Laboratório de programação
Arquivos de acesso Sequencial

Universidade Estadual Vale do Acaraú – UVA

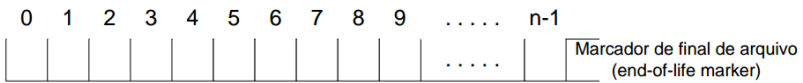
Paulo Regis Menezes Sousa

paulo_regis@uvanet.br

Arquivos e Streams

Arquivos de acesso Sequencial

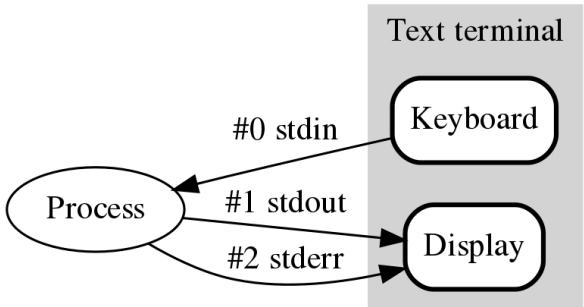
- O armazenamento de dados em variáveis e arrays é **temporário**; todos os dados são perdidos quando um programa termina.
- Os arquivos são usados para conservação **permanente** de grandes quantidades de dados.
- A linguagem C visualiza cada arquivo simplesmente como um fluxo de bytes (*streams*)



- Streams fornecem *canais de comunicação* entre arquivos e programas.
- Existem dois tipos de streams: texto e binária.

- Em C, um *arquivo* pode ser qualquer coisa, desde um arquivo em disco até um terminal ou uma impressora.
- Você associa um *stream* com um *arquivo* específico realizando uma operação de abertura.
- Um arquivo é desassociado de uma stream específica pode meio de uma operação de fechamento.
- Cada stream associada a um arquivo tem uma estrutura de controle de arquivo do tipo `FILE` definida em <`stdio.h`>) que contém informações usadas para processar o arquivo.

- Três arquivos e seus respectivos fluxos são abertos automaticamente quando a execução de um programa se inicia:
  - o de entrada padrão (`stdin`),
  - o de saída padrão (`stdout`) e
  - o de erro (ou exibição de erros) padrão (`stderr`).
- A linguagem C não impõe estrutura a um arquivo.
- O programador deve fornecer qualquer estrutura de arquivo para satisfazer as exigências de uma aplicação específica.


Text terminal
Keyboard
#0 stdin
Process
#1 stdout
Display
#2 stderr

- Função para abertura de um arquivo (retorna um ponteiro para o arquivo caso o arquivo seja aberto com sucesso, NULL caso contrário):

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

- Função de fechamento de um arquivo (retorna 1 caso o arquivo seja fechado com sucesso, 0 caso contrário):

```
int fclose(FILE *stream);
```

- Função para verificação de fim de arquivo (retorna 1 quando o fim do arquivo é encontrado, 0 caso contrário):

```
int feof(FILE *stream);
```

- Modos de abertura de um arquivo.

r	abre um arquivo existente para leitura (read)
w	abre um arquivo para escrita (write). Se o arquivo já existe, seu conteúdo é descartado. Senão, um novo arquivo vazio é criado
a	abre um arquivo para concatenação (append). Se o arquivo já existe, seu conteúdo é preservado e as escritas serão concatenadas no final do arquivo. Senão, um novo arquivo vazio é criado
r+	abre um arquivo existente para leitura e escrita. O conteúdo anterior do arquivo é preservado e o ponteiro é posicionado no início do arquivo
w+	abre um arquivo para leitura e escrita. Se o arquivo já existe, seu conteúdo é descartado. Senão, um novo arquivo vazio é criado
a+	abre um arquivo para escrita e concatenação. Se o arquivo já existe, seu conteúdo é preservado e as escritas serão concatenadas no final do arquivo. Senão, um novo arquivo vazio é criado. O ponteiro de leitura é posicionado no início do arquivo; as escritas são efetuadas no seu final

- Abertura em modo texto:

```
1                    FILE *arquivo;
2                    arquivo = fopen("arquivo.txt", "r");
```

- Fechamento do fluxo de dados para o arquivo

```
1                    ...
2                    fclose(arquivo);
```

- As funções de leitura e gravação `scanf` e `printf` tem versões para leitura e gravação em arquivos de texto.
- Leitura a partir de um arquivo de texto.

```
1                    fscanf(arquivo, "%d", &numero);
```

- Escrita em um arquivo de texto.

```
1                    fprintf(arquivo, "%d", numero);
```

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

int main() {
    int num, i;
    FILE *f = NULL;

    f = fopen("numeros_pares.txt", "w");

    if (f != NULL) {
        for (i=0, num=2; i<100; i++, num += 2)
            fprintf(f, "%d\n", num);

        fclose(f);
    }

    return 0;
}
```

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

int main() {
    int num;
    FILE *f = fopen("numeros_pares.txt", "r");

    if (f != NULL) {
        while (!feof(f)) { // enquanto não atingir o fim do arquivo
            fscanf(f, "%d", &num);
            printf("%d ", num);
        }
        printf("\n");
        fclose(f);
    }

    return 0;
}
```

```c
#include <stdio.h>
int main() {
    int conta;
    char nome[30];
    float saldo;
    FILE *f = fopen("clientes.dat", "w");
    if (f != NULL) {
        printf("Digite a conta, o nome e o saldo.\n");
        printf("Digite EOF para encerrar a entrada de dados.\n");
        while (!feof(stdin)) {
            if (scanf("%d%s%f", &conta, nome, &saldo) == 3) {
                fprintf(f, "%d %s %.2f\n",conta, nome, saldo);
            }
        }
        fclose(f);
    }
    return 0;
}
```

- Em ambientes Linux e MacOS, você pode encerrar a entrada padrão emitindo `EOF` (fim do arquivo) usando o atalho de teclado `CTRL-D`.
- No prompt de comando do Windows ou no Powershell, o equivalente a `CTRL-D` requer uma sequência de duas etapas: `CTRL-Z` seguido de `<return>`

```c
#include <stdio.h>

int main(){
    int conta;
    char nome[30];
    float saldo;
    FILE *f = fopen("clientes.dat", "r");
    if (f != NULL) {
        printf("Conta      Nome            Saldo\n");
        while (!feof(f)) {
            if (fscanf(f, "%d%s%f", &conta, nome, &saldo) == 3)
                printf("%-10d%-13s%7.2f\n", conta, nome, saldo);
        }
        fclose(f);
    }
    return 0;
}
```

- Para recuperar dados sequencialmente de um arquivo, normalmente um programa começa a leitura no início do arquivo e lê todos os dados consecutivamente até chegar ao dado procurado.
- Pode ser desejável processar os dados sequencialmente várias vezes (desde o início do arquivo) durante a execução de um programa. Uma instrução como:

```
rewind(cfPtr);
```

faz com que o ponteiro de posição do arquivo do programa seja reposicionado no início do arquivo.

```c
#include <stdio.h>

int main() {
  int opcao, conta;
  float saldo;
  char nome[30];
  FILE *f = fopen("clientes.dat", "r");

  if (f != NULL) {
    printf("Opções:\n"
           " 1 -- Lista contas com saldo zero\n"
           " 2 -- Lista contas com saldo positivo\n"
           " 3 -- Lista contas com saldo negativo\n"
           " 4 -- Encerra o programa\n:");
    scanf("%d", &opcao);
```

```
16
17      while (opcao != 4) {
18        switch (opcao) {
19          case 1:
20            while (!feof(f)) {
21              if (fscanf(f, "%d%s%f",&conta, nome, &saldo) == 3)
22                if (saldo == 0)
23                  printf("%-10d%-13s%7.2f \n", conta, nome, saldo);
24            }
25            break;
26          case 2:
27            while (!feof(f)) {
28              if (fscanf(f, "%d%s%f",&conta, nome, &saldo) == 3)
29                if (saldo < 0)
30                  printf("%-10d%-13s%7.2f\n", conta, nome, saldo);
31            }
32            break;
```

```
33            case 3:
34              while (!feof(f)) {
35                if (fscanf(f, "%d%s%f",&conta, nome, &saldo) == 3)
36                  if (saldo > 0)
37                    printf("%-10d%-13s%7.2f\n", conta, nome, saldo);
38              }
39              break;
40          }
41          rewind(f);
42          printf("\n:");
43          scanf("%d", &opcao);
44        }
45        fclose(f);
46    }
47    return 0;
48  }
```

Exercício 33

Escreva uma função que receba um vetor inteiro e seu tamanho como parâmetros e grave os números em um arquivo texto de nome "`vetor.txt`". Cada valor do vetor deve ser salvo em uma linha do arquivo.

Exercício 34

Escreva um programa que leia do usuário o nome de um arquivo texto. Em seguida, mostre na tela quantas linhas esse arquivo possui (link para o arquivo `name_age.txt`).

Exercício 35

Escreva um programa para converter o conteúdo de um arquivo texto em caracteres maiúsculos. O programa deverá ler do usuário o nome do arquivo a ser convertido e o nome do arquivo a ser salvo (link para o arquivo `lero-lero.txt`).

Exercício 36

Faça um programa que leia o arquivo texto `lista_compras.txt` contendo uma lista de compras. Cada linha do arquivo possui nome, quantidade e valor unitário do produto. O programa então exibe o total da compra.

Exercício 37

Escreva uma função que receba como parâmetro o nome de um arquivo de texto e retorne quantas vogais esse arquivo possui (link para o arquivo `lero-lero.txt`).

Exercício 38

Crie um tipo `Pessoa` para armazenar o nome e a idade de uma pessoa. Agora carregue em um vetor de 10 pessoas a partir dos dados contidos no arquivo `name_age.txt`. Você deve ordenar o vetor do tipo `Pessoa` pela idade das pessoas, da mais nova para a mais velha, depois gravar em um novo arquivo chamado `name_age_sorted.txt` os dados ordenados.