

Introdução aos Sistemas Operativos

Paulo Sérgio Almeida

Grupo de Sistemas Distribuídos
Departamento de Informática
Universidade do Minho



- **Introdução aos Sistemas Operativos**
 - O que é um sistema operativo
 - Evolução dos sistemas operativos
 - Conceitos envolvidos num sistema operativo



Sistema operativo é um intermediário

- Intermediário entre utilizadores e hardware
 - Interpretador de comandos (*shell*)
 - Sistemas de janelas
- Intermediário entre programas e hardware
 - Os principais clientes de um SO são os programas
 - Apesar de útil, interface gráfica nem sempre é necessária
 - Servidores podem nem ter interacção com utilizadores



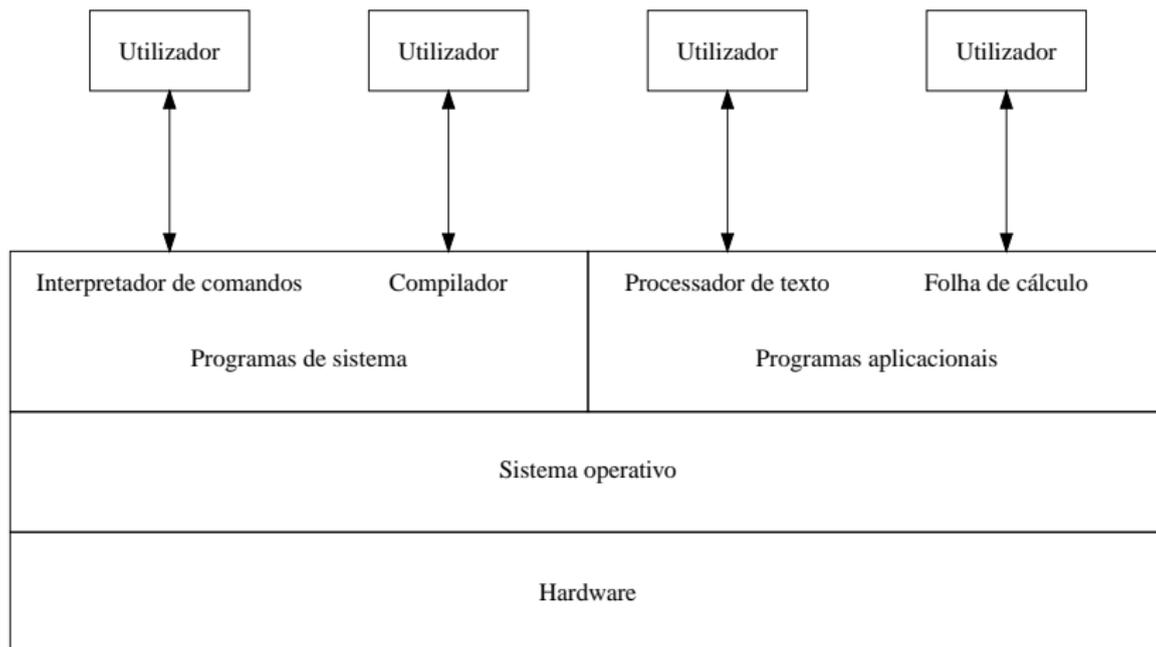
Estrutura de um sistema computacional

Um sistema computacional pode ser dividido em:

- **Hardware:** CPU, memória, dispositivos I/O
- **Sistema Operativo:** controla e gere o uso do hardware pelas aplicações e utilizadores
- **Programas de sistema:** interpretadores de comandos, gestores de janelas, compiladores, bibliotecas genéricas
- **Programas aplicativos:** resolvem problemas aos utilizadores, como processadores de texto, folhas de cálculo, jogos, ...
- **Utilizadores:** pessoas ou computadores que funcionam como clientes dos programas



Estrutura de um sistema computacional



Sistema operativo fornece abstracção sobre hardware

- Esconde detalhes, complexidade, diferenças
 - e.g. que placa gráfica, que disco
 - e.g. Linux existe em imensas arquitecturas de hardware
- Oferece API uniforme e conveniente
 - gestão de processos: `fork()`, `exec()`, ...
 - gestão de memória: `sbrk()`, ...
 - sistemas de ficheiros; `open()`, `read()`, `write()`, ...
 - ...

Um objectivo de um SO é oferecer conveniência



Sistema operativo é um gestor de recursos

- Isolamento entre processos
 - Cada processo executa numa máquina protegida
 - Um processo não deverá poder corromper memória dos outros
 - Se um processo entrar em ciclo infinito o computador não deverá bloquear
- Controlo no acesso a recursos
 - Se vários programas tentarem imprimir ao mesmo tempo . . .
 - Multiplexação no tempo: cada processo usa o recurso à vez
 - Recursos como memória e disco podem ser partilhados; exige técnicas de alocação de espaço

Outro objectivo de um SO é a eficiência na gestão que faz



Sistema operativo define a “personalidade” de um computador

- O SO define a “personalidade” de um computador
- O mesmo computador (hardware) comporta-se de modo diferente ao arrancar diferentes SO:
 - MS-DOS
 - Windows 95
 - Windows XP
 - Linux
 - Mac OS
- Só algumas combinações de hardware e SO são possíveis; ponto forte do Linux



E afinal o que faz parte do Sistema Operativo?

- Não existe definição universal
- Há quem tente dizer que “é tudo o que vem com o computador”; e.g. a Microsoft até disse que o browser fazia parte do sistema operativo
- Uma visão oposta é dizer que é apenas o **kernel**: o programa que corre em modo supervisor, com funções de controlo e gestão fundamentais (memória, processos, dispositivos de I/O)
- Um meio termo será dizer que é o kernel juntamente com os programas de sistema genéricos.



Evolução dos sistemas de computação

- 1^a geração (1945/1955) - Válvulas, placas programáveis
- 2^a geração (1955/1965) - Transistores, sistemas *batch*
- 3^a geração (1965/1980) - Circuitos integrados, time-sharing
- 4^a geração (1980/) - PCs, workstations, servidores
- ?? - PDAs, smartphones, web, p2p, grid



No início: o utilizador soberano

- Acesso livre ao computador
 - utilizador podia fazer tudo
 - utilizador tinha de fazer tudo
- Eficiência no uso do computador era baixa
 - elevado tempo de preparação
 - tempo despendido com debugging

Problemático devido ao custo muito elevado do computador



Introdução de um operador

- Introduziu-se um operador especializado
 - utilizador entrega fita perfurada ou cartões
 - operador carrega o programa, executa-o e devolve resultados
- Ganhou-se **eficiência**, perdeu-se em **conveniência**
 - operador é especialista em operação, não em programação
 - pode haver escalonamento (alteração da ordem de execução)
 - utilizador deixou de interagir com o seu programa



Introdução de programa de controlo

- Tarefas do operador podem ser automatizadas
- Programa toma o controlo:
 - controla a operação do computador
 - encadeia tarefas
 - operador passa a apenas carregar e descarregar
- Utilizadores incentivados a utilizar rotinas de I/O do sistema

Embrião de sistema operativo



Riscos de conflitos e erros

- Risco de o programa de controlo ... perder o controlo ...
- Erros nos programas dos utilizadores podem:
 - levar a ciclos infinitos
 - levar à destruição do programa de controlo
 - levar a erros no acesso a periféricos

Será que o programa de controlo pode proteger-se?



Hardware em auxílio do programa de controlo

Problemas anteriores motivam evolução do hardware:

- Interrupções:
 - permitem interromper os programas, mesmo que estejam num ciclo infinito
 - podem ser originadas por tempo, ou por um evento do sistema (e.g. fim de transferencia do disco).
- Modos de execução protegida:
 - instruções privilegiadas só executam em modo protegido
 - só o programa de controlo corre em modo protegido
- Protecção de memória:
 - evita que os programas do utilizador, por erro ou malícia, escrevam onde não devam
 - o programa de controlo já não pode ser destruído

Um SO moderno só é possível com hardware adequado



Aparecimento de Sistemas de batch

Processador auxiliar faz I/O de periféricos lentos:

- Uma máquina recebe cartões, que copia para banda magnética
- A banda é colocada na máquina principal, que executa os programas
- A banda com resultados é colocada noutra máquina, que os envia para a impressora



Multiprogramação

- Permite ter várias tarefas carregadas em memória central
- O tempo de CPU é repartido entre elas
- Motiva gestão da memória
- Exige ajuda do hardware para evitar acessos indevidos



Time-sharing

- Vários terminais são ligados ao computador central
- Permitem que os utilizadores voltem a interagir directamente com os seus programas
- Sistema operativo reparte o tempo de CPU pelos programas **prontos** a executar

Com time-sharing obtemos conveniência



Computador Pessoal

- Com os computadores pessoais volta tudo ao início:
 - utilizador soberano
 - monoprogramação, baixa eficiência
- No entanto:
 - grande conveniência para utilizadores
 - como são baratos a eficiência no uso do hardware já não é prioridade



Sistemas distribuídos

- Nos anos 80 aparecem as redes locais
- Permite partilha:
 - recursos caros
 - inconvenientes de replicar
- Permite padrões de cooperação
- Complica imenso a escrita de software:
 - que protocolos de comunicação
 - que modelos? cliente-servidor?
 - possibilidade de falhas independentes
- E por fim veio a Web ...



Mainframes e virtualização

- SOs para mainframes desenvolvidos nos anos 60
- IBM MVS, IBM VM/CMS
- Muitos deles ainda em operação
- A ideia de usar uma máquina poderosa par correr várias instâncias isoladas de um SO é apelativa
- Pode ser útil mesmo em máquinas pequenas, melhorando a flexibilidade e segurança

A virtualização está na moda (e.g. VMware)



Sistemas operativos para fins específicos

- SO de tempo real:
 - para controlo de sistemas industriais, sistemas de vôo, automóveis, robôs, ...
 - Permitem dar **garantias** de tempo de resposta que um SO normal não consegue
- SO para computadores com recursos limitados:
 - aparecimento de PDAs, telemóveis, sensores
 - hardware oferece menos suporte
 - outras preocupações; e.g consumo de energia

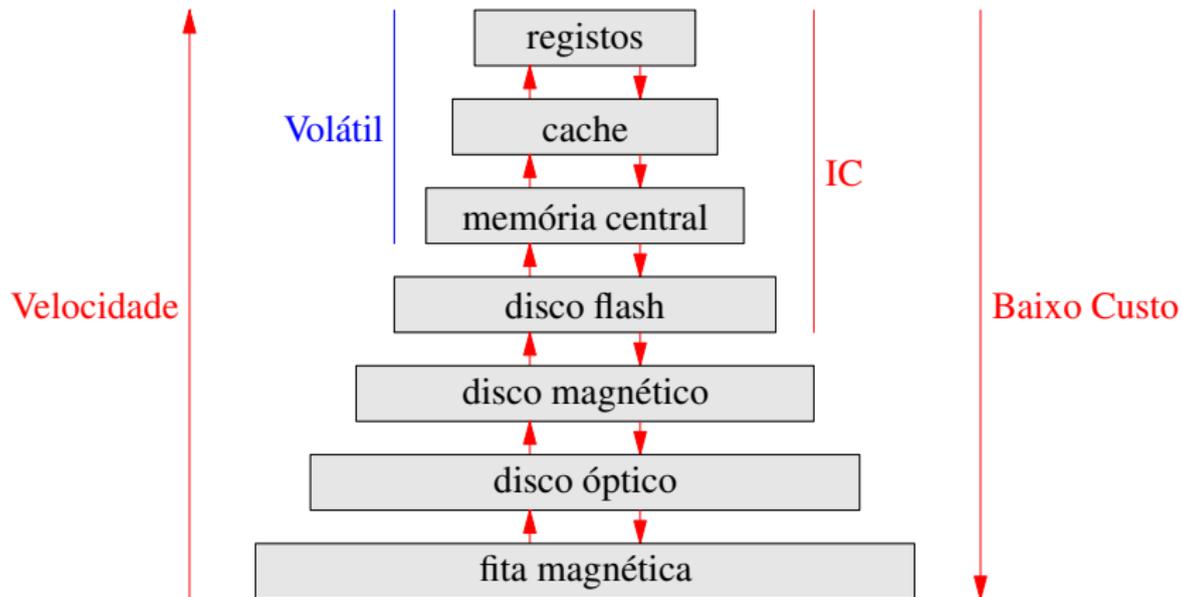


Arranque do computador

- O programa de **bootstrap** reside em ROM ou EPROM; é um exemplo de **firmware**
- é carregado no (re)arranque do computador
- Inicializa o hardware do sistema: controladores de dispositivos, memória, CPU
- carrega o **kernel** do sistema operativo e executa-o



Hierarquia de armazenamento



Interrupções

- Sinalização ao CPU de um evento externo ocorrido
- CPU pára o que estava a fazer e executa uma rotina de tratamento: o **interrupt handler**
- Do ponto de vista do programa a correr, a interrupção é assíncrona, acontece num momento não controlável
- Interrupções podem ser originadas por diferentes dispositivos
- É guardado em memória um **interrupt vector**, indexado pelo número do dispositivo, com os endereços dos handlers
- A rotina de tratamento deve ser curta; depois de terminada o processador recomeça onde estava
- A rotina deverá guardar e restaurar os registos do processador que necessitar de modificar



Polling

- Polling consiste em testar repetidamente se um evento aconteceu; e.g. a transferência do disco já terminou?

```
terminou = false;
while(!terminou)
    terminou = testa();
```

- Requer uso continuado de tempo de CPU
- Polling é de evitar em geral
- Particularmente grave se o evento demora a acontecer

Polling provoca desperdício; as interrupções permitem evitar o polling



Input/Output

- Um **device controller**:
 - está associado a um ou mais dispositivos de I/O
 - contém registos e um buffer local
- Um **device driver**:
 - existe no SO, associado a cada device controller
 - apresenta uma interface uniforme para aceder aos dispositivos
 - envia ordens ao controller; quando a operação termina, recebe interrupção
- O sistema de **DMA – direct memory access**:
 - permite a transferência de grandes quantidades de dados de um modo eficiente
 - os dados são transferidos directamente do buffer do controller para a memória central
 - não envolve intervenção do CPU
 - quando transferência termina, o CPU recebe interrupção



Multiprocessamento

- Utilização de vários CPUs numa máquina (partilhando memória)
- Vantagens:
 - aumenta o **throughput**: permite executar mais programas por unidade de tempo
 - economia: partilham recursos como periféricos, alimentação, disco
 - melhor fiabilidade: a falha de um CPU apenas diminui performance
- Variantes:
 - multiprocessamento **assimétrico**: um CPU distinguido, onde estão associados periféricos e SO
 - multiprocessamento **simétrico**: todos os CPU correm código do SO; exige maior sofisticação do SO
- **Clusters** agrupam máquinas individuais ligadas em rede



Multiprogramação e Time-sharing/multitasking

- Em ambos existem várias tarefas carregadas em memória central
- Quando uma tarefa não consegue prosseguir (e.g. requer I/O) o CPU é comutado para outra tarefa
- Melhoram a eficiência, aumentando a **utilização do CPU**
- Na multiprogramação uma tarefa pode prosseguir durante tempo considerável, até necessitar I/O
- Time-sharing/multitasking extensão da multiprogramação:
 - existe interacção com utilizadores: tempos de resposta baixos
 - a comutação entre tarefas ocorre a elevada frequência
 - os vários utilizadores têm a ilusão de ter a máquina só para si
 - com o multitasking temos **conveniência**



Escalonamento

- As tarefas à espera de virem para memória residem inicialmente em disco no **job pool**
- Em geral, apenas um subconjunto destas encontram-se em execução
- A escolha de quais vêm para memória é feita pelo escalonador de tarefas: **job scheduler**
- A atribuição do CPU em cada momento às tarefas em memória é feita pelo escalonador de CPU: **CPU scheduler**

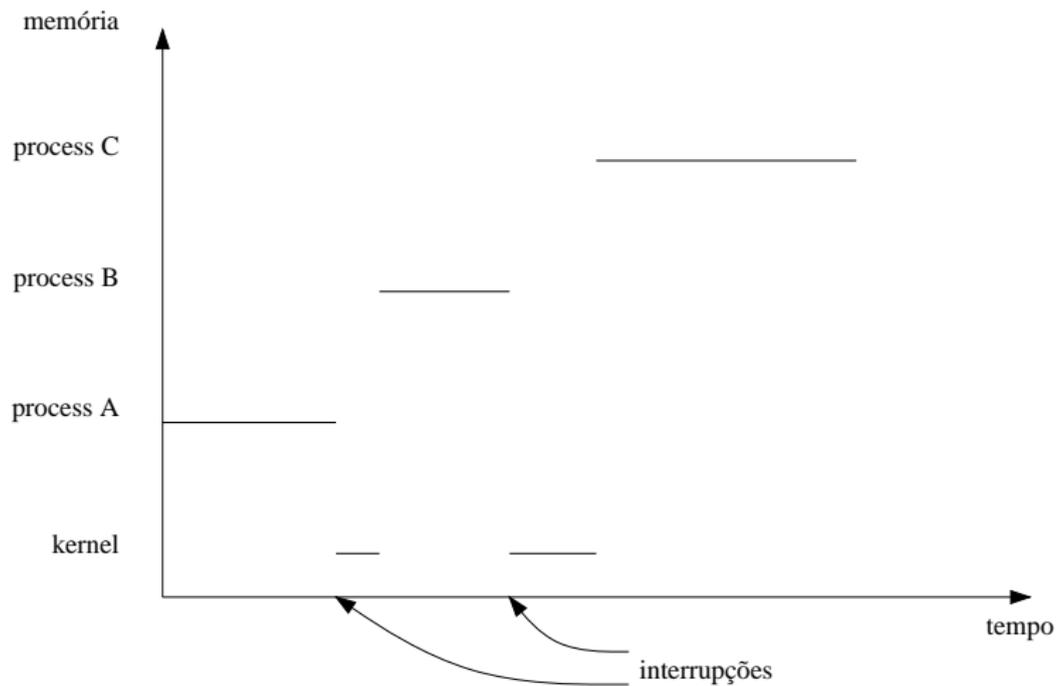


Um Sistema operativo é interrupt-driven

- O código do SO (kernel) corre tipicamente como resposta a um **interrupt** ou **trap**
- Uma trap é uma interrupção de software causada por:
 - erro numa instrução; e.g. divisão por zero, acesso inválido a memória
 - um pedido de código utilizador a um zerviço do sistema operativo
- Depois de correr algum código em resposta a uma interrupção:
 - quando o kernel cede o controlo do CPU não é necessariamente para voltar onde estava antes da interrupção
 - o normal é voltar ao modo utilizador a correr código de outro processo

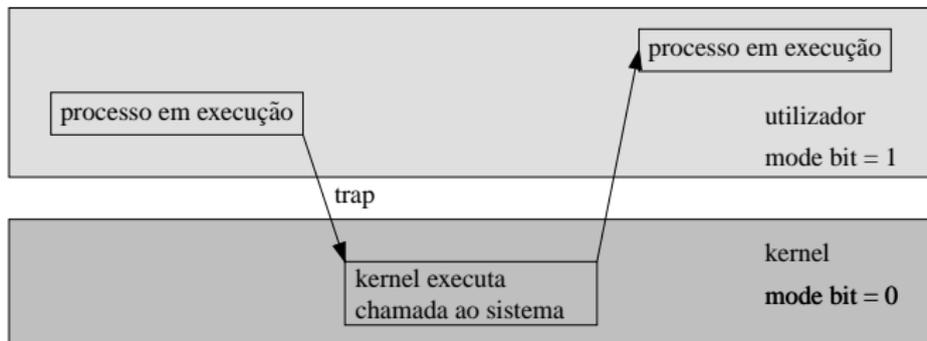


Um Sistema operativo é interrupt-driven



Modo utilizador e modo supervisor

- Um sistema operativo moderno necessita de pelo menos dois modos de execução:
 - modo utilizador: no qual corre o código das aplicações
 - modo kernel/supervisor/protégido: no qual corre o código do kernel
- O CPU toma conta do modo num **mode bit**
- Quando um processo pede um serviço ao SO por uma **chamada ao sistema / system call**, o CPU passa ao modo kernel



Instruções privilegiadas

- Algumas instruções só correm em modo kernel
- Se forem tentadas em modo utilizador resulta numa **trap**
- Exemplos de instruções privilegiadas:
 - mudar de modo de execução
 - controlo de I/O
 - gestão de timers
 - gestão de interrupções
 - acesso a registos de gestão de memória



Temporizador

- Uma aplicação não deverá poder correr para sempre; e.g. ciclo infinito
- E se a aplicação não ceder controlo ao kernel voluntariamente; e.g. via chamada ao sistema?
- O problema resolve-se usando um temporizador: **timer**
- Este pode ser programado para provocar uma interrupção passado um tempo especificado
- Antes de voltar ao modo utilizador, o kernel armadilha o timer . . .
- . . . o que faz com que eventualmente o kernel toma o controlo outra vez



Programas, processos e threads

- Um programa é uma entidade estática; um conjunto de instruções
- Um processo é uma entidade activa: um programa a correr
- Um processo necessita de recursos como: memória, tempo de CPU, ficheiros
- Os recursos podem ser atribuídos na criação ou alocados durante a execução
- Um processo **singlethreaded** é sequencial: contém um único **program counter**
- Um processo **multithreaded** contém várias **threads** concorrentes que partilham os recursos do processo, mas tendo cada uma o seu PC



Gestão de processos

Um sistema operativo é responsável por:

- Criar e apagar processos do sistema e utilizador
- Suspende e recomeçar processos
- Disponibilizar primitivas de sincronização de processos
- Disponibilizar primitivas de comunicação de processos



Gestão de memória

- O sistema operativo necessita de:
 - tomar conta que partes da memória estão em uso e por quem
 - decidir que partes de processos colocar em memória ou remover de memória
 - alocar ou desalocar memória para processos
- Um sistema operativo moderno:
 - distingue a memória de diferentes processos, confinando os acessos
 - disponibiliza segmentação, onde os processos podem utilizar vários segmentos de memória para fins diferentes
 - disponibiliza memória virtual, onde cada processo tem a ilusão de um espaço de endereçamento lógico, que pode ser muito maior do que a memória física



Gestão de ficheiros

- SO oferece vista uniforme sobre armazenamento: o conceito de ficheiro abstrai sobre propriedades físicas
- Ficheiros são organizados em directórios
- Sistemas de **controlo de acesso** determinam quem pode aceder a que ficheiro
- SO oferece serviços para:
 - criar e apagar ficheiros e directórios
 - manipular ficheiros e directórios
 - armazenar ficheiros em memória não volátil



Gestão de armazenamento

- Discos são usados para armazenar ficheiros e no funcionamento da memória virtual
- Discos são lentos: muita influência sobre performance global do computador
- SO trata de:
 - gerir espaço livre e alocar espaço
 - escalonamento do disco
- SO trata ainda do armazenamento terciário:
 - inclui discos ópticos e fita magnética
 - mais lento
 - diferentes modos: WORM (write-once, read-many), RW



Sistema de I/O

- Esconde peculiaridades de dispositivos de hardware
- Trata da gestão de memória de I/O:
 - **buffering**: armazenamento temporário
 - **caching**: armazenamento para reuso mais rápido
 - **spooling**: sobreposição de I/O de diferentes tarefas
- Oferece interface genérica com device-drivers
- Disponibiliza drivers para diferente hardware

