Gymnázium Brno, Vídeňská, příspěvková organizace

Leek - Správce souborů

Maturitní projekt - VPI

Otakar Kočí

Otakar Kočí
VPI
6. března 2022

Obsah

1 Úvod

Správce souborů jsou v dnešní době nepostradatelnými pomocníky při práci na počítačích a na trhu jich existuje veliké množství. Každý z nich nabízí uživateli mírně odlišné rozhraní a různorodé funkce. Některé se již vyvinuly ve velice složité a komplexní programy nabízející vše potřebné ke správě souborů na dané platformě. Otázkou se ale stává, jak složité by bylo v dnešní době vytvořit vlastní program pro správu souborů, který by měl alespoň základní funkce typu zobrazování souborů, složek a disků. A dále umožňoval kopírování, přesouvání, přejmenování, mazání, řazení, vyhledávání a další. Právě tomuto se ve svém projektu věnuji. Mým cílem je vytvoření jednoduchého správce souborů, který bude díky svému ovládání a množství funkcí použitelný a alespoň trochu komfortní pro uživatele. Zároveň se pokouším o co nejjednodušší a pokud možno nekomplexní řešení.

2 Analýza problému a návrh jeho řešení

Vytvoření správce souborů není úplně jednoduchým úkolem a přináší s sebou hned několik problémů, které mají výrazný dopad na funkčnost celé aplikace a její použití. Příkladem může být cílená platforma a její struktura souborů, kontroly existence vyhledaných souborů, nebo algoritmická složitost kódu správce souborů jako taková.

2.1 Problém cílené platformy

Existuje mnoho platforem a operačních systémů, na které by bylo vhodné s programem na správu souborů cílit. Například Microsoft Windows, operační systémy založené na Linuxu, Mac OS, Android a další. Tyto všechny operační systémy s sebou ale přináší malý problém a to ve formě zpracování a virtualizace souborového systému. Každý z těchto operačních systémů totiž správu a interakci se soubory, složkami a disky definuje a umožňuje trochu odlišně, což v případě správce souborů, který by je měl podporovat všechny vede k větší složitosti samotného kódu. Původně bylo mým plánem vytvoření správce souborů pro systém Microsoft Windows a pro systémy na bázi Linuxu. Ale z důvodu snahy o co nejjednodušší a nekomplexní řešení jsem se rozhodl zatím podporovat pouze systém Microsoft Windows.

2.2 Veliké množství kontrol

Programování správce souborů s sebou také nese veliké množství potřebných kontrol při práci se soubory. Například lze uvést problém vyhledaných souborů. V případě, že program zjistí a zobrazí výsledky vyhledávání, tak je stále nutné kontrolovat jejich existenci při práci s nimi, jelikož zdaleka ne všechny soubory musí existovat až do doby, kdy s nimi uživatel provádí nějakou operaci. Toto ale platí nejenom pro vyhledávání, ale i pro další funkce aplikace pro správu souborů. Dále musíme brát ohled na práci se samotnými soubory a složkami, jelikož pro práci s nimi například nemusíme mít dostatečná

práva, nebo nám systém nemusí dovolit s nimi pracovat z jakýchkoli jiných důvodů.

2.3 Složitost řešení

Dalším problémem je samotná složitost aplikace pro správu souborů. Jednak se jedná o složitost kódu logiky programu jako takového, ale i funkcí běžících na pozadí a rozhodně i složitost rozhraní.

2.3.1 Řešení hlavní logiky a funkcí běžících na pozadí

Jelikož jsem si již na začátku projektu stanovil cíl pokusit se o co nejjednodušší a pokud možno nekomplexní řešení, rozhodl jsem se využít programovacího jazyka Python, který se pro tyto účely velmi hodí. Je pravdou, že se jedná o skriptovací jazyk, který je výrazně pomalejší než například staticky typované jazyky typu C, ale díky nepřeberné sadě již vestavěných funkcí a jednoduchosti syntaxe se velmi hodí pro vytvoření tohoto typu programu v této velikosti. Zároveň nás ani nijak výrazně negativně neovlivňuje jeho nižší výkon, jelikož naším cílem je jednoduchá práce se soubory a využití vestavěných funkcí například pro kopírování souborů, které tak jako tak využívají knihoven systému a jiných rychlejších jazyků. Python nám tedy pro naše potřeby programu pro správu souborů postačí a díky jeho využití se nám zjednoduší a zpřehlední kód, jak hlavní logiky programu, tak i dalších podpůrných funkcí běžících na pozadí.

2.3.2 Řešení rozhraní

Původně bylo mým plánem vytvořit okenní aplikaci. Od tohoto rozhodnutí jsem ale nakonec odstoupil, jelikož jsem se začal zabývat možností použít pro potřeby programu zobrazení v konzoli, které je jednak mnohem jednoduší na vytvoření a i v některých ohledech praktičtější, i když zaostává v jiných. Zároveň jsem si stanovil výzvu pokusit se zjistit jaké možnosti vlastně zobrazení v konzoli nabízí a jaké jsou jeho limitace. Pro potřeby svého programu jsem objevil rozsáhlou knihovnu pro práci s rozšířeným výstupem do konzole pro Python jménem Rich. Tato knihovna umožňuje do konzole vypisovat v podobě tabulek, různých typů písma, zarovnání, může zobrazovat různé barvy, odrážky, stromové výpisy a mnoho dalšího. Zároveň je její použití velice jednoduché a intuitivní. Pro potřeby programu jsem, ale nakonec využil jen několik základních funkcí této knihovny.

3 Výsledné řešení a jeho popis

Z důvodů snahy o pokud možno co nejjednodušší a nekomplexní řešení jsem tedy zvolil programovací jazyk Python a konzoli jako formu výstupu. Pro vylepšení možností výpisu používám knihovnu Rich. Platformu jsem zvolil Microsoft Windows. Dále jsem použil program pyinstaller pro zabalení a vyexportování hotového programu se všemi závislostmi a knihovnami. Problémem totiž je, že všude, kde by se program spouštěl jen za pomocí skriptu by bylo potřeba mít nainstalovanou správnou verzi Pythonu a to Python 3.10 a také Rich ve verzi 11.2. Pomocí vyexportování se tento problém vyřešil. Vyexportovanou aplikaci by mělo jít spustit na jakémkoli zařízení s operačním systémem Microsoft Windows 8 nebo novějším. Potřebná architektura je 64 bitů.

3.1 Struktura programu

V programu se nachází podprogram `main` obstarávající hlavní logiku a zobrazení programu. Pracuje v tomto pořadí:

- Zaktualizuje informace o discích a zjistí existenci položek v aktuálním adresáři
  - Zavoláním funkce `refresh_all`
- Zaktualizuje zobrazení
  - Zavoláním `clear` vyčistí obrazovku
  - Zavoláním `create_table` nechá vytvořit nový tabulkový list
  - Tabulku vytiskne
  - Zavolá `print_page` pro výtisk informací o aktuální stránce a počtu souborů zobrazených a zobrazitelných
  - Zavolá `print_sortinfo` pro výtisk informací o způsobu řazení
  - Zavolá `print_errors` pro výtisk chybových hlášení, pokud jsou

    - Zavolá `print_info` pro výtisk informací o průběhu příkazů
- Nechá uživatele zadat nový příkaz a jeho parametry
- Vstup uživatele zpracuje a zavolá potřebné funkce pro zpracování příkazu

Po dokončení zpracování příkazu se tento proces opakuje až do ukončení programu.

3.2 Globální proměnné

Jelikož se jedná o skriptovací jazyk, tak jsem k ukládání veškerých dat použil globální proměnné. Tento krok vedl k dalšímu zjednodušení složitosti řešení.

4 Informace o programu a jeho ovládání

4.1 Základní informace o programu

Leek (pórek) je jednoduchý konzolový správce souborů ovládaný krátkými příkazy, jako každý správce souborů umí zobrazovat soubory, složky a základní informace o nich, veškeré zobrazování se děje na tabulkových listech, jejichž velikost si můžete sami upravit. Zároveň můžete volně mezi těmito listy přecházet. Soubory, disky a složky voláte dle jejich zobrazovaného ID. Leek je schopen volně přecházet mezi disky, vracet se zpět na root disku i na root celku. Zobrazení můžete kdykoli aktualizovat. Součástí leeku je i jednoduché sledování chyb a úspěšných operací, které uživateli umožní se jednoduše vyznat ve výsledcích zadaných příkazů. V leeku si můžete poznačovat soubory pro budoucí práci s nimi, soubory poznačené si také můžete nechat zobrazit, nebo jejich označení zrušit. Leek samozřejmě umí soubory a složky přesouvat, přejmenovávat, kopírovat, kopírovat a přejmenovávat, mazat, vyhledávat v nich a zobrazení různě seřazovat.

4.2 Seznam funkcí

Toto je seznam veškerých funkcí, které správce souborů Leek podporuje:

Funkce	**Příkaz**	**Parametry**
Nápověda programu	help	bez parametrů
Informace o programu	info	bez parametrů
Akutalizace	refresh	bez parametrů
Otevření složky či souboru	open	[id položky k otevření]
O úroveň výše	up	[číslo o kolik se přesunout (volitelné)]
Vytvoření složky	makedir	[název složky]
Velikost složky	dirsize	[id složky ke zjištění její velikosti]

Funkce	Příkaz	Parametry
Zpátky ke kořenům disků	roots	bez parametru
Zpátky ke kořeni disku	root	bez parametru
Přesunutí na jiný disk	drive	[písmeno disku]
Změna maximálního počtu položek k zobrazení na listu	rows	[číslo jako maximální počet zobrazitelných položek]
Přesune na další list	next	[číslo o kolik se posunout dopředu (je volitelné)]
Přesune na předchozí list	previous	[číslo o kolik se posunout dozadu (je volitelné)]
Ukončení programu	exit	bez parametrů
Poznačení položek bez předchozích	select	[číslo položky, nebo čísla položek, která jsou oddělená jednoduchou mezerou, nebo rozsah ve formátu číslo mezera pomlčka mezera a číslo]
Přidání k poznačeným položkám	add	[číslo položky, nebo čísla položek, která jsou oddělená jednoduchou mezerou, nebo rozsah ve formátu číslo mezera pomlčka mezera a číslo]
Odoznačí označené	unselect	[číslo položky, nebo čísla položek, která jsou oddělená jednoduchou mezerou, nebo rozsha ve formátu číslo mezera pomlčka mezera a číslo (je volitelné)]
Zobrazí označené	showselect	bez parametru
Vyhledávání v aktuálním adresáři a ve všech podadresářích	search	[hledaný výraz zadaný jako jednoslovný text]
Nastavení řazení	sort	[name/size/changed/created up/down]

Funkce	Příkaz	Parametry
Přejmenuje položku nebo položky nebo pracuje s označenými položkami	rename	[nový_název kde_bude_počítadlo začátek_počítadla zvyšování_počítadla počet_cifer_počítadla zadní_název (parametr 1 je povinný)]
Zkopíruje označené položky do aktuálního adresáře	copy	bez parametru
Zkopíruje a přejmenuje označenépoložky do aktuálního adresáře	copyrename	[nový_název kde_bude_počítadlo začátek_počítadla zvyšování_počítadla počet_cifer_počítadla zadní_název (parametr 1 je povinný)]
Přesune označené položky do aktuálního adresáře	move	bez parametru
Odstraní všechny označené položky	remove	bez parametru

4.3 Příklady vstupu

- `open 24`
- `help`
- `select 1 - 24`
- `search .log`
- `rename obrazek after 0 1 3 .png`
- `sort name up`

5 Závěr

Aplikace je schopna zprostředkovat veškeré funkce, které jsem plánoval implementovat. Zároveň doufám, že se mi podařilo vytvořit alespoň trochu intuitivní a originální ovládání a zobrazení. Na druhou stranu by ale aplikaci neuškodily úpravy vzhledu a výpisů, přidání více funkcí, detailnější chybová hlášení a verze pro další operační systémy. Z hlediska lepších výpisů by se mohly implementovat tzv. *progress bary*, které jsou součástí knihovny Rich. Dovedu si představit jejich využití například u kopírování a přesouvání souborů. Z hlediska verze pro další systémy by mělo být relativně jednoduché vytvořit verzi pro operační systémy na bázi Linuxu, jelikož bych musel pouze upravit komunikaci se strukturou souborového systému, který je používán Linuxovými distribucemi.

6 Informace o programu

- **Název aplikace - leek.py / leek.exe / Leek a Pórek**
- **Programovací jazyk - Python**
- **Počet řádků kódu - 1767**
- **Velikost skriptu - 86016 bajtů**
- **Velikost vyexportované .exe aplikace - 7616 kilobajtů**
- **Závislosti - knihovna Rich a její závislosti, základní knihovny Pythonu**
- **Balící nástroj - Pyinstaller**
- **Minimální verze Pythonu - 3.10**
- **Použitá verze Rich - 11.2**
- **Použitá verze Pyinstaller - 4.10**
- **Minimální verze operačního systému - Microsoft Windows 8 64 bit**
- **Příkaz pro zabalení v programu Pyinstaller**
  - `pyinstaller leek.py --hidden-import rich --hidden-import os --hidden-import shutil --onefile`