Arquivos

Programação e Desenvolvimento de Software I
Heitor Ramos

```
int main(int argc, char **argv)
```

int main(int argc, char **argv)

- Até o momento, nosso comando main era utilizado com void apenas
- O usuário na linha de comando pode passar argumentos para o main

```c
// argc: número de argumentos
// argv: ponteiro para ponteiro de char
//       em outras palavras, argc strings
int main(int argc, char **argv);

// argc: número de argumentos
// argv: ponteiro para vetor de char
//       em outras palavras, argc strings
int main(int argc, char *argv[]);
```

Exemplo

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
  printf("Recebi %d argumentos\n", argc);
  for (int i = 0; i < argc; i++)
       printf("Argumento %d foi %s\n", i, argv[i]);
  return 0;
}
```

Detalhes

- O argumento 0 sempre é o nome do programa
- A partir do 1 temos o que o usuário passa na linha de comando
- No geral são separados por espaço
    - Use " " na linha de comando caso tenha espaços

Arquivos de Texto

Arquivos

- Existem duas "grandes" memórias no computador
- Memória RAM
  - Onde seus programas vivem quando estão executando
- Disco
  - Onde seus arquivos vivem

Tipos de Arquivos

Tipo	Descrição	Binário ou Texto?
`.txt`	arquivo de texto	T
`.csv`	tabela separada por ,	T
`.tsv`	tabela separada por '\t'	T
`.pdf`	portable document	B
`.docx`	arquivo word	T
`.out, *, .exe`	executável	B

Caminhos

Windows

```
C:\Users\heitor\PDS1\Estudos\prog1.c
“C:\Users\heitor ramos\PDS1\Estudos\prog1.c"
```

Linux/MacOSx/Unix*

```
/home/heitor/pds1/estudos/prog1.c
/home/heitor/pds1/estudos\ prova1/prog1.c
“/home/heitor/pds1/estudos prova1/prog1.c"
```

Abrindo Arquivos (API)

```
#include <stdio.h>

FILE *

fopen(const char *restrict filename, const char *restrict
mode);
```

RETURN VALUES

Upon successful completion fopen(), fdopen(), and freopen() return a FILE pointer. Otherwise, NULL is returned and the global variable errno is set to indicate the error.

NULL

- NULL é um valor especial na linguagem C
- Indica que um ponteiro é vazio
- Aponta para NADA/NULL
- Então, se ao abrir o arquivo eu recebo NULL, qual o significado?

Problema 1

Fazer um programa que abre um arquivo e indica um erro caso o arquivo não exista.

Abrindo Arquivos

```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv) {
  if (argc < 2) {
        printf("Erro, favor passe um nome de
arquivo");
        return EXIT_FAILURE;
  }

  char *fpath = argv[1];
  FILE *file = fopen(fpath, "r");
  if (file == NULL) {
        printf("Arquivo %s não existe!", fpath);
        return EXIT_FAILURE;
  }
  // Sempre feche arquivos!
  fclose(file);
  return EXIT_SUCCESS;
}
```

Modos de Abertura

Modo	Significado	Se o arquivo existir?	Se não existir?	Ponto Inicial
`r`	Leitura	-	NULL (erro)	`0`
`w`	Escrita	“Apaga e cria novo”	Cria o arquivo	`0`
`a`	Append	-	Cria o arquivo	`EOF (end of file)`
`r+`	Leitura e Escrita	-	NULL (erro)	`0`
`w+`	Leitura e Escrita	“Apaga e cria novo”	Cria o arquivo	`0`
`a+`	Append e Leitura	-	Cria o arquivo	`EOF (end of file)`

Modo Abertura

Modo	*Arquivo*	*Função*
"r"	Texto	Leitura. Arquivo deve existir.
"w"	Texto	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a"	Texto	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"rb"	Binário	Leitura. Arquivo deve existir.
"wb"	Binário	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"ab"	Binário	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+"	Texto	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
"w+"	Texto	Leitura/Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a+"	Texto	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+b"	Binário	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
"w+b"	Binário	Leitura/Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a+b"	Binário	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").

Funções de Leitura e Escrita

- Trabalhar com arquivos é igual a trabalhar com a entrada/saída padrão
- Apenas indicamos qual arquivo estamos tratando no início

```
scanf(…)    vs  fscanf(arquivo, …)
getc(…)     vs  fgetc(arquivo, …)
putc(…)     vs  fputc(arquivo, …)
puts(…)     vs  fputs(arquivo, …)
printf(…)   vs  fprintf(arquivo, …)

gets(…) vs fgets(…,arquivo)
```

End of File

- O valor EOF indica o fim do arquivo
  - fgetc retorna EOF
  - fscanf não

- Também podemos usar a função feof
  - Útil quando usamos fscanf

Problema 2

Faça um programa que lê um arquivo e repete seu conteúdo na tela

Duas forma de imprimir na tela

```c
#include <stdio.h>
#include <stdlib.h>

// arquivo já aberto para leitura no main
void imprime_tela_fgetc(FILE *file) {
  char c;
  while ((c = fgetc(file)) != EOF) {
        printf("%c", c);
  }
}
// arquivo já aberto para leitura no main
void imprime_tela_feof(FILE *file) {
  char c;
  while (1) {
        fscanf(file, "%c", &c);
       if(feof(file))
          break;
        printf("%c", c);
  }
}
```

Problema 3

Faça um programa que lê um arquivo e cria uma cópia idêntica

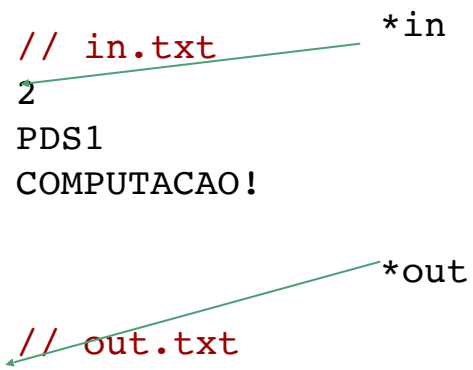
Copiando arquivos

```
// in.txt
2
PDS1
COMPUTACAO!


// out.txt
```

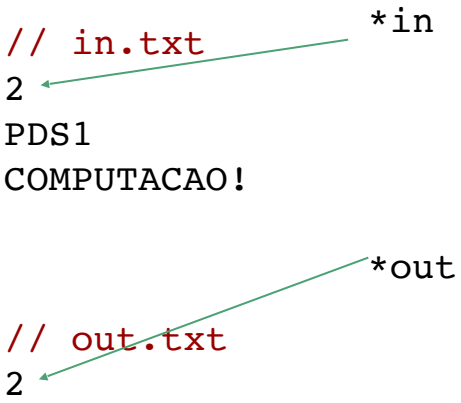
```c
#include <stdio.h>
#include <stdlib.h>
void copia(FILE *in, FILE *out) {
  char c;
  while (1) {
        fscanf(in, "%c", &c);
        if(feof(in))
           break;
        fprintf(out, "%c", c);
  }
}
int main(int argc, char **argv) {
  // . . . tratamento de entrada e erros
  FILE *in = fopen(argv[1], "r");
  // . . . tratamento de entrada e erros
  FILE *out = fopen(argv[2], "w");
  copia(in, out);
  fclose(in);
  fclose(out);
}
```

Copiando arquivos



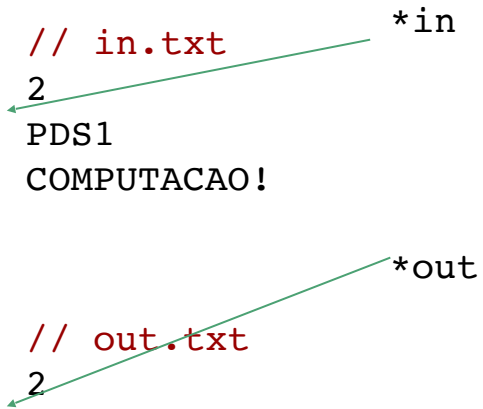

```c
#include <stdio.h>
#include <stdlib.h>
void copia(FILE *in, FILE *out) {
  char c;
  while (1) {
       fscanf(in, "%c", &c);
       if(feof(in))
          break;
       fprintf(out, "%c", c);
  }
}
int main(int argc, char **argv) {
  // . . . tratamento de entrada e erros
  FILE *in = fopen(argv[1], "r");
  // . . . tratamento de entrada e erros
  FILE *out = fopen(argv[2], "w");
  copia(in, out);
  fclose(in);
  fclose(out);
}
```

Copiando arquivos




```c
#include <stdio.h>
#include <stdlib.h>
void copia(FILE *in, FILE *out) {
  char c;
  while (1) {
       fscanf(in, "%c", &c);
       if(feof(in))
          break;
       fprintf(out, "%c", c);
  }
}
int main(int argc, char **argv) {
  // . . . tratamento de entrada e erros
  FILE *in = fopen(argv[1], "r");
  // . . . tratamento de entrada e erros
  FILE *out = fopen(argv[2], "w");
  copia(in, out);
  fclose(in);
  fclose(out);
}
```

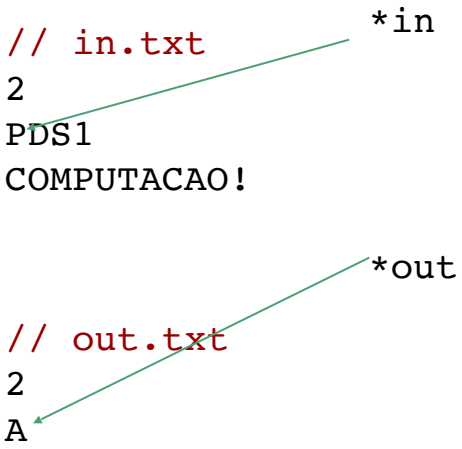
Copiando arquivos



Leu e imprimiu o '\n'

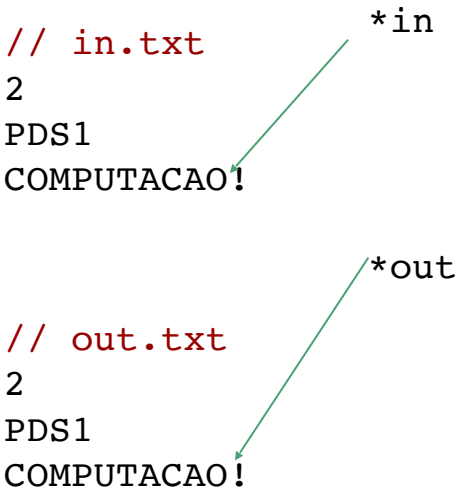
```c
#include <stdio.h>
#include <stdlib.h>
void copia(FILE *in, FILE *out) {
  char c;
  while (1) {
       fscanf(in, "%c", &c);
       if(feof(in))
          break;
       fprintf(out, "%c", c);
  }
}
int main(int argc, char **argv) {
  // . . . tratamento de entrada e erros
  FILE *in = fopen(argv[1], "r");
  // . . . tratamento de entrada e erros
  FILE *out = fopen(argv[2], "w");
  copia(in, out);
  fclose(in);
  fclose(out);
}
```

Copiando arquivos



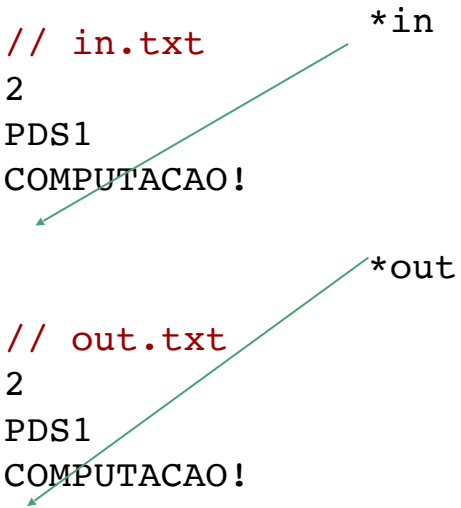

```c
#include <stdio.h>
#include <stdlib.h>
void copia(FILE *in, FILE *out) {
  char c;
  while (1) {
       fscanf(in, "%c", &c);
       if(feof(in))
          break;
       fprintf(out, "%c", c);
  }
}
int main(int argc, char **argv) {
  // . . . tratamento de entrada e erros
  FILE *in = fopen(argv[1], "r");
  // . . . tratamento de entrada e erros
  FILE *out = fopen(argv[2], "w");
  copia(in, out);
  fclose(in);
  fclose(out);
}
```

Copiando arquivos




```c
#include <stdio.h>
#include <stdlib.h>
void copia(FILE *in, FILE *out) {
  char c;
  while (1) {
       fscanf(in, "%c", &c);
       if(feof(in))
          break;
       fprintf(out, "%c", c);
  }
}
int main(int argc, char **argv) {
  // . . . tratamento de entrada e erros
  FILE *in = fopen(argv[1], "r");
  // . . . tratamento de entrada e erros
  FILE *out = fopen(argv[2], "w");
  copia(in, out);
  fclose(in);
  fclose(out);
}
```

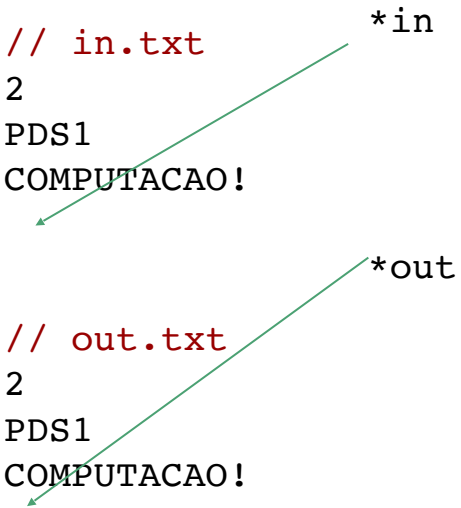
Copiando arquivos



Leu e imprimiu o '\n'

```c
#include <stdio.h>
#include <stdlib.h>
void copia(FILE *in, FILE *out) {
  char c;
  while (1) {
       fscanf(in, "%c", &c);
       if(feof(in))
          break;
       fprintf(out, "%c", c);
  }
}
int main(int argc, char **argv) {
  // . . . tratamento de entrada e erros
  FILE *in = fopen(argv[1], "r");
  // . . . tratamento de entrada e erros
  FILE *out = fopen(argv[2], "w");
  copia(in, out);
  fclose(in);
  fclose(out);
}
```

Copiando arquivos




```c
#include <stdio.h>
#include <stdlib.h>
void copia(FILE *in, FILE *out) {
  char c;
  while (1) {
       fscanf(in, "%c", &c);
       if(feof(in))
          break;
       fprintf(out, "%c", c);
  }
}
int main(int argc, char **argv) {
  // . . . tratamento de entrada e erros
  FILE *in = fopen(argv[1], "r");
  // . . . tratamento de entrada e erros
  FILE *out = fopen(argv[2], "w");
  copia(in, out);
  fclose(in);
  fclose(out);
}
```

Problema 4

Faça um programa que lê um arquivo e copia o mesmo para o fim de outro arquivo

Problema 4

Faça um programa que lê um arquivo e cópia o mesmo para o fim de outro arquivo

Basta abrir out com “a”

```c
#include <stdio.h>
#include <stdlib.h>
void copia(FILE *in, FILE *out) {
  char c;
  while (1) {
        fscanf(in, "%c", &c);
        if(feof(in))
           break;
        fprintf(out, "%c", c);
  }
}
int main(int argc, char **argv) {
  // . . . tratamento de entrada e erro
  FILE *in = fopen(argv[1], "r");
  // . . . tratamento de entrada e erro
  FILE *out = fopen(argv[2], "a");
  copia(in, out);
  fclose(in);
  fclose(out);
}
```

Problema 5

Temos um arquivo com o seguinte forma:
- A primeira linha indica o número de matrizes (n)
- A partir daqui o arquivo contém n matrizes
- Para cada matriz indicamos o número de linhas e colunas
- Depois indicamos os elementos

```
3
2 4
1 -1 90 20
0 -100 1 -7
3 5
0 0 0 -3 0
0 -7 0 0 0
0 0 1 5 0
1 1
7
```

Problema 5

Faça um programa que lê as n matrizes e imprime a soma de todos os elementos de cada.

```
3
2 4
1 -1 90 20
0 -100 1 -7
3 5
0 0 0 -3 0
0 -7 0 0 0
0 0 1 5 0
1 1
7
```

Problema 5

```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv) {
  // . . . tratamento de entrada e erros
  FILE *in = fopen(argv[1], "r");
  // . . . tratamento de entrada e erros
  int n = 0;
  fscanf(in, "%d", &n);
  int nl;
  int nc;
  for (int m = 0; m < n; m++) {
    fscanf(in, "%d %d", &nl, &nc);
    int matriz[nl][nc];
    for(int i = 0; i < nl; i++)
      for(int j = 0; j < nc; j++)
        fscanf(in, "%d", &matriz[i][j]);

    int normasoma = soma_tudo(nl, nc, matriz);
    printf("%d\n", normasoma);
  }
  fclose(in);
  return 0;
}
```

```c
int soma_tudo(int nl, int nc,
              int matriz[nl][nc])
  // . . . computa a norma
}
```

Escrita/Leitura de bloco de dados

- Além da leitura/escrita de caracteres e seqüências de caracteres, podemos ler/escrever blocos de dados.
- Para tanto, temos duas funções
  - **fwrite()**
  - **fread()**

Escrita/Leitura de bloco de dados

- A função **fwrite** recebe 4 argumentos
  - **buffer:** ponteiro para a região de memória na qual estão os dados;
  - **numero_de_bytes:** tamanho de cada posição de memória a ser escrita;
  - **count:** total de unidades de memória que devem ser escritas;
  - **fp**: ponteiro associado ao arquivo onde os dados serão escritos.

Exemplo de fwrite

- Vscode

Escrita/Leitura de bloco de dados

- A função **fread** funciona como a sua companheira **fwrite**, porém lendo do arquivo.
- Como na função **fwrite, fread** retorna o número de itens escritos. Este valor será igual a **count** a menos que ocorra algum erro.

Exemplo de fread

- Vscode

Movendo-se pelo arquivo

- A função **fseek** recebe 3 parâmetros
  - **fp**: o ponteiro para o arquivo;
  - **numbytes**: é o total de bytes a partir de **origem** a ser pulado;
  - **origem**: determina a partir de onde os **numbytes** de movimentação serão contados. Os valores possíveis são definidos por macros em **stdio.h** e são:

Movendo-se pelo arquivo

- Os valores possíveis para **origem** são definidos por macros em **stdio.h** e são:

Nome	Valor	Significado
SEEK_SET	0	Início do arquivo
SEEK_CUR	1	Ponto corrente no arquivo
SEEK_END	2	Fim do arquivo

- Portanto, para mover **numbytes** a partir do início do arquivo, **origem** deve ser SEEK_SET. Para mover da posição atual, SEEK_CUR, e a partir do final do arquivo, SEEK_END.
- A função devolve 0 quando bem sucedida.

Movendo-se pelo arquivo

- Vscode

Movendo-se pelo arquivo

- Outra opção de movimentação pelo arquivo é simplesmente retornar para o seu início.
- Para tanto, usa-se a função **rewind**:

    void rewind (FILE *fp);

Apagando um arquivo

- Além de permitir manipular arquivos, a linguagem C também permite apagá-lo do disco. Isso pode ser feito utilizando a função **remove**:

  int remove (char *nome_do_arquivo);

- Diferente das funções vistas até aqui, esta função recebe o caminho e nome do arquivo a ser excluído, e não um ponteiro para FILE.
- Como retorno temos um valor inteiro, o qual será igual a 0 se o arquivo for excluído com sucesso.