Sistemas Operativos

2025 – 2026

Conceito introdutório de sistema

Introdução ao sistema Unix

Tópicos

Conceito de sistema operativo – objetivos e organização

Funcionalidades principais de sistemas operativos

Núcleo *vs.* aplicações - modos de execução

Visão muito geral de um sistema operativo

Começando pelo início

-> Para que serve o sistema operativo?

3

Visão muito geral de um sistema operativo

-> Para que serve o sistema operativo?

O sistema operativo tem como objetivo

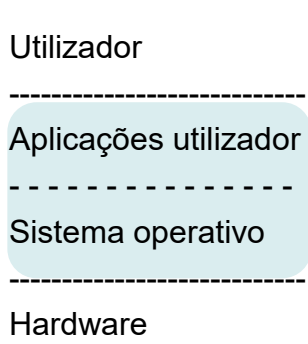
- **Proporcionar serviços às aplicações** (recursos e funcionalidades: ambiente de execução)
- **Virtualizar**, abstraindo e simplificando, a interação com periféricos
- **Garantir** que as aplicações e utilizadores **cumprem as regras** em vigor, proporcionando estabilidade e segurança
- **Gerir os recursos**, e **otimizar** a sua utilização

Ou seja, é o gestor da máquina, e portanto:

- O seu lugar é entre as aplicações e o hardware
- Vai precisar de privilégios que as meras aplicações não têm

4

Visão esquemática de um sistema operativo



- Cada camada interage apenas com as que estão imediatamente adjacentes
- As aplicações *utilizador*
  - Não fazem parte do sistema operativo
  - Inclui-se aqui a interface com o utilizador
  - Não têm capacidade de interação com o hardware
- O sistema operativo
  - Tem uma estrutura interna com maior ou menor complexidade, dependendo da arquitetura usada
  - Inclui os *device drivers*

Visão esquemática de um sistema operativo

- Cada camada interage apenas com as que estão imediatamente adjacentes
- As aplicações *utilizador*
  - Não fazem parte do sistema operativo
  - ***Inclui-se aqui a interface com o utilizador (Shell)***
  - Não têm capacidade de interação com o hardware
- O sistema operativo
  - Tem uma estrutura interna com maior ou menor complexidade, dependendo da arquitetura usada
  - Inclui os *device drivers*

Visão muito geral da estrutura um sistema

- O sistema operativo é constituído por **um conjunto** de programas
  - **Núcleo (**ou ***kernel*****).**
    - É a parte mais central do sistema. Controla todos os aspetos da máquina
    - **Corre em modo mais privilegiado (o processador reconhece mais instruções)**
    - Pode ser um programa monolítico, ou uma coleção de vários programas, dependendo da arquitetura.
  - **Gestores de dispositivos (*device drivers*).**
    - Podem ou não ser considerados como parte do núcleo.
    - **Correm normalmente (mas não sempre) também com privilégios elevados**
  - **Outros programas do sistema operativo.**
    - Fazem parte do sistema mas não são o núcleo.
    - Em termos de privilégios, são "meras" aplicações
    - Exemplo: a Shell (consola ou gráfica)

Os restantes programas a executar no computador são meras aplicações. Correm num modo não privilegiado. Se quiserem aceder aos recursos da máquina têm que passar pelo sistema operativo (API)

Visão muito geral da estrutura um sistema

Núcleo (kernel).

•Pode ser um único componente monolítico (um "mega programa"), que corre em modo de execução privilegiado

- A abordagem monolítica tem vantagens (+rápido, em teoria) e desvantagens (mais difícil de gerir o desenvolvimento do software)
- Exemplo: Linux

•Pode ser constituído por vários programas que são todos postos a correr no modo privilegiado, sendo assim mais modular

- Em teoria, mais lendo, mas mais organizado (mas isto depende muito de outros aspetos do sistema)
- Exemplo: Windows
- Neste caso, passa a ser: *tudo aquilo que corre no modo de execução privilegiado é o núcleo*

Visão muito geral da estrutura um sistema

Núcleo (kernel).

- Independentemente de ser monolítico ou vários programas, pode suportar a adição e remoção de componentes em *runtime*, conforme necessário
- Permite adaptar aquilo que está a correr em modo privilegiado àquilo que realmente é necessário face ao que a máquina está a fazer.
  - É uma característica bastante vantajosa
  - Exemplo: Linux ("módulos")
  - Comandos **insmod**, **rmmod**, **lsmod**
  - Sendo vários programas, aquilo que identifica o que é o núcleo passa simplesmente a ser: tudo aquilo que corre no modo de execução privilegiado é o núcleo

Visão muito geral da estrutura um sistema

Gestores de dispositivos (device drivers).

- São um exemplo de componentes autónomos mas que fazem parte do núcleo se correrem em modo de execução privilegiado
- Exemplo: drivers das placas gráficas – programados pelo fabricante e não pelos autores do sistema, correm normalmente em modo privilegiado e passam a fazer parte do núcleo quando estão em execução
  - Pergunta relacionada: "Por que razão havia tantos BSOD no Windows há uns tempos atrás?"
    - Isto para demonstrar que os *device drivers* correm mesmo em modo privilegiado, e que isso não é uma vantagem, mas tem que ser assim

Visão muito geral da estrutura um sistema

Outros programas do sistema operativo.

- Fazem parte do sistema mas não são o núcleo.
- Correm em modo não-privilegiado

Exemplos

- Comandos do utilizador (exemplos: ls, cp, mkdir), **inclusive os do super-utilizador (administrador, "root")** (exemplo: fdisk, mount)
- Utilitários de apoio ao utilizador ou ao sistema
- Bibliotecas de apoio à interface com o utilizador

Os restantes programas a executar no computador são meras aplicações. Correm num modo não privilegiado. Se quiserem aceder aos recursos da máquina têm que passar pelo sistema operativo (API). Exemplo: office, browser, jogos, etc.

Visão muito geral da estrutura um sistema

"Modo de execução"

- O processador permite a execução em dois (ou mais) modos
  (-> **Isto não tem nada a ver com tipo de utilizador**)

Modo privilegiado

- Neste modo o processador permite aos programas determinadas ações e instruções críticas
- Concretamente, permite executar determinadas instruções (código máquina) necessárias para aceder e manipular recursos da máquina
- Sem essas instruções um programa não consegue ver memória que não lhe pertence, interromper outros programas, modificar o próprio sistema, aceder diretamente aos periféricos (exemplo, disco), etc.

Visão muito geral da estrutura um sistema

"Modo de execução"

Modo não-privilegiado ("modo utilizador")

- Neste modo o processador não reconhece a existência das instruções (código máquina) necessárias para aceder a outras zonas de memória, aos periféricos, etc. (referidas no slide atrás)
- Um programa que esteja a executar neste modo é como se estivesse numa gaiola da qual não consegue sair, **mesmo que esteja a ser executado pelo super-utilizador**
  - Um programa que esteja a correr em modo utilizador não tem forma de passar a correr em modo privilegiado
- As aplicações do **utilizador (inclusive o root)** e os componentes não-núcleo do sistema correm neste modo ("modo **utilizador**")

Visão muito geral da estrutura um sistema

"Modo de execução"

- Um programa a correr **em modo utilizador**
  - Não dispõe de instruções que lhe permitam passar para modo privilegiado
  - Não dispõe de instruções nem meios para ver o que não lhe pertence, ou interferir com outros programas e utilizadores
    - Seja de propósito ou por bug
  - A única forma de aceder aos recursos do sistema/máquina é de **passar pelo sistema operativo** o qual valida a identificação e permissões
  - A distinção entre modos é feita pelo processador. Se o sistema operativo estiver bem feito e sem bugs, é possível ter a garantia de estabilidade e segurança no sistema

Visão muito geral da estrutura um sistema

"Modo de execução"

- Um programa a correr em modo privilegiado
  - Dispõe de instruções que lhe permitem fazer o que entender globalmente na máquina toda
  - Consegue ver todas as zonas de memória e todos os periféricos, aceder a todos os ficheiros etc.
  - Se tiver bugs ou código malicioso, poderá provocar estragos enormes no sistema

  Por isso
  - Apenas o núcleo corre neste modo (precisa desses privilégios para gerir a máquina).
  - O sistema quando arranca coloca o núcleo nesse modo. Após isto, apenas corre em modo privilegiado quem o núcleo entender

Visão muito geral da estrutura um sistema

"Modo de execução"

- A distinção do modo de execução é baseada nos mecanismos de gestão de memória
  - Certas zonas de memória estão "marcadas" como podendo ter instruções privilegiadas – são destinadas a código do núcleo (*kernel*)
  - A verificação é feita pelo processador quando vai buscar a próxima instrução a executar
    - Se for encontrada uma instrução privilegiada numa zona de memória não-privilegiada, o processador não a executa e avisa o sistema operativo (através de uma exceção)
    - Não implica nenhum *overhead* de performance adicional
  - A configuração é feita pelo sistema operativo
    - Esta configuração exige acesso a essas instruções privilegiadas. Isto significa que o código que esteja numa zona não privilegiada já não tem a capacidade para subverter esta configuração

Este assunto voltara a ser analisado no capítulo de gestão de memória

Visão muito geral da estrutura um sistema

Exemplo/demonstração - Modo de execução

- Demonstração prática
  - Este programa, executado por um utilizador, mesmo que seja o *root*, é interrompido na instrução hlt
  - Esta instrução é considerada privilegiada é restringida a código do núcleo
  - O processador não executa a instrução e notifica o sistema operativo que interrompe a execução do programa

```
#include <stdio.h>

int main(int argc, char ** argv) {
   printf("\n\nantes da instrução\n");
   /* tenta colocar o processador */
   /* no estado HALT */
    __asm__ ( "hlt;" );
   printf("\ndepois da instrução\n\n");
   return 0;
}
```

```
$ ./exemplo
antes da instrução
Segmentation fault
$
```

Este exemplo volta a surgir mais adiante

Visão muito geral da estrutura um sistema

Exemplo/Demonstração - Acesso restringido a memória

- Os sistema não permite aos programas em execução acederem a memória que não lhes pertence
  - O núcleo tem capacidade de aceder a toda a memória que existir na máquina
  - Mas os programas não-núcleo só conseguem ver a memória que o sistema lhes reservou
  - A configuração é feita pelo sistema, mas a verificação é feita pelo processador
    - Este assunto só é completamente explicado no capítulo de gestão de memória

Exemplo: este programa tenta aceder a um ponteiro não inicializado que, muito provavelmente, aponta para uma zona fora da memória atribuída ao programa e, por isso, na maioria dos casos, o programa é interrompido

```
#include <stdio.h>
int main() {
   int * p;
   printf("passei aqui\n");
   *p = 10;
   printf("cheguei aqui\n");
   return 0;
}
```

```
$ ./exemplo
passei aqui
Segmentation fault
$
```

Como fica o sistema funcionar e a gerir a máquina

Processo de *bootstraping*

- O código em memória não volátil inicia o processo, localizando o local (disco e partição) onde está o sistema operativo e dando inicio à colocação do *kernel* em memória
- Tendo o *kernel* em memória, este é executado e traz para memória os restantes componentes do sistema
- A partir daqui é o sistema que controlará tudo inclusive a execução de outros programas.
  - O lançamento de novos programas **passa sempre pelo sistema**, que já está em execução, e os coloca a correr em modo não-privilegiado – os novos programas já não conseguem adquirir mais privilégios

Como é dado acesso ao utilizador

Processo de login

- Um dos componentes que é colocado a executar no final do processo de *bootstrap* é o "processo login".
- Este processo aguarda por credenciais (username e password).
- Se as credenciais forem válidas, é lançada uma Shell "em nome" (com a identificação) do utilizador em questão.
- Qualquer aplicação lançada pelo utilizador é, na verdade, lançada através da Shell, herdando automaticamente a identificação do utilizador
  - A identificação da aplicação é herdada a partir da aplicação que desencadeou a sua execução
- As operações desencadeadas pelas aplicações terão a validade da permissão verificadas pelo sistema com base na identificação do processo ("programa a executar) que desencadeou a operação.

Ambiente de utilização do sistema

- Como em qualquer sistema, após o *login* bem sucedido, o sistema lança uma *shell* em nome do utilizador
- A ***shell*** é um programa que interage com o utilizador e o sistema operativo, encaminhando as ações do utilizador para operações sobre o sistema
- Pode ser consola ou gráfica
  - **Consola**: o utilizador especifica o que pretende através de **comandos** escritos
  - Exemplo: bash
  - **Gráfica**: o utilizador especifica o que pretende através de ícones e menús
  - Exemplos: Gnome, KDE
- A *shell* é um programa como os outros (tecnicamente não faz parte do sistema). Corre "em nome" do utilizador e todas as operações que desencadeia e programas que lança são implicitamente em nome desse utilizador

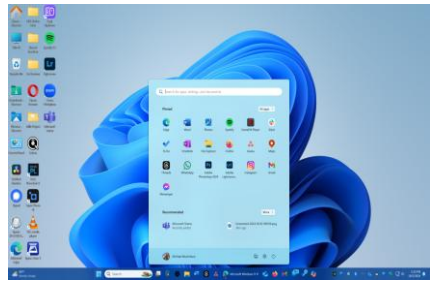
Ambiente de utilização Unix
Consola / linha de comandos

Exemplo

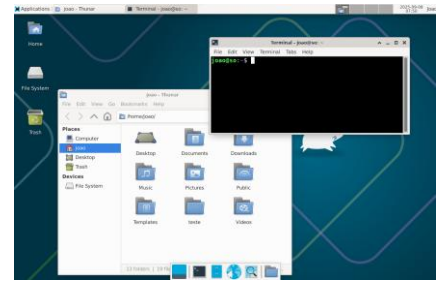
```
joao@so:~$ ls -l
total 36
drwxr-xr-x 2 joao joao 4096 Sep  7 23:50 Desktop
drwxr-xr-x 2 joao joao 4096 Sep  5 23:03 Documents
drwxr-xr-x 2 joao joao 4096 Sep  5 23:03 Downloads
drwxr-xr-x 2 joao joao 4096 Sep  5 23:03 Music
drwxr-xr-x 2 joao joao 4096 Sep  5 23:03 Pictures
drwxr-xr-x 2 joao joao 4096 Sep  5 23:03 Public
drwxr-xr-x 2 joao joao 4096 Sep  5 23:03 Templates
drwxrwxr-x 2 joao joao 4096 Sep  3 21:30 teste
drwxr-xr-x 2 joao joao 4096 Sep  5 23:03 Videos
joao@so:~$
joao@so:~$
```

Ambiente de utilização Unix
Consola/desktop gráfica

Exemplos



(Windows)



(Linux)