Universidade de Brasília

Departamento de Ciência da Computação

Lista de Exercício 1 - OA

Autores:

Gabriel Bessa 16/0120811

Thiago Veras 16/0146682

Disciplina:

Organização de Arquivos

Turma:

A

Professor:

Oscar Gaidos

Brasília

30 de Março de 2018

1 Exercícios:

Exercício 1: Implemente um programa que leia e concatene UM OU MAIS arquivos textos e grave o arquivo concatenado em DISCO ou o exibe no VÍDEO. Faça esse programa o mais seguro possível, isto é, após operações de E/S, verifique se não houve erro. Se houve, sinalize para o usuário e contorne a situação. O programa deverá ter a opção de ler caracter a caracter ou leitura de bloco.

Explicação do Código: Nosso programa consiste em ler 'n' arquivos de uma só vez. Primeiramente ele checa se os arquivos podem ser abertos, uma vez que os mesmos precisam estar no mesmo diretório que o programa em si. Se forem abertos o nome de cada arquivo é salvo num vetor dinâmico de string. Quando o programa começa a rodar, ele vai na ordem que o usuário colocou os programas, lê a string de cada arquivo, salva num vetor dinâmico e joga no arquivo final, assim concatenando todas as versões numa versão final (final.txt).

$bool\ open\_all(vector < string > files)$:

```
bool open_all(vector<string> files){
  // Return if all files could be opened
  bool ans = true;
  fstream arquivo;
  // Walk through files vector to get file name
  for(auto file : files){
      // Open file
    arquivo.open(file);
    //Check if opened  the file

    cout << "Opening " << file << " : ";
    if(!arquivo){
      cout << RED << "FAILED" << RESET << endl;
      ans = false;
    }
    else{
      cout << GREEN << "SUCCESS" << RESET << endl;
      // Close file if was opened
      arquivo.close();
    }
  }
  return ans;
}
```

Esta função é responsável por verificar se é possivel abrir todos os arquivos fornecidos, primeiramente se incializar a varável de retorno como verdadeira para poder retornar caso seja possível. se percorre todos as strings com o nome dos arquivos (linha 6) e tenta abrir o arquivo com aquele nome, se o arquivo não existir (linha 12) imprime na tela uma mensagem de falha e seta a variável de retorno como falsa, pois não conseguiu abrir o arquivo. Caso contrário, imprime uma mensagem em verde de sucesso e fecha o arquivo.

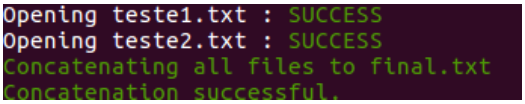

Figura 1: Two tests in this case, both being right.

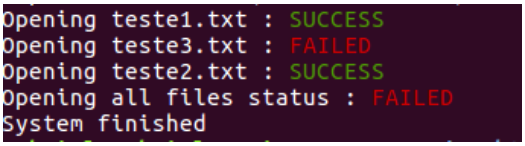

Figura 2: Three tests, this time the third archive doesn't exists, so it fails to open it.

$vector < string > read(int\ n)$:

```
1  vector<string> read(int n){
2
3    // Vector of string to store all files names
4    vector<string> files;
5
6    // String to get the file by file name
7    string file;
8
9    for(int i = 1; i <= n; i++){
10     cout << "Input file " << i << " name (With extention
         ) : ";
11     // Read file name
12     cin >> file;
13     // Push file to files vector
14     files.push_back(file);
15   }
16   return files;
17 }
```

A função read é responsável por simplesmente ler todos os nomes dos arquivos inseridos pelo usuário e retornar um vetor de string com todos os nomes.

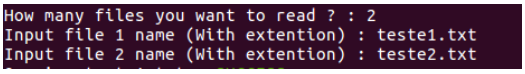

Figura 3: Two tests in this case, both being '.txt' archives.

$voidconcatenate(vector < string > files)$:

```
// Concatenate all files
void concatenate(vector<string> files){
  // String to read line by line in the file
  string line;
  // Vector to store all lines concatenated
  vector<string> all;
  fstream arquivo;
  // Walk through all files
  for(auto file : files){
    // Open the file
    arquivo.open(file);

    // Read all lines and store in vector all
    while(getline(arquivo,line))all.push_back(line);

    // Close file
    arquivo.close();
  }

  // Open the final file that have all files
     concatenated
  arquivo.open("final.txt");

  // Check if file exist
  if(!arquivo){

    // If the file don't exist create a file called
       final.txt
    cout << RED << "File final.txt don't exist ...
       Creating a new one..." << RESET << endl;

    // Clear all fstream content
    arquivo.clear();
```

```
31
32      // Create a file
33      arquivo.open("final.txt", ios::out);
34
35      cout << GREEN << "File final.txt created successfuly
            " << RESET << endl;
36      }
37      else{
38        // If already exist, close the final.txt file
39        arquivo.close();
40
41        // Reopen file with ios:app mode that all
             operations will be performed at the end of the
             file
42        arquivo.open("final.txt", ios::app);
43
44        cout << GREEN << "Concatenating all files to final
             .txt" << RESET << endl;
45      }
46
47      // Walk through all concatenated files and write at
          the end of final.txt file
48    for(auto line : all) arquivo << line;
49
50    // Close file
51      arquivo.close();
52
53    // Success message
54    cout << GREEN << "Concatenation successful.\n" <<
        RESET;
55  }
```

A função concatenate é a função mais importante de todo o programa, ela é reponsavel por concatenar todos os dados de todos os arquivos. Primeiramente se percorre o vetor com o nome de todos os arquivos forcenidos pela função *read*() (linha 9), depois itera por todas as linhas do arquivos e vai salvando cada linha no vetor dinâmico all (linha 14), depois fecha o arquivo. Para para concatenar todos os arquivos, primeiramente o programa teste se o arquivo final.txt (arquivo que irá ficar o resultado final da concatenação) já existe (linha 24), se o arquivo não existir, imprime na tela uma mensagem de erro ao abrir o arquivo, porém o próprio programa já cria uma novo arquivo (linha 33), caso o arquivo exista, o sistema o fecha e reabre com o método app, que possibilita a escrita no final do arquivo.

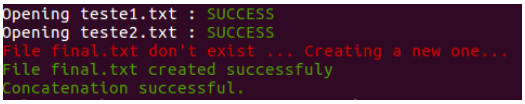

Figura 4: Two archives, being concatenated, but the final one doesn't exists yet "final.txt".

2 Outputs:

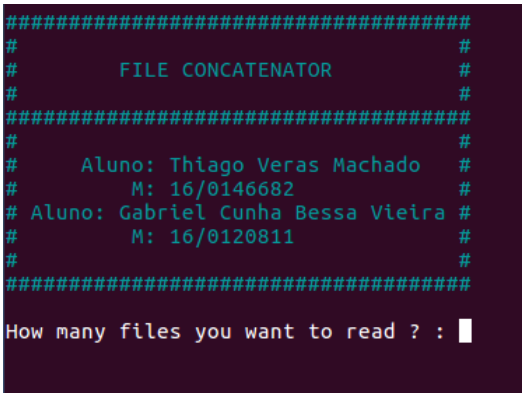

Figura 5: Introduction of the program.

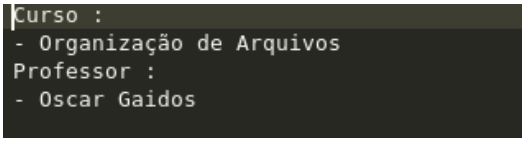

Figura 6: First test to concatenate.

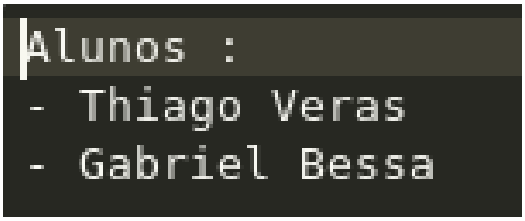

Figura 7: Second test to concatenate.

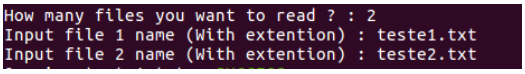

Figura 8: Two tests in this case, both being '.txt' archives.

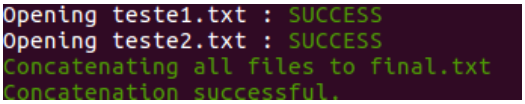

Figura 9: Two files being concatenated into a final version.

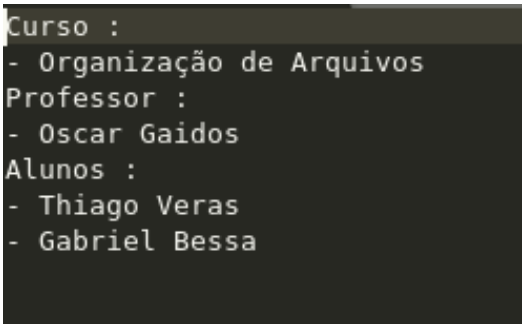

Figura 10: Final archive fully concatenated.

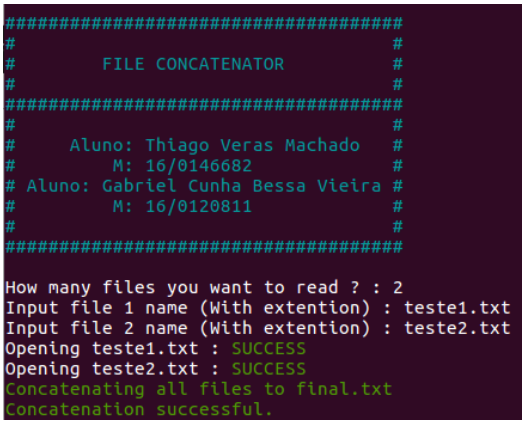

Figura 11: The whole compilation from the beginning to the end.

Exercício 2:

Pesquise e compare as diferenças entre um arquivo texto e um arquivo binário, do ponto de vista físico.

Arquivos físicos são uma coleção de bytes que são armazenados em um determinado dispositivo. No nível mais baixo de complexidade, tanto o arquivo texto e o arquivo binário são similares, pois ambos contém seus dados armazenados como uma série de bits 0's e 1's, mas suas estruturas internas são bem diferentes. Nos arquivos textos esses bits representam caracteres, já no arquivo binário, os bits representam dados personalizados, que se tentar abrir num arquivo texto, aparece uma sequência de símbolos.