Arquivador VINAc

Pietro Comin
GRR20241955
pietro.comin@ufpr.br

11 de maio de 2025

1 Descrição do Projeto

O VINAc é um arquivador com suporte a compressão, que permite armazenar múltiplos arquivos (membros) dentro de um único arquivo (archive) com extensão .vc. O programa oferece operações para inserção (com e sem compressão), movimentação, extração, remoção e listagem de membros.

2 Estrutura do Projeto

O projeto está organizado nos seguintes arquivos e diretórios:

2.1 Diretório `include/`

- `lz/`
  - `lz.h` - Cabeçalho para funções de compressão LZ
- `aux.h` - Funções auxiliares
- `options.h` - Manipulação de opções de linha de comando
- `types.h` - Definições de tipos de dados
- `utils.h` - Utilitários diversos
- `vina.h` - Cabeçalho principal

2.2 Diretório `src/`

- `lz/`
  - `lz.c` - Implementação da compressão LZ
- `aux.c` - Implementação de funções auxiliares

- `main.c` - Função principal
- `options.c` - Implementação das opções
- `utils.c` - Implementação dos utilitários
- `vina.c` - Implementação principal

2.3 Outros Arquivos

- `Makefile` - Script de compilação
- `A1 - O Arquivador VINAc.pdf` - Especificação do projeto
- Arquivos de teste: `texto.txt`, `texto2.txt`, `texto3.txt`, `texto4.txt`

3 Algoritmos e Estruturas de Dados

3.1 Estruturas Principais

- `struct arquivo`: Armazena metadados de cada membro
  - Nome, UID, tamanho original, tamanho comprimido
  - Ordem no archive, offset e data de modificação
- `struct diretorio`: Gerencia a coleção de membros
  - Quantidade de membros e vetor de ponteiros para `struct arquivo`

3.2 Principais Algoritmos

- **Manipulação de Arquivos**: Funções como `move()` e `move_sequencial()` permitem reorganizar eficientemente os membros no archive, mesmo com variações de tamanho e posição.
- **Compressão LZ**: Implementada conforme a biblioteca fornecida, com fallback para armazenamento não-comprimido quando a compressão não for eficiente.
- **Gerenciamento de Memória**: Alocação dinâmica cuidadosa para evitar vazamentos, com funções dedicadas para criação e destruição de estruturas.

4 Decisões de Implementação

- **Modularização**: O código foi dividido em funções pequenas e especializadas para facilitar depuração e manutenção.
- **Manipulação Direta em Disco**: Para evitar uso excessivo de memória, os dados dos membros são manipulados diretamente no arquivo.
- **Atualização de Metadados**: Funções como `atualiza_metadados()` garantem a consistência das informações após operações.

5 Dificuldades e Soluções

- **Manipulação de Arquivos Variáveis**: A principal dificuldade foi lidar com membros de tamanhos diferentes e suas posições no archive. As funções `move()` e `move_sequencial()` resolvem este problema movendo blocos de dados de forma segura.
- **Compressão Ineficiente**: Quando a compressão resulta em arquivo maior que o original, o programa automaticamente armazena os dados descomprimidos.
- **Valgrind Warnings**: O erro reportado pelo Valgrind (uninitialized bytes) foi considerado de baixo impacto, pois ocorre em operações de buffer interno da biblioteca padrão.

6 Bugs Conhecidos

- O Valgrind reporta acesso a bytes não inicializados durante operações de escrita, mas este comportamento parece ser inócuo e relacionado ao buffer interno da biblioteca padrão.

7 Compilação e Uso

7.1 Compilação

Execute `make` para compilar o programa, que será gerado em `login/vinac`.

7.2 Exemplos de Uso

```
./login/vinac -ip teste.vc texto.txt texto2.txt texto3.txt texto4.txt
./login/vinac -m teste.vc texto.txt texto2.txt
./login/vinac -ic teste.vc texto2.txt
./login/vinac -x teste.vc
```

8 Makefile

O Makefile fornecido:

- Compila cada módulo separadamente
- Gera o executável em `login/vinac`
- Suporta `make clean` para remover arquivos temporários
- Inclui dependências entre arquivos