UFABC

Leitura de Arquivos

Prof. Fabrício Olivetti de França

Leitura de Arquivos

Na linguagem C temos um tipo especial chamado **FILE** para trabalhar com arquivos.

UFABC

Abrindo e fechando um arquivo

Para abrir um arquivo utilizamos:

FILE * fopen(const char * filename, const char * mode);

Retorna NULL em caso de erro.

UFABC

Abrindo e fechando um arquivo

mode representa o modo de abertura do arquivo:

r - somente leitura

w - somente escrita, criando um arquivo novo e zerando os existentes

a - somente escrita, iniciando a partir do final do arquivo

UFABC

Abrindo e fechando um arquivo

mode também permite modificadores:

+ - abre para leitura E escrita

b - indica que a leitura/escrita é em arquivos binários

UFABC

Abrindo e fechando um arquivo

fopen(“arquivo”, “r+”);

Abre o arquivo para leitura e escrita, apontando para o início dele.

UFABC

Abrindo e fechando um arquivo

fopen(“arquivo”, “w+”);

Abre o arquivo para leitura e escrita, zerando o arquivo caso ele já exista.

UFABC

Abrindo e fechando um arquivo

fopen(“arquivo”, “a+”);

Abre o arquivo para leitura e escrita, iniciando do final do arquivo.

UFABC

Abrindo e fechando um arquivo

fopen(“arquivo”, “rb”);

Abre o arquivo para leitura no modo binário.

UFABC

Abrindo e fechando um arquivo

fopen(“arquivo”, “wb”);

Abre o arquivo para escrita no modo binário.

UFABC

Abrindo e fechando um arquivo

Após usar um arquivo, devemos fechá-lo com o comando:

int fclose(FILE *fp)

retorna 0 em caso de sucesso e EOF em caso de erro.

UFABC

Leitura de Arquivos Texto

A leitura do arquivo pode ser feita com as funções:

```
int fgetc( FILE * fp );
char *fgets( char *buf, int n, FILE *fp );
```

UFABC

Escrita de Arquivos Texto

A escrita do arquivo pode ser feita com as funções:

```
int fputc( int c, FILE * fp );
Int fputs( char *buf, FILE *fp );
```

UFABC

Caminhar dentro do arquivo

Para andar dentro do arquivo utilizamos a função:

int fseek(FILE *stream, long int offset, int whence)

offset é quantas posições a frente queremos andar

whence é **SEEK_SET**, **SEEK_CUR**, **SEEK_END**, para começo, posição atual e final de arquivo, respectivamente

UFABC

Leitura e Escrita em arquivos binários

```
size_t fread(void *ptr, size_t size, size_t number, FILE *f);

size_t fwrite(const void *ptr, size_t size, size_t number, FILE *f);
```

UFABC

Passagem de parâmetro do programa

A função **main** pode receber dois parâmetros:

int argc e **char ** argv**

argc indica o número de argumentos passados para o programa, no mínimo 1

argv é a lista de argumentos, em formato string, passadas ao programa

UFABC

Passagem de parâmetro do programa

Nota:

argv[0] é SEMPRE o nome do programa!

UFABC

getchar e putchar

int getchar (void); ← retorna um char (como um inteiro) capturado do teclado

int putchar (int); ← imprime um char na tela e retorna o mesmo char

UFABC

getchar e putchar

```
char c;
do {
    c = getchar();
    putchar(c);
} while (c != ‘\n’);
```

UFABC

gets e puts

char * gets (char *); ← captura uma string até que aperte enter.

int puts(char *); ← imprime uma string adicionando ‘\n’ no final.

UFABC

gets e puts

char * gets (char *); ← captura uma string até que aperte enter.

int puts(char *); ← imprime uma string adicionando ‘\n’ no final.

UFABC

gets e puts

```
char s[MAXLEN];

puts("Entre com um texto: ");
gets(s);
puts("Você digitou: ");
puts(s);
```

UFABC

gets: perigo!!

Defina MAXLEN como 5 e digite um texto maior que 5.

Segmentation fault! A função gets não limita a quantidade de dados de entrada.

Isso gera possibilidade de ataques do tipo **buffer overflow**.

UFABC

fgets e fputs

int fputs(const char *str, FILE *stream)

char *fgets(char *str, int n, FILE *stream)

Lê e escreve na entrada/saída de dados definido em stream e limita a quantidade de dados lido pelo parâmetro **n**

UFABC

fgets e fputs

```
char s[MAXLEN];

fputs("Entre com um texto: ", stdout);
fgets(s, MAXLEN, stdin);
fputs("Você digitou: ", stdout);
fputs(s, stdout);
putchar('\n');
```

UFABC

scanf e printf

int scanf(const char *format, ...) ← captura uma entrada formatada e armazena nas respectivas variáveis

int printf(const char *format, ...) ← imprime uma saída formatada

UFABC

scanf e printf

A string format é a formatação dos dados e pode conter:

%d, %u - número inteiro com ou sem sinal, respectivamente

%c - caractere

%f, %lf - float e double

%s - string

%o, %x - octal e hexadecimal

UFABC

scanf e printf

Também pode conter os caracteres especiais:

\n - próxima linha

\t - tabulação

\a - aviso sonoro

UFABC

Problemas com scanf

O scanf espera receber exatamente aquilo que foi especificado. Você não tem garantia que o usuário irá obedecer adequadamente!

Ao falhar, o scanf pode preencher o buffer com outras informações comprometendo toda a entrada de dados subsequente.

UFABC

Problemas com scanf

Para entradas do usuário é mais seguro utilizar a função fgets seguida de atoi, strtol ou strtok conforme necessário.

UFABC

atoi(), strtol(), strtok()

int atoi(const char *str) - converte uma string em número inteiro ou zero quando não for possível converter

long int strtol(const char *str, char **endptr, int base) - converte a parte inicial da string que contenha um inteiro em um int na base “base”. endptr aponta para o primeiro caractere não numérico da string.

UFABC

atoi(), strtol(), strtok()

char *strtok(char *str, const char *delim) - quebra a string em substrings separadas por **delim**.

Se o primeiro parâmetro for NULL ele retorna o ponteiro para a próxima substring.

UFABC

atoi(), strtol(), strtok()

Para float e double temos também: **atof** e **strtod**

Nota:

ato* e strtok se encontram na biblioteca <string.h>

strto* se encontram na biblioteca <stdlib.h>

UFABC

Exemplo 1

```
int main (int argc, char **argv )
{
    if (argc < 2) {
        printf("Uso: %s <nome do arquivo>\n", argv[0]);
        return -1;
    }
    cat(argv[1]);
    return 0;
}
```

UFABC

Exemplo 1

```
#define MAXLEN 5000

void cat (char *name)
{
    FILE * fp;
    char buf[MAXLEN];

    fp = fopen(name, "r");

    while ( fgets(buf,MAXLEN,fp) ) {
        fputs(buf, stdout);
    }
}
```

UFABC

Exemplo 2

```
#include <stdio.h>
#include <string.h>

int main ( )
{
    FILE * fp;
    char string[100] = "Ola Mundo!";

    fp = fopen("arquivo.bin", "wb");
    fwrite(string, sizeof(char), strlen(string), fp);

    return 0;
}
```

UFABC