Biblioteca de Apoio a Desenvolvimento Terminal

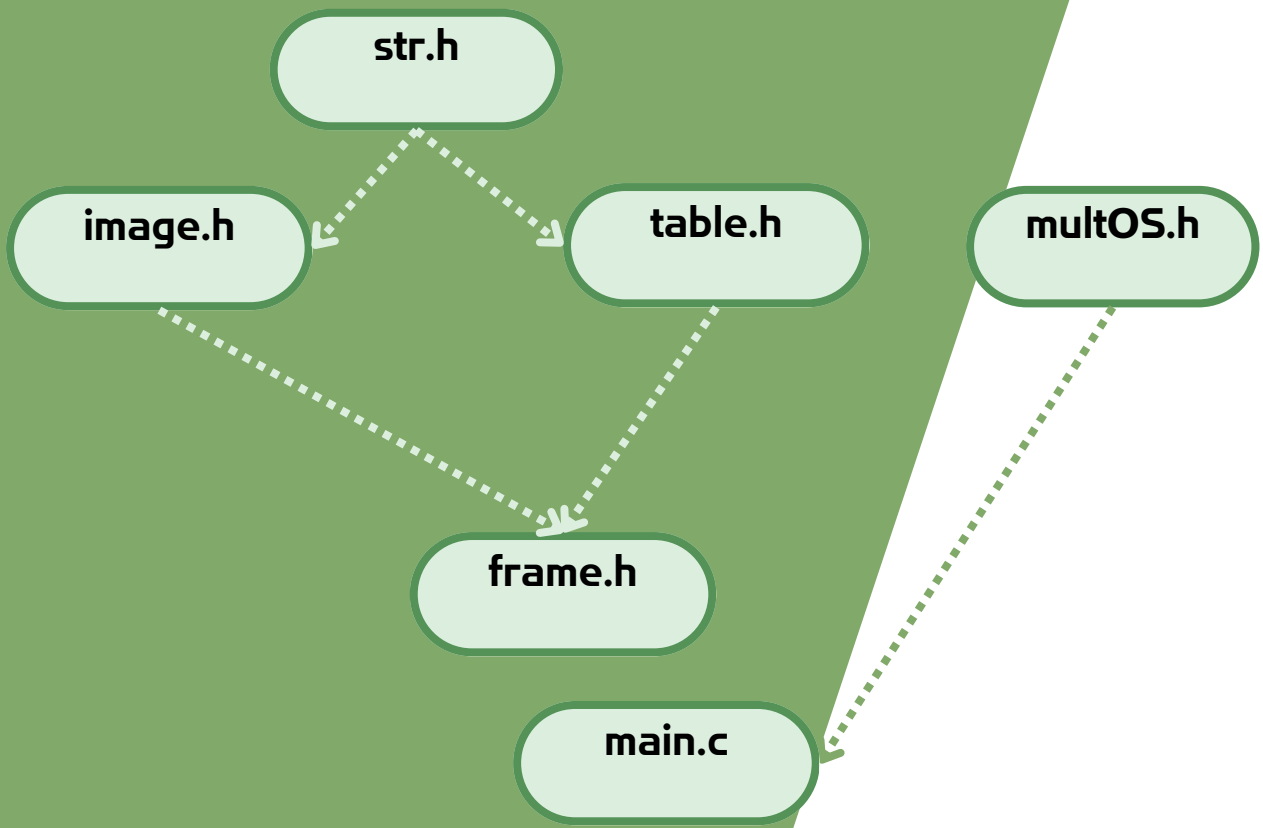
BADT 1.0

Definição

Essa biblioteca consiste na simplificação de desenvolvimento de aplicações em **C**. Contando principalmente com gerenciadores de janelas dinâmicas e manipulação -quase- direta de arquivos de forma não destrutiva.

Visão Geral

Dependências



Visão Geral Resumo dos Códigos

str.h

- Funções de manipulação de char * criadas principalmente para solucionar tratativas recorrentes das estruturas.

image.h

- A estrutura **Image** e seus métodos foram criados para gerenciar "imagens ACSII" de maneira eficiente.
- A imagem é armazenada em Image.pixels (char *)

table.h

- A estrutura **Table** e seus métodos foram criados para manipular arquivos. sua utilização depende unicamente de **str.h.**
- O arquivo todo fica armazenado em Table.registry (char *).

frame.h

- A estrutura **Frame** e seus métodos foram criados para desenvolver interfaces agradáveis para o usuário final de maneira intuitiva.
- Basicamente uma herança de **Image**, com mais funcionalidades.

MultOS.h

- Possui funções de apoio que necessitam ser escritas de maneiras diferentes para SO's diferentes.

INTRODUÇÃO

Para manipular arquivos é necessário importar **frame.h** e inicializar a tabela.

```c
#include "frame.h"

void main(){

    Table table;
    tableSetup(&table, "dscif","coluna 1,coluna 2,coluna 3,coluna 4,coluna 5");
}
```

Parâmetros:

- ponteiro de Table
- texto contendo o tipo de cada coluna
- nome das colunas separados por vírgula

```
    printf("%s", table.registry);
luan@luan-Lenovo-IdeaPad-S145-15IWL:~/Documentos/GITHub_EDITAVEL/BADT$ ./a.out
| | | |
```

Ao “printar” o registro da tabela é possível notar que uma linha vazia já foi automaticamente criada.

INSERINDO REGISTROS

Para inserir registros basta usar a coordenada da célula e um ponteiro do tipo adequado da coluna (definido na inicialização da tabela).

```c
    Table table;
    tableSetup(&table, "dscif","coluna 1,coluna 2,coluna 3,coluna 4,coluna 5");

    tableInsert(&table, 0, 0, &numero);
    tableInsert(&table, 1, 3, "texto");

    printf("%s", table.registry);
}
```

```
luan@luan-Lenovo-IdeaPad-S145-15IWL:~/Documentos/GITHub_EDITAVEL/BADT$ ./a.out
1| | | |
 | | | |
 | | | |
 |texto| | |
```

*note que novas linhas são criadas automaticamente quando necessário

DELETANDO REGISTROS

Basta passar o ponteiro , coluna, linha.

```c
    tableDelete(&table,1,3);
```

```
luan@luan-Lenovo-IdeaPad-S145-15IWL:~/Documentos/GITHub_EDITAVEL/BADT$ ./a.out
1| | | |
 | | | |
 | | | |
 | | | |
```

ATUALIZANDO REGISTROS

Basta passar o ponteiro , coluna, linha e o novo valor, da mesma maneira que seria feito em tableInsert.

```
13      tableUpdate(&table,1,3,"novo texto");
luan@luan-Lenovo-IdeaPad-S145-15IWL:~/Documentos/GITHub_EDITAVEL/BADT$ ./a.out
1| | | |
 | | | |
 | | | |
 |novo texto| | |
```

*Não é necessário usar tableDelete() previamente, apenas atualizar diretamente já basta.

ARQUIVOS...

A interação de tabelas com arquivos txt é feita por meio de duas funções:

- tableToTxt(Table *table, char *directory);
- txtToTable(Table *table, char *directory);

Na primeira execução é importante usar txtToTable() para salvar o arquivo formatado corretamente.
Nas próximas execuções segue a fórmula

- Setup
- txtToTable
- Modificações
- tableToTxt
- freeTable

```c
#include "frame.h"

void main(){

    int numero = 1;

    Table table;
    tableSetup(&table, "dscif","coluna 1,coluna 2,coluna 3,coluna 4,coluna 5");

    //txtToTable(&table, "./txtTabela");

    tableInsert(&table, 0, 0, &numero);
    tableInsert(&table, 1, 3, "texto");

    tableUpdate(&table,1,3,"novo texto");

    printf("%s", table.registry);

    tableToTxt(&table, "./txtTabela");

    tableFree(&table);
}
```

*Primeira execução, depois disso basta descomentar txtToTable.

Nunca esqueça de usar tableFree() antes de encerrar o programa. essa função é responsável por liberar as alocações dinâmicas necessárias para o funcionamento das tabelas.