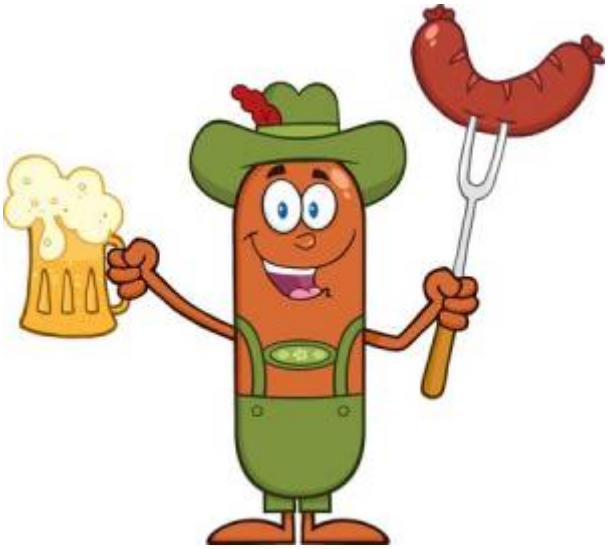
# O Arquivador VINA++

(contribuição do prof. Daniel Weingaertner)



O objetivo deste trabalho é implementar o programa `vina++`[1], que consiste de um arquivador básico (*archiver*), isto é, um programa que salva em sequência uma coleção de arquivos (denominados membros) dentro de outro arquivo (denominado *archive*) cuja estrutura permite recuperar os arquivos originais individualmente.

Os programas `tar`, `zip`, `rar` e `arj` são arquivadores populares. Além de arquivar arquivos, a maioria deles também comprime os dados dos mesmos, para ocupar menos espaço de armazenamento. Para simplificar, o arquivador VINA++ não comprime os arquivos, apenas os armazena.

## Execução do Programa

O pacote de software a ser construído deve gerar o executável chamado `vina++`, que deve ser executado da seguinte forma:

```
vina++ <opção> <archive> [membro1 membro2 ...]
```

Onde a opção pode ser:

- `-i` : insere/acrescenta um ou mais membros ao *archive*. Caso o membro já exista no *archive*, ele deve ser substituído. Novos membros são inseridos respeitando a ordem da linha de comando, ao final do *archive*;
- `-a` : mesmo comportamento da opção `-i`, mas a substituição de um membro existente ocorre APENAS caso o parâmetro seja **mais recente** que o arquivado;
- `-m target` : move o membro indicado na linha de comando para imediatamente depois do membro *target* existente em *archive*. A movimentação deve ocorrer na seção de dados do *archive*;
- `-x` : extrai os membros indicados de *archive*. Se os membros não forem indicados, **todos** devem ser extraídos. A extração consiste em ler o membro de *archive* e criar um arquivo correspondente, com conteúdo idêntico, em disco;
- `-r` : remove os membros indicados de *archive*;
- `-c` : lista o conteúdo de *archive* em ordem, incluindo as propriedades de cada membro (nome, UID, permissões, tamanho e data de modificação) e sua ordem no arquivo.
- `-h` : imprime uma pequena mensagem de ajuda com as opções disponíveis e encerra.

Caso sejam indicados nomes de arquivos com caminhos absolutos ou relativos, deve-se arquivar estes nomes SEMPRE com caminhos relativos. Por exemplo, se for indicado o arquivo `/home/inf/xyz00/texto.doc`, deve ser colocado no arquivo `vpp` como `./home/inf/xyz00/texto.doc`.

Ao extrair um membro, toda a hierarquia de diretórios contidos no nome do arquivo deve ser criada (caso não existam).

Para manipulação de diretórios veja o material a respeito do assunto aqui.

Exemplos de comandos:

```
// inclui os arquivos "arq.txt", "foto.jpg" e "despesas.ods" em backup.vpp
vina++ -i backup.vpp arq.txt foto.jpg despesas.ods

// atualiza "despesas.ods" em backup.vpp, se o arquivo externo for mais recente
vina++ -a backup.vpp despesas.ods

// extrai o arquivo arq.txt de backup.vpp
vina++ -x backup.vpp arq.txt

// extrai todos os arquivos de backup.vpp
vina++ -x backup.vpp

// move o arquivo "arq.txt" para depois de "despesas.ods" em backup.vpp
vina++ -m despesas.ods backup.vpp arq.txt

// inclui os arquivos "xy/dir/arq.txt", "/dir/foto.jpg" em novo_backup.vpp como
"./xy/dir/arq.txt", "./dir/foto.jpg"
vina++ -i novo_backup.vpp xy/dir/arq.txt, /dir/foto.jpg

// extrai todos os arquivos de novo_backup.vpp, criando (se não existir)
// toda a hierarquia de diretórios indicada para o arquivo.
// Por exemplo se não existir "./xy/dir", deve ser criado o diretório ''xy''
// e abaixo dele o diretório ''dir'' e extrair o arquivo ''arq.txt'' para o
diretório ''./xy/dir''
vina++ -x novo_backup.vpp
```

## Formato do Arquivo

Cada aluno é livre para inventar seu próprio formato para o *archive*, desde que respeitando as seguintes restrições:

- O início ou final do *archive* (`.vpp`) deve ter uma área de diretório, como mostra a figura:


- Todas as informações sobre os membros, necessárias para a manipulação do *archive*, devem estar armazenadas única e exclusivamente na área de diretório. A parte restante deve conter **apenas** os

dados dos membros.
- Para cada membro, devem ser armazenadas as seguintes informações: nome (sem espaços), UID (*user ID*), permissões, tamanho, data de modificação, ordem no arquivo e localização.
- A ordem de um membro no arquivo é dada pela ordem de inserção, e pode ser alterada pela opção `-m`.
- O conteúdo da área de diretório pode ser manipulado em memória RAM ou em disco, a critério do aluno.
- O conteúdo dos membros do *archive* **deve ser manipulado diretamente em disco**, não sendo permitida a alocação de mais de 1.024 bytes de memória para manipulação dos conteúdos dos membros.

Uma abordagem interessante consiste em colocar a área de diretório no **final** do *archive*, mantendo no início do deste apenas um inteiro indicando a posição do *archive* onde inicia essa área. Essa organização torna mais simples as operações de alteração do conteúdo do arquivo.

As operações de leitura e escrita nos arquivos devem **sempre** ser feitas em formato **binário**, ou seja, usando as funções `fread()` e `fwrite()`.

Se estiver usando as funções `fprintf()`, `fscanf()`, `fputc()`, `fgetc()` e similares, **provavelmente está fazendo errado**!

## Erros

Em caso de erros, uma mensagem explicando o ocorrido deve ser impressa em `stderr` e a execução do programa deve ser encerrada com código de saída **diferente de 0**. Caso o programa possa recuperar-se automaticamente do erro, deve fazê-lo.

## Produto a ser Entregue

Deve-se entregar um pacote de software completo contendo os fontes em linguagem C. O pacote deve ser arquivado e compactado com `tar` e `gzip` em um arquivo chamado `login.tar.gz` (substitua `login` por seu login/identificador na rede do departamento).

O pacote deve ter a seguinte estrutura de diretório e arquivos:

```
login/           diretório principal
     /src/...    diretório dos arquivos fonte (*.c, *.h)
     /LEIAME     arquivo
     Makefile    arquivo
```

Note que a extração dos arquivos de `login.tar.gz` deve criar o diretório `login/` contendo todos os arquivos e diretórios acima. Os arquivos fonte também devem estar contidos no sub-diretório `src`.

O arquivo de texto `LEIAME` deve conter as seguintes informações:

- autoria do software (GRR e nome(s) do(s) autor(es));
- lista dos arquivos e diretórios contidos no pacote e sua descrição (breve);
- uma seção descrevendo os algoritmos e as estruturas de dados utilizadas, as alternativas de implementação consideradas e/ou experimentadas e os motivos que o levaram a optar pela versão entregue, as dificuldades encontradas e as maneiras pelas quais foram contornadas.
- bugs conhecidos;

O arquivo `Makefile` deve possuir as regras necessárias para compilar os módulos individualmente e gerar o programa executável. As seguintes regras devem estar presentes:

- `all`: compila e produz um executável chamado **vina++** no diretório `login/`;
- `clean`: remove todos os arquivos temporários e os arquivos gerados pelo Makefile (*.o, executável, etc.).

## Avaliação

Os itens de avaliação do trabalho e respectivas pontuações são:

- Qualidade da documentação: arquivo `LEIAME` (15 pontos)
- Funcionamento: corretude das respostas nos testes executados (40 pontos)
- Eficiência: algoritmos e estruturas de dados utilizados para obter um melhor desempenho, desde que devidamente justificados no arquivo `LEIAME` (45 pontos)

[1)]
https://pt.wiktionary.org/wiki/vina