Frequência II

- Gestão de Memória
  - Endereçamento Real (ER)
  - Endereçamento Virtual (EV)
    - Segmentação
    - Paginação
    - Proteção de Memória
  - Proteção
- Algoritmos de Gestão de Memória
  - Algoritmos de Reserva de Memória
  - Algoritmos de Transferência de Memória
    - Transferência de Segmentos
    - Transferência de Páginas
  - Algoritmos de Substituição de Memória
    - Algoritmos de *Swapping* de Segmentos
    - Algoritmos de *Swapping* de Páginas
- Comunicação entre Processos
  - Comunicação entre Processos
    - Inter Process Communication, IPC
    - Modelo do Canal de Comunicação
    - Caraterísticas do Canal de Comunicação
    - Ligação
    - Sincronização
    - Estrutura da Informação Trocada
    - Direcionalidade da Comunicação
  - Implementação do Canal de Comunicação
- Entradas e Saídas
  - Modelo Conceptual
    - Objetivos
    - Arquitetura
    - Modelo das I/O
  - Modelo Computacional
    - Modelo de Programação das I/O
    - Partilha de Periféricos
  - Programação de Gestores de Periféricos
    - Arquitetura de Comunicação
    - Arquitetura de I/O
    - Input/Output Request Block (IORB)
    - Funções do Gestor de Periféricos

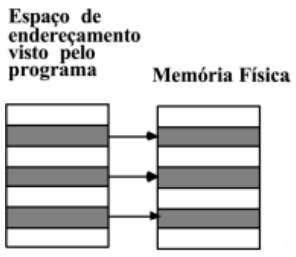
        Programação de um Gestor de Periféricos
        Modos de transferência de dados
    Implementação de GP
        Gestores de Periféricos dentro ou fora do *kernel*
Sistemas de Ficheiros
    Memória Persistente
        Discos Magnéticos
        Otimização dos Acessos ao Disco Magnético
    Sistema de Ficheiros
        Ficheiro
        Diretório
        Organização do Nome dos Ficheiros
        Tipos de Ficheiros
        Proteção
        Relação entre Sistema de Ficheiros e I/O
    Estrutura Interna
        File Allocation Table, FAT:
        Disco e Partição
        Acesso a Ficheiros

Gestão de Memória

> 💡 **Espaço de Endereçamento:** zona(s) de memória atribuída(s) a um processo e que este pode referenciar.

Endereçamento Real (ER)

> 💡 **Endereçamento Real:** endereço indicado no programa e é aqueçe que é acedido na memória principal.

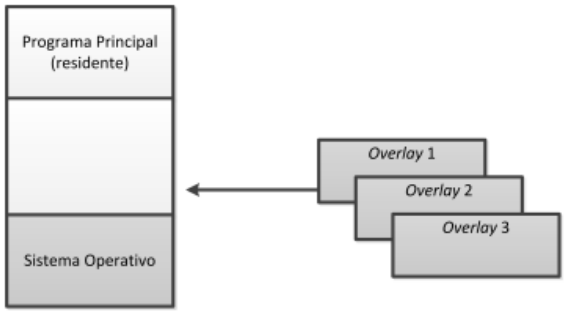


Limitações:

- A dimensão é limitada pela dinmensão da RAM do computador;
- Um programa só funciona nos endereços reais para onde foi gerado pelo compilador;
- A multiprogramação é dificultada, pois neste tipo de endereçamento não é possível executar simultaneamente programas que tivessem sido compilados para os mesmos endereços físicos.

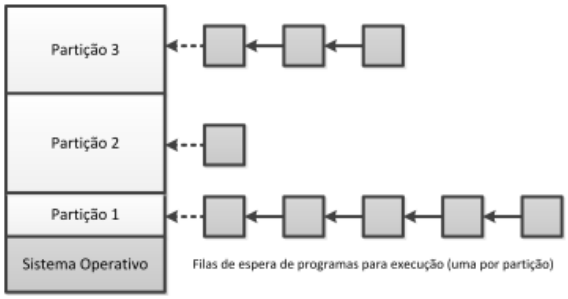
Sistemas Monoprogramados:

- Mecanismo de *Overlays*:
  - Possibilidade de executar programas com dimensão superior à RAM;
  - As *overlays* são dimensionadas para a memória física disponível;
  - O carregamento das *overlays* é explícito.



Sistemas Multiprogramados:

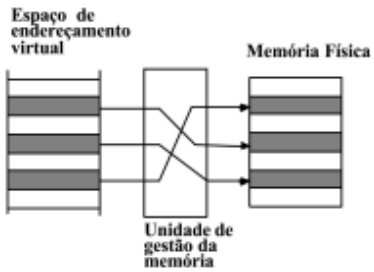
- Memória principal dividida em partições:
  - **fixas:** menor capacidade de recolocação; fragmentação interna; paragem da máquina para alterar dimensão das partições; utilização de *overlays*;
    - Memória dividida em partições de dimensão fixa, onde é carregado um programa (recolocável);
    - **Fragmentação interna:** dimensão dos programas não coincide exatamente com a dimensão das partições;



    - **variáveis:** maior capacidade de recolocação, fragmentação externa;
        - Alocação sucessiva de partições com dimensões variáveis;
        - Dimensão dos programas limitada pela memória física;
        - **Fragmentação externa:** muitas partições de dimensão muito reduzida;
        - **Recompactação** (*garbage collection*);
- Proteção através da utilização de registos `base` e `limite`;

Endereçamento Virtual (EV)

> 💡 **Endereçamento Virtual:** endereço indicado no programa é convertido em tempo de execução através da Unidade de Gestão de Memória (**MMU/UGM**).



- Espaço de endereçamento dos processos não linearmente relacionado com a RAM;
- Quando os endereços virtuais são utilizados, são convertidos pela MMU para endereços reais;

- A memória virtual é logicamente dividida em blocos contíguos:

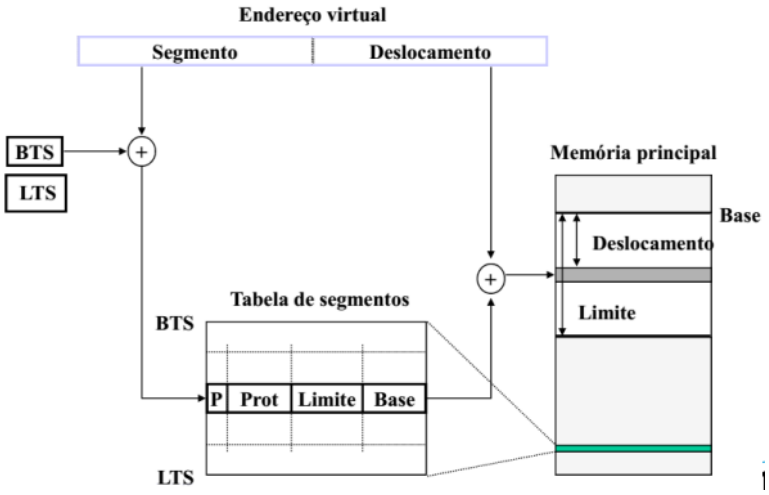
Endereço Virtual
(bloco, deslocamento)

- Na RAM estão apenas os blocos que estão a ser utilizados por cada processo;
- **Propriedade da Localidade:** a probabilidade de um programa se executar na mesma região é muito elevada, o que permite ter apenas uma parte reduzida do programa carregada em memória (simplifica a gestão de memória).

Segmentação

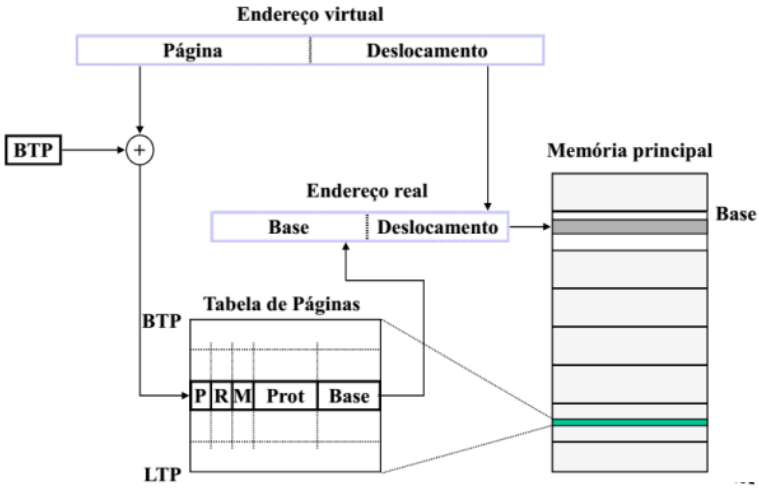
- Divisão dos programas em segmentos que refletem a sua estrutura funcional (funções, código, dados, *stack*);
- Tenta suportar diretamente na gestão de memórias as abstrações comuns nas linguagens de programação (carregamento em memória, proteção, eficiência);
- A dimensão dos segmentos é limitada pela arquitetura e não pode exceder a dimensão da RAM;
- **Fragmentação externa;**
- **Proteção:**
  - Verificação de limites de endereçamento intra-segmentos;
  - Verificação e limitação dos tipos de acesso ao segmento;
  - Processos diferentes têm tabelas de segmentos diferentes.
- **Partilha:**
  - Basta colocar na TS dos processos em questão o ER do segmento a partilhar;
  - Os EV usados para aceder ao segmento partilhado podem ser diferentes nos processos;
  - A proteção de um segmento partilhado é definida para cada processo através da TS;

- **Tamanhos variáveis.**



Paginação

- Na RAM são mantidas apenas algumas páginas do programa, sendo carregadas de memória secundária as demais quando ocorre uma **page fault**;
- A dimensão da páginas é constante, e é normalmente muito menor que a da memória principal, o que influencia:
  - **fragmentação interna**;
  - número de **page faults** e o tempo da sua resolução;
  - a dimensão das tabelas de páginas (**PTE**).



Proteção de Memória

- **Arquitetura Segmentada**:
  - Processos diferentes têm TS diferentes;
  - O número de segmento e o deslocamento são verificados, comparando-os respetivamente com a dimensão da tabela e com a dimensão do segmento;
  - Cada segmento tem associado um código de acesso (**R, W, X**);
- **Sistemas Paginados:**
  - Processos diferentes têm tabelas independentes;
  - O número de página é validado com o tamanho da tabela;
  - Cada página tem um código de acesso;
  - A granularidade mínima da proteção é a da página.

Proteção

- Para a partilha de memória virtua, basta que os descritores de segmento ou página em 2 processos independentes referenciem o mesmo endereço base;
- A proteção no acesso (**RWE**) nos 2 processos não têm de ser idêntica;
- Na paginação, a partilha tem que ter em conta a granularidade das páginas (apenas se partilham páginas ou múltiplos de páginas);
- Na segmentação, a entidade partilhada é uma divisão lógica do programa pelo que a partilha é natural.

Algoritmos de Gestão de Memória

- **Reserva:** onde colocar um bloco de dados;
- **Transferência:** quando transferir um bloco de memória secundária para RAM e vice-versa;
- **Substituição:** quando não existe mais memória livre, qual o bloco a retirar da RAM para satisfazer o pedido.

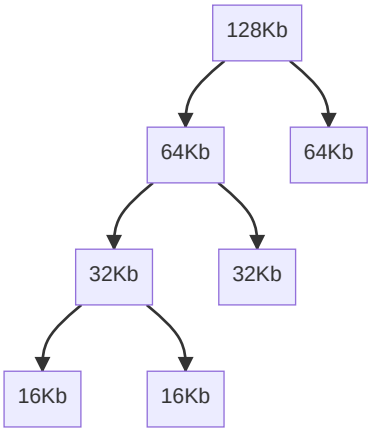
Algoritmos de Reserva de Memória

- Ocorre na **criação de processos** (reservado espaço para o código, dados e *stack* do programa) e na **extensão** do espaço de endereçamento;
- Reserva da **páginas**:
  - Basta reservar uma página entre todas as que estão livres;
- Reserva de **segmentos**:
  - Analisar uma lista de blocos livros para encontrar um com a dimensão igual ou superior à desejada;
  - Escolher o bloco livre que provocar menor fragmentação externa;
  - Reservar parte/totalidade do bloco escolhido tendo em conta uma dimensão mínima para o fragmento gerado (o que causa fragmentação interna mas reduz a lista de blocos livres);
  - **Critérios de Escolha de Blocos Livres:**
    - **Best Fit (o menor possível):** gera elevado número de pequenos fragmentos; em média, percorre-se metade da lista na procura;
    - **Worst Fit (o maior possível):** pode impossibilitar a reserva de blocos de grandes dimensões;
    - **First Fit (o primeiro possível):** minimiza o tempo gasto a percorrer a lista de blocos livres; gera muita fragmentação externa;
    - **Next Fit (o primeiro possível):** espalha os blocos pequenos por toda a memória;
    - **Buddy**:
      - A memória livre é dividida em blocos de dimensão b^n;
      - Para satisfazer um pedido de dimensão D percorre-se a lista à procura de um bloco de dimensão R tal

que $2^{k-1} < R <= 2^k$;

- Se não for encontrado, procura-se um bloco de dimensão $2^{k+1}, i > 0$, que será dividido em 2 partes iguais (*buddies*);
- Um dos *buddies* será subdividido quantas vezes for necessário até se obter um bloco de dimensão 2^k;

- Se possível, na libertação, um bloco é recombinado com o seu *buddy*, sendo a associação entre *buddies* repetida até se obter um bloco com a maior dimensão possível;
- Problemas de **fragmentação externa, fragmentação interna** e de **complexidade em reservar e libertar segmentos**.



Algoritmo Buddy, b=2

Algoritmos de Transferência de Memória

- A transferência pode ser feita em 3 situações:
  - **A Pedido**: o programa/SO determinam quando se deve carregar o bloco na RAM;
  - **Por Necessidade**: o bloco é acedido e gera-se uma *fault* (*page fault* ou *segment fault*), sendo necessário carregá-lo para a memória principal;

- **Por Antecipação:** o bloco é carregado na RAM pelo SO porque este considera provável que ele venha a ser acedido nos próximos instantes.

Transferência de Segmentos

- A transferência de segmentos faz-se a **pedido**;
- Um processo precisa de um **segmento de código, de dados e de *stack*** em memória;
- Caso haja escassez de memória, os processos são transferidos (*swapping*) na íntegra para o disco;
- Os segmentos são guardados numa zona separada do disco (*swap area)*;

Transferência de Páginas

- A transferência de páginas faz-se **por necessidade**:
  - Páginas de um programa que não sejam acedidas durante a execução de um processo não chegam a ser carregadas na RAM;
- Usam-se políticas de transferência **por antecipação**:
  - Diminuir o número de *page faults*;
  - Otimizar acessos a disco;
- As páginas retiradas da RAM são guardadas na **área de paginação**;
- As páginas modificadas são **transferidas em grupos** para o disco de modo a otimizar os acessos a disco;

Algoritmos de Substituição de Memória

Algoritmos de *Swapping* de Segmentos

- Critérios de decisão sobre qual o processo/segmento a transferir para disco:
  - **Dimensão dos segmentos**: os segmentos *swapped out* devem ter dimensão suficiente de modo que a memória livre seja

suficiente para acomodar os segmentos *swapped in*;

- **Estado e prioridade do processo**: processos bloqueados e pouco prioritários são candidatos preferenciais;
- **Tempo de permanência na RAM:** um processo tem que permanecer um determinado tempo a executar-se antes de ser novamente enviado para disco;

- Quando é necessário libertar espaço na RAM, o Sistema Operativo copia páginas para disco;

Algoritmos de *Swapping* de Páginas

- Manutenção de duas listas de páginas:
  - **livres**, que podem ser utilizadas a qualquer momento;
  - **livres modificadas**, que têm de ser passadas para disco antes de serem usadas;
- Quando o # de páginas livres é insuficiente, o processo paginador irá transferir as páginas livres para disco;
- Páginas livres modificadas são periodicamente escritas em disco, passando para a lista de páginas livres;
- O algoritmo de escolha das páginas a libertar segue o princípio
  da localidade de referência:
  - **Least Recently Used, LRU**:
    - Eficaz segundo o princípio acima;
    - Latência associada à sua implementação rigorosa;
    - Utilização de um contador por página que indique a que “grupo etário” ela pertence;
    - Atualizado regularmente pelo processo paginador;
    - Quando atingir um valor máximo, a página passa para as lista das páginas livres/livres modificadas;
  - **Not Recently Used, NRU**
    - Agrupamento das páginas em 4 grupos (**R**: referenciada; **M:** modificada):

        - **0 (R=0, M=0)**
        - 1 **(R=0, M=1)**
        - 2 **(R=1, M=0)**
        - 3 **(R=1, M=1)**
    - O **processo paginador percorre** as tabelas de páginas e coloca o bit **R** a 0;
    - Libertam-se primeiro as páginas dos grupos de número mais baixo;
- **First In First Out, FIFO**:
    - Eficiente mas não atende ao grau de utilização das páginas (apenas ao tempo de permanência em RAM);
- **Working Sets**:
    - É o conjunto de páginas acedidas pelo processo num intervalo de tempo:
        - Para intervalos de tempo razoáveis, o **working set de um processo mantém-se constante** e menor que o espaço de endereçamento;

Comparação: Segmentação vs. Paginação

Segmentação

- **Vantagens**:
    - Adapta-se à estrutura lógica dos programas;
    - Permite a realização de sistemas simples sobre hardware simples;
    - Permite realizar eficientemente operações que agem sobre segmentos inteiros.
- **Desvantagens**:
    - O programador tem de ter algum conhecimento dos segmentos subjacentes;
    - Os algoritmos de gestão de memória são complicados;

        - O tempo de transferência de segmentos entre RAM e disco torna-se incomportável para segmentos muito grandes;
        - A dimensão máxima dos segmentos é limitada.

Paginação

- **Vantagens**:
    - O programador não se preocupa com a gestão de memória;
    - Os algoritmos de gestão de memória são mais simples e eficientes;
    - O tempo de leitura de uma página de disco é razoalmente pequeno;
    - A dimensão dos programas é virtualmente ilimitada.
- **Desvantagens**:
    - O hardware é mais complexo que o da memória segmentada;
    - Operações sobre estruturas lógicas são mais complexas e menos elegantes;
    - O tratamento das *page faults* representa uma sobrecarga adicional de processamento.

Comunicação entre Processos

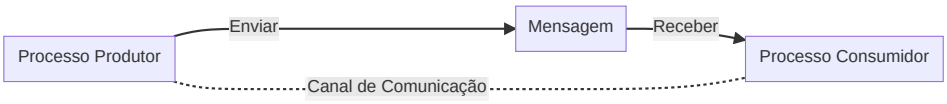
Comunicação entre Processos

- A sincronização entre processos permitiu a cooperação na execução de um algoritmo:
    - No entanto, a cooperação por vezes implica a transferência de informação;

Inter Process Communication, IPC

- Os processos precisam de se sincronizar e trocar dados:

    - A transferência de dados é suportada por um canal de comunicação disponibilizado pelo SO;
- Os processos que comunicam estabelecem a estrutura das mensagens trocadas:
    - O SO considera as mensagens como sequências de octetos;
    - Numa visão OSI, estes mecanismos correspondem às camadas de Transporte e Sessão;



Canal de Comunicação

Modelo do Canal de Comunicação

- Propriedades: `nome`, `estado` e `metadados`;
- Operações: `criar`, `terminar`, `eliminar` , `associar`, `enviar` e `receber`;

Caraterísticas do Canal de Comunicação

- Propriedades;
- Tipo de ligação;
- Capacidade de armazenamento;
- Sincronização (*unbuffered*, *buffered*, *client-server*);
- Segurança;
- Fiabilidade.

Ligação

- Antes de um canal de comunicação ser usado, um processo tem de:
    - Saber se o mesmo existe (**nome dos objetos de comunicação**), para o qual existem 2 soluções:
        - Dar nomes explícitos aos canais: o espaço de nomes é gerido pelo SO e pode assumir diversas formas;

        - Os processos terem implicitamente associado um canal de
          comunicação: o canal é implicitamente identificado usando os IDs dos processos;
    - Indicar, ao SO, que se pretende associar;

Sincronização

- Envio de mensagem:
    - **Assíncrona** (buffered): o cliente envia o pedido e continua a execução;
    - **Síncrona** (*unbuffered*): o cliente fica bloqueado até que o processo servidor leia a mensagem;
    - *Client-Server*: o cliente fica bloqeuadoa té que o servidor envie uma mensagem de resposta;
- Receção de mensagem:
    - Bloqueante na ausência de mensagens;
    - Testa se há mensagens e retorna;
- Capacidade de armazenamento de informação:
    - Um canal pode ou não armazenar várias mensagens;
    - Permite desacoplar os ritmos de produção e consumo de informação, tornando a sincronização flexível;

Estrutura da Informação Trocada

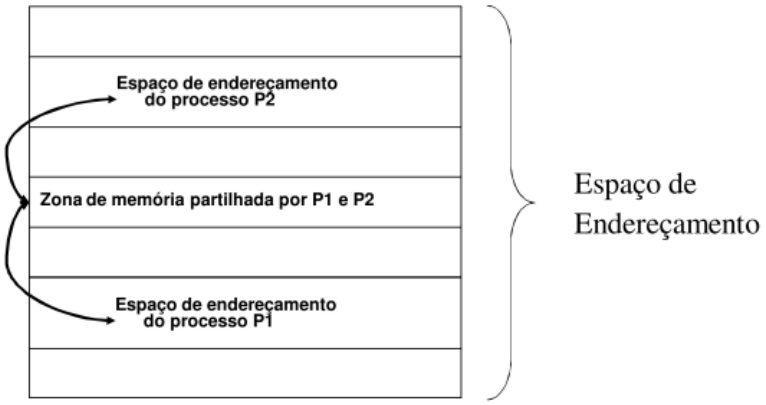
- Fronteiras das mensagens:
    - Mensagens individualizadas;
    - Sequências de octetos;

Direcionalidade da Comunicação

- Unidirecional: o canal apenas permite enviar informação num sentido que fica definido na sua criação (*e.g. pipes*);
- Bidirecional: o canal permite enviar mensagens nos dois sentidos (*e.g. sockets*)

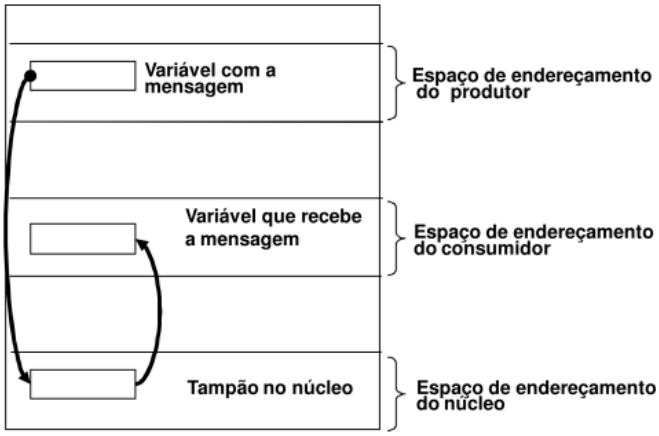
Implementação do Canal de Comunicação

- Define-se como são transferidos os dados entre os espaços de endereçamento dos processos;
- Pode ser implementado com dois tipos de mecanismos:
  - **Memória partilhada**:
    - Os processos acedem a uma zona de memória que faz parte do espaço de endereçamento dos processos comunicantes;
    - Pode ser considerada como extensão à gestão da memória virtual;
    - São necessários mecanismos de sincronização para:
      - Garantir *MutEx* sobre a zona partilhada;
      - Sincronizar a cooperação dos processos produtor e consumidor;
    - **Mecanismo mais eficiente; a sincronização tem de ser explicitamente programada (complexa);**



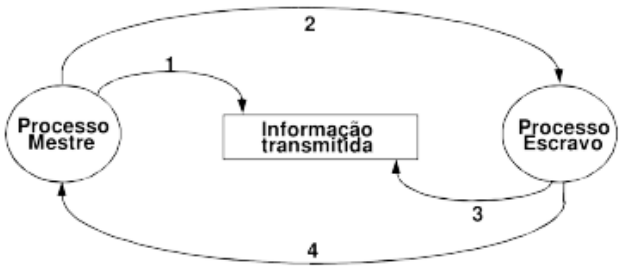
  - **Transferidos através do *kernel:***
    - Os dados são sempre copiados para o *kernel* antes de serem transferidos;
    - Semelhante à gestão de outros objetos do sistema (*e.g.* ficheiros);
    - **Velocidade de transferência limitada pelas duas cópias da informação e pelo uso das *syscalls* para**

`send` e `recv` ; sincronização implícita.

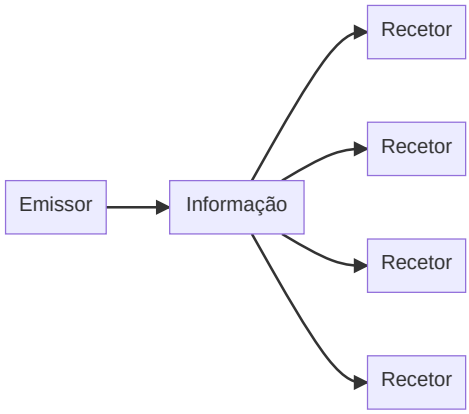


Modelos de Comunicação

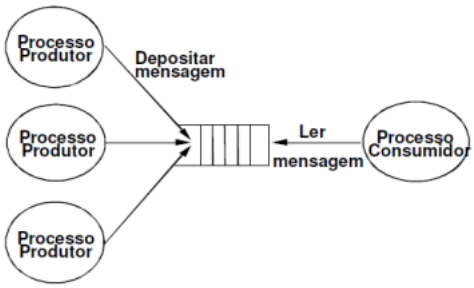
- *One-to-One* (***Master/Slave***): O processo consumidor (escravo) tem a sua ação controlada por um processo produtor (mestre);
  - O *master* não necessita de autorização para utilizar o *slave*;
  - A atividade do *slave* é controlada pelo *master*;



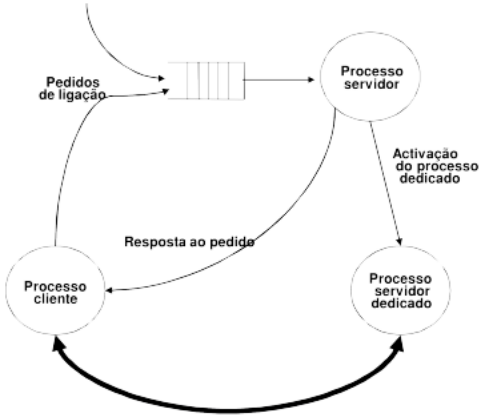
- *One-to-Many* (***Diffusion***): Envio da mesma informação a um conjunto de processos consumidores;



- *Many-to-One* (***Mailbox***): Transferência assíncrona de mensagens, de vários processos produtores, para um canal de comunicação associado a um processo consumidor;
  - Os emissores não controlam diretamente a atividade do(s) recetor(es);
  - Não existe ligação direta entre os processos;
  - A *mailbox* permite memorizar as mensagens quando estas são produzidas mais rapidamente do que consumidas;



- *One-to-One of Many* (***Dialogue***): Um processo que pretende interatuar com outro negoceia o estabelecimento de um canal dedicado, temporário, de comunicação entre ambos;
  - Estabelecido um canal de comunicação entre o cliente e servidor;
  - O servidor pode gerir váriso clientes, mas dedica a cada um deles uma atividade independente;



- *Many-to-Many*: Transferência assíncrona de mensagens, de vários processos produtores para um canal de comunicação associado a vários processos consumidores.

Entradas e Saídas

Modelo Conceptual

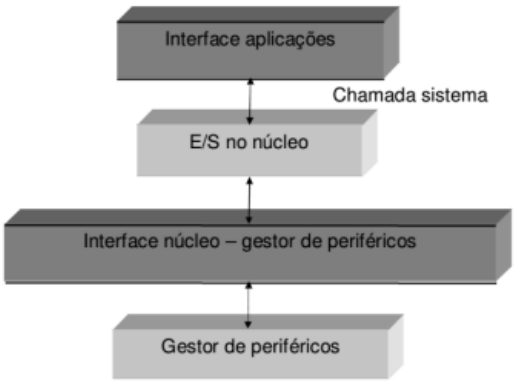
- Parte do SO que permite a um programa interagir com o ambiente que o rodeia;
- Cria canais virtuais entre os programas e os periféricos;
- Diferentes caraterísticas:
  - Operações aceites;
  - Unidade de transferência de informação;
  - Velocidade de transferência;
  - Representação de dados;
  - Ação em caso de erro;

Objetivos

- Estender o núcleo do SO permitindo a inclusão de periféricos;
- Modelo de programação de I/O uniforme, evitando que os programadores tenham de lidar com os detalhes dos diferentes periféricos;

- Modelos de programação do gestor de periféricos que simplifque a programação.

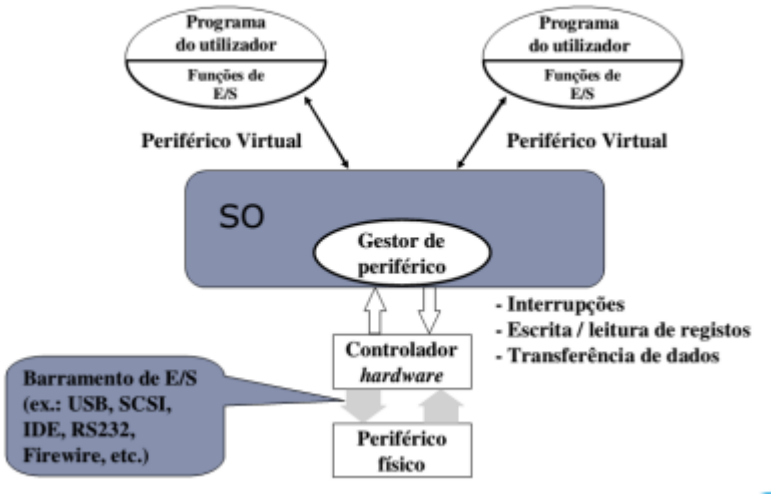
Arquitetura



Objetivos	Soluções
Uniformização da designação e do acesso	Periférico virtual
Independência do periférico do SO	Decomposição da interação com os periféricos
Redireção das I/O	Uniformização dos mecanismos de comunicação e I/O
Adaptação a novos periféricos	Carregamento dinâmico de módulos no núcleo

Modelo das I/O

- **Periféricos Virtuais**: Entidades abstratas sobre as quais se realizam todas as operações de I/O;
- **Funções de I/O:** Conjunto reduzido e uniforme de funções necessárias à interação com qualquer periférico;
- **Gestor de Periféricos (GP):** Efetua a interação real com o periférico.



Modelo Computacional

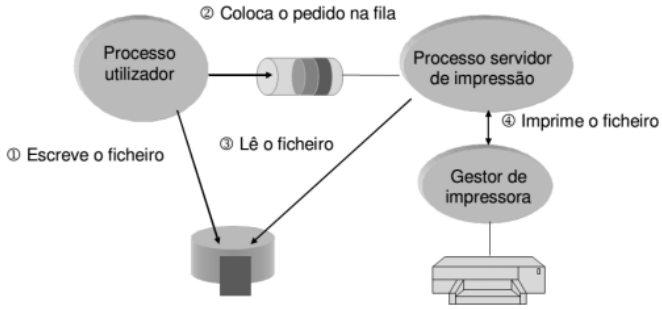
Modelo de Programação das I/O

- Um periférico virtual é idêntico a um canal de comunicação: são trocadas informações entre o processo utilziador e o gestor de periféricos;
- Associação ao periférico virtual: estabelecimento de um canal de comunicação entre o processo e o GP;
- Designação dos periféricos: varia consoante os sistemas (Windows: geridos por um gestor de nomes autónomo; UNIX: identificados como ficheiros);
- Suporte de vários tipos de sincronização associados à operação;
- Suporta dois tipos de transferência de dados:
  - *byte stream*;
  - bloco de dados de tamanho fixo;
- Sincronização:
  - **Escrita:** o processo cliente fica bloqeuado até que os dados tenham sido transferidos para o periférico ou para estruturas de dados internas ao SO;
  - **Leitura:** o processo cliente fica bloqueado até que os dados pedidos lhe tenha msido transferidos; há sistemas

que permitem evitar o bloqueio e efetuam leituras assíncronas.

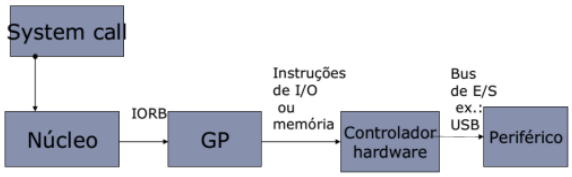
Partilha de Periféricos

- ***Spooling***: processo de transferência de dados colocando-os numa área de trabalho temporária onde outro programa pode acessá-lo para processá-lo num tempo futuro;

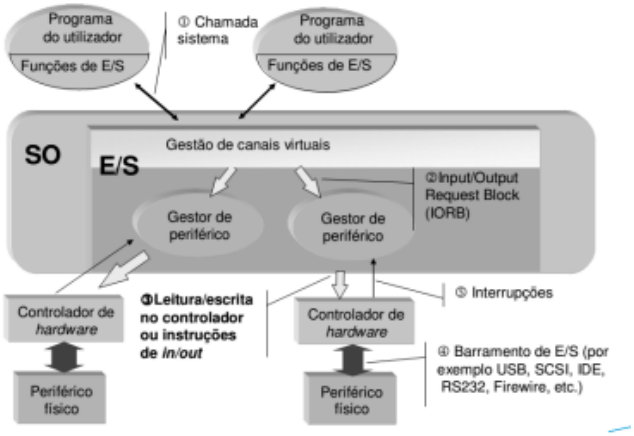


Programação de Gestores de Periféricos

Arquitetura de Comunicação



Arquitetura de I/O



Input/Output Request Block (IORB)

- Mensagem normalizada que transmite informação dos processos cliente para um GP:
  - O GP está bloqueado à espera dos IORB e, quando os recebe, desencadeia a operação pedida introduzindo a sequência de controlo adequada no controlador do periférico;
  - A partilhade um periférico pode realizar-se à custa uma *mailbox* onde o GP recebe os diversos IORB.
- Geradas pelas funções de I/O após validação da coerência dos parâmetros:
  - Validade do modo de operação;
  - Validade do modo de transmissão.

Funções do Gestor de Periféricos

- Esperar por um IORB;
- Validar os parâmetros;
- Programar o controlador do periférico de acordo com o IORB;
- Esperar pela conclusão da operação e analisar/processar o resultado da mesma:
  - A conclusão é assinalada por uma interrupção;
  - As rotinas de interrupção devem executar um conjunto reduzido de instruções no mais curto espaço de tempo possível;
- Tratamento de condições de erro;
- Transferir dados entre tampões próprios e o espaço de endereçamento do processo do utilizador;
- Alertar o processo utilizador do término da operação.

Programação de um Gestor de Periféricos

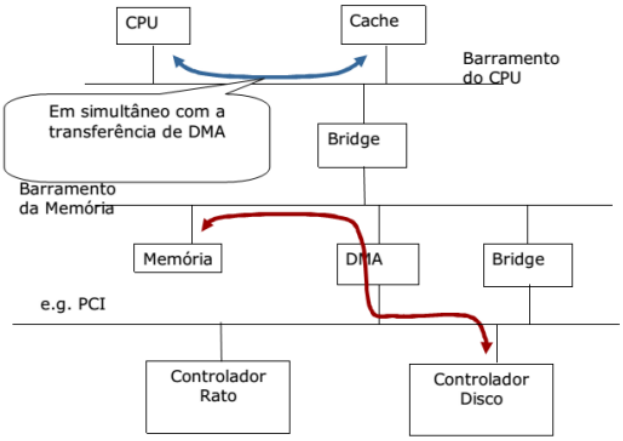
- Armazenamento temporário de dados: conjunto fixo ou variável de tampões, tendo em conta um fluxo médio

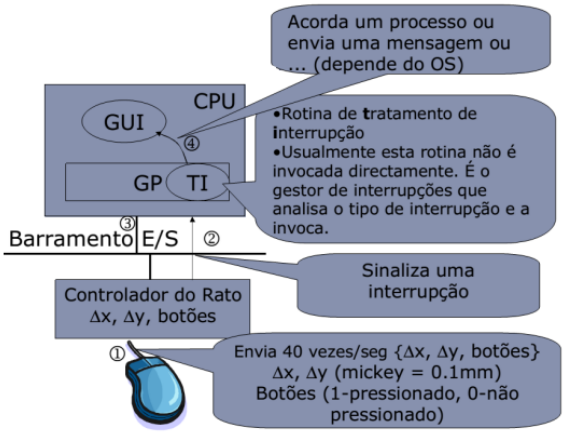
expectável ou a otimização de transferências;

- Interação com a gestão de memória: fixação de páginas dos processos utilizadores na RAM; utilização de ER em vez de EV;
- Rotina de interrupção: determinar a causa; executar as operações necessárias; assinalar ao gestor o respetivo acontecimento.

Modos de transferência de dados

- **I/O programáticas**: o CPU executa um conjunto de instruções do GP que escrevrem/lêem para/do periférico;
- **Direct Memory Access (DMA)**: unidade de hardware especializada que transfere blocos da memória para a memória do periférico; permite libertar o CPU para computação interna e com acesso à cache;
- **Processador de periféricos**: CPU especialziado que partilha a memória com o CPU principal egere o acesso aos controladores.

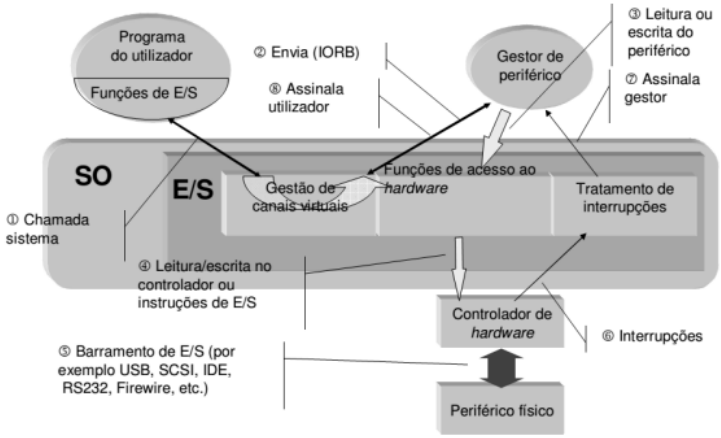




Implementação de GP

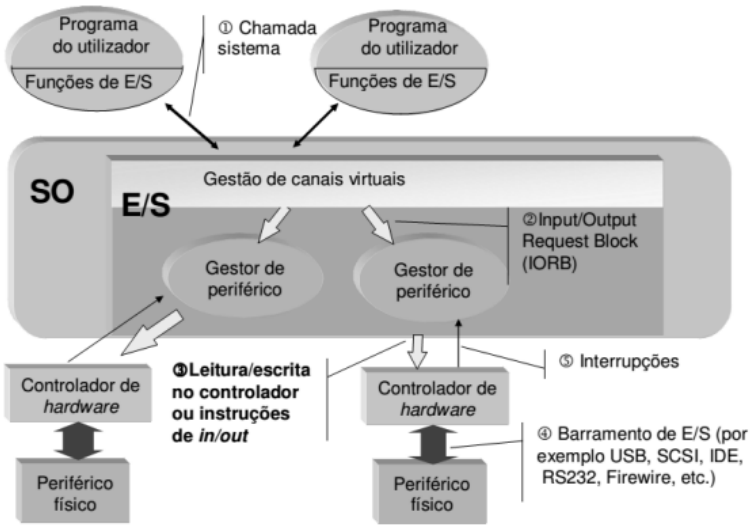
Gestores de Periféricos dentro ou fora do *kernel*

- **Processo independente**:
  - Maior flexibilidade;
  - Necessidade de privilégios especiais que permitam ultrapassar as proteções (alta prioridade, controlo de interrupções, gestão de memória);
  - Isolamento de espaços de endereçamento entre os gestores e o *kernel* do SO;



- **Integrado no núcleo**:
  - Caso mais comum;
  - Poupa tempo gasto na comutação entre processos;

- Maior simplicidade e redução das operações do *kernel*;



Sistemas de Ficheiros

Memória Persistente

- Dispositivos lógicos:
  - Subdivisão de um dispositivo físico em segmentos contíguos;
  - O sistema de ficheiros (SF) gere os acessos a estes.
  - Pode ser considerado um *vector* de blocos:
    - Dimensão múltipla da dos sectores (512B ou 1024B) e, em memória paginada, igual à das páginas;
    - A informação dos ficheiros é guardada em grupos de blocos (segmentos/*extents*);
    - A fragmentação é diminuída reaproveitando parte dos segmentos para outros ficheiros;
  - Superbloco: informação geral de descrição do SF do dispositivo lógico.

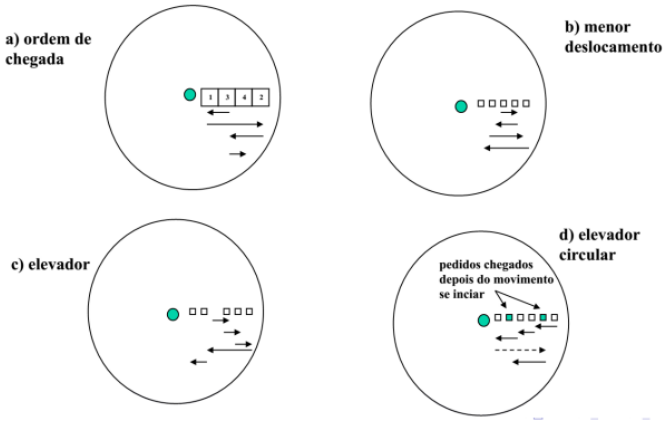
Discos Magnéticos

- Composição:

    - Conjunto de pratos sobrepostos;
    - Em cada face, a informação é escrita em pistas concêntricas, e cada pista é composta por setores;
    - O conjunto das pistas com o mesmo raio forma um cilindro;
- O tempo de leitura/escrita é composto por:
    - *Seek time*: tempo de deslocação das cabeças até ao cilindro desejado;
    - Tempo de latência: tempo de espera pelo setor ($t_{\text{médio}} = t_{\text{meia rotação}}$);
    - Tempo de transferência: tempo para transferir um setor entre o disco e RAM ($t_{\text{revolução}} / \#\,\text{sectores por pista}$);
- Tempo médio de acesso: dado pela soma dos tempos médios acima.

Otimização dos Acessos ao Disco Magnético

- Minimização da frequência de acesso: cache de blocos em RAM;
- Minimização do tempo de posicionamento:
    - Ordenação dos pedidos segundo a ordem mais favorável:
        - **Ordem de chegada:** simples, justo, não otimiza as operações mais demoradas (parar e mover cabeças);
        - **Menor deslocamento:** menor tempo de posicionamento, maior desempenho quando os pedidos estão relacionados, pode ser injusto para os cilindros nas extremidades pois estes são preteridos;
        - **Elevador:** análogo ao anterior mas aplica-se apenas aos pedidos situados no sentido do deslocamento das cabeças, visita menos os cilindros na periferia;
        - **Elevador circular:** cabeças leem apenas num sentido, quando não há mais pedidos ou chegam à extremidade do disco as cabeças deslocam-se para o cilindro mais distante para o qual haja pedidos.

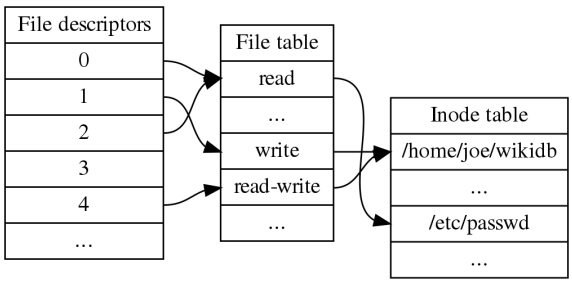


Sistema de Ficheiros

- Conjunto de ficheiros, diretórios, descritores e estruturas de dados auxiliares geridos por um módulo do SO;
- Permitem estruturar o armazenamento e a recuperação de dados persistentes em 1/+ disco;

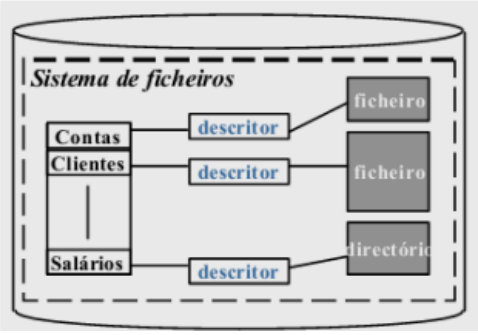
Ficheiro

- Conjunto de dados persistentes composto por **nome**, **descritor** (estrutura de dados em disco com informação sobre o ficheiro) e **informação** (dados guardados em disco).
- Visão global:
  - Um ficheiro é identificado pro um *pathname* podendo eventualmente existir diversos caminhos de acesso para o mesmo ficheiro;
  - Cada ficheiro tem associado um único *index node* que contém informação de localização e de gestão;
  - O sistema encarrega-se de a partir do *pathname* determinar o *index node* correspondente para todas as operações que envolvam ficheiros;



Diretório

- Catálogo de nomes de ficheiros (ou diretórios) que estabalece a associação entre os nomes e os seus descritores (*inodes*);
- Pode conter descritores ou apenas os seus identificadores;
- É composto por um nome, descritor e informação.



Organização do Nome dos Ficheiros

- Um nome por cada ficheiro num diretório único a nível da máquina;
- Um nome por cada ficheiro num diretório único para cada utilizador;
- Organização hierárquica:
  - Os ficheiros e diretorias vazias são nós terminais;
  - Nomes absolutos: caminho de acesso desde a raíz;
  - Nomes relativos: caminho de acesso desde o diretório corrente.
- Diretório corrente mantido para cada processo como parte do seu contexto.

Tipos de Ficheiros

- O tipo de um ficheiro depende do seu conteúdo e forma de acesso:
  - **Conteúdo:** conjunto de registos;
  - **Forma de acesso aos registos:**

- Sequencial:
  - Para se ler o registo N é necessário ler os $N-1$ registos anteriores;
  - Para alterar um registo é preciso ler o ficheiro todo e escrevê-lo de novo com o registo alterado.
- Direto:
  - Um registo pode ser acedido diretamente;
  - Não se pode inserir um novo registo entre outros dois.
- Por chave:
  - Os registos são identificados por chaves alfanuméricos reconhecidas pelo sistema de ficheiros.

Proteção

- Estabelecimento da proteção quando o ficheiro é criado:
  - A informação de proteção é guardada no descritor do ficheiro e está relacionada com o utilizador responsável pela sua criação;
  - O dono de um ficheiro pode ser alterado durante o tempo de vida de um ficheiro.
- Verificação dos direitos de acesso de um processo a um ficheiro pelas funções sistema do sistema de ficheiros;
- Especificação dos direitos de acesso a um ficheiro:
  - **Listas de acesso**;
  - Definição de **grupos de utilizadores e de direitos de acesso.**

Relação entre Sistema de Ficheiros e I/O

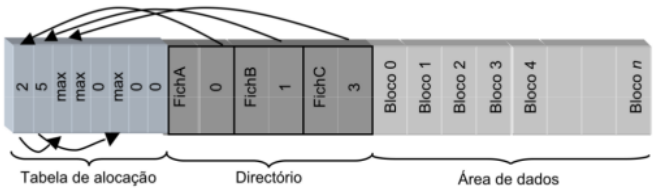
- O sistema de ficheiro situa-se num nível hierárquico acima das I/O, sendo estas usadas para aceder aos periféricos de memória de massa;

- As I/O estão ao mesmo nível do sistema de ficheiros, sendo dispositivos virtuais vistos como quaisquer outros ficheiros;

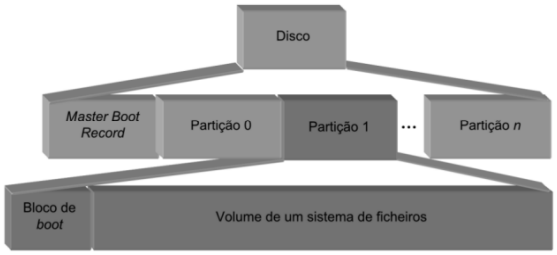
Estrutura Interna

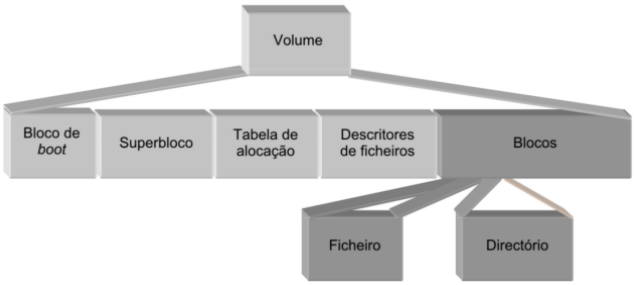
File Allocation Table, FAT:

- A tabela de alocação é partilhada por todos;
- Ficheiros grandes ocupam mais espaço na tabela de alocação.



Disco e Partição





- Blocos grandes optimizam o I/O mas conduzem a desperdício de espaço por fragmentação interna;

Acesso a Ficheiros

- 3 etapas para aumentar o desempenho:

- **Abertura** do ficheiro: dado do nome, é pesquisado o diretóriado, copiado o descritor do ficheiro para memória e é devolvido ao utilizador um ID de ficheiro aberto;
- **Leitura/Escrita** de informação dado o ID do ficheiro aberto;
- **Fecho** do ficheiro.