdoucí zaváděcí fáze. Zejména u plochových nasazení se uživatel často setkává s interaktivním menu. Zde je možné vybrat spouštěcí konfiguraci včetně operačního systému, který má být spuštěn.
Fáze 1, označovaná jako Stage 1, není schopna pracovat se systémem souborů. Odkazuje pouze na místo na disku, kde je uložen zavaděč pro druhou fázi.
Fáze 2, je nahrát jádro do paměti. Předá parametry, zkontroluje a pak inicializuje jádro. Pokud je používáno UEFI (Unified Extensible Firmware Interfac, nahrazuje Basic Input/Output System (BIOS)), zavaděč není potřeba a přenos dat linuxového jádra probíhá přímo. Primárním úkolem je však vždy zkopírovat spouštěcí kód OS do paměti a pak jej spustit.

4.3 Inicializace jádra

V této fázi je hlavním cílem nahrát jádro do paměti. Konkrétně je načten komprimovaný obraz jádra. Komprimovaný formát jádra je většinou zImage nebo bzImage. Nástroje pro práci s tímto formátem jsou v knihovně zlib. Jádro se musí před použitím dekomprimovat z tohoto formátu. Metoda Head provede základní nastavení a poté dekomprimuje obrazový soubor jádra do fyzické paměti.

Po načtení se jádro spustí, A jsou vytvořeny stránkovací tabulky pro stránkování paměti (swapper). Jádro musí při startu také rozpoznat typ procesoru, včetně dalších součástí. Jakmile je identifikace hotova a jsou nastaveny potřebné HW, stránkování paměti a systémové funkce, je přepnuta samotná funkčnost jádra, která není přímo závislá na konkrétní architektuře CPU. Zde se zastaví činnost hlavní spouštěč jádra

4.4 Inicializační proces

Proces Init je poslední fází startu. Většinou je umístění v /sbin/init. Je to první proces spuštěný v uživatelském prostoru. Jeho PID číslo (identifikační číslo procesu) je vždy 1. Proces Init zavádí uživatelské prostředí a pracuje pouze v uživatelském prostoru s podporou systémových volání.

Úkolem iniciačního procesu (Init) je poskytnout funkční systém pro běžné potřeby a využití. Kontroluje systém souborů, spouští jednotlivé démony nebo vyhledává závislosti jednotlivých modulů jádra. Služby jako Systemd nebo Upstarter používají jiný přístup tím, že uživateli poskytují konfiguraci přes konfigurační soubory a poté tuto konfiguraci předat svým binární součásti při příštím spuštění.

V iniciačním procesu je nutné zajistit chod jednotlivých služeb a démonů. Tyto služby a démoni jsou umístěni v několika úrovních nebo cílech. Programy a Služby jsou spouštěny v jednotlivých úrovních, které se nazývají runlevels. Tyto runlevely využívají modernější přístupy, jako je Systemd. Pak běží programy v jednotlivých runlevelech číslovaných od 0 do 6.

- Úroveň 0 představuje cíl zastavení.
- Úroveň 1 Režim jednoho uživatele.
- Úroveň 2 představuje víceuživatelský režim bez NFS (síťový souborový systém).
- Úroveň 3 je používána víceuživatelským systémem.
- Úroveň 4 se nepoužívá.

- Úroveň 5 představuje X11 (X Window System).
- Úroveň 6 představuje cíl restartu.

Uživatel si tuto fázi vizualizuje v podobě seznamů jednotlivých služeb při spuštění a démonů ostatních programů a pomocných výpisů. Po dokončení této fáze se zobrazí přihlašovací dialog. V případě vypnutí systému je volán init s požadavkem na uzavření všech funkcí uživatelského prostoru. Init je poté ukončen a jádro provede řízené vypnutí počítače [14].

5 Programové moduly

V této části bude uveden seznam modulů Pythonu potřebných pro úspěšné spuštění vytvořené aplikace. Moduly Pythonu jsou balíčky předem napsaného kódu, které mohou být importovány do programu za účelem provedení specifických úkolů a mohou značně zjednodušit proces vývoje tím, že poskytují vestavěné funkce[19].

5.1 Modul Tkinter

Tkinter je standardní balíček Pythonu pro vytváření grafických uživatelských rozhraní (GUI). Poskytuje sadu nástrojů a widgetů, které umožňují vytvářet okna, tlačítka, textová pole, popisky a mnoho dalších typů interaktivních prvků, se kterými mohou uživatelé komunikovat. Jednou z hlavních výhod Tkinteru je, že je obsažen ve standardní Python distribuci, takže pro jeho použití není nutné instalovat žádný další software nebo knihovny. Dalším důvodem, proč je Tkinter výkonný, je to, že je založen na Tcl/Tk GUI sadě nástrojů, která existuje již mnoho let a má vyspělý a stabilní kódový základ. Tkinter je také vysoce přizpůsobitelný a rozšiřitelný. Vývojáři mohou vytvářet vlastní widgety a upravovat chování stávajících widgetů tak, aby odpovídaly jejich specifickým potřebám. Navíc existuje mnoho balíčků a knihoven třetích stran, které rozšiřují funkčnost Tkinteru a poskytují další nástroje a widgety [15].

5.2 Modul PIL

PIL (Python Imaging Library) je populární balíček Pythonu pro práci s digitálními obrázky. Poskytuje širokou škálu možností zpracování obrázků, jako je změna velikosti obrázku, ořez, úprava barev, filtrování a transformace. PIL podporuje mnoho populárních formátů souborů s obrázky, včetně JPEG, PNG, GIF, BMP a TIFF. Jednou z hlavních výhod PIL je jeho snadnost použití. Poskytuje jednoduché a intuitivní Aplikační programovací rozhraní (API) pro práci s obrazy. Navíc je PIL čistě Python balíček, což znamená, že je nezávislý na platformě a může být použit na jakémkoli operačním systému, který podporuje Python. PIL je také vysoce přizpůsobitelný a rozšiřitelný, poskytuje řadu funkcí pro práci s jednotlivými pixely a pro vytváření vlastních obrazových filtrů a transformací.

Výpis 5.1: Ukázkový kód pro použití modulu PIL

```
# function to resize image
 def resizeImage(image,width,height):
    resizedImage=image.resize((int((screen_width/2)*0.9),int((screen_height/2)*0.9)
    ))
    newImage=ImageTk.PhotoImage(resizedImage)
    return newImage
```

Kód výše 5.1 definuje funkci *resizeImage()*, která vezme "obrázek, šířku a výšku", jako vstupní parametry a vrátí obrázek s upravenou velikostí jako objekt "ImageTk.PhotoImage".
Funkce *resize()* mění velikost obrazu na zadané rozměry, které se vypočítají pomocí šířky a výšky obrazovky. Metoda bere jako vstup nové velikosti, kde první prvek je šířka a druhý prvek je výška. V tomto případě se rozměry vypočítají jako 90% poloviční šířky obrazovky a výšky, což umožňuje umístění čtyř tlačítek na celou obrazovku.
Funkce *ImageTk.PhotoImage()* slouží k vytvoření nového objektu PhotoImage z objektu Image se změněnou velikostí. Tento objekt lze použít k zobrazení obrázku s upravenou velikostí v grafickém uživatelském rozhraní (GUI) pomocí "Label" nebo jiných podobných widgetů.

5.3 Modul pygame

Pygame je balíček Pythonu, který poskytuje sadu nástrojů a knihoven pro vytváření videoher a multimediálních aplikací. Je postaven na vrcholu knihovny SDL (Simple DirectMedia Layer), která poskytuje přístup k audio, klávesnici, myši a grafickému hardwaru [17]. Pygame je také vysoce přizpůsobitelný a rozšiřitelný, poskytuje řadu funkcí pro vytváření vlastní grafiky, zvukových efektů a hudby a také pro práci s uživatelskými vstupy a ovládání herní logiky. Vývojáři mohou Pygame využít také k vytváření komplexních herních enginů a rámců, které poskytují pokročilé funkce pro vytváření her a multimediálních aplikací. Ve spouštěči aplikací pro seniory (*srun*), bude pygame použit pro přidání zvuků do programu, tyto zvuky budou popisovat tlačítko (hlasová asistence), na kterém bude kurzor myši, seniorům může poskytnout několik výhod, včetně:

- **Lepší přístupnost:** Pro seniory se zrakovým postižením nebo kognitivními omezeními může přidání zvuku k tlačítku usnadnit orientaci a interakci s grafickými uživatelskými rozhraními. Díky zvukovému podnětu mohou senioři rychle rozpoznat, které tlačítko si vybrali, čímž se sníží zmatek.

- **Zvýšená důvěra uživatelů:** Senioři, kteří se méně vyznají v počítačích, mohou váhat s klikáním na tlačítka nebo s interakcí s grafickými prvky na obrazovce. Přidání zvuku k tlačítku může poskytnout uklidňující zvukovou zpětnou vazbu, která potvrdí jejich výběr.
- **Snížená chybovost:** Pokud senioři kliknou na tlačítko, aniž by obdrželi zpětnou vazbu, nemusí mít jistotu, zda byl jejich výběr zaregistrován počítačem. To může vést k chybám, jako klepnutí na špatné tlačítko. Přidání zvuku k tlačítku poskytuje jasnou zpětnou vazbu, že tlačítko bylo kliknuto, čímž se snižuje pravděpodobnost chyb a zlepšuje se celková efektivita.

Výpis 5.2: Ukázkový kód pro použití modulu pygame

```
pygame.mixer.init()
def play_shutdown(event):
    if SOUND:
        pygame.mixer.music.load(SHUTDOWN_WAV)
        pygame.mixer.music.play()

shutdownButton.bind('<Enter>',play_shutdown)
```

Kód uvedený výše 5.2 inicializuje modul pygame.mixer pro přehrávání zvuku pomocí knihovny Pygame. Funkce *pygame.mixer.init()* inicializuje modul mixeru s výchozími parametry, což programu umožňuje přehrávat zvuky.
Funkce *play_shutdown()* je funkce zpětného volání, která je volána, když ukazatel myši vstoupí do oblasti tlačítka "shutdownButton". Pokud je proměnná "SOUND" nastavena na True, načte zvukový soubor určený proměnnou "SHUTDOWN_WAV" pomocí funkce *pygame.mixer.music.load()* a přehraje ji pomocí funkce *pygame.mixer.music.play()*.

5.4 Modul hashlib

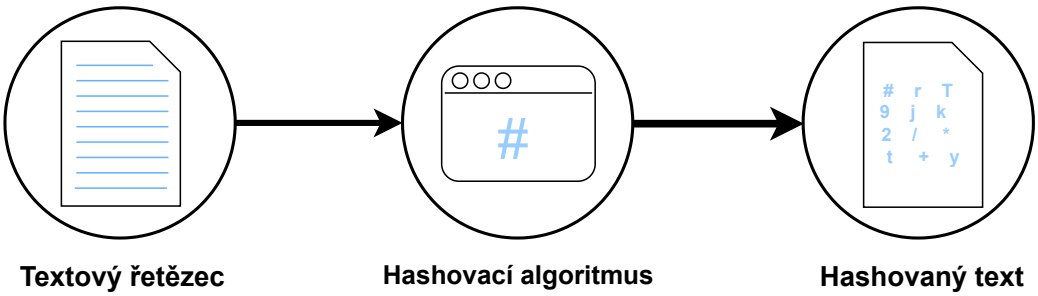
Balíček hashlib v Pythonu poskytuje sadu hashovacích algoritmů, které mohou být použity pro generování unikátních hashových hodnot pro vstupní data [16]. Hashing je proces, při kterém se vezme vstup (nebo zpráva) libovolné délky a vygeneruje řetězec znaků s pevnou délkou, nazývaný hashová hodnota, který představuje vstupní data unikátním způsobem. Balíček hashlib poskytuje implementace několika široce používaných hashovacích algoritmů, jako jsou SHA1, SHA224, SHA256, SHA384, SHA512, MD5 a další. Tyto algoritmy jsou navrženy tak, aby byly rychlé, bezpečné a odolné vůči kolizím (tj. když dva různé vstupy generují stejnou hodnotu hash).

Ve spouštěči aplikací pro seniory (*srun*), byl použit hash SHA256 pro hašování uživatelského hesla před uložením do souboru JSON s názvem "user_data.json". V hashovací funkci 5.3 je vstup řetězec, ten je zakódován do UTF-8 a poté je pomocí

funkce *hash SHA-256* vygenerována 64 znaková hexadecimální řetězcová reprezentace hashovaného hesla.

Výpis 5.3: Ukázkový kód pro hashovací funkci

```
#function to hash password
def hash_fun(string):
    hashed_string = hashlib.sha256(string.encode('utf-8')).hexdigest()
    return hashed_string
```



Obr. 5.1: Přehled hashovacího algoritmu

5.5 Modul os

Balíček os je vestavěný modul Python, který poskytuje způsob interakce s operačním systémem. Poskytuje přenosný způsob používání funkcí závislých na operačním systému, jako je čtení nebo zápis do systému souborů, správa procesů a práce s proměnnými prostředí [20].

Mezi běžné úkoly, které lze provádět pomocí os modulu, patří:

- **Správa souborů a adresářů:** Modul os poskytuje funkce pro vytváření, mazání, přejmenování a kopírování souborů a adresářů. Tyto funkce lze použít k manipulaci se systémem souborů.
- **Správa procesů:** Modul os poskytuje funkce pro správu procesů, jako je spouštění a ukončování procesů, získávání informací o běžících procesech a odesílání signálů procesům.
- **Práce s proměnnými prostředí:** Modul os poskytuje funkce pro práci s proměnnými prostředí, jako je získávání a nastavování proměnných prostředí a rozšiřování proměnných v cestách.
- **Práce s cestami:** Modul os poskytuje funkce pro práci s cestami, jako je spojování a dělení cest, získání aktuálního pracovního adresáře a získání absolutní cesty souboru.

Výpis 5.4: Ukázkový kód pro použití modulu os

```
# shutdown function
def shutdown_system():
    os.system("systemctl poweroff")

#enable window key
def enable_window_key():
    os.system("gsettings set org.gnome.mutter overlay-key 'Super'")

# function to reboot the operating system
def reboot_sys():
    os.system('systemctl reboot')
```

Výše uvedený kód 5.4 definuje tři funkce pro provádění systémových operací v linuxovém operačním systému:

- ***Funkce shutdown_system()***: Tato funkce vypíná systém pomocí metody *os.system()* pro vykonání příkazu "systemctl poweroff". Tento příkaz se používá k okamžitému vypnutí systému.
- ***Funkce enable_window_key()***: Tato funkce povoluje klávesu window (Super klávesa) na klávesnici pomocí metody *os.system()* k provedení příkazu "gsettings set org.gnome.mutter overlay-key 'Super' ". Tento příkaz nastaví překryvný klíč ke klíči Super, který se používá k otevření přehledu aktivit v desktopovém prostředí GNOME.
- ***Funkce reboot_sys()***: Tato funkce restartuje systém pomocí metody *os.system()*, která vykoná příkaz systemctl reboot. Tento příkaz se používá k restartování systému.

6 Vytvoření spouštěče aplikací pro seniory

Vizí bakalářské práce je upravit operační systém s linuxovým jádrem a vytvořit jednoduché uživatelské rozhraní přizpůsobené pro seniory ve věkové kategorii 90 let. Operační systém je spustitelný z USB disku. Vytvoření uživatelského rozhraní v Pythonu s názvem Spouštěč aplikací pro seniory (*srun*) musí být snadno použitelné pro seniory.

Vzhledem k jednoduchému designu (Spouštěč aplikací Senior), viz obrázek 6.2, který nevyžaduje složité požadavky, jsou potřeba pouze tlačítka a štítky, byl zvolen modul Tkinter. Aplikace Spouštěč aplikací pro seniory se skládá z pěti rozhraní, z nichž dvě jsou určené pro seniory. První rozhraní slouží jako primární okno, které umožňuje seniorům spouštět další aplikace. Druhé rozhraní umožňuje seniorům snadno vypnout systém. Pro zajištění snadného ovládaní (*srun*) je vždy otevřeno pouze jedno okno. Zbývající tři rozhraní jsou rezervována pro správce. První rozhraní slouží jako přihlašovací okno pro správce. Druhé rozhraní poskytuje rozšíření nastavení umožňující správcům konfigurovat další funkce. Třetí rozhraní umožňuje správcům měnit uživatelská hesla, což poskytuje další úroveň zabezpečení. Tato rozhraní specifická pro administrátora zvyšují funkčnost a řízení dostupné v aplikaci.

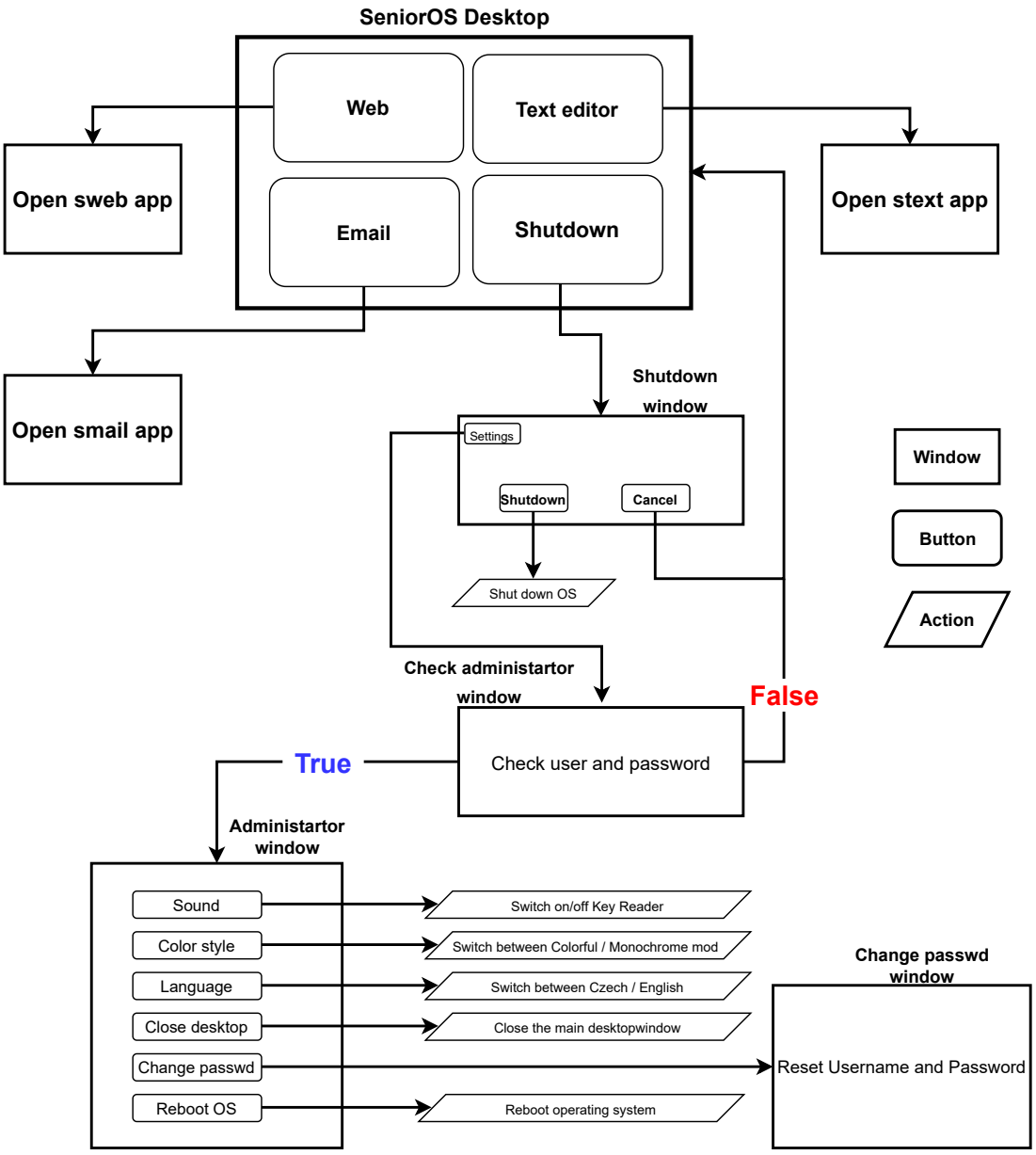
V rámci rozšíření nastavení mají správci možnost konfigurovat různé možnosti. Pro funkci hlasové asistence mohou přepínat mezi různými jazyky, což seniorům umožňuje interakci s aplikací v jimi preferovaném jazyce. Dále si administrátoři mohou vybrat mezi barevnými režimy a vybrat si buď barevný, nebo černobílý režim, viz obrázek 6.1 v závislosti na preferencích seniorů nebo zrakovém postižení. Konečně, správci mají možnost program (Spouštěč aplikací pro seniory) v případě potřeby ukončit, což jim poskytuje flexibilitu a kontrolu nad fungováním aplikace.

Dalším aspektem, který byl řešen, je umožnit seniorům automatické přihlašování a automatické spouštění aplikace (Spouštěč aplikací pro seniory). Senioři budou při spuštění systému automaticky přihlášeni ke svým účtům, čímž odpadne nutnost manuálních přihlašovacích postupů. To zjednodušuje proces a umožňuje seniorům rychlý přístup k personalizovanému prostředí pracovní plochy bez dalších kroků.

Obr. 6.1: Barevný a černobílý režim hlavního rozhraní

6.1 Základní funkce

Program je desktopové rozhraní napsané v Pythonu. Jedinečná věc na tomto rozhraní je, že je jednoduché a snadno ovladatelné, stačí jen stisknout ikony myší. Na obrázku 6.2 je podrobné vysvětlení struktury programu. Hlavní funkcí programu je poskytnout seniorům snadný způsob, jak spustit tři různé aplikace, konkrétně e-mail , textový editor a prohlížeč. Tyto aplikace jsou navrženy tak, aby odpovídaly specifickým potřebám a preferencím seniorů a umožňovaly jim zjednodušený přístup k e-mailu (*smail*), textovému editoru (*stext*) a základnímu prohlížení internetu (*sweb*). Program *srun* navíc poskytuje jednoduché rozhraní pro vypínání operačního systému, čímž zajišťuje, že senioři mohou snadno a bezpečně opustit systém, když jej přestanou používat.



Obr. 6.2: Architektura programu Spouštěč aplikací pro seniory (*srun*)

Hlavní uživatelské rozhraní Spouštěcí aplikace pro seniory

Popis tohoto okna je jasný z jeho názvu, je to hlavní okno, které se po spuštění programu *srun* objeví před uživatelem, tlačítka jsou velká a černobílá aby to bylo vhodné pro lidi s barvoslepostí , obrázek 6.3. Po najetí na tlačítka se změní barva pozadí tlačítka tak, aby uživatel viděl, na kterém tlačítku se nachází ukazatel myši. Pro ty, kteří preferují, aby byla tlačítka barevná, mohou barvy upravit v nastavení.

Funkce tlačítek:

- Tlačítko Web, vlevo nahoře, stisknutím tohoto tlačítka se spustí program sweb.
- Tlačítko Email, vlevo dole, stisknutím tohoto tlačítka se spustí program smail.
- Tlačítko Textový editor, vpravo nahoře, stisknutím tohoto tlačítka se spustí program stext.
- Tlačítko Vypínání, vpravo dole, stisknutím tohoto tlačítka se otevře Vypínací okno

Obr. 6.3: Hlavní uživatelské rozhraní pro aplikace *srun*

Výpis 6.1: Ukázkový kód pro vytvoření hlavního uživatelského rozhraní

```
window = Tk()                                        #create window
screen_width = window.winfo_screenwidth()            #get screen width
screen_height= window.winfo_screenheight()           #get screen height
button_width = int(screen_width/2 - 10)              #set button width
button_height = int(screen_height/2 - 10)            #set button height
window.geometry("%dx%d" % (screen_width,screen_height))
window.attributes('-fullscreen',True)                #disable the close button on
    window
window.resizable(0,0)                                #disable resize the window
```

Pro vytvoření grafického rozhraní spouštěcí aplikace pro seniory (*srun*) byla vybrána knihovna Tkinter. V ukázkovém kódu 6.1 bylo vytvořeno hlavní objekt okna tkinter pomocí funkce *Tk()*. Metoda *winfo_screenwidth()* slouží k nastavení šířky okna na šířku obrazovky a metoda *winfo_screenheight()* slouží k nastavení výšky okna na výšku obrazovky. Dvě proměnné *button_width* a *button_height* jsou definovány pro nastavení rozměrů tlačítka, které bude do okna přidáno později. Funkce *int()* slouží k zaokrouhlení rozměrů na celá čísla.

Metoda *geometry()* je funkce poskytovaná modulem Tkinter v Pythonu, která nastavuje velikost a polohu okna Tkinter. Metoda vyžaduje jeden argument, řetězec ve formátu "šířka x výška + xpozice + ypozice", kde šířka a výška jsou rozměry okna v pixelech a xposition a yposition jsou souřadnice levého horního rohu okna v pixelech. Respektive, formát řetězce "%dx%d" je nahrazen proměnnou "screen_width" šířka obrazovky a "screen_height" výška obrazovky.
Metoda *attributes()* v Tkinter slouží k nastavení nebo úpravě atributů okna. Vyžaduje řadu klíčových argumentů, které odpovídají určitým atributům okna, například :

- **-alpha:** Nastaví průhlednost okna, od 0,0 (zcela průhledné) do 1,0 (zcela neprůhledné).
- **-disabled:** Pokud je nastavena hodnota True, okno a všechny jeho potomky budou vypnuty, takže nebudou reagovat na uživatelské vstupy. Pokud je nastavena hodnota False (výchozí), bude okno zapnuto.
- **-fullscreen:** Je-li nastaveno na True, okno vyplní celou obrazovku a skryje hlavní panel a další prvky plochy. Pokud je nastavena hodnota False (výchozí), bude okno zobrazeno normálně.
- **-topmost:** Je-li nastaveno na True, zobrazí se okno nad všemi ostatními okny na obrazovce bez ohledu na to, zda má zaměření. Pokud je nastavena hodnota False (výchozí), zobrazí se okno v normálním pořadí skládání.

Metoda *resizable()* v Tkinteru slouží ke kontrole, zda uživatel může změnit velikost okna. Vyžaduje dva argumenty "width and height" šířka a výška, obojí jsou booleovské hodnoty. Pokud je šířka nastavena na True (1), uživatel bude moci změnit velikost okna vodorovně, pokud je nastavena na False (0), bude okno nastaveno na aktuální šířku. Stejně tak, pokud je výška nastavena na True, uživatel bude moci změnit velikost okna vertikálně, pokud je nastavena False, bude okno opraveno na aktuální výšku. V tomto řešení jsou obě proměnné nastaveny na hodnotu false (0), což zabrání uživateli změnit velikost okna.
Níže bude popsáno a ukázáno, jak byla tlačítka hlavního rozhraní vytvořena.

Tlačítko webového prohlížeče

Výpis 6.2: Ukázkový kód tlačítka swebButton

```
swebButton = Button(window, image = img_sweb_resized,
        activebackground = SWEB_ACTIVEBACKGROUND, background = SWEB_BACKGROUND,
        bd = 5, overrelief = GROOVE, command = sweb_window)
```

Výpis 6.3: Ukázkový kód funkce *sweb_window()*

```
    #function to open sweb app
    def sweb_window():
        os.chdir("/home/sweb")
        try:
            if os.system('/home/sweb/env/bin/python3 /home/sweb/main.py') != 0 :
                raise Exception()
        except (Exception) :
            messagebox.showerror('Error','Can not open sweb app')
```

Posledním krokem vytvoření hlavního okna je přidání čtyř požadovaných tlačítek. První tlačítko bylo vytvořeno pod názvem "swebButton" s obrázkem a určenými atributy tlačítka, výpis 6.2. Tlačítko je spojeno s funkcí *sweb_window()* jako příkazem, výpis6.3, který má být proveden po kliknutí. Funkce *sweb_window()* je určena k otevření webové aplikace v jejím virtuálním prostředí, které se nachází v adresáři "/home/sweb/env". Funkce používá os a messagebox, takže by měly být importovány na začátku zdrojového kódu. Parametr "image" nastavuje obrázek, který se zobrazí na tlačítku, které je uloženo v proměnné "img_sweb_resized". "Activebackground" a "background" nastavují barvu tlačítka při kliknutí, respektive bez kliknutí. Parametr "bd" nastavuje velikost okraje tlačítka a parametr "overrelief" určuje typ efektu, který se použije, když se kurzor myši pohybuje nad tlačítkem.

Tlačítko email

Výpis 6.4: Ukázkový kód tlačítka smaiButtonl

```
smailButton = Button(window, image = img_smail_resized,
        activebackground = SMAIL_ACTIVEBACKGROUND, background = SMAIL_BACKGROUND,
        overrelief = GROOVE, bd = 5, command = smail_window)
```

Výpis 6.5: Ukázkový kód funkce *smail_window()*

```
#function to open smail app
def smail_window():
    os.chdir("/home/smail")
    try:
        if os.system('/home/smail/env/bin/python3 /home/smail/main.py') != 0 :
            raise Exception()
    except (Exception) :
        messagebox.showerror('Error','Can not open smail app')
```

Ve výpisu 6.4 bylo vytvořeno tlačítko s názvem smailButton s obrázkem a určenými atributy tlačítka. Tlačítko je spojeno s funkcí *smail_window()* jako příkazem, který má být proveden po kliknutí. Parametr "image" nastavuje obrázek, který se zobrazí na tlačítku, které je uloženo v proměnné "img_smail_resized". "Activebackground" a "background" nastavují barvu tlačítka při kliknutí, respektive bez kliknutí. Parametr "bd" nastavuje velikost okraje tlačítka a parametr "overrelief" určuje typ efektu, který se použije, když se kurzor myši pohybuje nad tlačítkem. Parametr "command" nastavuje funkci, která bude vyvolána po kliknutí na tlačítko, což je v tomto případě *smail_window()*. Funkce *smail_window()* je určena k otevření mailové aplikace v jejím virtuálním prostředí, které se nachází v adresáři "/home/smail/env", výpis 6.5 .

Tlačítko textového editoru

Výpis 6.6: Ukázkový kód tlačítka stextButton

```
stextButton = Button(window,image = img_stext_resized,
        activebackground = STEXT_ACTIVEBACKGROUND, background = STEXT_BACKGROUND,
         bd = 5,overrelief = GROOVE, command = stext_window)
```

Výpis 6.7: Ukázkový kód funkce *stext_window()*

```
    #function to open stext app
    def stext_window():
        os.chdir("/home/stext")
        try:
            if os.system('/home/stext/env/bin/python3 /home/stext/main.py') != 0 :
                raise Exception()
        except (Exception) :
            messagebox.showerror('Error','Can not open stext app')
```

Jako předchozí tlačítka. Tlačítko stextButton je spojeno s funkcí *stext_window()* jako příkaz, který se provede po kliknutí. Funkce *stext_window()* je určena k otevření aplikace textového editoru v jeho virtuálním prostředí umístěném v adresáři "/home/stext/env", výpis 6.7.

Tlačítko vypnutí

Výpis 6.8: Ukázkový kód tlačítka shutdownButton

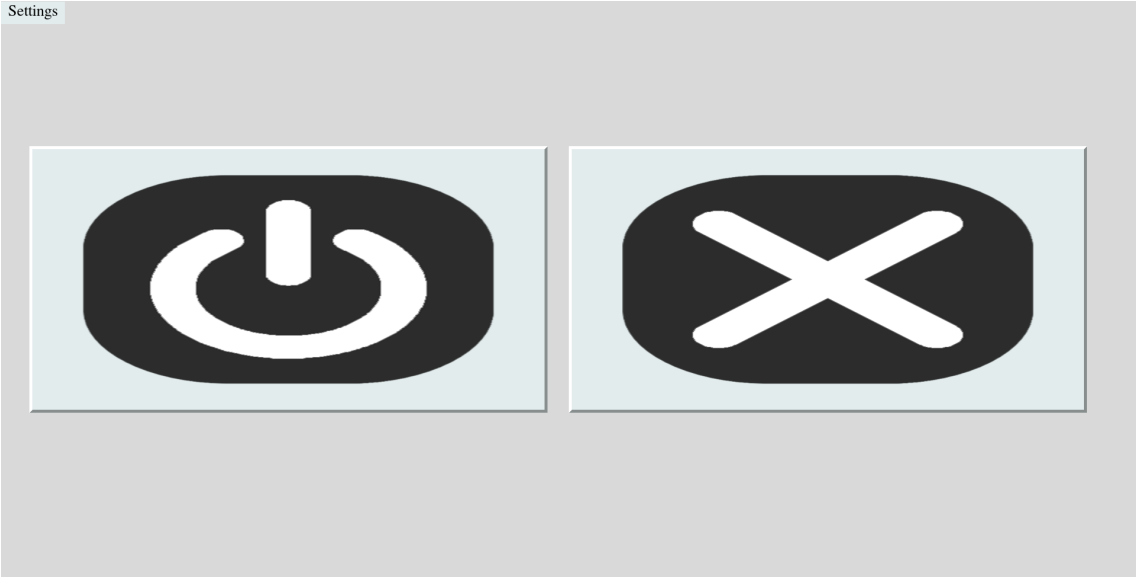
```
shutdownButton = Button(window,image = img_shutdown_resized, activebackground =
    SHUTDOWN_ACTIVEBACKGROUND, background = SHUTDOWN_BACKGROUND,
    overrelief = GROOVE, bd = 5,command = shutdown_window)
```

Jako předchozí tlačítka. Tlačítko shutdownButton je spojeno s funkcí *shutdown_window()* jako příkaz, který se provede po kliknutí. Funkce *shutdown_window()* je určena k otevření nového okna s názvem "confirm-win", kde uživatel může potvrdit vypnutí, výpis 6.9.

Vypnutí operačního systému

Jakmile senioři skončí se svými pracovními nebo volnočasovými aktivitami na počítači, měli by počítač vypnout kliknutím na tlačítko vypnutí "shutdownButton", které otevře okno vypnutí Obr6.4 voláním funkce *shutdown_window()*, kde mohou:

- Potvrdit vypnutí, kliknutím na tlačítko (Potvrdit), v levém středu.
- Zrušit vypnutí, kliknutím na tlačítko (Zrušit), v pravém středu.
- Otevřít pokročilé nastavení, kliknutím na tlačítko (Settings), vlevo nahoře.



Obr. 6.4: Rozhraní pro potvrzení vypnutí operačního systému

Výpis 6.9: Ukázkový kód pro vytvoření vypínacího okna

```
def shutdown_window():
        confirm_win = Toplevel(window)
        confirm_win.title('Shut down window')
        confirm_win.geometry("%dx%d" % (screen_width,screen_height))
        confirm_win.attributes('-fullscreen',True)
        confirm_win.resizable(0,0)
```

V ukázkovém kódu 6.9 bylo vytvořeno nové okno s názvem "confirm_win" a je nastaveno jako vyskakovací okno (Toplevel), které je potomkem hlavního okna (window). Funkce nastaví název nového okna na "Shut down windowo" a nastaví geometrii okna tak, aby byla stejná jako šířka a výška obrazovky. Také nastaví okno na celou obrazovku nastavením atributu "-fullscreen" na hodnotu True. Nakonec nastaví velikost okna tak, že nastaví atribut "resizable" na hodnotu (0,0), což znamená, že uživatel nebude moci změnit velikost okna.

Tlačítko pro potvrzení vypnutí

Výpis 6.10: Ukázkový kód tlačítka potvrzení vypnutí

```
#ok button
ok_btn = Button(confirm_win, bd = 5,overrelief = SUNKEN,
        image = img_shutdown_resized, background = SHUTDOWN_BACKGROUND,
        activebackground = SHUTDOWN_ACTIVEBACKGROUND,
        command = shutdown_system)
ok_btn.image = img_shutdown_resized
```

V ukázkovém kódu 6.10 bylo vytvořeno tlačítko "OK" v potvrzovacím okně pro vypnutí systému. Tlačítko je Tkinter Button s následujícími atributy:

- bd = 5: Tlačítko má šířku okraje 5 pixelů.
- overrelief = SUNKEN: Tlačítko se zdá být stisknuto, když je kurzor myši nad ním.
- image = img_shutdown_resized: Tlačítko zobrazuje obrázek (uložený v proměnné img_shutdown_resized)
- background = SHUTDOWN_BACKGROUND: Tlačítko má barvu pozadí definovanou proměnnou SHUTDOWN_BACKGROUND.
- activebackground=SHUTDOWN_ACTIVEBACKGROUND: Barva pozadí tlačítka se po klepnutí změní na barvu SHUTDOWN_ACTIVEBACKGROUND.
- command = shutdown_system: Funkce shutdown_system() je volána po kliknutí na tlačítko.

Výpis 6.11: Ukázkový kód funkce vypnutí

```
# shutdown function
def shutdown_system():
    os.system("systemctl poweroff")
```

Funkce 6.11 používá modul *os* k provedení systémového příkazu k vypnutí systému pomocí příkazu "systemctl poweroff". Příkaz "shutdown now" nebyl použit, protože pro jeho spuštění vyžaduje práva sudo.

Tlačítko pro zrušení vypnutí

Výpis 6.12: Ukázkový kód tlačítka zrušení vypnutí

```
#cancel button
cancel_btn = Button(confirm_win, bd = 5,overrelief = SUNKEN,
        image = image_cancel_resized, background = CANCEL_BACKGROUND,
        activebackground = CANCEL_ACTIVEBACKGROUND,
        command = confirm_win.destroy)
cancel_btn.image = image_cancel_resized
```

Tlačítko "cancel_btn" bylo vytvořeno stejným způsobem jako tlačítko "ok_btn". Po kliknutí na cancel_btn se zavře vypínací okno (shutdown_window()) a to pomocí příkazu "confirm_win.destroy".

Tlačítko pro otevření nastavení

Výpis 6.13: Ukázkový kód tlačítka otevření nastavení

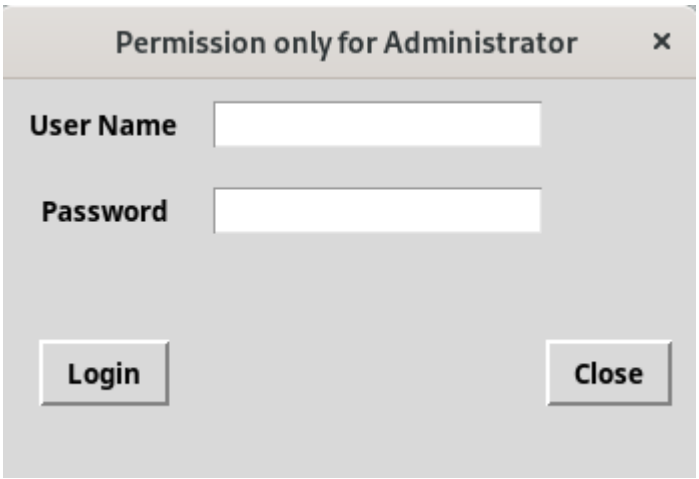
```
#settings button
settings_btn = Button(confirm_win, text = 'Settings',
        font = ('Times', 20), bd = 0,bg = CANCEL_BACKGROUND,
        activebackground = CANCEL_ACTIVEBACKGROUND,
        command = lambda:[confirm_win.destroy(),check_administrator_window()])
```

Celkově tento kód vytváří tlačítko "settings_btn", na které lze kliknout a otevřít nové okno "check_administrator_window()" pro kontrolu oprávnění správce. Tlačítko má vlastní velikost písma (20) a barvu pozadí a po stisknutí mění barvu pozadí. Funkce lambda umožňuje spustit více příkazů při kliknutí na tlačítko. Funkce lambda volá dvě metody, "confirm_win.destroy()" pro zavření aktuálního okna a "check_administrator_window" pro otevření nového okna pro kontrolu, zda má uživatel administrátorská oprávnění.

6.2 Rozšířené nastavení

Rozšířené nastavení poskytují správci další funkce, které nejsou normálním uživatelům (seniory) k dispozici. Rozšířené nastavení by měla být chráněna heslem, protože poskytují uživatelům zvýšená oprávnění a přístup ke kritickým systémovým prostředkům a nastavení, která mohou mít významný dopad na bezpečnost a stabilitu systému. Bez hesla nebo jiné formy ověřovacího mechanismu může kdokoli s přístupem k počítači získat administrátorská oprávnění a provádět změny v systému bez řádného autorizace. Požadavek hesla pro přístup k administrátorským právům navíc může chránit seniory před chybami nebo před tím, aby se stali terčem škodlivých útoků. Ochrana heslem pomáhá zajistit, aby do systému měli přístup a mohli v něm provádět změny pouze oprávnění jedinci, což může pomoci zabránit náhodnému nebo úmyslnému poškození systému a chránit citlivé informace.

Otevření rozšíření nastavení



Obr. 6.5: Přihlášení správce

Výpis 6.14: Ukázkový kód pro vytvoření přihlašovacího okna správce

```
def check_administrator_window():
    check_root_win = Toplevel(window)
    check_root_win.title('Permission only for Administrator')
    check_root_win_width = 400
    check_root_win_height = 200

  #set window to be in center
    x_point=(screen_width/2)-(check_root_win_width/2)
    y_point=(screen_height/2)-(check_root_win_height/2)
    check_root_win.geometry("%dx%d+%d+%d"%(check_root_win_width,
    check_root_win_height,x_point,y_point))
    check_root_win.resizable(0,0)

    #Labels
    username_label = Label(check_root_win, text="User Name",font=("None 10 bold"))
    username_label.grid(row=0, column=0, pady = 10)
    password_label = Label(check_root_win,text="Password",font=("None 10 bold"))
    password_label.grid(row=1, column=0, pady = 8)
    info_label = Label(check_root_win)
    info_label.grid(row = 2 ,column = 1, pady =10)

    #Entry
    username_entry = Entry(check_root_win, textvariable=check_username)
    username_entry.grid(row=0, column=1, pady = 10)
    password_entry = Entry(check_root_win, textvariable=check_password, show='*')
    password_entry.grid(row=1,column=1, pady = 8)
```

Výše uvedený kód 6.14 je funkce pro vytvoření nového okna, která uživatele vyzve k zadání uživatelského jména a hesla, aby zkontroloval, zda má oprávnění správce. Zde je rozpis kódu:

- Šířka a výška okna jsou nastaveny na 400, respektive 200 pixelů a jsou přiřazeny proměnným “check_root_win_width” a “check_root_win_height”.
- Okno se vycentruje na obrazovku pomocí metody *geometry* a souřadnice x a y se vypočítají na základě šířky a výšky obrazovky.
- Okno je vyrobeno tak, aby nešlo změnit velikost pomocí metody *resizable*.
- Pomocí metody *Label* jsou vytvořeny dva popisky, jeden pro uživatelské jméno a jeden pro heslo. Text popisků je nastaven pomocí parametru “text”.
- Štítky jsou umístěny pomocí metody *grid* s parametry řádku a sloupce.
- Pro zobrazení informací uživateli se vytvoří prázdný štítek. Tento štítek je umístěn také metodou “grid”.
- Metodou *Entry* se vytvoří dvě vstupní pole, jedno pro uživatelské jméno a jedno pro heslo. Text zadaný do polí je uložen v proměnné “check_username” respektive “check_password”.
- Vstupní pole jsou umístěna pomocí metody *grid* s parametry řádků a sloupců.

Přihlašovací tlačítko

Výpis 6.15: Ukázkový kód pro přihlašovací tlačítko

```
#login button
login_button = Button(check_root_win, text="Login", bd = 2, font = ("None 10 bold")
    , overrelief = SUNKEN, command = get_data)
login_button.grid(row = 3, column = 0, sticky = W ,padx = 20)
```

Celkově kód 6.15 vytvoří tlačítko s popiskem "Login" a specifikuje některé vizuální atributy a chování. Po kliknutí na tlačítko se zavolá funkce "get_data". Zde je rozpis kódu:

- Funkce *Button* slouží k vytvoření nového tlačítka, který je pak přiřazen proměnné "login_button".
- Volba "text" je nastavena na "Login", které se zobrazí na tlačítku.
- Volba "bd" nastaví šířku okraje okolo tlačítka na 2 pixely.
- Volba "font" nastaví písmo a velikost textu na tlačítku na "None 10 bold".
- Volba "overrelief" je nastavena na "SUNKEN", což znamená, že tlačítko se bude jevit jako stisknuté, když je myš nad ním.
- Volba "command" je nastavena na "get_data", což je funkce, která bude vyvolána po kliknutí na tlačítko.
- Metoda *grid* se používá k umístění tlačítka v okně. Nachází se v řádku 3 a sloupci 0.
- Volba "sticky" je nastavena na "W", což znamená, že tlačítko bude v buňce zarovnáno vlevo.

Funkce smazat záznam

Výpis 6.16: Ukázkový kód pro funkci smazání záznamu

```
def clear_entry():
    username_entry.delete(0,'end')
    password_entry.delete(0,'end')
```

Funkce *clear_entry* 6.16 odstraní obsah "username_entry" a "password_entry" vyvoláním jejich metod "delete" s argumenty 0 a "end". To při volání efektivně vyčistí vstupní pole.

Funkce pro kontrolu přihlašovacích údajů

Výpis 6.17: Ukázkový kód pro funkci kontroly přihlášení

```
global count
count = 0
def get_data():
    global count
    count =count + 1
    with open(pwd_json+'/user_data.json') as f:
        data_user = json.load(f)

    #get values from files
    users_from_file = data_user["user"]
    passwords_from_file=data_user["password"]

    #Get values from entry
    username= check_username.get()
    password=check_password.get()
    if users_from_file == username and passwords_from_file == hash_fun(password):
        check_root_win.destroy()
        administrator_window()
        return

    if count >= 3 :
        info_label.config( text = 'Oops', fg = 'red',font=("Helvetica",12))
        check_root_win.destroy()
        messagebox.showwarning("Warning", "Entering 3-times incorrect password")
        return

    if username == '' or password =='':
        info_label.config( text = 'All fields requried !', fg = 'red',font=("Helvetica",12))
        clear_entry()
    else:
        info_label.config(text = 'Incorrect inputs',fg = 'red',font=("Helvetica",12))
        clear_entry()
```

Funkce *get_data* 6.17 je příkaz, který se spustí po klepnutí na tlačítko “login_button”. Tato funkce nejprve načte hodnoty “user” a “password” ze souboru “user_data.json” umístěného v cestě zadané proměnnou “pwd_json”. Poté načte hodnoty zadané uživatelem do vstupu “username_entry” a “password_entry” vyvoláním jejich metod *get*. Pokud zadané hodnoty odpovídají hodnotám v souboru JSON, funkce zničí přihlašovací okno správce a vyvolá funkci “administrator_window” pro otevření rozšíření nastavení. V opačném případě, pokud je zadané heslo nesprávné a počet pokusů je menší než 3, funkce aktualizuje popisek “info_label” zprávou, která informuje uživatele o nesprávném pokusu o heslo. Pokud je počet pokusů větší než 3, funkce zničí přihlašovací okno správce, a zobrazí varovnou zprávu pomocí metody “messagebox.showwarning”. Pokud jsou pole pro zadání uživatelského jména nebo hesla prázdná, funkce aktualizuje popisek “info_label” zprávou, která informuje uži-

vatele, že jsou požadována všechna pole, a vymaže pole pro zadání pomocí funkce *clear_entry*. Pokud je uživatelské jméno a nebo heslo nesprávné, funkce aktualizujepopisek "info_label" zprávou, aby uživatele informovala, že vstup není správný, a vymaže pole záznamu pomocí funkce *clear_entry*.

Tlačítko pro zavření přihlašovacího okna správce

Výpis 6.18: Ukázkový kód pro tlačítko zavřít

```
#close button
close_button = Button(check_root_win, text = "Close",bd = 2,font = ("None 10 bold")
    ,
            overrelief = SUNKEN, command = check_root_win.destroy)
close_button.grid(row = 3, column = 3, sticky =W )
```

Tlačítko uzavření "close_button" bylo vytvořeno stejným způsobem jako přihlašovací (Login) tlačítko, ale po kliknutí na něj zavolejte metodu"check_root_win. destroy", čímž se zavře opřihlašovací okno správce (check_administrator_window).

Rozšíření nastavení

Okno správce je typicky specializované grafické uživatelské rozhraní (GUI), které je navrženo pro uživatele s oprávněními správce pro přístup a správu nastavení a konfigurací systému. Okno správce se může lišit v závislosti na používaném operačním systému nebo softwaru, ale obecně poskytne přístup k pokročilým funkcím a nastavením, které nejsou dostupné běžným uživatelům, v tomto případě se jedná o seniory. K oknu správce Obr 6.6 v *srun* lze přistupovat zadáním správného hesla a uživatelského jména.

Výpis 6.19: Ukázkový kód pro vytvoření rozhraní pro rozšíření nastavení

```
Administrator_win = Toplevel(window)
Administrator_win.title('Administrator window')
root_win_width = 430
root_win_height = 500
x_point_root_win = (screen_width/2)-(root_win_width/2)
y_point_root_win = (screen_height/2)-(root_win_height/2)
Administrator_win.geometry("%dx%d+%d+%d"%(root_win_width,
    root_win_height, x_point_root_win, y_point_root_win))
Administrator_win.resizable(0,0)
```

Kód 6.19 vytvoří nový objekt okna s pevnou velikostí a umístěním, které uživatel nemůže změnit. Toto okno obsahuje následující tlačítka:

Obr. 6.6: Rozhraní pro rozšíření nastavení

Tlačítko přepínání hlasová asistence

Umožňuje správci povolit/zakázat hlasovou asistenci5.3.

Výpis 6.20: Ukázkový kód pro tlačítko zvuk

```
sound_lbl = Label(Administrator_win, text = "- Sound:", font = "none 12 bold")
sound_lbl.grid(row = 1, column = 0, sticky = W, pady = 8, padx = 10)
sound_btn = Button(Administrator_win, image = sound_toggle_img, borderwidth = 0,
    command = toggle_json_sound)
sound_btn.grid(row = 1, column = 2, sticky = E, padx = 10, pady = 8)
sound_btn.image = sound_toggle_img
```

Ukázkový Kód6.20 vytvoří popisek a tlačítka v objektu okna s názvem “Administrator_win” pomocí nástroje Tkinter GUI v Pythonu. Zde je rozpis kódu:

- Popisek (label), je vytvořen s textem “- Sound:” a písmem o velikosti 12.
- Tlačítko (button), je vytvořen pomocí obrázkového souboru “sound_toggle_img” a šířky ohraničení 0.
- Parametr příkazu (command), je nastaven na funkci jménem “toggle_json_sound”, která bude zavolána po kliknutí na tlačítko.

Výpis 6.21: Ukázkový kód pro funkci přepínání zvuku v souboru JSON

```
def toggle_json_sound():
    #read the JSON data from the file
    with open(pwd_json+'/mainConstants.json') as f:
        data = json.load(f)

    #toggle the value of the "toggle_value" key
        data['SOUND'] = not data['SOUND']

    #write the modified data back to the file
    with open(pwd_json+'/mainConstants.json', 'w') as f:
        json.dump(data, f)

    #update the label to display the new value
        if data['SOUND']:
            sound_btn.config(image = on_img)
        else:
            sound_btn.config(image = off_img)
```

Ukázkový kód 6.21 definuje funkci s názvem “toggle_json_sound()”, která přepíná hodnotu klíče s názvem “SOUND” v souboru JSON s názvem “mainConstants.json”. Aktualizuje také obrázek tlačítka zobrazený v GUI okně, aby odrážel novou hodnotu “SOUND”.

Funkce začíná otevřením souboru “mainConstants.json” pomocí funkce *open()* a načtením jeho obsahu pomocí funkce *json.load()*. Obsah souboru je uložen v proměnné jménem “data”. Funkce poté přepíná hodnotu klíče “SOUND” v datovém “data” slovníku pomocí operátoru “not”, který převrací booleovskou hodnotu. To znamená, že pokud jsou data['SOUND'] aktuálně True, budou nastavena na False a naopak. Dále je upravený datový slovník zapsán zpět do souboru “mainConstants.json” pomocí funkce *json.dump()*. Nakonec funkce aktualizuje obrázek zobrazený na tlačítka “sound_btn”, aby odrážel novou hodnotu dat['SOUND']. Pokud jsou data['SOUND'] 'True', je obrázek tlačítka nastaven na “on_img”, a pokud je 'False', je obrázek tlačítka nastaven na “off_img”. K tomu slouží metoda *config()* tlačítko, která umožňuje dynamicky upravovat jeho vlastnosti.

Tlačítko pro přepínání barevného stylu

Výpis 6.22: Ukázkový kód pro tlačítko barevný styl

```
color_lbl = Label(Administrator_win, text="- Color style:", font = "none 12 bold")
color_lbl.grid(row = 3, column = 0, sticky = W, pady = 8, padx = 10)
color_btn = Button(Administrator_win, image= color_toggle_img, borderwidth=0,
    command = toggle_json_color)
color_btn.grid(row = 3, column = 2, sticky = E, padx = 10, pady = 8)
color_btn.image = color_toggle_img
```

Tlačítko Barevný styl (Color style) bylo vytvořeno stejným způsobem jako tlačítko zvuku (Sound), ale po jeho stisknutí vyvolá funkci “toggle_json_color”, která dělá totéž jako “toggle_json_sound” funkce, místo změny hodnoty SOUND změní hodnotu COLORFUL. je-li hodnota COLORFUL True, bude barevný styl vybarven, bude-li False, bude černobílý.

Tlačítko pro přepínání jazyků

Výpis 6.23: Ukázkový kód pro tlačítko jazyk

```
language_lbl = Label(Administrator_win, text = "- Language:", font = "none 12 bold"
    )
language_lbl.grid(row = 5, column = 0, sticky = W, pady = 8, padx = 10)
language_btn = Button(Administrator_win, image = language_toggle_img, borderwidth =
     0, command = toggle_json_language)
language_btn.grid(row = 5, column = 2, sticky = E, padx = 10, pady = 8)
language_btn.image = language_toggle_img
```

Tlačítko Jazyk (Language) bylo vytvořeno stejným způsobem jako tlačítko Styl barev a zvuk, ale kliknutím na něj vyvolá funkci “toggle_json_language”. terá dělá totéž jako “toggle_json_sound” funkce, místo změny hodnoty SOUND změní hodnotu LANGUAGE. Je-li hodnota LANGUAGE True, bude jazykem systému čeština, bude-li False jazykem angličtina.

Tlačítko pro zavření plochy

Funkce tohoto tlačítka je zavřít hlavní okno (zavřít aplikaci *srun*), což správci umožní používat operační systém v desktopovém správci Gnome.

Výpis 6.24: Ukázkový kód pro tlačítko zavřít plochu

```
close_lbl = Label(Administrator_win, text = "- Close desktop:",  font = "none 12
    bold")
close_lbl.grid(row = 7, column = 0, sticky = W, pady = 8, padx = 10)
close_btn = Button(Administrator_win, image = close_img, borderwidth = 0,
             command = lambda:[enable_window_key(), window.destroy()])
close_btn.grid(row = 7, column = 2,s ticky = E, padx = 10, pady = 8)
close_btn.image = close_img
```

Výpis 6.25: Ukázkový kód pro funkci povolit super klíč

```
#enable window key
def enable_window_key():
    os.system("gsettings set org.gnome.mutter overlay-key 'Super'")
```

Kód 6.24 vytvoří tlačítka s názvem “close_btn” s obrázkem “close_img”, který slouží jako jeho pozadí. Tlačítko nemá viditelné ohraničení, protože argument “Borderwidth” je nastaven na 0. Po kliknutí na tlačítko spustí funkci lambda, která dělá dvě věci:

1. Volá funkci “enable_window_key()” 6.25, Která je definována pro vykonání příkazu shellu, který mění nastavení Gnome související s klíčem “Super”. Příkaz “gsettings set org.gnome.mutter overlay-key ’Super’” je shellový příkaz, který používá nástroj gsettings k úpravě hodnoty klíče “overlay-key” ve schématu “org.gnome.mutter”. Nová hodnota klíče je nastavena na “Super”, což odkazuje na “Super” klávesu na klávesnici (také známou jako Windows klávesa na klávesnicích Windows).
2. Zničí okno “window”, což je hlavní okno programu *srun*, který ho zavře.

Tlačítko pro změnu uživatelského hesla

Funkce tohoto tlačítka je otevřít okno pro změnu hesla uživatele (change_password()), což správci umožní změnit heslo.

Výpis 6.26: Ukázkový kód pro vytvoření okna správce

```
change_lbl = Label(Administrator_win, text = "- Change user's password",  font = "
    none 12 bold")
change_lbl.grid(row = 9, column = 0, sticky = W, pady = 8, padx = 10)
change_btn = Button(Administrator_win, image = change_img,borderwidth = 0,
             command = lambda:[Administrator_win.destroy(), change_password()])
change_btn.grid(row = 9, column = 2, sticky = E, padx = 10, pady = 8)
change_btn.image = change_img
```

Tlačítko Změna hesla (Change user’s password) bylo vytvořeno stejným způsobem jako tlačítko Zavřít plochu (Close desktop). Po kliknutí na tlačítko spustí funkci lambda, která vykonává dvě věci:

1. Zničí objekt okna Okno správce (Administrator_win), fakticky zavře aktuální okno.
2. Volá funkci *change_password()*, což otevře okno pro změnu hesla uživatele 6.7.

Výpis 6.27: Ukázkový kód pro funkci restartování operačního systému

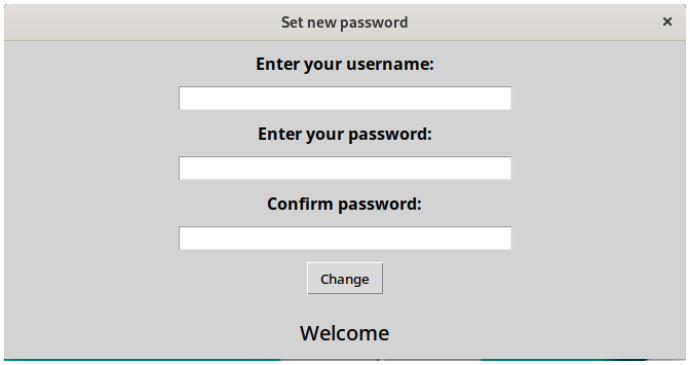
```
# function to reboot the system
def reboot_sys():
    os.system('systemctl reboot')
```

Tlačítko restartovat operační systém (Reboot the OS) bylo vytvořeno stejným způsobem jako tlačítko Zavřít plochu (Close desktop). Po kliknutí na tlačítko spustí funkci lambda, která vykonává dvě věci:

1. Zničí objekt okna Okno správce (Administrator_win), fakticky zavře aktuální okno.
2. Volá funkci *reboot_sys()*6.27, která je zodpovědná za restartování systému. Funkce používá funkci *os.system()* k vykonání příkazu shellu, který spustí restart systému. Konkrétne vykonávaný príkaz je “systemctl reboot”, což je linuxový príkaz sloužící k restartování systému. Příkaz “reboot” nebyl použit, protože pro jeho spuštění vyžaduje práva sudo.

Změnit heslo uživatele

Funkčnost okna 6.7 je umožnit správci změnit uživatelské jméno a heslo. Tyto změny se uloží do JSON souboru “user_data”, heslo bude před uložením hashované hashem SHA 256, uživatelské jméno ne.



Obr. 6.7: Rozhraní pro změnu hesla správce

Výpis 6.28: Ukázkový kód pro vytvoření rozhraní pro změnu hesla správce

```
change_password_win = Toplevel(window)
change_password_win.title("Set new password")
change_password_win.configure(bg=COLOR_BG)
change_password_win.resizable(0,0)
user_lbl = Label(change_password_win, text="Enter your new username:", bg=COLOR_BG,
    font="none 12 bold")
user_lbl.grid(row=1, column=0, sticky=EW, pady=8)
pass_lbl = Label(change_password_win, text="Enter your password:", bg=COLOR_BG,
    font="none 12 bold")
pass_lbl.grid(row=3, column=0, sticky=EW, pady=8)
cpass_lbl = Label(change_password_win, text="Confirm password:", bg=COLOR_BG, font=
    "none 12 bold")
cpass_lbl.grid(row=5, column=0, sticky=EW, pady=8)
username_box = Entry(change_password_win, width=40, bg="white")
username_box.grid(row=2, column=0)
pw_box = Entry(change_password_win, width=40, bg="white",show='*')
pw_box.grid(row=4, column=0)
pw_confirm_box = Entry(change_password_win, width=40, bg="white",show='*')
pw_confirm_box.grid(row=6, column=0)
```

```
sub_btn=Button(change_password_win, text="Change", width=6, command=click)
sub_btn.grid(row=7, column=0, pady=10)
output_info = Label(change_password_win, text="Welcome", bg=COLOR_BG, font="none 15
    ", width=60)
output_info.grid(row=8, column=0, sticky=EW, pady=10)
```

Ukázkový kód 6.28 definuje funkci *change_password()*, která vytvoří nové okno, aby uživatel mohl změnit své uživatelské jméno a heslo. Okno je vytvořeno pomocí Toplevel, tři popisky a tři vstupní pole. Tlačítko "Změnit" (Change) je vytvořeno pomocí widgetu Tlačítko s funkcí *click()* jako příkazem, který se spustí po kliknutí na tlačítko. Nakonec je vytvořen popisků pro zobrazení zpráv uživateli.
Funkce *click()* je definována uvnitř funkce *change_password()* a je spuštěna po kliknutí na tlačítko "Change". Funkce načte hodnoty z polí pro zadání uživatelského jména a hesla, ověří zadání pomocí série kontrol, aktualizuje informace o uživateli v souboru JSON a zobrazí rámeček se zprávou označující úspěch nebo neúspěch změny hesla.

Výpis 6.29: Ukázkový kód pro funkci změnit hesla správce

```
def change_password():
    def clear_entry_password():
        pw_box.delete(0,'end')
        pw_confirm_box.delete(0,'end')
    def click():
        username = username_box.get()
        password = pw_box.get()
        confirmed_pw = pw_confirm_box.get()
        caps = string.ascii_uppercase
        numbers = []
        for i in range(10):
            numbers.append(str(i))
        special_chars = ['!', '@', '#', '$', '%']
        name_check = False        def number_found(name):
            found = False
            for number in numbers:
                if name.find(number) > -1:
                    found = True
            return found

        #Capital letter
        def caps_found(pw):
            found = False
            for letter in caps:
                if pw.find(letter) > -1:
                    found = True
            return found        def special_found(pw):
            found = False
            for char in special_chars:
                if pw.find(char) > -1:
                    found = True
            return found
        if len(username) < 4:
```

```
34                output_info.configure(text="New user name must be longer than 3 letters
    ",font="green 15 bold")
35                username_box.delete(0,'end')
36                return
37            if not username.isalpha():
38                output_info.configure(text="Please use only letters for Username",font=
    "green 15 bold")
39                username_box.delete(0,'end')
40                return
41            if password != confirmed_pw and len(password) > 7:
42                output_info.configure(text="Passwords do not match")
43                clear_entry_password()
44                return
45            if not number_found(password):
46                output_info.configure(text="Password must contain a number")
47                clear_entry_password()
48                return
49            if not special_found(password):
50                output_info.configure(text="Password must contain a special character
    (!, @, #, $, %)")
51                clear_entry_password()
52                return
53            if not caps_found(password):
54                output_info.configure(text="Password must contain a capital letter")
55                clear_entry_password()
56                return
57            if len(password) < 8:
58                output_info.configure(text="Password must be at least 8 characters long
    ")
59                clear_entry_password()
60                return
61            with open(pwd_json+'/user_data.json') as f:
62                data_user = json.load(f)
63
64            data_user["user"] = username
65            data_user["password"] = hash_fun(password)
66            with open(pwd_json+'/user_data.json', 'w') as f:
67                json.dump(data_user, f)
68            change_password_win.destroy()
69            messagebox.showinfo("info", "User and Password changed successful")
```

Funkce *change_password()* nejprve definuje dvě vnořené funkce, *clear_entry_password()* a *click()*, které slouží k vymazání polí s hesly a ke zpracování požadavku uživatele na změnu hesla.

Funkce *click()* nejprve načte uživatelský vstup ze tří polí s heslem, pole s uživatelským jménem "username_box", pole s heslem "pw_box" a pole s potvrzením hesla "pw_confirm_box". Poté ověří zadání uživatele pomocí několika kontrol. Kontroly zahrnují:

- Uživatelské jméno musí být dlouhé alespoň 4 písmena a musí obsahovat pouze abecední znaky.
- Pole pro potvrzení hesla a hesla se musí shodovat a musí obsahovat alespoň 8 znaků.

- Heslo musí obsahovat alespoň jedno číslo, jeden speciální znak (!, @, #, $, %) a jedno velké písmeno.

Pokud některá z těchto kontrol selže, funkce zobrazí chybovou zprávu a vymaže pole s hesly. Pokud všechny kontroly projdou, funkce otevře soubor JSON "user_data", který ukládá data uživatele, aktualizuje pole uživatelského jména a hesla novým vstupem uživatele a poté uloží změny do souboru JSON. Nakonec funkce zavře okno grafického uživatelského rozhraní a zobrazí zprávu potvrzující, že heslo uživatele bylo úspěšně změněno.

6.3 Spouštění ve virtuálním prostředí

Protože aplikace Spouštěč aplikací pro seniory (*srun*) umožňuje uživatelům spouštět tři různé aplikace Pythonu (textový editor (stext), webový prohlížeč (sweb), email (smail)), je pro každou aplikaci vyžadováno virtuální prostředí.
Virtuální prostředí je nástroj v Pythonu, který umožňuje vytvářet izolovaná prostředí Pythonu. Každé virtuální prostředí může mít vlastní sadu balíčků Pythonu, aniž by zasahovalo do jiných projektů Pythonu na stejném počítači. Užitečnost virtuálního prostředí spočívá v jeho schopnosti vyhnout se konfliktům závislosti mezi různými projekty. Různé projekty mohou vyžadovat například verze 1.0 a 2.0 stejného balíčku. V takových případech lze pro každý projekt vytvořit samostatné prostředí s příslušnou verzí balíčku nainstalovanou pomocí virtuálních prostředí. Vestavěný modul zvaný **venv** je poskytován Pythonem, který umožňuje vytvářet virtuální prostředí. Virtuální prostředí lze také vytvářet pomocí nástrojů třetích stran, jako je **virtualenv** nebo **conda**.

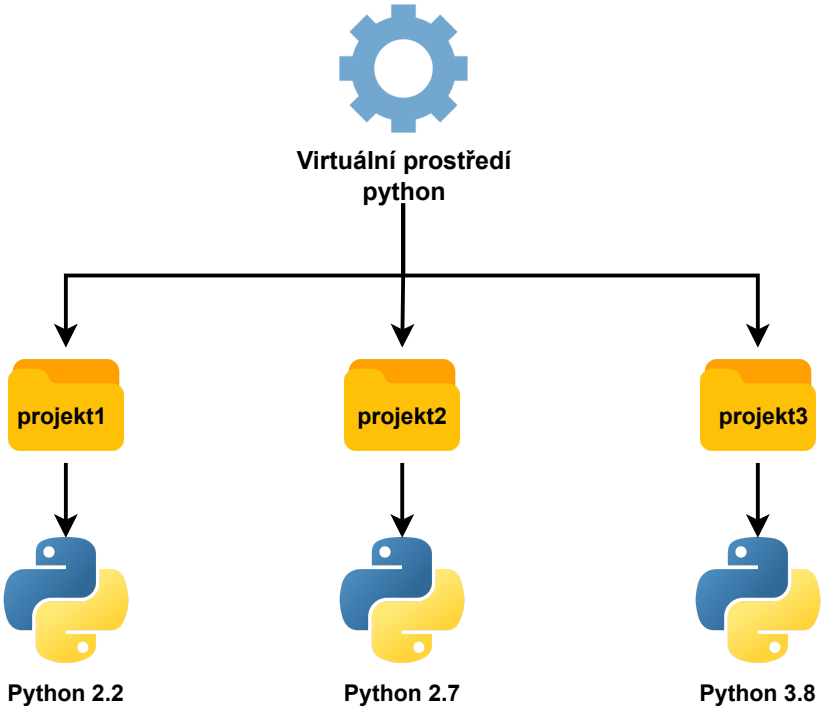
6.3.1 Vytvoření virtuálního prostředí

Složka s vlastní instalací Pythonu, která obsahuje vlastní adresář balíčků, je vytvořena při vytvoření virtuálního prostředí. Do tohoto adresáře lze nainstalovat knihovny třetích stran a balíky specifické pro projekt, na kterém se pracuje. Po aktivaci virtuálního prostředí budou všechny spuštěné příkazy nebo nainstalované balíčky specifické pro toto prostředí.
Pro vytvoření virtuálního prostředí Pythonu ve Fedoře pomocí nástroje virtualenvu lze použít následující kroky:

1. Instalace "pip" spuštěním následujícího příkazu v terminálu.

   ```
   $ sudo dnf install python3-pip
   ```

   Toto nainstaluje "pip", což je správce balíčků pro Python.



Obr. 6.8: Virtuální prostředí Python

2. Instalace virtualenvu spuštěním následujícího příkazu v terminálu.

   ```
   $ sudo pip3 install virtualenv
   ```

   Toto nainstaluje "virtualenv", což je nástroj třetí strany používaný k vytváření virtuálních prostředí v Pythonu.
3. Vytvoření nového virtuálního prostředí spuštěním následujícího příkazu.

   ```
   $ virtualenv env
   ```

   V aktuálním adresáři tak vznikne nové virtuální prostředí s názvem "env".
4. Aktivace virtuálního prostředí spuštěním následujícího příkazu:

   ```
   $ source env/bin/activate
   ```

   Tímto se aktivuje virtuální prostředí (env) a všechny spuštěné příkazy Pythonu budou nyní používat interpreter Pythonu a balíčky nainstalované ve virtuálním prostředí.
5. Balíčky lze nainstalovat pomocí příkazu pip. Pro instalaci balíku "pygame" lze například použít následující příkaz:

   ```
   (env)$ pip install pygame
   ```

   Tím bude balíček "pygame" nainstalován do virtuálního prostředí (env).
6. Virtuální prostředí lze deaktivovat pomocí následujícího příkazu.

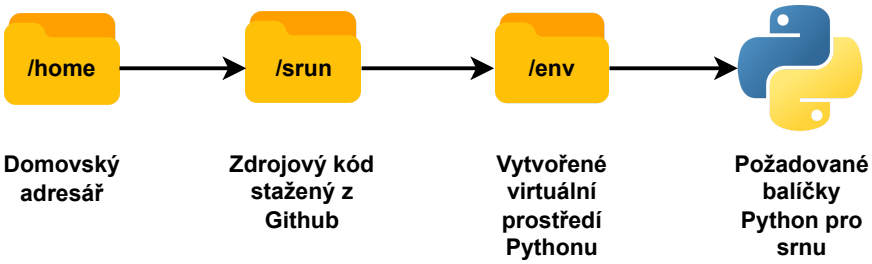
```
(env)$ deactivate
```

Tím dojde k deaktivaci virtuálního prostředí (env).

6.3.2 Spuštění aplikace ve virtuálním prostředí

Pokud již byl nainstalován virtualenv jako předchozí krok, lze provést následující kroky pro spuštění *srun* ve virtualenv:

1. Stáhnout zdrojový kód z GitHub do adresáře /home. Pořadí adresářů by mělo být stejné jako na obrázku 6.9.



Obr. 6.9: Struktura adresářů aplikace *srun*

2. Vytváření nové virtuální prostředí pod názvem "env" v adresáři /home/srun pomocí následujícího příkazu.

   ```
   $ virtualenv /home/srun/env
   ```

3. Aktivace virtuálního prostředí spuštěním následujícího příkazu.

   ```
   $ source /home/srun/env/bin/activate
   ```

4. Nainstalovat potřebné moduly (Tkinter, pygame, PIL) spuštěním následujících příkazů.

   ```
   (env)$ pip install pillow pygame
   (env)$ sudo dnf install python3-tkinter -y
   ```

5. Spustit spouštěč aplikací pro seniory (*srun*) spuštěním následujícího příkazu.

   ```
   (env)$ python main.py
   ```

   Tím se spustí skript "main.py" pomocí interpreteru Pythonu nainstalovaného ve virtuálním prostředí "env". Script "main.py" je hlavní spouštěcí kód pro aplikaci *srun*. skript mian.py lze spustit ve virtuálním prostředí (env), aniž by byl aktivován, a to spuštěním "main.py" přímo pod interpretem Python, který je nainstalován v (env), pomocí následujícího příkazu:

   ```
   $ /home/srun/env/bin/python3 /home/srun/main.py
   ```

7 Instalace operačního systému GNU/Linux na USB disk

V této sekci se vytvoří bootovatelný USB disk pomocí aplikace Rufus. Po vytvoření bootovatelného USB disku bude použit ke spuštění počítače a instalaci operačního systému na jiný USB disk. Tento proces zahrnuje nastavení spouštěcí priority počítače na USB disk tak, aby se počítač spouštěl z něj a ne z interního pevného disku. Po zavedení počítače z USB disku bude instalace pokračovat a jako cíl operačního systému bude vybrán druhý USB disk. To umožní instalaci operačního systému na USB disk a jeho následné využití jako přenosného operačního systému.

7.1 Stažení operačního systému

Pro instalaci operačního systému na USB disk je nejprve nutné stáhnout operační systém z oficiálních stránek dodavatele. V tomto případě bylo pro stažení vybráno Fedora workstation 36, a to kvůli výhodám popsaným v bodě 2.6. Pro stažení Fedory lze postupovat následujícím způsobem:

1. Otevření stránky pro stažení Fedora Workstation: https://getfedora.org/en/workstation
2. Kliknutí na zelené tlačítko s nápisem “Stáhnout nyní” pod sekcí Fedora workstation.
3. Stránka pak přesměruje uživatele na stránku, kde lze zvolit verzi Fedora Workstation, která se má stáhnout, v tomto případě byla verze 36.
4. Po výběru požadované verze bude uživatel vyzván, aby vybral příslušnou architekturu (32 bitovou nebo 64 bitovou), která odpovídá procesoru jeho počítače.

Obr. 7.1: Verze operačního systému Fedora

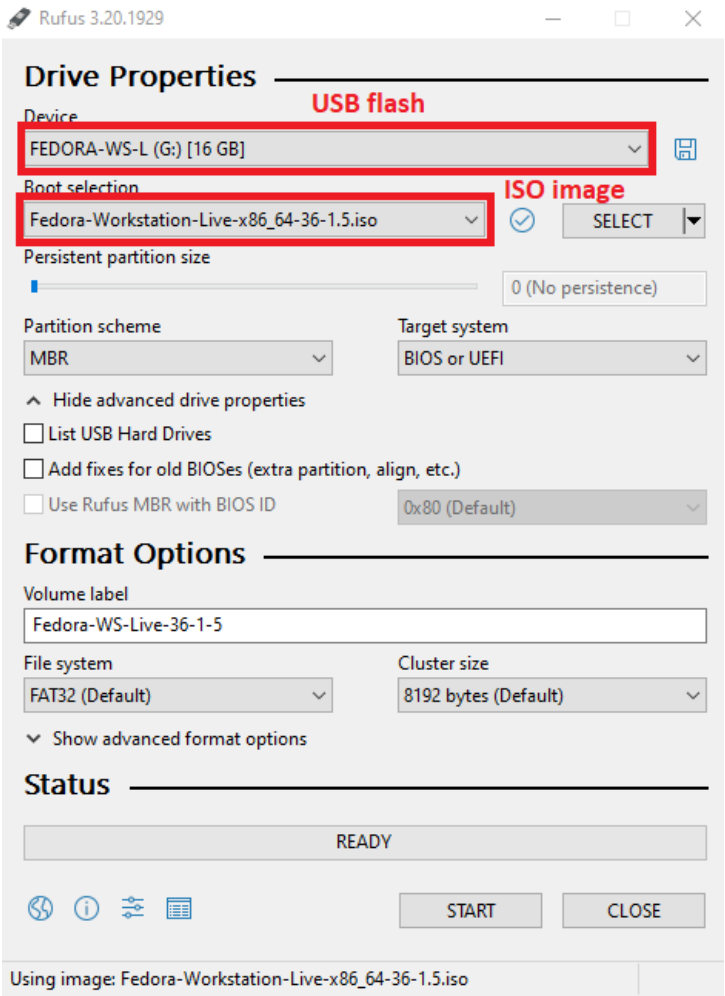
5. Jakmile je architektura vybrána, uživatel bude vyzván, aby si vybral umístění pro stahování s možností vybrat umístění.
6. Kliknutím na tlačítko “Stáhnout” se spustí stahování proces.

Po dokončení stahování lze pomocí softwaru, jako je Rufus pro Windows nebo Fedora media writer pro Fedora, vytvořit bootovatelný USB disk pomocí staženého obrazu Fedora Workstation.

7.2 Vytvoření spustitelného obrazu na USB disku

V tomto případě byl pro vytvoření bootovatelného USB disku zvolen software Rufus. Rufus je svobodný a open-source softwarový nástroj používaný pro vytváření bootovatelných USB disků. Jedná se o malý nástroj, který pomáhá uživatelům formátovat a vytvářet bootovatelné USB disky z ISO obrazových souborů. Je kompatibilní s různými operačními systémy, jako jsou Windows, Linux a MacOS. Pro vytvoření zaváděcího USB pomocí Rufus lze postupovat následujícím způsobem:

1. **Stáhnout Rufus:** Rufus lze stáhnout z oficiálních stránek na adrese https://rufus.ie/. Stáhnout by se měla nejnovější verze softwaru.
2. **Spustit Rufus:** Po stažení aplikace Rufus ji lze spustit dvojitým kliknutím na stažený spustitelný soubor. Instalace aplikace Rufus není nutná a lze ji spustit přímo ze staženého souboru.
3. **Vložení jednotky USB:** Připojením USB disku k počítači, Rufus automaticky rozpozná USB disk.
4. **Výběr USB disku:** V aplikaci Rufus v části “Zařízení” (Device) lze z rozbalovací nabídky vybrát USB disk, který byl vložen. Rozhodující je zvolit správný USB disk, protože Rufus smaže všechna data na vybraném disku.
5. **Výběr iso obrazu:** V části “Výběr spouštění”(Boot selection) klikněte na tlačítko “SELECT” a vyhledejte umístění bootovatelného souboru s obrazem (obvykle ISO soubor, nap “Fedora-Workstation-Live-x86_64-36-1.5”) ve vašem počítači, pro který chcete vytvořit bootovatelný USB. Vyberte soubor a klepněte na “Otevřít”.



Obr. 7.2: Nastavení aplikace Rufus

6. **Nastavení aplikace Rufus:** Ostatní nastavení v Rufus by mělo zůstat na svých výchozích hodnotách.
7. **Start Rufus:** Jakmile je nastavení nakonfigurováno, je třeba kliknout na tlačítko “START”, čímž se spustí vytvoření spouštěcího USB disku. USB disk bude naformátován, soubory z bootovatelného obrazu budou zkopírovány na USB disk.

Jakmile Rufus dokončí instalační proces, zobrazí se oznámení “READY”. Nyní lze USB disk bezpečně vysunout z počítače. Po ukončení výše uvedeného procesu. Výsledkem je, že pomocí Rufuse byl úspěšně vytvořen bootovatelný USB disk.

7.3 Zavedení z USB disku

V prvním kroku je potřeba vložit bootovací USB disk do USB portu počítače. Poté po zapnutí počítače je třeba okamžitě stisknout klávesu pro přístup k BIOSu. Klávesa pro stisknutí se může lišit v závislosti na počítačích výrobce, ale společné klávesy jsou F12, F2, F10 a Delete. Jakmile se dostanete k systému BIOS, přejděte pomocí šipek na klávesnici do sekce "Boot". Na kartě "Boot" je potřeba změnit pořadí spouštění tak, aby USB disk byl nahoře seznamu. Tím je zajištěno, že počítač před pokusem o zavedení z pevného disku nebo jiné zařízení bude spouštěno z USB disku. po uložení změn do systému BIOS. Počítač bude nyní restartováno a spouštěno z usb disku. V závislosti na typu spouštěče na USB disku můžete být vyzváni k stisknutí klávesy pro spuštění z USB disku.

7.4 Instalace operačního systému na USB disk

Instalaci operačního systému z bootovatelného USB disku na jiný USB disk lze provést následujícími obecnými kroky:

1. Výběr instalace operačního systému: Po spuštění počítače z bootovatelného USB disku se obvykle zobrazí nabídka nebo rozhraní, které vám umožní vybrat možnost instalace operačního systému, který chcete nainstalovat.
2. Výběr cílovou jednotku USB: Jakmile vyberete možnost instalace, můžete být vyzváni k volbě cílového USB disku, kam chcete operační systém nainstalovat. Vyberte USB disk, který chcete použít jako instalační cíl. Při výběru správného USB disku buďte opatrní, protože instalační proces smaže všechna data na vybraném USB disku.
3. Spustit instalaci: Proces instalace zahajte podle pokynů na obrazovce. To může zahrnovat výběr požadovaného nastavení instalace, jako je jazyk, rozložení klávesnice, časové pásmo a další možnosti konfigurace. Potvrďte výběr a pokračujte v instalaci.
4. Čekat na dokončení instalace: Počkejte na dokončení instalace, což může nějakou dobu trvat v závislosti na velikosti operačního systému a rychlosti vašich USB disků. Po dokončení instalace můžete být vyzváni k restartování počítače.
5. Odebrat spouštěcí jednotku USB: Po dokončení instalace počítač vypněte a spouštěcí USB disk z počítače můžete bezpečně vyjmout.
6. Zavedení z nainstalované jednotky USB: Vložte USB disk, na který jste nainstalovali operační systém, do USB portu v počítači. Zapněte počítač a zajistěte, aby byl v nastavení systému BIOS/UEFI nastaven pro spouštění z USB. Po spuštění počítače z USB disku by měl být operační systém nainstalován a připraven k použití.

8 Instalace aplikací pro seniory

V této kapitole budou popsány kroky pro instalaci všech seniorských aplikací, spouštěč aplikací pro seniory (*srun*), textový editor (*stext*), webový prohlížeč (*sweb*) a mail (*smail*), pro vybrané operační systémy (Fedora workstation 36 nebo Debian 11). Před zahájením procesu instalace je však třeba se postarat o některé předpoklady. Mezi tyto předpoklady patří aktualizace správce balíčků, instalace potřebných balíčků a konfigurace prostředí pro zajištění plynulé instalace a provozu aplikací pro seniory. Dodržováním těchto předpokladů lze zajistit bezproblémový proces instalace aplikací pro seniory.

8.1 Příprava prostředí pro Fedora workstation 36

V této části bude operační systém Fedora připraven k instalaci seniorských aplikací tak, že bude funkce "overview on startup" vypnuta. "Overview on startup" je funkce desktopového prostředí Fedora Gnome, které poskytuje vizuální shrnutí všech otevřených pracovních prostorů a umožňuje mezi nimi rychle přepínat. Ve výchozím nastavení je funkce "overview on startup" zapnuta při přihlášení uživatele do Fedory a je přístupná stisknutím "Super" kláves (známých také jako klávesa Windows). Zakázání "overview on startup" zajistí, že funkce nebude standardně zapnuta při přihlášení. Níže uvedené kroky vysvětlují, jak to provést:

1. Vypnutí automatického uzamčení obrazovky následováním příkazu.

   ```
   $ gsettings set org.gnome.desktop.screensaver lock-enabled false
   ```

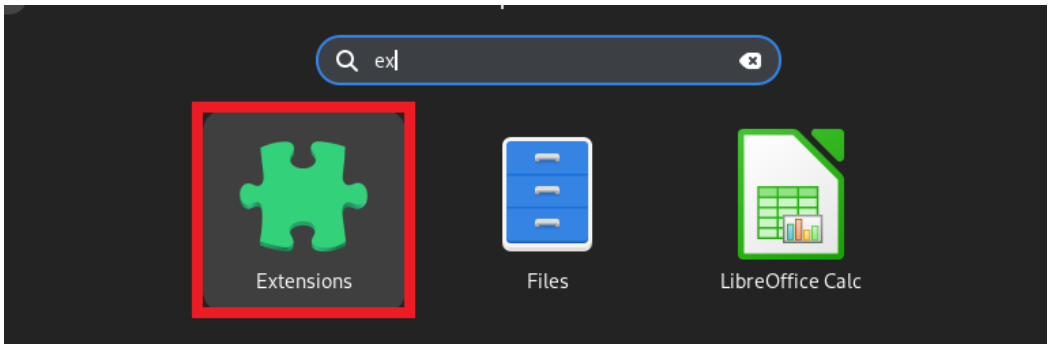
2. Stažení rozšiřujícího modulu prostředí gnome (gnome-shell-extensios) pomocí následujícího příkazu.

   ```
   $ sudo dnf install gnome-shell-extension* -y
   ```

3. Reloadnout správce plochy Gnome.

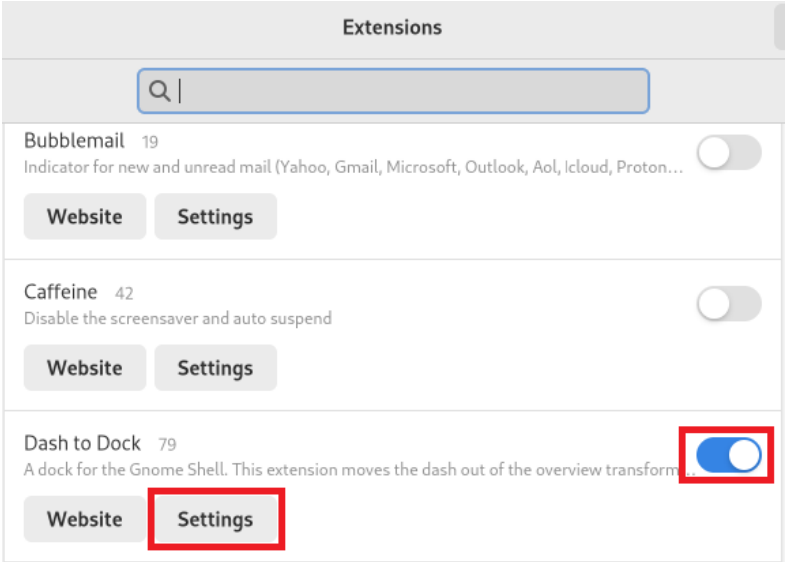
   ```
   $ killall -u username
   ```

4. Otevření rozšíření GNOME (Extensions).



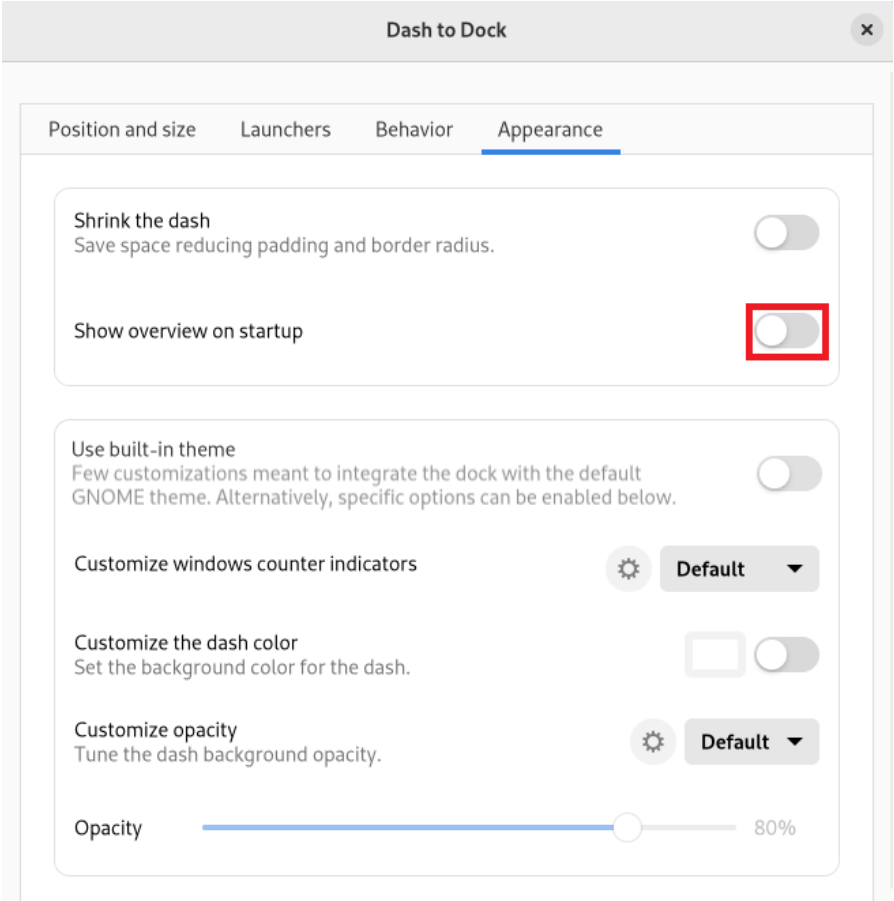
Obr. 8.1: Otevření rozšíření Gnome

5. Povolení “Dash to Dock” funkce v seznamu nainstalovaných rozšíření.



Obr. 8.2: Seznam nainstalovaných rozšíření Gnome

6. Kliknutím na tlačítko nastavení (Settings) vedle “Dash to Dock”, otevře se jeho nastavení. V záložce “Vzhled” (Appearance) je třeba vypnout volbu “Zobrazit přehled při spuštění” (Show overview on startup).



Obr. 8.3: Vypnout funkci (overview on startup)

Po provedení výše uvedených kroků bude Fedora připraven pro instalaci seniorských aplikací.

8.2 Instalace aplikace pro seniory

Skript "install_seniorOS.sh" byl navržen tak, aby zjednodušil proces instalace všech aplikací pro seniory, což zahrnuje spouštěč aplikací pro seniory, textový editor, email a webový prohlížeč. Spuštěním tohoto skriptu mohou uživatelé snadno nainstalovat všechny tyto aplikace pouze s jedním příkazem, místo aby museli procházet procesem instalace pro každou aplikaci zvlášť. Spuštěním skriptu "install_seniorOS.sh" mohou mít uživatelé jistotu, že mají všechny aplikace pro seniory nainstalované a připravené k použití. Následující kroky popisují, jak stáhnout a spustit skript "install_seniorOS.sh".

1. Stahování "install_seniorOS.sh" script z Github: kliknutím na odkaz "install_seniorOS.sh" se otevře repozitář, které obsahuje skript ke stažení.

Jakmile je soubor otevřen, kliknutím na tlačítko “Raw” umístěné v pravém horním rohu se zobrazí nezpracovaná verze souboru. Soubor se uloží do počítače kliknutím pravým tlačítkem myši kdekoli na stránce a výběrem možnosti “Uložit jako” a poté výběrem umístění v počítači, kde bude soubor uložen.

2. Spuštění skriptu pod oprávněním sudo: jakmile je skript stažen, může být spuštěn pod sudo zadáním níže uvedeného příkazu do terminálu. Uživatel je vyzván k zadání hesla a skript je poté spuštěn se zvýšenými oprávněními.

```
$ sudo sh install_seniorOS.sh
```

Popis skriptu install_seniorOS.sh

Výpis 8.1: Ukázkový kód skript install_seniorOS.sh

```
#!/bin/bash
#user who run the script
user=$(echo $SUDO_USER)
#installation directory
dir_to_install="/home/"$user
echo The user is $user
echo Installation direction $dir_to_install
#check if the Operating system is Linux
if [ "$(uname -s)" = "Linux" ]
then
  if [ -f /etc/fedora-release ]
    then
  echo "The running operating system is Fedora"
  #update the system
  dnf update -y
        #install virtualenv for python
  check_command_success "dnf install virtualenv -y" "Install virtualenv"
        #check if the Senior system was installed
  check_if_system_exists
        #check if the source code from Github is downloaded
  check_if_os_exists
  #download source from Githut
  check_command_success "git clone https://github.com/forsenior/os.git /tmp/
    $new_dir" "Download         source code"

  #move srun, stext, sweb, smail to /home/
  check_command_success "sudo mv /tmp/$new_dir/sinstall/srun /home" "move srun to /
    home"
  check_command_success "sudo mv /tmp/$new_dir/sinstall/stext /home" "move stext to
     /home"
  check_command_success "sudo mv /tmp/$new_dir/sinstall/sweb /home" "move sweb to /
    home"
  check_command_success "sudo mv /tmp/$new_dir/sinstall/smail /home" "move smail to
     /home"

  #create virtual env inside srun,stext,sweb,smail.
  check_command_success "virtualenv /home/srun/env" "Create srun env"
  check_command_success "virtualenv /home/stext/env" "Create stext env"
  check_command_success "virtualenv /home/sweb/env" "Create sweb env"
```

```
35    check_command_success "virtualenv /home/smail/env" "Create smail env"
36
37    #change OS owner
38    check_command_success "sudo chown -R $user /home/srun /home/stext /home/sweb /
        home/smail" "Change     owner to $user"
39
40    #install needed packages for stext app
41    source /home/stext/env/bin/activate && pip install pyqt5 xhtml2pdf  bs4 markdown
        && deactivate
42    #install needed packages for srun app
43    source /home/srun/env/bin/activate && pip install pillow pygame && dnf install
        python3-tkinter -y     && deactivate
44    #install needed packages for sweb app
45    source /home/sweb/env/bin/activate && dnf install python3-tkinter -y &&
        deactivate
46    #install needed packages for smail app
47    source /home/smail/env/bin/activate && dnf install python3-tkinter -y &&
        deactivate
48
49    #create autostart file to run srun
50    auto_start_srun
51    #change autostart owner
52    check_command_success "sudo chown -R $user /home/$user/.config/autostart" "Change
         autostart owner     to $user"
53    #allow the user to automatically log in to their account without having to
        manually enter their       username and password
54    auto_login_to_gnome "gdm" "custom.conf"
55
56    #command to disable the automatic Screen Lock in Fedora Workstation
57    dconf write /org/gnome/desktop/screensaver/lock-enabled false
58
59    #ask for restart
60    request_restart
61
62    else
63        echo "The System is not Linux"
64    fi
65  fi
```

Skript 8.1 zkontroluje, zda je operační systém Linux, konkrétně Fedora. Pokud je to Fedora, pak provede několik akcí:

1. Systém se aktualizuje spuštěním příkazu “dnf update -y”.
2. Virtuální prostředí (virtualenv) bude nainstalováno spuštěním příkazu “dnf install virtualenv -y”.
3. Ověření, zda aplikace pro seniory jsou již nainstalovány voláním funkce *check_if_system_exists()*.
4. Ověření, zda je zdrojový kód z GitHub již staže voláním funkce *check_if_os_exists()*.
5. Stažení zdrojový kód z GitHub spuštěním příkazu “git clone https://github.com/forsenior/os.git /tmp/$new_dir”.
6. Přesunutí čtyř aplikace (*srun*), (*stext*), (*sweb*) a (*smail*) do adresáře /home

spuštěním příkazu “sudo mv /tmp/$new_dir/sinstall/srun /home” a podobných příkazů pro ostatní aplikace.

7. Vytvoření virtuálního prostředí uvnitř každé ze čtyř aplikací spuštěním příkazu “virtualenv /home/srun/env” a podobných příkazů pro ostatní aplikace.
8. Změna vlastníka čtyř aplikací na uživatele, který spustil skript, a to spuštěním příkazu “sudo chown -R $user /home/srun /home/stext /home/sweb /home/smail”.
9. Instalace požadovaných modulů pro každou ze čtyř aplikací uvnitř jejich virtuálního prostředí. Architektura složky instalovaných aplikací bude jako na obrázku 8.4.
10. Vytváření souboru automatického spuštění pro spouštěč aplikací pro seniory (*srun*) pomocí funkce *auto_start_srun()*.

Výpis 8.2: Ukázkový kód pro funkci automatické spuštění *srun* po přihlášení

```
auto_start_srun(){
mkdir -p /home/$user/.config/autostart && echo "[Desktop Entry]
Type=Application
Name=Pyapp
Exec=/home/srun/env/bin/python /home/srun/main.py
Terminal=false" >/home/$user/.config/autostart/autostart.desktop
chmod +x /home/$user/.config/autostart/autostart.desktop
}
```

Funkce *auto_start_srun()* nejprve vytvoří adresář “/home/$user/.config/autostart” pomocí příkazu “mkdir” s příznakem -p, který vytvoří adresář, pouze pokud ještě neexistuje. V adresáři pak vytvoří soubor s názvem “autostart.desktop”. obsah souboru “autostart.desktop”, určuje, že se jedná o aplikaci, která by se měla spouštět automaticky po přihlášení, její název je “Pyapp” a určuje příkaz k vykonání “/home/srun/env/bin/python /home/srun /main.py”. Také nastaví Terminal=false pro spuštění aplikace bez otevření okna terminálu.

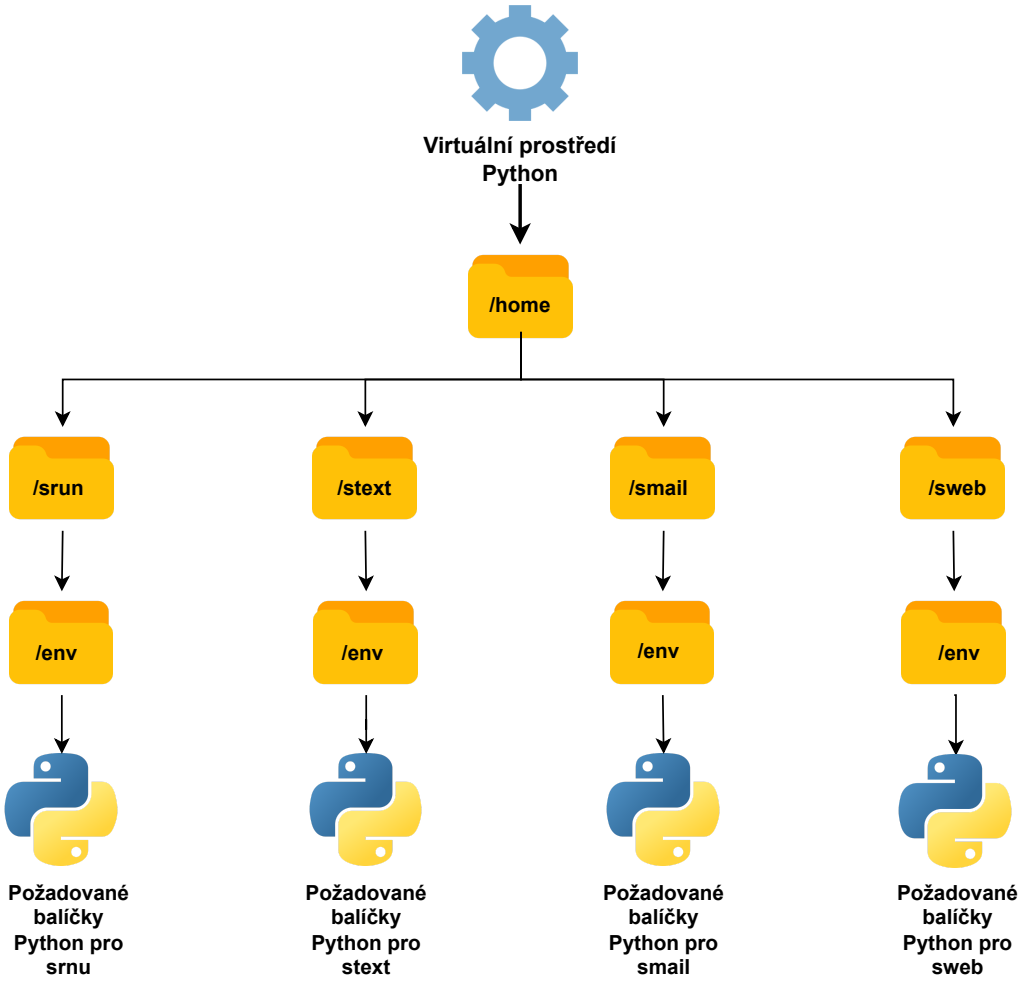
11. Změna vlastníka souboru “autostart.desktop” na uživatele, který spustil skript, spuštěním příkazu “sudo chown -R $user /home/$user/.config/autostart”.
12. Povolit uživateli, aby se automaticky přihlásil ke svému účtu, aniž by musel ručně zadat své uživatelské jméno a heslo vyvoláním funkce *auto_login_to_gnome()*.

Výpis 8.3: Ukázkový kód pro funkci automatické přihlášení do desktopového prostředí GNOME

```
auto_login_to_gnome(){
gdm="$1"
custom="$2"
custom_path="/etc/$gdm/$custom"
Config1="AutomaticLoginEnable=true"
Config2="AutomaticLogin=$user"
#if the file custom.conf exists
if [ -f $custom_path ]
then
  echo "File custom  found"
  if grep -E "$Config1" $custom_path && grep -E "$Config2" $custom_path
     then
      echo "Automatic login has been set"
  else
      echo "Set auto login"
      echo "[daemon]
            AutomaticLoginEnable=true
            AutomaticLogin=$user" >> $custom_path
  fi
else
    echo "File custom NOT found,
    Create custom.conf file in directory /etc/$gdm/$custom, and run this
    script again"
    exit 0
fi
}
```

Funkce *auto_login_to_gnome()* přidá do souboru s názvem "custom.conf" dva řádky konfigurace. Účelem těchto řádků je umožnit automatické přihlášení do desktopového prostředí Gnome, aniž by uživatel musel zadat uživatelské jméno a heslo.

13. Požádat uživatele o restartování systému vyvoláním funkce *request_restart()*.



Obr. 8.4: Architektura nainstalovaných aplikací pro seniory

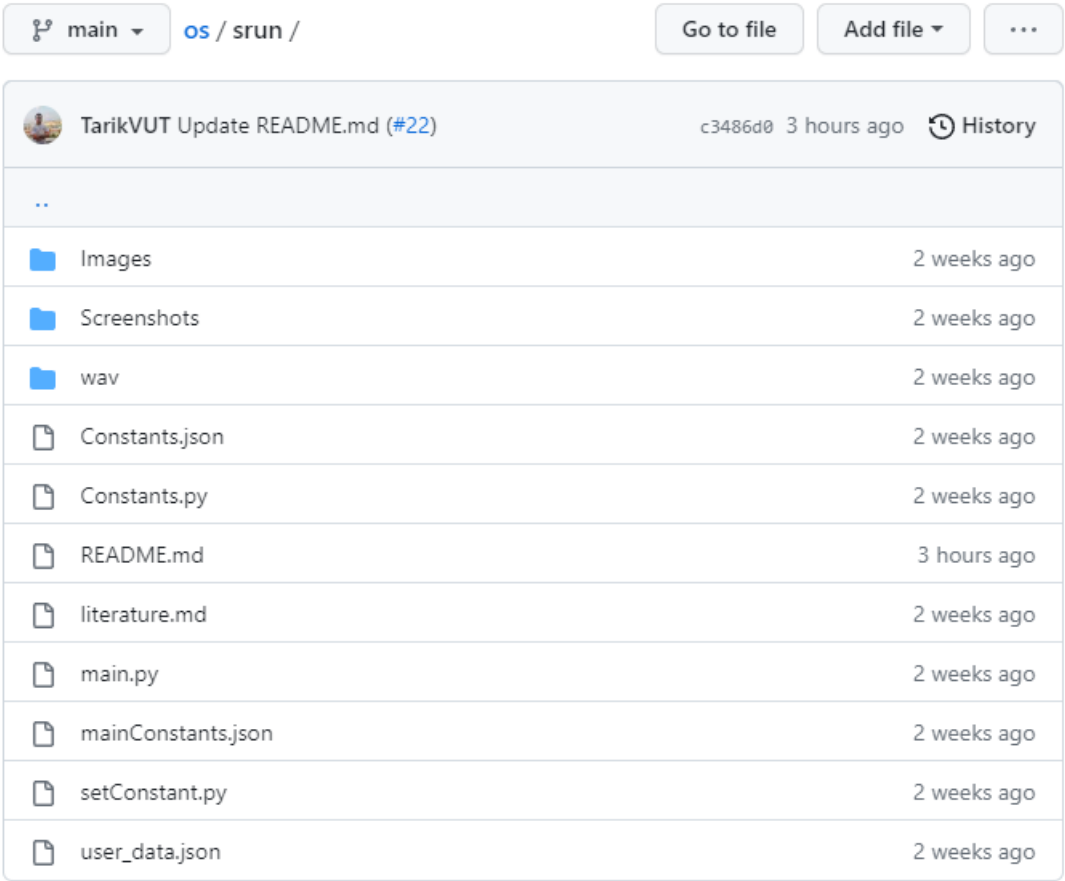
8.3 Oblasti pro budoucí zlepšení

Mezi možné oblasti budoucího zlepšení by mohly patřit:

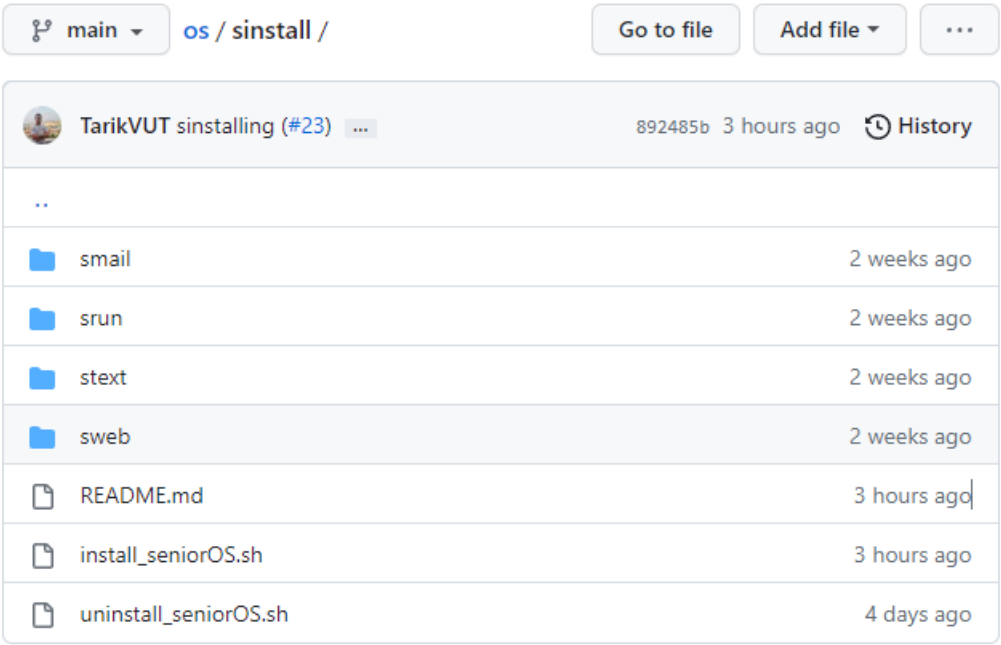
- Zmenšení velikosti operačního systému odstraněním nepotřebných programů umožňujících instalaci na menších USB discích.
- Využití live-CD verze linuxové distribuce a nahrazení výchozího Desktop manažera aplikací Spouštěč aplikací pro seniory.
- Do Spouštěče aplikací pro Senior by mohly být přidány další funkce, jako je přizpůsobení rozhraní a snadná instalace nových aplikací.
- Uživatelské jméno a heslo aplikace by také mohlo být propojeno s uživatelským účtem operačního systému.
- Rozšíření funkce hlasové asistence o další jazyky s jasnou výslovností.

9 Zveřejnění zdrojového kódu na repozitáři GitHub

Výsledky bakalářské práce, programový kód aplikace *srun* a skript pro instalaci aplikací pro seniory jsou publikovány na repozitáři Github pod licencí MIT. Součástí úložiště je zdrojový kód aplikace a také podrobná dokumentace, jak ji nainstalovat a spustit. Pro instalaci všech aplikací pro seniory (*srun*, *stext*, *sweb*, *smail*), je třeba postupovat podle kroků uvedených v repozitáři sinstall. Spouštěč aplikací pro seniory lze také nainstalovat nezávisle stažením všech souborů z jeho adresáře a následnou instalací potřebných modulů Pythonu pomocí příkazů uvedených v repozitáři *srun*.



Obr. 9.1: Struktura repozitáře GitHub *srun*



Obr. 9.2: Struktura repozitáře GitHub *sinstall*

Závěr

Cílem této bakalářské práce bylo upravit linuxovou distribuci tak, aby byla spustitelná z USB disku a měla jednoduché a přívětivé grafické uživatelské rozhraní, které vyhovuje seniorům ve skupině 90 let. Cílem tohoto projektu bylo poskytnout seniorům snadno použitelnou počítačovou zkušenost a řešit problémy, kterým čelí při používání počítačů a internetu. Výsledná aplikace nazvaná (Spouštěč aplikací pro seniory) umožňuje seniorům spustit tři různé aplikace (textový editor, webový prohlížeč, e-mail) a snadno vypnout operační systém, a to vše prostřednictvím jednoduchého a intuitivního rozhraní.

Teoretická část bakalářské práce poskytla podrobný přehled operačních systémů se zvláštním zaměřením na systémy s linuxovým jádrem. Dále byl důkladně prodiskutován systém X Window spolu s populárními desktopovými manažery jako Gnome a KDE. Tyto znalosti byly klíčové pro pochopení základní struktury a složek operačního systému a byly zásadní pro vývoj aplikace.

Praktická část obsahovala podrobné vysvětlení kroků, které byly při vytváření aplikace (Spouštěč aplikací pro seniory) podniknuty, včetně výzev, které se objevily během vývojového procesu. Byly popsány hlavní funkce poskytované seniorům prostřednictvím aplikace, kterými jsou spouštění programů vytvořených v projektu (Operační systém pro seniory) a snadná cesta k vypnutí operačního systému, i další rozšiřující funkce dostupné administrátorům, které jim umožňují měnit jazyky hlasové asistence, přepínat mezi barevnými režimy, měnit uživatelské heslo a uzavírat aplikaci *srun*. Poté byly zdůrazněny výhody využití virtuálního prostředí pro spuštění vyvinutého programu spolu s tím, jak aplikaci spustit ve virtuálním prostředí. Dále praktická část popisovala podrobné pokyny pro instalaci operačního systému Fedora na USB disk pomocí programu Rufus, včetně nezbytných nastavení, která je třeba nastavit v systému BIOS, aby bylo možné spouštění z USB disku.

Poslední praktická část vysvětluje, jak připravit operační systém Fedora pro instalaci všech seniorských aplikací pomocí shellového skriptu. Tento skript byl navržen tak, aby automatizoval proces instalace všech potřebných balíčků a modulů pro hladký chod aplikací.

Programový kód aplikací Spouštěč aplikací pro seniory a skript pro instalaci všech vytvořených aplikací pro seniory v projektu (Operační systém pro seniory) jsou dostupné na repositáři GitHub https://github.com/forsenior/os/.

Literatura

[1] *SILBERSCHATZ, Abraham, Peter Baer GALVIN a Greg GAGNE. Operating system concepts: international student version. 8th ed. Hoboken, NJ: John Wiley, c2010. ISBN 978-0-470-23399-3.*

[2] *ARPACI-DUSSEAU, Remzi H.; ARPACI-DUSSEAU, Andrea C. Operating systems: Three easy pieces. Arpaci-Dusseau Books, LLC, 2018.*

[3] *FOX, Richard. Linux with operating system concepts. Second edition. Boca Raton: CRC Press, Taylor & Francis Group, 2022. ISBN 978-1-032-06345-4.*

[4] *JOHNSON, Michael K. Linux Kernel: /Kernelova príručka/. Slovakia: mffexpert, 1994.*

[5] *KUČERA, František. Linux a systém GNU [online].[cit.2022-12-4].Dostupné z: https://www.gnu.org/gnu/linux-and-gnu.htm*

[6] *MUSTER, John. Introduction to unix and linux. New York: McGraw-Hill/Osborne, 2003. ISBN 0-07-222695-1.*

[7] *ABHISHEK, Prakash. Linux Directory Structure Explained [online].[cit.2022-12-2]. Dostupné z: https://linuxhandbook.com/linux-directory-structure/*

[8] *BLUM, Richard. Linux for dummies. 10th Edition. Hoboken, NJ: John Wiley, [2020]. ISBN 978-1-119-70425-6.*

[9] *QUERCIA, Valerie; O'REILLY, Tim. X Window system user's guide for X11 R3 and R4. O'Reilly & Associates, Inc., 1990.*

[10] *ZLOSKÝ, Ondřej. Linux Fedora Core 5: instalační a uživatelská příručka. Brno: Computer Press, 2006. ISBN 80-251-1056-7.*

[11] *PETERSEN, Richard. Linux: The complete reference. McGraw-Hill, Inc., 2007.*

[12] *SOBELL, Mark G. Mistrovství v RedHat a Fedora Linux: pro verze Fedora Core 2 až 5 a RedHat 3 a 4. Brno: Computer Press, 2006. ISBN 80-251-1152-0.*

[13] *Artem S. Tashkinov. Linux Desktop Environments System Usage (Gnome, KDE, XFCE, LXQT, Cinnamon, Mate) [online].[cit.2022-12-3]. Dostupné z: https://itvision.altervista.org/linux-desktop-environments-system-usage.html*

[14] *SIMMONDS, Chris. Mastering Embedded Linux Programming. First published. Birmingham: Packt Publishing Ltd., 2015. ISBN 978-1-78439-253-6.*

[15] *Tkinter. Python interface to Tcl/Tk. [online].[cit.2022-12-5]. Dostupné z: https://docs.python.org/3/library/tkinter.html*

[16] *NIELSON, Seth James; MONSON, Christopher K. Practical Cryptography in Python: Learning Correct Cryptography by Example. Apress, 2019.*

[17] *VIAZANICA, Branislav. Prostredie PyGame na vývoj hier: diplomová práca. Banská Bystrica, 2021.*

[18] *SARMIENTO, Matthew. A Guide to Python's Virtual Environments. [online].[cit.2022-12-5]. Dostupné z: https://towardsdatascience.com/virtual-environments-104c62d48c54*

[19] *LUTZ, Mark. Learning python: Powerful object-oriented programming. "O'Reilly Media, Inc.", 2013.*

[20] *HELLMANN, Doug. The Python Standard Library by Example: PYTH STAND LIB BY EXAM -p1. Addison-Wesley Professional, 2011.*

Seznam symbolů a zkratek

srun	Aplikace spouštěč aplikací pro seniory
stext	Textový editor pro seniory
sweb	Webový prohlížeč pro seniory
smail	E-mailová aplikace pro seniory
GNU	GNU's not Unix, GNU není Unix
GUI	Graphical user interface, Grafické uživatelské prostředí
Gnome	GNU Network Object Model Environment, Prostředí síťového objektového modelu GNU
KDE	K Desktop Environment, K Desktopové prostředí
Bios	Basic input/output system, Základní vstupní/výstupní systém
MBR	Master Boot Record, Hlavní spouštěcí záznam
GRUB	Grand Unified Bootloader
Init	Initialization, Inicializace
sudo	Substitute user do