UNIVERSIDADE ESTADUAL DE CAMPINAS
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Estrutura de Arquivos - Arquivador

Amigos de Baunilha
Alfredo Albélis - 193532
Cléofas Santos - 195741
Vinícius Pereira - 206681

Limeira - 2018

VISÃO GERAL

Projeto desenvolvido para disciplina ST562 – Estrutura de Arquivos. O trabalho consiste na implementação de um arquivador (programa gerenciador de archives). A solução foi desenvolvida usando a linguagem de programação C para ser executado via prompt do Windows.

Nas próximas seções serão apresentado os objetivos do trabalho e suas especificações, bem como os casos de uso implementados e instruções para executar o arquivador.

ESPECIFICAÇÃO

Um *archive* é um arquivo que contém um conjunto de arquivos, em uma estrutura que permite recuperar cada arquivo separadamente. O arquivador é um programa que oferece as funcionalidades básicas para gerenciamento de archives, possibilitando a criação de archives e para estes permite inserir, listar, extrair e remover arquivos.

SOLUÇÃO

A solução proposta pelo grupo visa economizar acessos a disco (seeks). Partindo desta premissa, foram implementadas as seguintes funções:

criaArchive

Esta funcionalidade recebe o nome do archive que será criado e um conjunto de arquivos que o archive deve conter. Para cada arquivo na lista, é chamada a função insereArquivo.

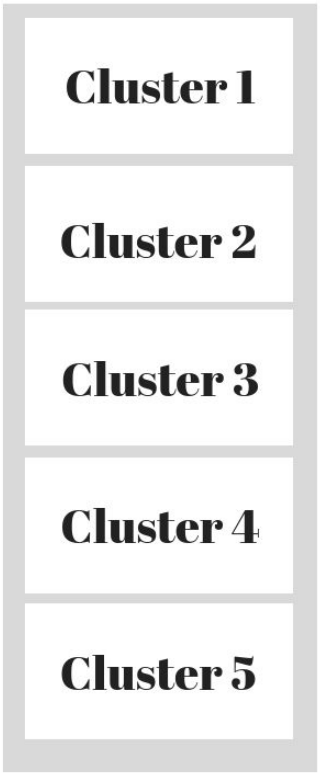
O archive criado conterá somente os arquivos válidos, ou seja, se um dado arquivo não existir, o usuário é informado que este não pôde ser inserido e a execução prossegue na tentativa de inserir o próximo arquivo da lista.

insereArquivo

Para diminuir o número de seeks, os arquivos são divididos em blocos de 4KB podendo ocorrer fragmentação interna, pois a inserção de um novo arquivo começa sempre no início de um cluster livre.

A política escolhida para a inserção foi a de primeiro encaixe (First Fit), ou seja, ao inserir um novo arquivo, percorre-se o archive até encontrar o primeiro espaço que caiba o novo arquivo, diminuindo a fragmentação interna.

Imagine o seguinte cenário: temos o archive vazio e três arquivos a serem incluídos.



Após incluir os arquivos no Archive:

*Note que não estamos aproveitando o restante do Cluster 4, gerando a fragmentação interna.

Agora, suponha que o arquivo 2 tenha sido excluído, aumentando a fragmentação interna.

O próximo arquivo a ser inserido seguirá a política de First Fit, ocupando o espaço antes alocado ao arquivo 2 caso o espaço disponível seja suficiente, diminuindo a fragmentação interna.

O mesmo ocorre com os demais arquivos a serem incluídos no archive.

listaArquivos

Esta funcionalidade recebe o nome de um archive e caso ele exista, o percorre informando ao usuário o nome dos arquivos que existem em seu interior.

extraiArquivo

Esta funcionalidade recebe o nome de um archive e caso ele exista, o percorre verificando se existe em seu interior um arquivo com o nome informado. Caso exista, a função tenta fazer uma cópia deste arquivo, concluindo com sucesso caso não exista um arquivo com o mesmo nome no diretório atual.

removeArquivo

Esta funcionalidade recebe o nome de um archive e caso ele exista, o percorre verificando se existe em seu interior um arquivo com o nome informado. Caso exista, todos os seus clusters são marcados como livre e serão reutilizados nas próximas inserções, seguindo a política de First Fit.

A solução proposta segue a ideia inicial de economizar seeks, evitando o custoso trabalho de criar arquivos intermediários e reescrever todo o conteúdo do archive e optando por desperdiçar temporariamente espaço de armazenamento que será reutilizado futuramente, propondo assim uma solução mais eficiente.

Funções auxiliares

Foram criadas ainda duas funções auxiliares chamadas getTamanhoArquivo e procuraArquivo, que servem respectivamente para informar o tamanho de um dado arquivo e verificar se o archive já contém o arquivo informado, garantindo que cada arquivo seja inserido uma única vez no archive.

ESTRUTURA DO ARQUIVO

XXARQUIVO.TXT|YYYYCONTEÚDO

XX - 2 bytes indicando espaco disponivel no cluster

ARQUIVO.TXT- Nome do arquivo no archive

| - Separador

YYYY - 4 bytes indicando o tamanho do arquivo

CONTEÚDO - Arquivo propriamente dito

Caso de uso 1: Criar archive

Comando: -c <novoarchive.arq> <arq1> <arq2> ...

Este caso de uso é responsável por criar um archive com os arquivos informados no comando. Após a execução desse caso de uso o Archive criado fica localizado na pasta do projeto.

Execução:

```
C:\Users\Alfredo Albélis\Desktop\TrabalhoSt562>executavel.exe -c trabalho.arq balão.jpg neve.jpg video.mp4
balÒo.jpg inserido com sucesso!
neve.jpg inserido com sucesso!
video.mp4 inserido com sucesso!
```

Código:

```
int criaArchive(int argc, char *argv[]) {
    char nomeArchive[220]; //verificar tamanho máximo de caracteres em um arquivo
    strcpy(nomeArchive, argv[2]);

    //insere os arquivos informados dentro do archive
    for(int i = 3; i < argc; i++) {
        int retorno = insereArquivo(nomeArchive, argv[i]);
        if(retorno == 0) {
            printf("\n%s inserido com sucesso!\n", argv[i]);
        }
        else if(retorno == 1) {
            printf("Erro ao abrir %s!\n", nomeArchive);
            return 1;
        }
        else if(retorno == 2) printf("\nErro ao abrir %s\nVerifique se o arquivo existe e tente inseri-lo novamente!\n", argv[i]);
        else if(retorno == 3) printf("\n%s ja contem %s!\n", nomeArchive, argv[i]);
    }
    return 0;
}
```

Caso de uso 2: Inserir arquivo no archive

Comando: -i <archive.arq> <arq>

Este caso de uso é responsável por inserir um arquivo no archive. Caso um arquivo com o mesmo nome do novo arquivo já exista no archive, uma mensagem é apresentada. Caso não exista, o arquivo é inserido seguindo a política First Fit.

Execução:

```
C:\Users\Alfredo Albélis\Desktop\TrabalhoSt562>executavel.exe -i trabalho.arq balão.jpg
balÒo.jpg inserido com sucesso!
```

Código:

```c
int insereArquivo(char nomeArchive[], char nomeArquivo[]) {
    char c, stringNome[220];
    int tamanhoArchive = getTamanhoArquivo(nomeArchive), inicioArquivo, i, deslocamento;
    int tamanhoArquivo = getTamanhoArquivo(nomeArquivo);
    int posicaoAtual, posicaoRelativa;
    short int bytesDisponiveis, exit = 0, cont = 0;
    short int qtdBytesNome = strlen(nomeArquivo);
    FILE *archive;

    //abre o arquivo informado em modo leitura
    FILE *arquivo = fopen(nomeArquivo, "rb");
    if(arquivo == NULL) return 2;

    if(procuraArquivo(nomeArchive, nomeArquivo) == 0) return 3;

    if(tamanhoArchive == 0) {
        archive = fopen(nomeArchive, "wb");
        if(archive == NULL) return 1;
    }
    else {
        archive = fopen(nomeArchive, "r+b");
        if(archive == NULL) return 1;
    }

    //quantidade total de bytes que as informações do arquivo gastarão no archive
    int bytesNecessarios = qtdBytesNome + 1 + sizeof(int) + tamanhoArquivo;

    //quantidade de clusters que o arquivo ocupará no archive além do primeiro
    int clustersNecessarios = bytesNecessarios / (4096 - sizeof(short int));
```

```
//quantidade de bytes que ficarão no último cluster
int bytesExcedentes = bytesNecessarios - (4096 - sizeof(short int)) * clustersNecessarios;

//entra somente se o archive não estiver vazio e executa enquanto exit == 0 (false)
while(exit == 0 && tamanhoArchive != 0) {
    inicioArquivo = (int) ftell(archive); //guarda a posição do primeiro cluster de cada iteração
    fread(&bytesDisponiveis, sizeof(short int), 1, archive);
    cont = 0;

    //se o cluster atual estiver 'vazio' (significa que o arquivo que ocupava este espaço foi excluído)
    if(bytesDisponiveis == 4096 - sizeof(short int)) {

        //enquanto houver clusters vazios
        while(bytesDisponiveis == 4096 - sizeof(short int)) {
            //entrará no if caso o arquivo caiba no buraco
            if(cont == clustersNecessarios) {
                fseek(archive, inicioArquivo, SEEK_SET); //volta para o primeiro cluster livre
                exit++;
                break;
            }
            else {
                cont++;
                fseek(archive, 4096 - sizeof(short int), SEEK_CUR);
                fread(&bytesDisponiveis, sizeof(short int), 1, archive);
            }
        }
    }
    if(exit == 1) break;

    //caso o cluster esteja ocupado
    if((bytesDisponiveis != 4096 - sizeof(short int))) {

        //lê o nome do arquivo para ler seu tamanho e calcular o deslocamento
        i = 0;
        while((c = getc(archive)) != '|') {
            stringNome[i] = c;
            i++;
        }
        stringNome[i] = '\0';
        fread(&tamanhoArquivo, sizeof(int), 1, archive);
        posicaoAtual = (int) ftell(archive); //pega a posição atual do arquivo
        posicaoRelativa = posicaoAtual - ((posicaoAtual / 4096) * 4096); //calcula a posição relativa
        //deslocamento para o final do conteúdo do arquivo
        deslocamento = tamanhoArquivo + sizeof(short int) * (tamanhoArquivo / (4096 - sizeof(short int)));
        fseek(archive, deslocamento, SEEK_CUR);
        c = getc(archive);
    }
    //caso esteja no final do arquivo
    if(feof(archive)) {
        exit++;
    }
    posicaoAtual = (int) ftell(archive);
    posicaoRelativa = posicaoAtual - ((posicaoAtual / 4096) * 4096);
    //deslocamento para o próximo cluster
    fseek(archive, 4096 - posicaoRelativa, SEEK_CUR);
}
tamanhoArquivo = getTamanhoArquivo(nomeArquivo);
```

```
    //armazena a quantidade de bytes disponíveis no cluster atual
    bytesDisponiveis = 4096 - sizeof(short int); //cada cluster começa com 4096 menos o tamanho da variáve
    bytesDisponiveis -= (qtdBytesNome + 1); //desconta a quantidade de bytes do nome do arquivo mais o pip
    bytesDisponiveis -= sizeof(int); //desconta a quantidade de bytes da variável tamanhoArquivo (int)

    if(clustersNecessarios == 0) { //arquivo cabe em um único cluster
        bytesDisponiveis -= tamanhoArquivo; //desconta a quantidade de bytes do conteúdo do arquivo

        //gravando dados no primeiro cluster
        fwrite(&bytesDisponiveis, sizeof(short int), 1, archive);
        fprintf(archive, "%s", nomeArquivo);
        fprintf(archive, "|");
        fwrite(&tamanhoArquivo, sizeof(int), 1, archive);

        while((c = getc (arquivo)) != EOF)
            putc(c, archive);
    }
    else { //arquivo utiliza mais que um cluster
        short int i, j, zero = 0;
        for(i = 0; i < clustersNecessarios; i++) { //grava o cluster completo
            fwrite(&zero, sizeof(short int), 1, archive); //grava bytesDisponiveis = 0 (cluster completo)
            if(i == 0) { //somente na primeira iteração
                fprintf(archive, "%s", nomeArquivo);
                fprintf(archive, "|");
                fwrite(&tamanhoArquivo, sizeof(int), 1, archive);
                for(j = 0; j < bytesDisponiveis; j++) {
                    c = getc (arquivo);
                    putc(c, archive);
                }
            }

            else { //grava o restante dos clusters completos
                for(j = 0; j < 4096 - sizeof(short int); j++) {
                    c = getc (arquivo);
                    putc(c, archive);
                }
            }
        }
        //grava o excedente
        zero = 4096 - sizeof(short int) - bytesExcedentes;
        fwrite(&zero, sizeof(short int), 1, archive);
        for(j = 0; j < bytesExcedentes; j++) {
            c = getc (arquivo);
            putc(c, archive);
        }
    }
    fclose(arquivo);
    fclose(archive);
    return 0;
}
```

Caso de uso 3: Listar arquivos de um archive

Comando: -l <archive.arq>

Este caso de uso é responsável por listar todos os arquivos presentes no Archive.

Execução:

```
C:\Users\Alfredo Albélis\Desktop\TrabalhoSt562>executavel.exe -l trabalho.arq
balÒo.jpg
neve.jpg
video.mp4
```

Código:

```c
int listaArquivos(char nomeArchive[]) {
    short int bytesDisponiveis, cont = 0;
    int tamanhoArquivo, deslocamento, posicaoRelativa, posicaoAtual, i;
    char stringNome[220];
    char c;
    FILE *archive = fopen(nomeArchive, "rb");
    if(archive == NULL) return 1;

    while(!feof(archive)) {
        fread(&bytesDisponiveis, sizeof (short int), 1, archive);

        //se o cluster atual estiver 'vazio' (significa que o arquivo que ocupava este espaço foi excluído)
        while(bytesDisponiveis == 4096 - sizeof(short int)) {
            if(feof(archive)) {
               if(cont == 0) return 2;
               return 0;
            }
            fseek(archive, 4096 - sizeof(short int), SEEK_CUR);
            fread(&bytesDisponiveis, sizeof (short int), 1, archive);
        }

        i = 0;
        while((c = getc(archive)) != '|') {
            stringNome[i] = c;
            i++;
        }
        stringNome[i] = '\0';
        printf("%s\n", stringNome);
        cont++; //incrementa a quantidade de arquivos printados
        fread(&tamanhoArquivo, sizeof(int), 1, archive);

        deslocamento = tamanhoArquivo + sizeof(short int) * (tamanhoArquivo / (4096 - sizeof(short int)));
        fseek(archive, deslocamento, SEEK_CUR);
```

```
        c = getc(archive);
        if(feof(archive)) {
            if(cont == 0) return 2;
            return 0;
        }
        else {
            posicaoAtual = (int) ftell(archive); //pega a posição atual do arquivo
            posicaoRelativa = posicaoAtual - ((posicaoAtual / 4096) * 4096); //calcula a posição relativa
            fseek(archive, 4096 - posicaoRelativa, SEEK_CUR);
        }
    }
    return 0;
}
```

Caso de uso 4: Extrair arquivo de um archive

Comando: -e <archive.arq> <arq>

Execução:

```
C:\Users\Alfredo Albélis\Desktop\TrabalhoSt562>executavel.exe -e trabalho.arq neve.jpg

Arquivo extraido com sucesso!
```

Código:

```
int extraiArquivo(char nomeArchive[], char nomeArquivo[]) {
    short int bytesDisponiveis;
    int tamanhoArquivo, deslocamento, posicaoRelativa, posicaoAtual, i;
    char stringNome[220];
    char c;
    FILE *archive = fopen(nomeArchive, "rb");

    //tenta abrir em modo leitura para verificar se já existe um arquivo com o nome informado
    FILE *file = fopen(nomeArquivo, "rb");
    if(file != NULL) {
        fclose(file);
        return 1;
    }

    while(!feof(archive)) {
        fread(&bytesDisponiveis, sizeof(short int), 1, archive);

        //se o cluster atual estiver 'vazio' (significa que o arquivo que ocupava este espaço foi excluído)
        while(bytesDisponiveis == 4096 - sizeof(short int)) {
            if(feof(archive)) return 2;
            fseek(archive, 4096 - sizeof(short int), SEEK_CUR);
            fread(&bytesDisponiveis, sizeof (short int), 1, archive);
        }

        i = 0;
        while((c = getc(archive)) != '|') {
            stringNome[i] = c;
            i++;
        }
        stringNome[i] = '\0';
        fread(&tamanhoArquivo, sizeof(int), 1, archive);
```

```
        posicaoAtual = (int) ftell(archive); //pega a posição atual do arquivo
        posicaoRelativa = posicaoAtual - ((posicaoAtual / 4096) * 4096); //calcula a posição relativa

        if(strcmp(stringNome, nomeArquivo) == 0) {
            //cria o arquivo que irá receber o conteúdo
            FILE *novoArquivo = fopen(stringNome, "wb");

            if(tamanhoArquivo <= (4096 - posicaoRelativa)) { //arquivo ocupa somente o cluster atual
                for(i = 0; i < tamanhoArquivo; i++) {
                    c = getc(archive);
                    putc(c, novoArquivo);
                }
            }
            else { //arquivo ocupa mais que o cluster atual
                for(i = 0; i < 4096 - posicaoRelativa; i++) { //copia o pedaço do primeiro cluster que o arquivo ocupa
                    c = getc(archive);
                    putc(c, novoArquivo);
                }
                fread(&bytesDisponiveis, sizeof(short int), 1, archive);
                while(bytesDisponiveis == 0) { //copia todos os clusters completos que o arquivo ocupa
                    for(i = 0; i < 4096 - sizeof(short int); i++) {
                        c = getc(archive);
                        putc(c, novoArquivo);
                    }
                    fread(&bytesDisponiveis, sizeof(short int), 1, archive);
                }
                for(i = 0; i < (4096 - sizeof(short int) - bytesDisponiveis); i++) {
                    c = getc(archive);
                    putc(c, novoArquivo);
                }
            }

            fclose(archive);
            fclose(novoArquivo);
            return 0;
        }
        else {
            deslocamento = tamanhoArquivo + sizeof(short int) * (tamanhoArquivo / (4096 - sizeof(short int)));
            fseek(archive, deslocamento, SEEK_CUR);
            c = getc(archive);
            if(feof(archive))
                return 2;
            else {
                posicaoAtual = (int) ftell(archive); //pega a posição atual do arquivo
                posicaoRelativa = posicaoAtual - ((posicaoAtual / 4096) * 4096); //calcula a posição relativa
                fseek(archive, 4096 - posicaoRelativa, SEEK_CUR);
            }
        }
    }
    return 2;
}
```

Caso de uso 5: Remover arquivo de um archive

Comando: -r <archive.arq> <arq>

Execução:

```
C:\Users\Alfredo Albélis\Desktop\TrabalhoSt562>executavel.exe -r trabalho.arq neve.jpg

Arquivo removido com sucesso!
```

Código:

```c
int removeArquivo(char nomeArchive[], char nomeArquivo[]) {
    short int bytesDisponiveis, max = 4096 - sizeof(short int);
    int tamanhoArquivo, deslocamento, posicaoRelativa, posicaoAtual, inicioArquivo;
    char stringNome[220];
    char c;

    FILE *archive = fopen(nomeArchive, "r+b");

    while(!feof(archive)) {
        inicioArquivo = (int) ftell(archive); //guarda a posição do primeiro cluster de cada iteração
        fread(&bytesDisponiveis, sizeof(short int), 1, archive);

        //se o cluster atual estiver 'vazio' (significa que o arquivo que ocupava este espaço foi excluído)
        while(bytesDisponiveis == 4096 - sizeof(short int)) {
            if(feof(archive)) return 1;
            fseek(archive, 4096 - sizeof(short int), SEEK_CUR);
            fread(&bytesDisponiveis, sizeof (short int), 1, archive);
        }

        int i = 0;
        while((c = getc(archive)) != '|') {
            stringNome[i] = c;
            i++;
        }
        stringNome[i] = '\0';
        fread(&tamanhoArquivo, sizeof(int), 1, archive);

        //quantidade total de bytes que as informações do arquivo gastarão no archive
        int bytesNecessarios = strlen(nomeArquivo) + 1 + sizeof(int) + tamanhoArquivo;

        //quantidade de clusters que o arquivo ocupará no archive além do primeiro
        int clustersNecessarios = bytesNecessarios / (4096 - sizeof(short int));
```

```
        posicaoAtual = (int) ftell(archive); //pega a posição atual do arquivo
        posicaoRelativa = posicaoAtual - ((posicaoAtual / 4096) * 4096); //calcula a posição relativa

        if(strcmp(stringNome, nomeArquivo) == 0) {
            if(bytesDisponiveis == max) return 1; //caso arquivo existe mas esteja apagado
            fseek(archive, inicioArquivo, SEEK_SET);
            fwrite(&max, sizeof(short int), 1, archive);

            if(clustersNecessarios == 0) return 0;

            else { //arquivo ocupa mais que o cluster atual
                for(i = 0; i < clustersNecessarios; i++) {
                    fseek(archive, 4096 - sizeof(short int), SEEK_CUR);
                    fwrite(&max, sizeof(short int), 1, archive);
                }
                return 0;
            }
            fclose(archive);
            return 0;
        }

        else {
            deslocamento = tamanhoArquivo + sizeof(short int) * (tamanhoArquivo / (4096 - sizeof(short int)));
            fseek(archive, deslocamento, SEEK_CUR);
            c = getc(archive);
            if(feof(archive))
                return 1;
            else {
                posicaoAtual = (int) ftell(archive); //pega a posição atual do arquivo
                posicaoRelativa = posicaoAtual - ((posicaoAtual / 4096) * 4096); //calcula a posição relativa
                fseek(archive, 4096 - posicaoRelativa, SEEK_CUR);
            }
        }
    }
    return 1;
}
```

Código das funções auxiliares

```c
int getTamanhoArquivo(char nomeArquivo[]) {
    int tamanhoArquivo;

    FILE *arquivo = fopen(nomeArquivo, "rb");
    if(arquivo == NULL) return 0; //verifica se o arquivo não existe

    fseek(arquivo, 0, SEEK_END); //caminha até o final do arquivo
    tamanhoArquivo = ftell(arquivo); //pega a posição atual, que indica o tamanho do arquivo
    fclose(arquivo);
    return tamanhoArquivo;
}

int procuraArquivo(char nomeArchive[], char nomeArquivo[]) {
    short int bytesDisponiveis;
    int tamanhoArquivo, deslocamento, posicaoRelativa, posicaoAtual, i;
    char stringNome[220];
    char c;

    //verifica se o archive existe
    FILE *archive = fopen(nomeArchive, "rb");
    if(archive == NULL) return 1;
    while(!feof(archive)) {
        fread(&bytesDisponiveis, sizeof (short int), 1, archive);

        //se o cluster atual estiver 'vazio' (significa que o arquivo que ocupava este espaço foi excluído)
        while(bytesDisponiveis == 4096 - sizeof(short int)) {
            if(feof(archive)) return 2;
            fseek(archive, 4096 - sizeof(short int), SEEK_CUR);
            fread(&bytesDisponiveis, sizeof (short int), 1, archive);
        }

        i = 0;
        while((c = getc(archive)) != '|') {
            stringNome[i] = c;
            i++;
        }
        stringNome[i] = '\0';
        if(strcmp(stringNome, nomeArquivo) == 0) return 0; //arquivo encontrado
        fread(&tamanhoArquivo, sizeof(int), 1, archive);

        deslocamento = tamanhoArquivo + sizeof(short int) * (tamanhoArquivo / (4096 - sizeof(short int)));
        fseek(archive, deslocamento, SEEK_CUR);
        c = getc(archive);
        if(feof(archive)) return 2;
        else {
            posicaoAtual = (int) ftell(archive); //pega a posição atual do arquivo
            posicaoRelativa = posicaoAtual - ((posicaoAtual / 4096) * 4096); //calcula a posição relativa
            fseek(archive, 4096 - posicaoRelativa, SEEK_CUR);
        }
    }
    return 2;
}
```

Clique aqui para acessar o código.