MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Alfenas . UNIFAL-MG
Rua Gabriel Monteiro da Silva, 700 . Alfenas/MG . CEP 37130-000
Fone: (35) 3299-1000 . Fax: (35) 3299-1063

Bacharelado em Ciência da Computação
Trabalho – Estrutura de Dados I
Prof. Luiz Eduardo da Silva

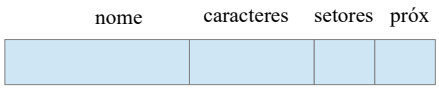
Objetivo:
Utilizar a estrutura de lista encadeada para simular um sistema de arquivo

Procedimento:
Leia com atenção o enunciado do problema abaixo, desenvolva um algoritmo e implemente o programa em C para resolver o problema descrito. Após a implementação, teste o programa para diferentes entradas de dados.

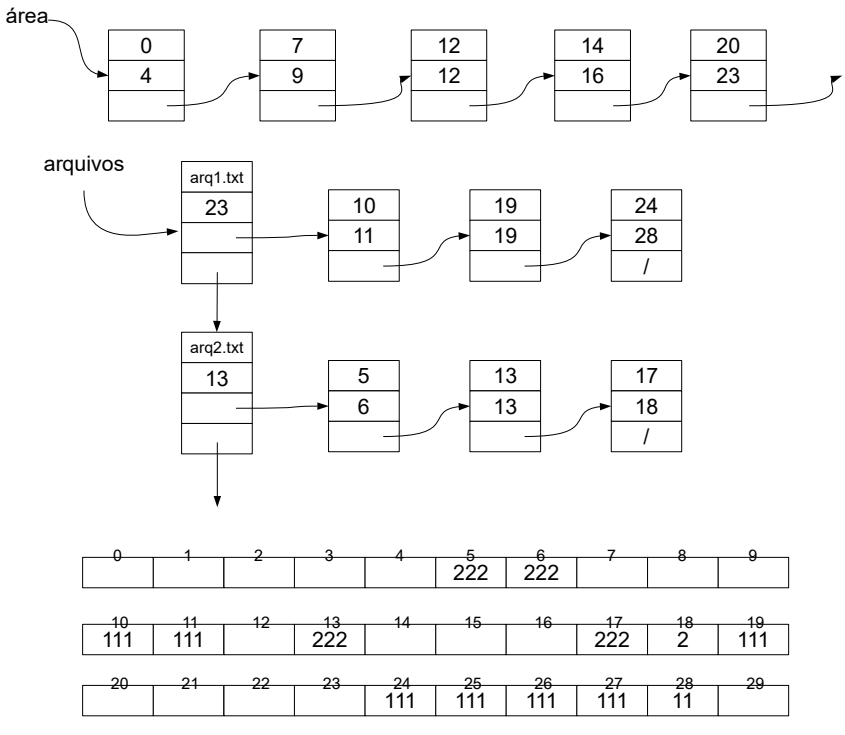
Problema:
Uma estratégia utilizada pelo Sistema Operacional para o gerenciamento dos arquivos em disco utiliza um estrutura baseada em lista encadeada chamada FAT (File Allocation Table). Neste trabalho você terá que simular o gerenciamento de arquivos realizada pelo Sistema Operacional. Para esta simulação, represente o disco como um vetor unidimensional de tamanho *TAM_MEMORIA * TAM_GRANULO*, onde *TAM_GRANULO* indica o número de bytes (caracteres) que podem ser armazenados em cada setor do disco. (Para simplificar a simulação e a depuração, considere o *TAM_GRANULO* um número pequeno, por exemplo, *TAM_GRANULO= 3*).

A **área** de setores disponíveis do disco deve ser mantida numa lista ligada ordenada de nós compostos de três campos: os campos *início* e *fim* para indicar o intervalo de setores disponíveis e um campo *próximo* para promover o encadeamento dos nós da lista ligada. Assim:

Os **arquivos** devem ser mantidos numa lista ligada ordenada de nós compostos de quatro campos: o *nome* do arquivo, o número de caracteres do arquivo, um ponteiro para uma lista ligada de *setores* ocupados pelo arquivo, e o campo *próximo* que aponta para o próximo arquivo do sistema de arquivos. Assim:

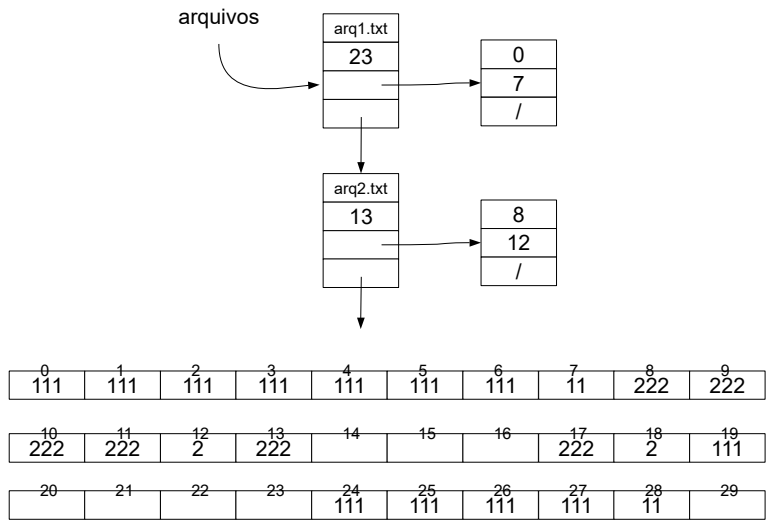


Uma situação exemplo para o sistema de arquivos proposto:



MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Alfenas . UNIFAL-MG
Rua Gabriel Monteiro da Silva, 700 . Alfenas/MG . CEP 37130-000
Fone: (35) 3299-1000 . Fax: (35) 3299-1063

1. De início implemente uma forma de gravar e deletar arquivos no sistema de arquivos. Para gravação é necessário determinar um número de setores suficiente para o armazenamento do arquivo. Por exemplo, se o tamanho do setor é 3 e será feita a gravação de um arquivo com 22 caracteres então são necessários 8 setores. Os setores não precisam ser contíguos, assim a lista atribuída ao arquivo pode conter vários nós. O conteúdo dos arquivos tem que ser escrito para os setores atribuídos aos arquivos. A deleção de um arquivo implica em remover o arquivo da lista arquivo e devolver os setores atribuídos ao arquivo removido para a área de setores disponíveis. Para fins de visualização, sugere-se limpar os setores ocupados pelo arquivo deletado.
2. Um problema deste sistema é a fragmentação dos arquivos o que torna a sua recuperação mais lenta numa sistema de arquivos real. O ideal é que o grupo de setores atribuído a cada arquivo ocupasse lugares contíguos no disco. Estenda o programa que simula o gerenciamento de arquivo para realizar a defragmentação dos arquivos em disco, conforme ilustrado na figura abaixo:



Roteiro:

1. Desenvolva um programa que utiliza a estrutura de dados lista encadeada para simular o sistema de arquivos conforme ilustrado no enunciado do problema.
2. Nesse simulador o usuário deverá ter a opção de: (a) **gravar** arquivo e, neste caso, o usuário deve definir o nome e o conteúdo do arquivo, (b) **deletar** o arquivo a partir do nome do arquivo passado pelo usuário e (c) **apresentar** o conteudo de um arquivo. As outras opções do programa (**mostrar** estruturas, apresentar **ajuda** para o usuário e **fim de operações**) já estão disponíveis no projeto inicial em anexo).
3. Estenda o projeto inicial para incluir a operação de **defragmentação** do disco.
4. Experimente o programa para várias sequências de utilização diferentes.

Observação:

1. Inclua um comentário no cabeçalho do programa fonte com o seguinte formato:

```
(*+------------------------------------------------------------+
  |          UNIFAL - Universidade Federal de Alfenas.         |
  | BACHARELADO EM CIENCIA DA COMPUTACAO.                      |
  | Trabalho..: SIMULACAO DE SISTEMA DE ARQUIVOS FAT           |
  | Disciplina: Estrutura de Dados I                           |
  | Professor.: Luiz Eduardo da Silva                          |
  | Aluno(s)..:  Fulano da Silva                               |
  |              Beltrano da Silva. (MAXIMO 3 ALUNOS).         |
  | Data......: 99/99/9999                                     |
  +------------------------------------------------------------+*)
```

2. Inclua comentários no programa fonte para explicar a lógica desenvolvida.
3. Anexe o código fonte num arquivo zipado com o nome de um integrante do grupo na opção de ENVIO DA ATIVIDADE do Moodle.

Anexo: Código inicial com a definição das estruturas que deverão ser utilizadas:

```
/*-------------------------------------------------
 * Simulador de FAT - File Allocation Table
 * Luiz Eduardo da Silva
 *-------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#define TAM_GRANULO 3
#define TAM_MEMORIA 30
#define TRUE 1
#define FALSE 0

typedef struct noSet * ptnoSet;
typedef struct noSet {
    int inicio, fim;
    ptnoSet prox;
} noSet;

typedef struct noArq *ptnoArq;
typedef struct noArq {
    char nome[13];
    int caracteres;
    ptnoSet setores;
    ptnoArq prox;
} noArq;

typedef char memoria[TAM_MEMORIA][TAM_GRANULO];

void mostraSetores(ptnoSet S, char *n) {
    printf("%s = [", n);
    while (S) {
        printf("(%d,%d)", S->inicio, S->fim);
        S = S->prox;
        if (S) printf(",");
    }
    printf("]\n");
}

void mostraArquivos(ptnoArq A) {
    printf("Arquivos:\n");
    while (A) {
        printf("  %12s, %2d caracter(es).  ", A->nome, A->caracteres);
        mostraSetores(A->setores, "Setores");
        A = A->prox;
    }
    printf("\n");
}

void mostraMemoria(memoria Memo) {
    int i, j;
    for (i = 0; i < TAM_MEMORIA; i++) {
        printf("%3d:[", i);
        for (j = 0; j < TAM_GRANULO - 1; j++)
            printf("%c,", Memo[i][j]);
        printf("%c]", Memo[i][TAM_GRANULO - 1]);
        if ((i + 1) % 10 == 0)
            printf("\n");
    }
}

void inicia(ptnoSet *Area, ptnoArq *Arq, memoria Memo) {
    int i, j;
    *Area = (ptnoSet) malloc(sizeof (noSet));
    (*Area)->inicio = 0;
    (*Area)->fim = TAM_MEMORIA - 1;
    (*Area)->prox = NULL;
    *Arq = NULL;
    for (i = 0; i < TAM_MEMORIA; i++)
        for (j = 0; j < TAM_GRANULO; j++)
            Memo[i][j] = ' ';
}
```

```
/*--------------------------------------------
 * Implementar as rotinas para simulacao da FAT
 *--------------------------------------------*/

void ajuda() {
    printf("\nCOMANDOS\n");
    printf("--------\n");
    printf("G <arquivo.txt> <texto><ENTER>\n");
    printf(" -Grava o <arquivo.txt> e conteúdo <texto> no disco\n");
    printf("D <arquivo.txt>\n");
    printf(" -Deleta o <arquivo.txt> do disco\n");
    printf("A <arquivo.txt>\n");
    printf(" -Apresenta o conteudo do <arquivo.txt>\n");
    printf("M\n");
    printf(" -Mostra as estruturas utilizadas\n");
    printf("H\n");
    printf(" -Apresenta essa lista de comandos\n");
    printf("F\n");
    printf(" -Fim da simulacao\n");
}

/*-----------------------------------------
 * CORPO PRINCIPAL DO PROGRAMA
 *-----------------------------------------*/
int main(void) {
    ptnoSet Area, set;
    ptnoArq Arq, ant;
    memoria Memo;
    char com[3];
    char nome[13];
    char texto[TAM_MEMORIA * TAM_GRANULO];

    inicia(&Area, &Arq, Memo);

    do {
        printf("\n=> ");
        scanf("%3s", com);
        com[0] = toupper(com[0]);
        switch (com[0]) {
            case 'G':
                scanf("%s %s", nome, texto);
                printf("nome = %s\n", nome);
                printf("texto = %s\n", texto);
                /*
                 * Implementar as chamadas das funcoes pra GRAVAR arquivo
                 */
                break;
            case 'D':
                scanf("%s", nome);
                printf("nome = %s\n", nome);
                /*
                 * Implementar as chamadas das funcoes pra DELETAR arquivo
                 */
                break;
            case 'A':
                scanf("%s", nome);
                printf("nome = %s\n", nome);
                /*
                 * Implementar as chamadas das funcoes pra APRESENTAR arquivo
                 */
                break;
            case 'M':
                mostraSetores(Area, "Area");
                mostraArquivos(Arq);
                printf ("Memoria:\n");
                mostraMemoria(Memo);
                break;
            case 'H':
                ajuda();
                break;
        }
    } while (com[0] != 'F');
    printf("\nFim da Execucao\n\n");
    return (EXIT_SUCCESS);
}
```

Exemplo de uso do Simulador

```
=> G arquivo1.txt 11111111
nome = arquivo1.txt
texto = 11111111

=> m
Area = [(3,29)]
Arquivos:
arquivo1.txt, 8 caracter(es). Setores = [(0,2)]

Memoria:
0:[1,1,1] 1:[1,1,1] 2:[1,1, ] 3:[ , , ] 4:[ , , ] 5:[ , , ] 6:[ , , ] 7:[ , , ] 8:[
, , ] 9:[ , , ]
10:[ , , ] 11:[ , , ] 12:[ , , ] 13:[ , , ] 14:[ , , ] 15:[ , , ] 16:[ , , ] 17:[ ,
, ] 18:[ , , ] 19:[ , , ]
20:[ , , ] 21:[ , , ] 22:[ , , ] 23:[ , , ] 24:[ , , ] 25:[ , , ] 26:[ , , ] 27:[ ,
, ] 28:[ , , ] 29:[ , , ]

=> g arquivo2.txt 222222222222222222222
nome = arquivo2.txt
texto = 22222222222222222222

=> m
Area = [(11,29)]
Arquivos:
arquivo1.txt, 8 caracter(es). Setores = [(0,2)]
arquivo2.txt, 22 caracter(es). Setores = [(3,10)]

Memoria:
0:[1,1,1] 1:[1,1,1] 2:[1,1, ] 3:[2,2,2] 4:[2,2,2] 5:[2,2,2] 6:[2,2,2] 7:[2,2,2] 8:
[2,2,2] 9:[2,2,2]
10:[2, , ] 11:[ , , ] 12:[ , , ] 13:[ , , ] 14:[ , , ] 15:[ , , ] 16:[ , , ] 17:[ ,
, ] 18:[ , , ] 19:[ , , ]
20:[ , , ] 21:[ , , ] 22:[ , , ] 23:[ , , ] 24:[ , , ] 25:[ , , ] 26:[ , , ] 27:[ ,
, ] 28:[ , , ] 29:[ , , ]

=> g arquivo3.txt 333333333
nome = arquivo3.txt
texto = 333333333

=> m
Area = [(14,29)]
Arquivos:
arquivo1.txt, 8 caracter(es). Setores = [(0,2)]
arquivo2.txt, 22 caracter(es). Setores = [(3,10)]
arquivo3.txt, 9 caracter(es). Setores = [(11,13)]

Memoria:
0:[1,1,1] 1:[1,1,1] 2:[1,1, ] 3:[2,2,2] 4:[2,2,2] 5:[2,2,2] 6:[2,2,2] 7:[2,2,2] 8:
[2,2,2] 9:[2,2,2]
10:[2, , ] 11:[3,3,3] 12:[3,3,3] 13:[3,3,3] 14:[ , , ] 15:[ , , ] 16:[ , , ] 17:[ ,
, ] 18:[ , , ] 19:[ , , ]
20:[ , , ] 21:[ , , ] 22:[ , , ] 23:[ , , ] 24:[ , , ] 25:[ , , ] 26:[ , , ] 27:[ ,
, ] 28:[ , , ] 29:[ , , ]

=> g arquivo4.txt 444444444444
nome = arquivo4.txt
texto = 444444444444

=> g arquivo5.txt 555555555555555555555
```

```
nome = arquivo5.txt
texto = 555555555555555555

=> g arquivo5.txt 6666666666666666
nome = arquivo5.txt
texto = 6666666666666666
Arquivo já existe

=> g arquivo6.txt 66666666666666666
nome = arquivo6.txt
texto = 66666666666666666
Memoria cheia

=> m
Area = [(25,29)]
Arquivos:
arquivo1.txt, 8 caracter(es). Setores = [(0,2)]
arquivo2.txt, 22 caracter(es). Setores = [(3,10)]
arquivo3.txt, 9 caracter(es). Setores = [(11,13)]
arquivo4.txt, 12 caracter(es). Setores = [(14,17)]
arquivo5.txt, 21 caracter(es). Setores = [(18,24)]

Memoria:
0:[1,1,1] 1:[1,1,1] 2:[1,1, ] 3:[2,2,2] 4:[2,2,2] 5:[2,2,2] 6:[2,2,2] 7:[2,2,2] 8:
[2,2,2] 9:[2,2,2]
10:[2, , ] 11:[3,3,3] 12:[3,3,3] 13:[3,3,3] 14:[4,4,4] 15:[4,4,4] 16:[4,4,4] 17:
[4,4,4] 18:[5,5,5] 19:[5,5,5]
20:[5,5,5] 21:[5,5,5] 22:[5,5,5] 23:[5,5,5] 24:[5,5,5] 25:[ , , ] 26:[ , , ] 27:[ ,
, ] 28:[ , , ] 29:[ , , ]

=> d arquivo2.txt
nome = arquivo2.txt

=> m
Area = [(3,10),(25,29)]
Arquivos:
arquivo1.txt, 8 caracter(es). Setores = [(0,2)]
arquivo3.txt, 9 caracter(es). Setores = [(11,13)]
arquivo4.txt, 12 caracter(es). Setores = [(14,17)]
arquivo5.txt, 21 caracter(es). Setores = [(18,24)]

Memoria:
0:[1,1,1] 1:[1,1,1] 2:[1,1, ] 3:[ , , ] 4:[ , , ] 5:[ , , ] 6:[ , , ] 7:[ , , ] 8:[
, , ] 9:[ , , ]
10:[ , , ] 11:[3,3,3] 12:[3,3,3] 13:[3,3,3] 14:[4,4,4] 15:[4,4,4] 16:[4,4,4] 17:
[4,4,4] 18:[5,5,5] 19:[5,5,5]
20:[5,5,5] 21:[5,5,5] 22:[5,5,5] 23:[5,5,5] 24:[5,5,5] 25:[ , , ] 26:[ , , ] 27:[ ,
, ] 28:[ , , ] 29:[ , , ]

=> d arquivo3.txt
nome = arquivo3.txt

=> m
Area = [(3,13),(25,29)]
Arquivos:
arquivo1.txt, 8 caracter(es). Setores = [(0,2)]
arquivo4.txt, 12 caracter(es). Setores = [(14,17)]
arquivo5.txt, 21 caracter(es). Setores = [(18,24)]

Memoria:
0:[1,1,1] 1:[1,1,1] 2:[1,1, ] 3:[ , , ] 4:[ , , ] 5:[ , , ] 6:[ , , ] 7:[ , , ] 8:[
, , ] 9:[ , , ]
10:[ , , ] 11:[ , , ] 12:[ , , ] 13:[ , , ] 14:[4,4,4] 15:[4,4,4] 16:[4,4,4] 17:
[4,4,4] 18:[5,5,5] 19:[5,5,5]
20:[5,5,5] 21:[5,5,5] 22:[5,5,5] 23:[5,5,5] 24:[5,5,5] 25:[ , , ] 26:[ , , ] 27:[ ,
```

```
, ] 28:[ , , ] 29:[ , , ]

=> g arquivo2.txt 2222
nome = arquivo2.txt
texto = 2222

=> m
Area = [(5,13),(25,29)]
Arquivos:
arquivo1.txt, 8 caracter(es). Setores = [(0,2)]
arquivo2.txt, 4 caracter(es). Setores = [(3,4)]
arquivo4.txt, 12 caracter(es). Setores = [(14,17)]
arquivo5.txt, 21 caracter(es). Setores = [(18,24)]

Memoria:
0:[1,1,1] 1:[1,1,1] 2:[1,1, ] 3:[2,2,2] 4:[2, , ] 5:[ , , ] 6:[ , , ] 7:[ , , ] 8:[
, , ] 9:[ , , ]
10:[ , , ] 11:[ , , ] 12:[ , , ] 13:[ , , ] 14:[4,4,4] 15:[4,4,4] 16:[4,4,4] 17:
[4,4,4] 18:[5,5,5] 19:[5,5,5]
20:[5,5,5] 21:[5,5,5] 22:[5,5,5] 23:[5,5,5] 24:[5,5,5] 25:[ , , ] 26:[ , , ] 27:[ ,
, ] 28:[ , , ] 29:[ , , ]

=> a arquivo5.txt
nome = arquivo5.txt

Setores | Conteudo
--------+----------------
(18,24) | 555555555555555555

=> d arquivo1.txt
nome = arquivo1.txt

=> g arquivo3.txt 33333333333333333
nome = arquivo3.txt
texto = 3333333333333333

=> m
Area = [(9,13),(25,29)]
Arquivos:
arquivo2.txt, 4 caracter(es). Setores = [(3,4)]
arquivo3.txt, 20 caracter(es). Setores = [(0,2),(5,8)]
arquivo4.txt, 12 caracter(es). Setores = [(14,17)]
arquivo5.txt, 21 caracter(es). Setores = [(18,24)]

Memoria:
0:[3,3,3] 1:[3,3,3] 2:[3,3,3] 3:[2,2,2] 4:[2, , ] 5:[3,3,3] 6:[3,3,3] 7:[3,3,3] 8:
[3,3, ] 9:[ , , ]
10:[ , , ] 11:[ , , ] 12:[ , , ] 13:[ , , ] 14:[4,4,4] 15:[4,4,4] 16:[4,4,4] 17:
[4,4,4] 18:[5,5,5] 19:[5,5,5]
20:[5,5,5] 21:[5,5,5] 22:[5,5,5] 23:[5,5,5] 24:[5,5,5] 25:[ , , ] 26:[ , , ] 27:[ ,
, ] 28:[ , , ] 29:[ , , ]

=> a arquivo3.txt
nome = arquivo3.txt

Setores | Conteudo
--------+----------------
( 0, 2) | 333333333
( 5, 8) | 33333333333

=> f

Fim da Execucao
```