Linux Operating System Utilities

Engenharia Informática

ANO/SEMESTRE: 2023-2024 / 6º Semestre

Unidade Curricular: Sistemas Operativos
Professor: Pedro Rosa

Carolyne Silva - 20210046
Gustavo Farinha – 20211115
Grupo 7
github.com/CMS77/Linux-Operating-System-Utilities

Índice

Descrição do Problema

Atualmente, a capacidade de interagir eficazmente com sistemas Linux é fundamental para uma variedade de utilizadores e administradores de sistemas. Embora o Linux ofereça uma vasta gama de utilitários e comandos prontos a usar, há sempre espaço para melhorias e personalizações para satisfazer as necessidades específicas de cada indivíduo ou organização, promovendo assim uma maior produtividade e controlo sobre o sistema. Esta capacidade de personalização e melhoria contínua realça a natureza dinâmica e adaptável do ecossistema Linux, tornando-o uma escolha poderosa para utilizadores que valorizam a flexibilidade e a liberdade de personalização nos seus sistemas operativos.

Para este projeto o objetivo é desenvolver utilitários para o Linux em Linguagem C, que proporciona um excelente exercício de programação, permitindo uma compreensão mais profunda sobre como as ferramentas funcionam. Para além de contribuir para a comunidade e disponibilizar um conjunto de utilitários open-source, oferecendo alternativas aos utilitários existentes, além de promover a colaboração e o compartilhamento de conhecimento.

Alterações Realizadas Durante o Projeto

Durante o projeto, a principal alteração em relação à proposta inicial foi a inclusão da opção `--help` em todos os comandos utilizados. Esta modificação permitiu uma melhor compreensão dos comandos e as suas opções, facilitando a execução das tarefas previstas.

Casos de Uso

ls: Este utilitário será desenvolvido para listar os arquivos e diretórios presentes no diretório atual.

1. `-l`: Exibe informações detalhadas, incluindo permissões, tamanho e data de modificação dos arquivos.
2. `-a`: Mostra todos os arquivos, incluindo aqueles ocultos.
3. `-h`: Exibe tamanhos de arquivo legíveis por humanos (por exemplo, 1K, 234M, 2G).
4. `--s`: Ordena os arquivos por data de modificação.

cat: O utilitário cat será desenvolvido para unir e exibir o conteúdo de arquivos.

1. `-n`: Numera todas as linhas de saída.
2. `-b`: Numera somente as linhas não vazias de saída.
3. `-s`: Suprime repetições de linhas vazias.
4. `-E`: Exibe um caractere $ ao final de cada linha.

rm: O utilitário rm será desenvolvido para remover arquivos e diretórios do sistema de arquivos.

1. `-f`: Força a remoção dos arquivos sem confirmação.
2. `-r`: Remove diretórios e seus conteúdos recursivamente.
3. `-v`: Exibe informações verbais sobre as operações executadas.
4. `-i`: Pede confirmação antes de remover cada arquivo.

cp: O utilitário cp será desenvolvido para copiar arquivos e diretórios para um novo local no sistema de arquivos.

1. `-r`: Copia diretórios recursivamente.
2. `-p`: Preserva os atributos originais, incluindo timestamps e permissões.
3. `-f`: Sobrescreve o destino sem confirmação.
4. `-v`: Exibe informações verbais sobre as operações executadas.

mv: O utilitário mv será desenvolvido para mover arquivos e diretórios para um novo local no sistema de arquivos.

1. `-i`: Pede confirmação antes de sobrescrever o destino.
2. `-n`: Não sobrescreve o destino se o arquivo existir.
3. `-v`: Exibe informações verbais sobre as operações executadas.
4. `--backup`: Faz backup dos arquivos sobrescritos.

grep: O utilitário grep será desenvolvido para pesquisar por padrões em arquivos de texto.

1. `-i`: Realiza uma pesquisa insensível a maiúsculas e minúsculas.
2. `-v`: Exibe linhas que não correspondem ao padrão especificado.
3. `-r`: Pesquisa recursivamente em diretórios.
4. `-n`: Exibe números de linha juntamente com as correspondências.

tail: O utilitário tail será desenvolvido para exibir as últimas linhas de um arquivo de texto.

1. `-n <num>`: Exibe as últimas `<num>` linhas do arquivo.
2. `-f`: Acompanha as adições ao arquivo em tempo real.
3. `-q`: Não exibe cabeçalhos de arquivo ao usar `-f`.
4. `-r`: Exibe linhas em ordem inversa.

head: O utilitário head será desenvolvido para exibir as primeiras linhas de um arquivo de texto.

1. `-n <num>`: Exibe as primeiras `<num>` linhas do arquivo.
2. `-q`: Não exibe cabeçalhos de arquivo ao exibir mais de um arquivo.
3. `-c <num>`: Exibe os primeiros `<num>` bytes do arquivo.
4. `-v`: Sempre exibe nomes de arquivos ao exibir mais de um arquivo.

Enquadramento nas áreas da Unidade Curricular

Este projeto está alinhado com os objetivos da unidade curricular de Sistemas Operativos, onde teremos a oportunidade de desenvolver um entendimento prático dos conceitos fundamentais de sistemas operativos e aprofundar nosso conhecimento através da implementação de utilitários básicos.

Requisitos Técnicos para desenvolvimento do projeto

- Conhecimento básico em Linguagem C.
- Compreensão dos conceitos fundamentais de sistemas operativos, como manipulação de arquivos, gerenciamento de processos e comunicação entre processos.
- Ambiente de desenvolvimento Linux para compilar e testar os utilitários.

Arquitetura de Solução

Implementação Individual:

- Desenvolver cada utilitário no seu próprio arquivo de código-fonte (Linguagem C);
- Cada arquivo contém a implementação completa do utilitário, incluindo lógica e manipulação de entrada/saída;

Compilação:

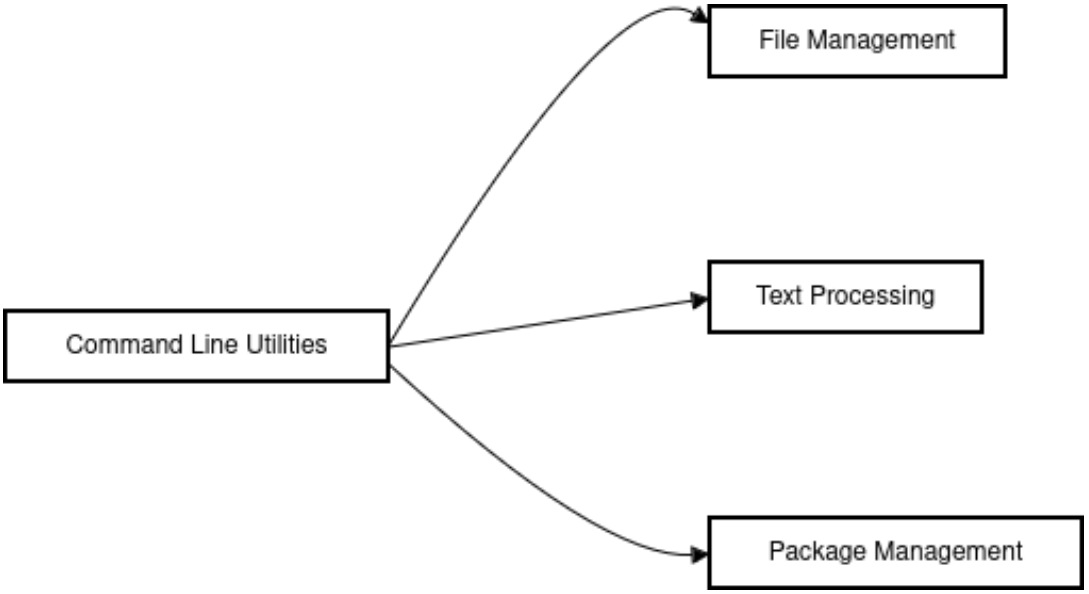
- Utilização de uma ferramenta de compilação, como GCC, para compilar cada arquivo de código-fonte separadamente;
- Resultará na geração de um executável independente para cada utilitário;

Execução na Linha de Comando:

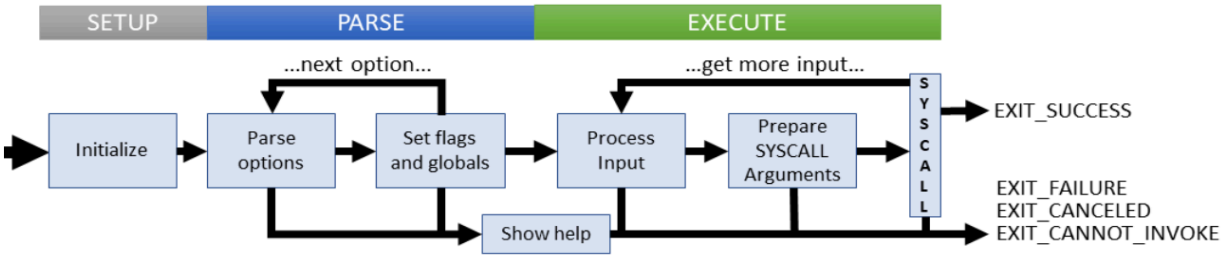
- Após a compilação, os utilitários podem ser executados diretamente na linha de comando do Linux.

Testes:

- Implementação de testes para cada utilitário;
- Estes testes permitem garantir o funcionamento correto do utilitário;



Usaremos como guia a arquitetura disponibilizada no site da GNU. De acordo com os mesmos, esta arquitetura é utilizada para a maioria dos utilitários.



Tecnologias a utilizar

Linguagem de Programação - Linguagem C para o desenvolvimento dos utilitários visto a eficiência e capacidade de interagir diretamente com o sistema operacional.

Editor de Texto - Visual Studio Code

Compilação - GCC (GNU Compiler Collection)

Controle de versão do código fonte - GitHub

Planeamento

Resultados

O projeto atingiu os seus objetivos principais de manipulação e gestão de arquivos utilizando os comandos criados para Linux como proposto. Foram realizadas com sucesso operações de listagem, visualização, remoção, pesquisa, monitoramento, cópia e movimentação de arquivos.

Pontos Fortes e Fracos

Pontos Fortes:

- Simplicidade e eficiência na utilização dos comandos;
- Flexibilidade proporcionada pelos utilizadores de GNU;
- Robustez e confiabilidade do sistema Linux.

Pontos Fracos:

- Necessidade de conhecimento prévio dos comandos e suas opções para saber qual deve utilizar para o fim pretendido.

Exemplo de Uso

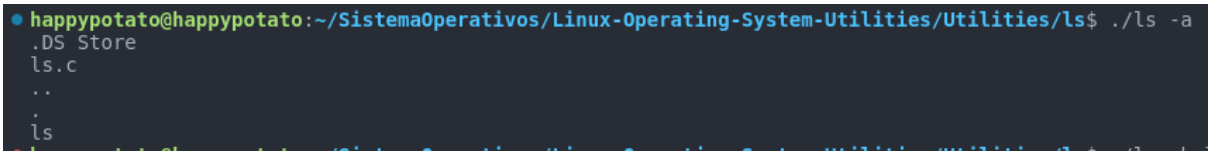
Ls

Input:

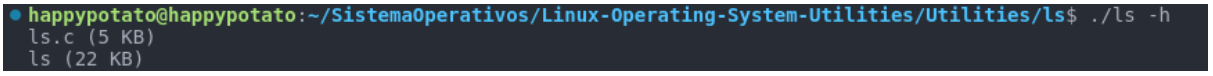
Parâmetro	Descrição	Exemplo de Uso
`-a`	Mostra arquivos ocultos	`./ls -a`
`-h`	Mostra os tamanhos dos arquivos em formato legível (human-readable)	`./ls -h`
`-l`	Mostra informações detalhadas dos arquivos (tipo, dono, grupo, tamanho, data de modificação, nome)	`./ls -l`
`--s`	Ordena os arquivos pelo tamanho	`./ls --s`
`--help`	Exibe a mensagem de ajuda com a descrição das opções disponíveis	`./ls --help`

Output:

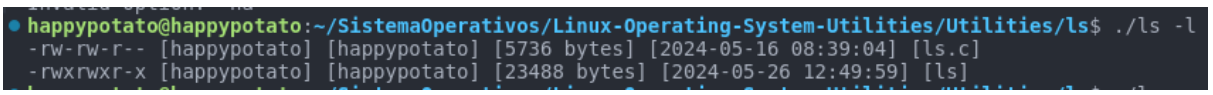
-a:



-h:



-l:



–s:

```
happypotato@happypotato:~/SistemaOperativos/Linux-Operating-System-Utilities/Utilities/ls$ ./ls --s
ls.c                5736 bytes
ls                  23488 bytes
```

–help:

```
happypotato@happypotato:~/SistemaOperativos/Linux-Operating-System-Utilities/Utilities/ls$ ./ls --help

Options:
  -a          : Show hidden files
  -h          : Show file sizes in human-readable format
  -l          : Show detailed file information(Type, Owner, Group, Size, Modification time, Name)
  --s         : Sort files by size
  --help      : Display this help message
```

Tail

Input:

Parâmetro	Descrição	Exemplo de Uso
`-n <num> <file>`	Especifica o número de linhas a serem exibidas para cada arquivo	`./tail -n 10 arquivo1`
`-f <file>...`	Exibe continuamente as últimas linhas do arquivo enquanto ele é modificado	`./tail -f arquivo1`
`-q <file>...`	Nunca exibe cabeçalhos de arquivos	`./tail -q arquivo1`
`-r <file>...`	Exibe o conteúdo dos arquivos em ordem reversa	`./tail -r arquivo1`
`--help`	Exibe uma mensagem de ajuda detalhando os parâmetros disponíveis	`./tail --help`

Output:

-n:

```
gustavofarinha@MBP-de-Gustavo tail % ./tail -n 1 helloworld.txt
HELLO WORLD
```

-f:

```
gustavofarinha@MBP-de-Gustavo tail % ./tail -f helloworld.txt
==========
Hello World
hello world
HELLO WORLD
==========
Hello World
hello world
HELLO WORLD
A
```

-q:

```
gustavofarinha@MBP-de-Gustavo tail % ./tail -q helloworld.txt
Hello World
hello world
HELLO WORLD
```

-r:

```
gustavofarinha@MBP-de-Gustavo tail % ./tail -r helloworld.txt
HELLO WORLD
hello world
Hello World
```

help:

```
gustavofarinha@MBP-de-Gustavo tail % ./tail --help

Options:

  <file>...             : Output the last 5 lines
  -n <num> <file>...    : Output the last <num> lines
  -f <file>...          : Output file while modified
  -q <file>...          : Never output headers file
  -r <file>...          : Output file in reverse
  --help                : Display this help message
```

Bibliografia

https://chat.openai.com/

https://www.canva.com/

https://pt.wikipedia.org/wiki/GNU_Core_Utilities

https://www.gnu.org/software/coreutils/

Arquitetura GNU: https://www.maizure.org/projects/decoded-gnu-coreutils/