

Componentes de um Sistema de Operação

Em sistemas modernos é habitual ter-se as seguintes componentes ou módulos:

- Gestor de processos
- Gestor da memória principal
- Gestor da memória secundária
- Gestor do sistema de I/O
- Gestor de ficheiros
- Sistema de protecção
- Rede
- Interpretador de comandos

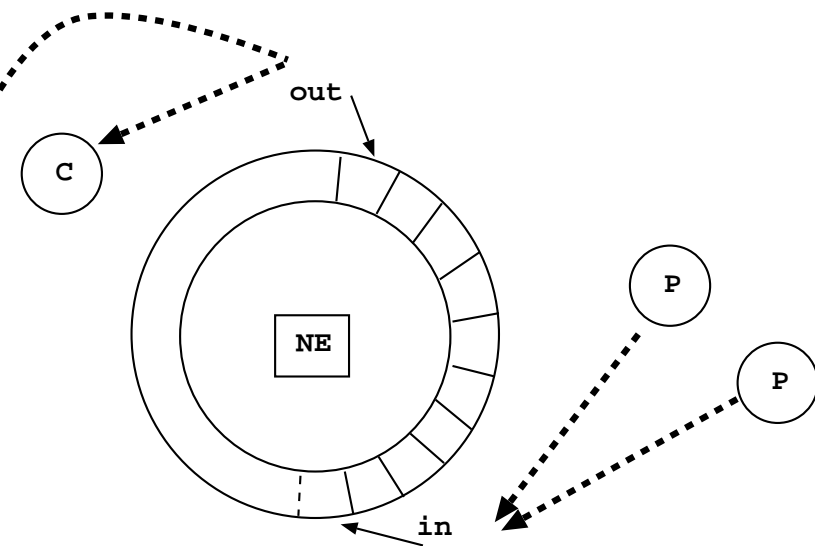
O que é um processo?

- *Um processo é um programa em execução.*
 - Um programa por si só não é um processo, é uma entidade passiva.
 - Um processo é uma entidade activa, com um *program-counter* a indicar qual a próxima instrução a executar.
 - Um processo para realizar a sua tarefa precisa de recursos: tempo de CPU, memória, ficheiros e periféricos de I/O.

Gestor de Processos

- O gestor de processos do SO é responsável por:
 - criar e remover processos.
 - suspender e retomar a execução de processos.
 - providenciar mecanismos que permitam:
 - sincronização entre processos
 - comunicação entre processos
- Um SO executa concorrentemente um grande número de processos:
 - processos do utilizador: correspondem à execução de programas do utilizador.
 - processos do sistema: correspondem à execução de programas do SO.

Sincronização e Comunicação entre Processos: exemplo



```
produtor:
  while (1) {
    ...
    novo=produz_item();
    ...
    while (ctr==N) ;
    buffer[in]= novo;
    in= (in+1)%N;
    ctr++;
  }
```

```
consumidor:
  while (1) {
    while (ctr==0) ;
    item=buffer[out];
    out= (out+1)%N;
    ctr--;
    ...
    usar(item);
    ...
  }
```

- várias instâncias dos processos Produtor e Consumidor.
- sincronizar actualização da variável `ctr`. Por que?
- comunicação entre processos Consumidor e Produtor dá-se através do `buffer`.
- `ctr` e `buffer` têm de estar numa zona de memória acessível aos processos (memória partilhada).

Gestor de Memória Principal

- A memória pode ser vista como um vector grande de palavras (ou bytes), cada uma com o seu endereço.
É um repositório de dados de acesso rápido, partilhado pelo CPU e periféricos de I/O.
- A memória principal é volátil, perde o seu conteúdo no caso de ocorrer uma falha no sistema.
- O gestor de memória do SO é responsável por:
 - saber que partes da memória estão a ser usadas e por quem.
 - decidir que processos é que devem ser carregados para a memória quando existir espaço livre.
 - atribuir e retirar espaço de memória

Gestor da Memória Secundária

- Memória de maior dimensão, não volátil, acessível on-line, que serve de apoio à memória principal (pequena para armazenar todos os programas e dados).
- Os computadores actuais usam discos como principal meio de armazenamento de dados e programas.
- O gestor de memória secundária (ou auxiliar) do SO, também visto como gestor do disco, é responsável por:
 - inicializar o disco, i.e. organizar o disco em blocos.
 - gerir o espaço livre, i.e. saber localizar os blocos livres.
 - atribuir espaço (memória) em disco. e.g. para guardar um programa.
 - scheduling do disco: aceder a um bloco do disco depende do movimento do disco

Gestor de Ficheiros

- visão independente do meio de armazenamento.
- ficheiro – conjunto de informação (sequência de bits ou bytes) relacionada, definida pelo seu utilizador.
- O gestor de ficheiros do SO é responsável por:
 - criar e remover ficheiros
 - criar e remover directórios
 - fornecer funções de manipulação de ficheiros e directórios.
 - mapeamento de ficheiros em disco
 - backup de ficheiros

Sistema de Protecção

- refere-se a mecanismos de controlo no acesso a recursos do sistema ou do utilizador, por parte de programas, processos, ou utilizadores.
- são necessários mecanismos que garantam que os ficheiros, segmentos de memória, CPU, e outros recursos podem ser usados apenas pelos processos que possuem autorização do SO para o fazer.
- por exemplo:
 - a MMU assegura que um processo apenas executa dentro do seu espaço de endereçamento.
 - o temporizador assegura que nenhum processo acede ao CPU por tempo ilimitado.
 - os registos de controlo dos periféricos não estão acessíveis aos utilizadores, sendo mais simples proteger a integridade dos vários periféricos.
- O mecanismo de protecção deve:
 - distinguir entre uso autorizado e não-autorizado.
 - especificar os controlos a serem impostos
 - fornecer meios de os impor

Gestor do Sistema de I/O

- esconde os detalhes dos periféricos de I/O do restante SO.
- O gestor do sistema de I/O envolve:
 - uma componente de gestão de memória que inclui: buffering, caching, e spooling.
 - uma interface geral para controladores de periféricos.
 - controladores (drivers) para periféricos específicos.

Interpretador de Comandos

- é a interface entre o utilizador e o SO.
- um dos programas mais importantes do SO (shell em Unix).
- a função do interpretador de comandos é muito simples:
ler o próximo comando e executá-lo
e deve fazê-lo o mais amigável possível: X-windows, Gnome, Kde, Motif, MacOS, Windows, etc.
- os comandos são meios para lidar com: criação e gestão de processos, I/O, acesso a ficheiros, gestão do disco, etc.

Serviços de um Sistema de Operação

Um SO fornece um conjunto de serviços que visam simplificar a tarefa de um programador, nomeadamente permite:

- Execução de programas – carregar um programa para memória e executá-lo.
- Operações de I/O – realizar operações de I/O a pedido dos programas dos utilizadores (não podem fazê-lo directamente).
- Manipulação do sistema de ficheiros – lêr, escrever, criar e remover ficheiros.
- Comunicação – troca de informação entre processos a executar na mesma máquina ou em máquinas numa rede. *Memória partilhada* ou *troca de mensagens*.
- Detecção de erros – assegurar correcto funcionamento através de detecção de erros no CPU e memória, nos periféricos de I/O, ou nos programas.

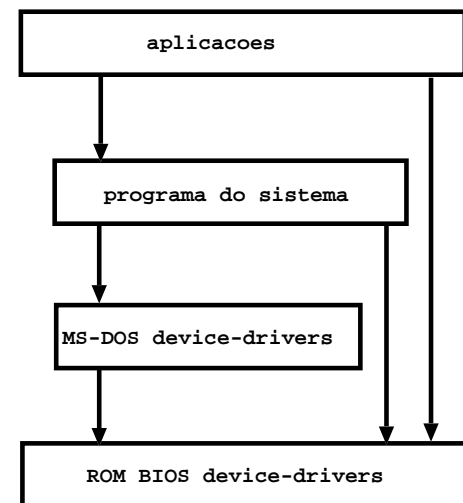
Funções adicionais do SO com vista a melhorar a sua eficiência:

- Atribuição de recursos – a vários utilizadores ou programas a executar ao mesmo tempo.
- Contabilização do uso de recursos – memória usada por utilizador, número de páginas impressas, número de acessos, etc.
- Protecção – assegurar que todos os acessos aos recursos do sistema estão controlados.

Estrutura dos SOs: sistemas monolíticos

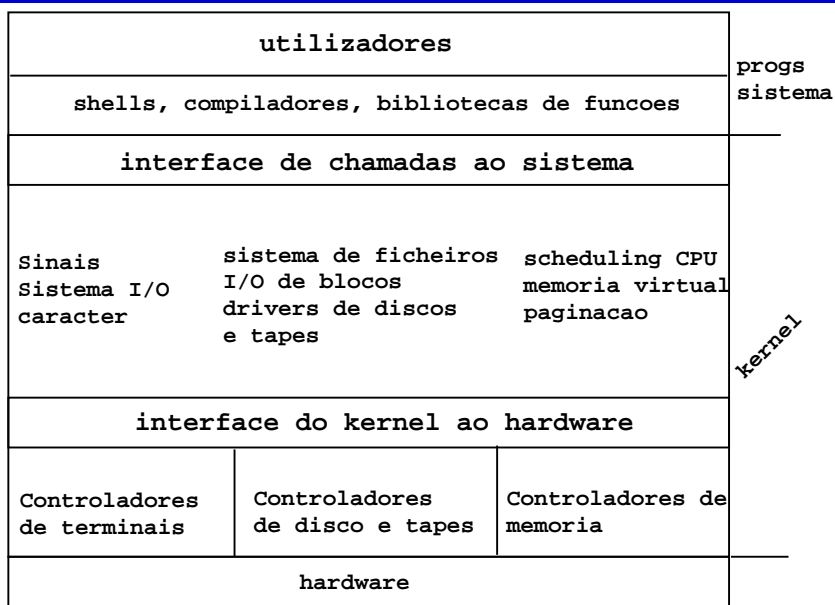
Sistemas monolíticos – sem estrutura – o SO é constituído por um conjunto de programas todos ao mesmo nível, em que qualquer programa pode chamar um dos outros.

→ MS-DOS é um sistema monolítico. Foi concebido com o intuito de proporcionar a maior funcionalidade no menor espaço (lembram-se do limite dos 640KB?).



- não está dividido em módulos
- interfaces e níveis de funcionalidade não estão bem separados. E.g. é possível a programas aceder directamente a rotinas básicas de I/O para escrever directamente para o ecrã ou impressora.
- estava também limitado pelo hardware. O Intel 8088 não proporcionava protecção ao nível do hardware.

Estrutura dos SOs: O Unix original



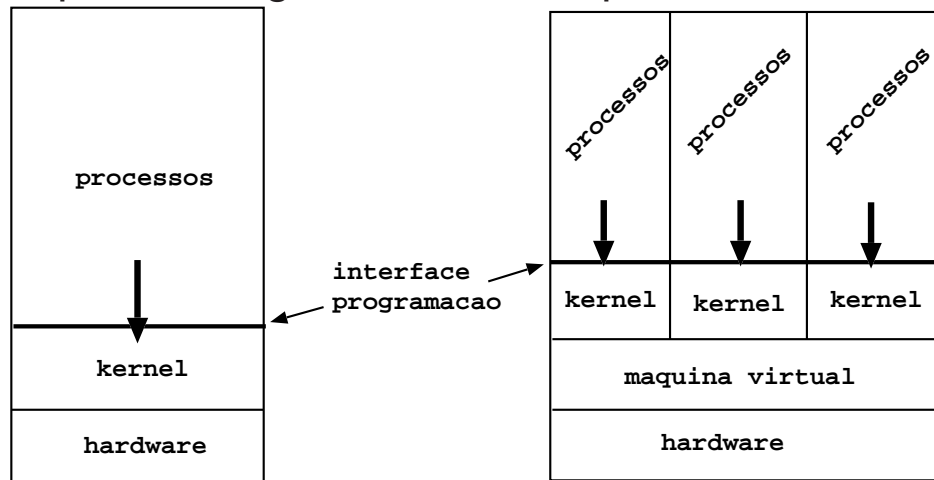
- estrutura limitada pela funcionalidade do hardware
- duas partes: kernel e aplicações de sistema
- kernel muito grande e pouco estruturado
- muita funcionalidade via chamadas de sistema

Sistemas por camadas (layered systems)

- O SO está dividido em camadas empilhadas. A camada 0, mais baixa, corresponde ao hardware e a mais alta, N, à interface com o utilizador.
- Garante modularidade: funções de uma camada apenas podem chamar outras funções mais básicas definidas em camadas mais interiores.
- Estrutura em camadas do sistema THE – 1o a usar esta estrutura.
 - camada 5: programas do utilizador
 - camada 4: buffering para periféricos de I/O
 - camada 3: controlador do periférico consola
 - camada 2: gestor de memória
 - camada 1: scheduling de processos
 - camada 0: hardware
- inconvenientes: planeamento de camadas e menor eficiência (I/O obriga a atravessamento de camadas).
- Exemplos: OS/2 (IBM) e primeira versão do Windows-NT.

Sistemas de máquina virtual

- *Máquina virtual* (VM) permite abstrair o hardware do kernel.
- Uma VM permite suportar vários SOs na mesma máquina.
- Sistema proposto por investigadores da IBM para o VM/370.



- VM é perfeito para investigação e desenvolvimento, embora difícil de implementar.
- VM está novamente popular, e.g. vmware