Arquitetura de Computadores

MiEI – 2016/17

DI-FCT/UNL

Aula 12

Biblioteca C

- Canais standard (abertos no inicio):
  - stdin, stdout, stderr
  - (em Java: System.in, System.out, System.err)
- Abrir/fechar:
  - FILE *fopen(char*name, char*mode);
  - fclose(FILE*stream);
- Escrever/Ler:
  - printf() / fprintf() / fwrite()
  - scanf() / fscanf() / fread()

Helloworld.c

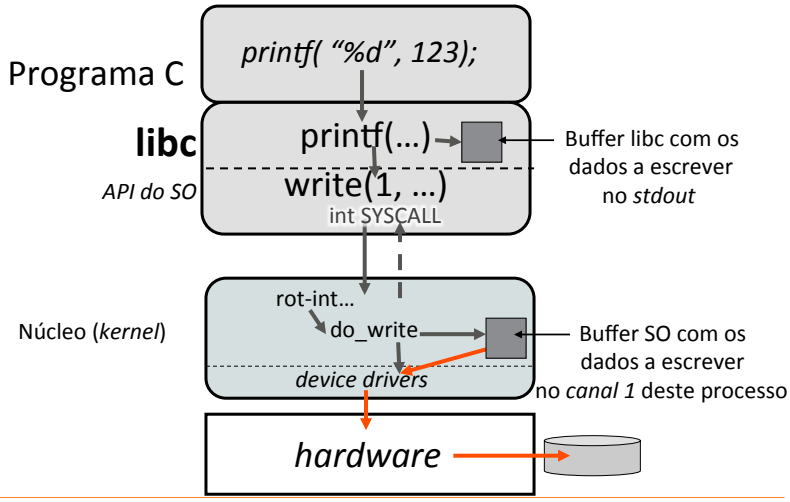
```
#include <stdio.h>
int main( int argc, char*argv[]) {
   printf(”Hello world!\n”);
   return 0;
}
```

Dentro da libc:

```
printf(...)
     fprintf( stdout, ... )
          write( 1, ... )  // interface
               int 0x80    // com SO
```


Vários níveis no IO - exemplo

- Níveis e Buffers (exemplo de escrita):



Ficheiro

- Um ficheiro é um espaço de logicamente contíguo, contendo um conjunto de dados acessível através de um **identificador único** (nome ou caminho)
- A informação é normalmente guardada de forma **persistente**.
- Um ficheiro pode conter qualquer **sequencia de bytes**, sejam dados (texto, imagem, som, etc.) ou programas.
- O SO encarrega-se de gerir os ficheiros e os periféricos em que estes residem. A organização interna do periférico é escondida dos programas, programadores e utilizadores.

Atributos de um ficheiro (meta-dados)

Mantidos pelo SO:

- **Nome** – identificador em formato legível para o utilizador
- **Tamanho** – tamanho corrente do ficheiro
- **Dono e Protecção** – quem é o dono e quem pode ler, criar, alterar, apagar, executar, …
- **Data, hora**– registo da criação, alteração e utilização
- **Localização** – informação sobre a localização física do ficheiro no respectivo suporte

Tipos de ficheiros em C

- **Ficheiros de texto**
  - Contêm apenas bytes que podem ser interpretados como caracteres imprimíveis: letras maiúsculas e minúsculas, algarismos e sinais de pontuação.
  - A codificação usada é a ASCII ou uma derivada desta.
- **Ficheiros binários**
  - A interpretação do seu conteúdo está a cargo do programa que o utiliza; contém normalmente bytes não imprimíveis.
  - Podem ser dados (inteiros, reais), programas (códigos máquina de um dado CPU), sons (segundo uma codificação como o MP3), imagens (codificadas num formato como o JPEG), filmes (no formato MPEG, por exemplo), etc.

Exemplo – escrever texto

```
#include <stdio.h>

int main(int ac, char*av[])
{    float x = 3.14;
     FILE*f = fopen("fich.txt", "w");
     fprintf(f, "Hello world!\n");
     fprintf(f, "%f", x);
     fclose(f);
     return 0;
}
```

Exemplo – lêr texto

```
#include <stdio.h>
#define SZ  100
int main(int ac, char*av[])
{    char line[SZ];  float x;
     FILE*f = fopen("fich.txt", "r");
     fgets( line, SZ, f );
     fscanf(f, "%f", &x);
     fclose(f);
     printf("%s%f\n", line, x);
     return 0;
}
```


Exemplo – escrever bytes genéricos

```
#include <stdio.h>

int main(int ac, char*av[])
{    float arr[6] = {1.2, 2.3, 3.4,
4.5, 5.6, 6.7};
     FILE*f = fopen("fich", "w");
     fwrite(arr, sizeof(float), 6, f );
     fclose(f);
     return 0;
}
```

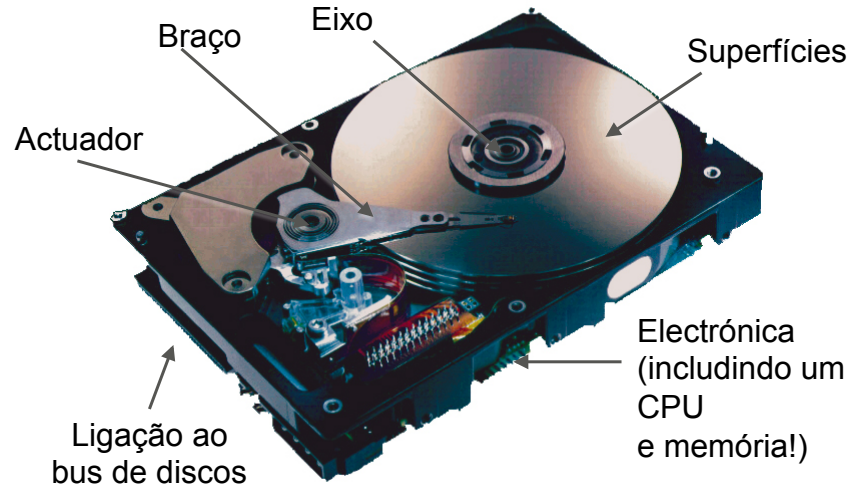
Exemplo – lêr bytes genéricos

```
#include <stdio.h>

int main(int ac, char*av[])
{    float arr[6] ;
     FILE*f = fopen("fich", "r");
     fread(arr, sizeof(float), 6, f );
     fclose(f);
     for (int i=0; i<6; i++)
          printf("%f ", arr[i]);
     return 0;
}
```

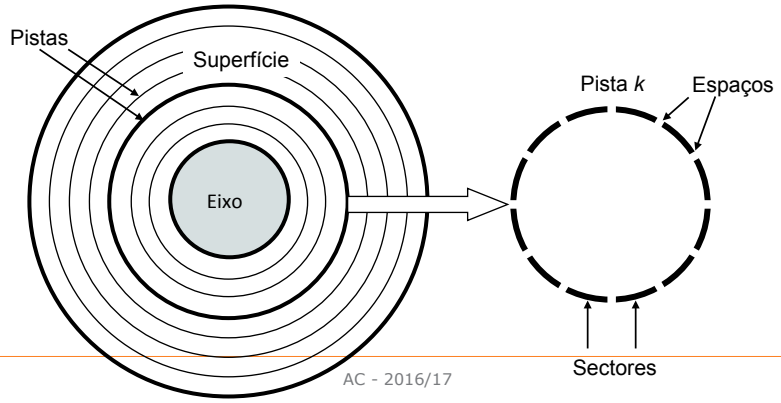

- Como o SO garante a persistência da informação?
  - Usa uma dispositivo que tenha *memória persistente*: discos, memória de estado sólido, ...
- Como escreve e lê os dados e os meta-dados?
  - Mantém estruturas de dados nessas *memórias* com os dados e meta-dados (ex. Discos formatados)

O que há dentro de um disco magnético?

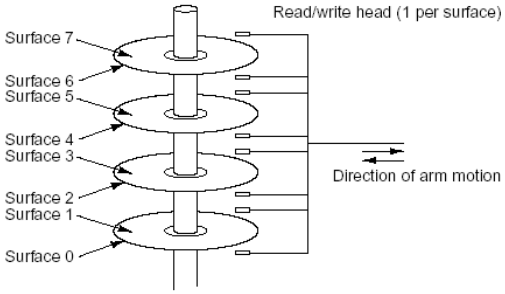


Geometria de um disco

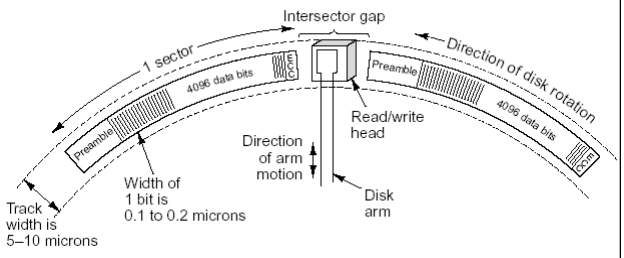
- Os discos são compostos por pratos, cada com duas superfícies.
- Cada pista contém aneis concêntricos chamados pistas.
- Cada pista é constituída por sectores separados por espaços.



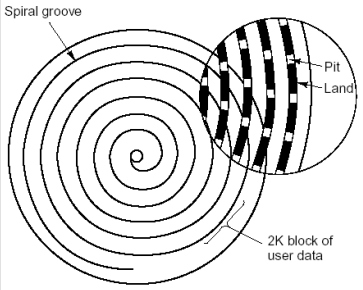
Discos – periféricos orientados ao bloco



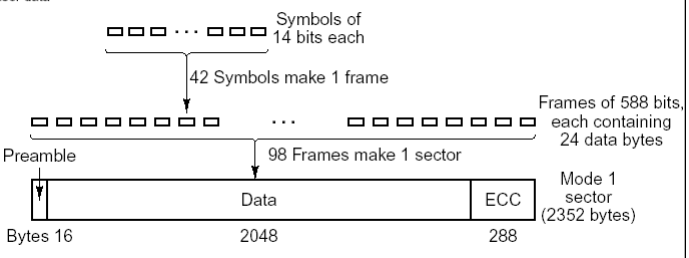
- A unidade de transferência é o sector (512bytes, 1K, etc)



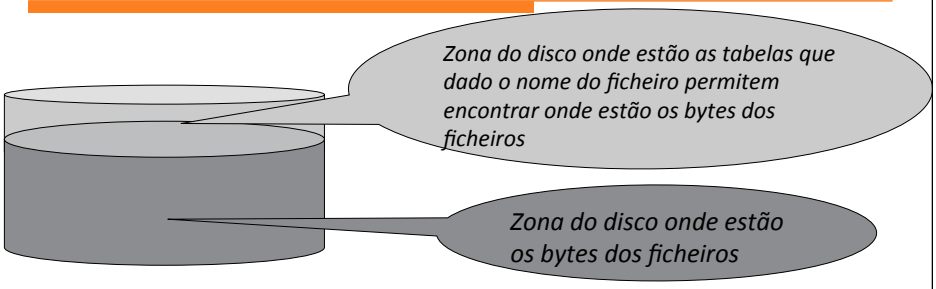
Exemplo de CD e DVD – orientados ao bloco



- A unidade de transferência é o sector de 2K



SO como gestor de recursos



- SO grava os dados, meta-dados e informação de gestão em sectores do disco.
- Um programa utilizador não pode ter livre acesso ao disco; há que garantir a partilha sem corromper os meta-dados ou os dados; também é mais conveniente para o programador.

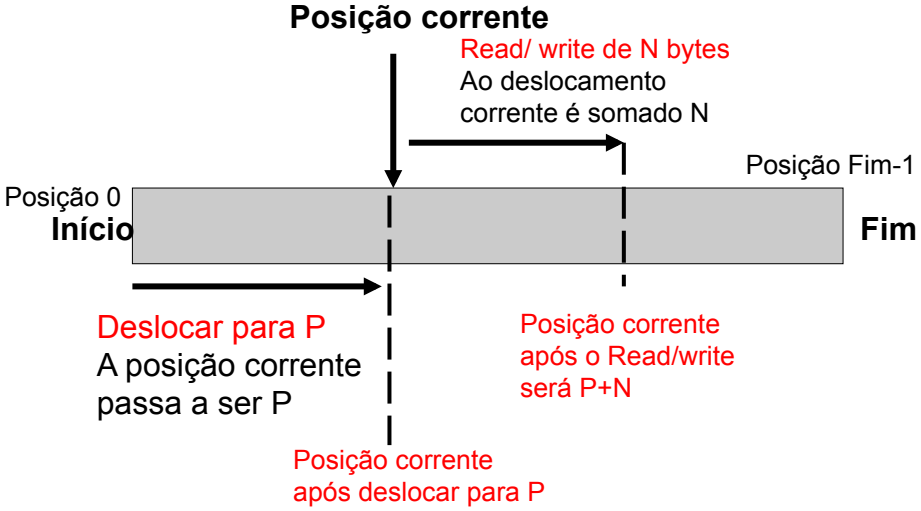
Abertura e fecho de um ficheiro

- Open (identificação do ficheiro, leitura/escrita)
  - O SO é informado que vão ser feitas leituras ou escritas no ficheiro
  - O SO cria estruturas de dados em memória e traz do disco para a RAM os meta-dados do ficheiro
  - Isto permite aceder aos dados de forma eficiente
- Close (Identificação do ficheiro)
  - O SO é informado que não vai haver mais leituras ou escritas no ficheiro
  - Já não é precisa a informação sobre o ficheiro que está em RAM. Pode escrever o que falta e libertar memória.

Métodos de acesso

- Existe sempre uma posição corrente (deslocamento); se o ficheiro tem N bytes, essa posição pode ir de 0 a N-1
- Quando se abre um ficheiro, a posição no ficheiro é 0
- **Acesso Sequencial**: A leitura ou escrita de B bytes, soma B à posição corrente
- Os bytes na zona da posição corrente são mantidos num buffer em memória (janela para o ficheiro em disco)
- **Acesso Direto**: Por alteração da posição corrente
  - Pode existir uma operação que permita mudar a posição corrente sem ler ou escrever

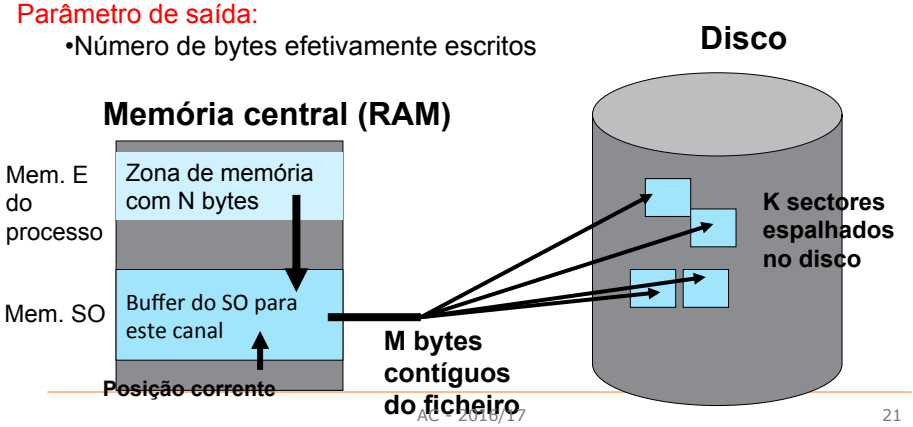
Acesso ao ficheiro



Escrita no ficheiro

Parâmetros de entrada:
- Identificação do ficheiro que passa a conter os dados escritos
- E: Endereço de memória de onde se escreve
- N: Número de bytes a escrever

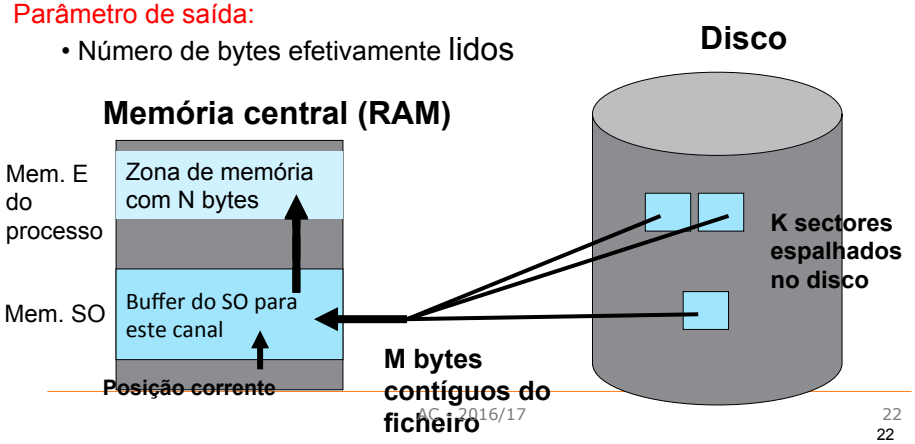
Parâmetro de saída:
- Número de bytes efetivamente escritos



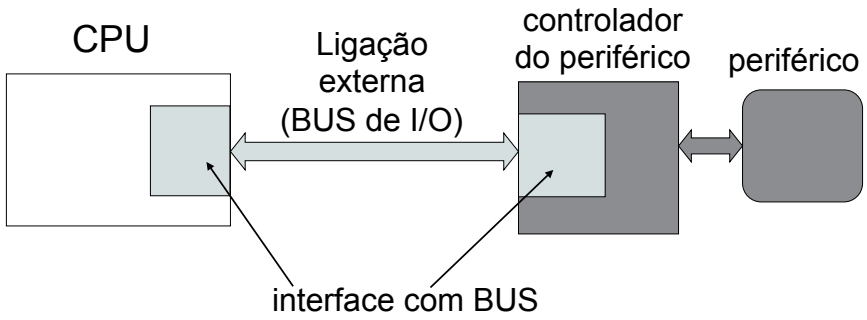
Leitura do ficheiro

Parâmetros de entrada:
- Identificação do ficheiro que contém a informação que se quer ler
- E: Endereço de memória para onde se lê
- N: Número de bytes a ler

Parâmetro de saída:
- Número de bytes efetivamente lidos



Interligação CPU-Periférico

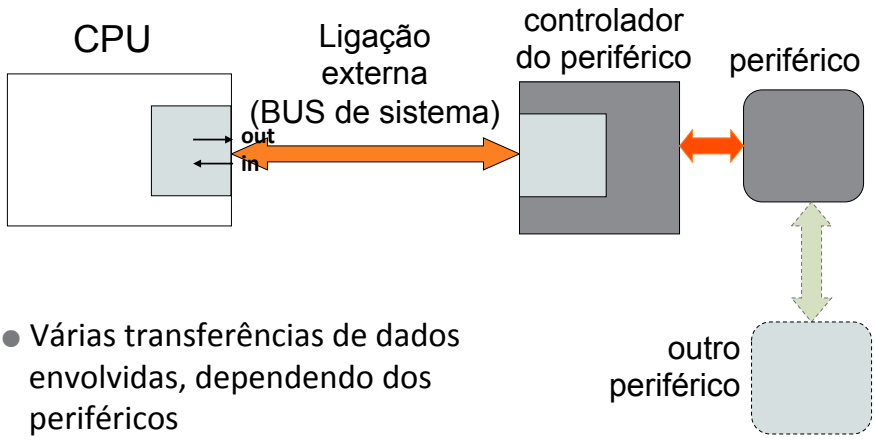


- Para transferir dados entre o CPU e o periférico
  - Quem inicia a transferência?
  - Que protocolos (nível eléctrico, ritmo, ...)?
  - Que quantidade de dados transferir?

Visão do CPU em relação às entradas/saídas

- O CPU não acede diretamente aos periféricos
- Usa uma interface programável para passar pedidos ao controlador de interface e para receber as repostas a esses pedidos
- Os programas só podem controlar o I/O usando o bus de sistema para "falar" com o controlador
- Este é que interage diretamente com o dispositivo físico periférico ou com outros dispositivos via interfaces / bus próprios e especializados

Transferências nas Entradas/Saídas



- Várias transferências de dados envolvidas, dependendo dos periféricos