Introdução

O sistema operativo providencia um meio/ambiente no qual outros programas podem realizar trabalho útil. Para além disso funciona como um intermediário entre o utilizador e o *hardware* do computador. Pode-se ver o sistema operativo como uma extensão da máquina, *i.e,* simula uma máquina virtual por acima da máquina real escondendo os detalhes do *hardware* através de *API's* mais fáceis de usar.

Analogia: O sistema operativo é como um governo, em si não tem nenhuma nenhuma função de útil. Simplesmente fornece um ambiente em que os outros programas possam fazer trabalho útil.

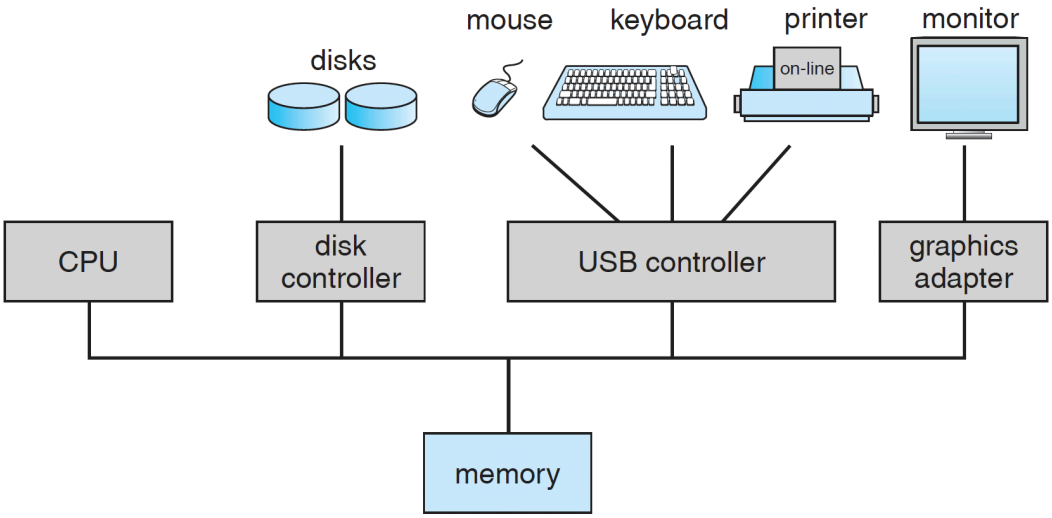
Um sistema operativo é composto por:

- **Hardware:** composto por uma unidade central de processamento (CPU), memória e dispositivos I/O (recursos básicos para o sistema). O sistema operativo deve colocar o *hardware* à disposição dos programas e utilizadores de uma forma conveniente, protegida, eficiente e justa.
- **Aplicações:** processadores de texto, compiladores, *browsers, etc.* (definem a maneira em como os recursos são usados para resolver os problemas dos utilizadores)
- **Sistema operativo:** controla o *hardware* e coordena o seu uso entre as diferentes aplicações
- **Utilizadores do sistema**

Pode-se o sistema operativo de dois pontos de vista:

- **Utilizador:** o objectivo é maximizar o trabalho que o utilizador está a executar. Nesta perspectiva, o sistema operativo, é concebido para facilitar trabalho
- **Sistema:** do ponto de vista do computador, o sistema operativo é o programa mais envolvido com o *hardware,* é um programa de controlo. Nesta perspectiva, é visto como um locador de recursos. Esses recursos podem ser tempo de *CPU,* espaço de memória, espaço de armazenamento de ficheiros, dispositivos

I/O, e por aí fora.



Analogia: é o sistema operativo que define a 'personalidade' de um computador.

Um computador consiste num ou mais *CPU's* e um conjunto de controladores de dispositivos conectados através de um *bus* que fornece acesso a memória partilhada. Para garantir acesso ordenado à memória, é fornecido um controlador de memória cuja função é sincronizar acessos à memória.

Quando um computador arranca ou é reiniciado necessita de um programa inicial para correr. Este programa tende a ser simples e designa-se por ***bootstrap program.*** Tipicamente, está guardado em memória apenas de leitura (***ROM*** - *Read-Only Memory*). O programa inicializa todos os aspectos do sistema, desde os registos de *CPU,* controladores de dispositivos, conteúdo de memória, *etc.* Este programa tem que saber como carregar o sistema operativo e como começar a executá-lo. Para isto, o programa *bootstrap* localiza e carrega para memória o *kernel* do sistema operativo. O sistema operativo começa então a executar o primeiro processo e espera que um evento ocorra.

Os controladores têm alguma capacidade de processamento?
Tem capacidade de processamento. Tem registos. É o sistema operativo que trata da gestão.

Interrupções vs Polling

A ocorrência de um evento é normalmente sinalizado por uma interrupção que pode ser proveniente do *hardware* ou do *software.* O *hardware* pode desencadear uma interrupção a qualquer momento enviando um sinal ao *CPU,* normalmente pelo *bus* do sistema. O *software* pode desencadear uma interrupção executando uma operação especial chamada s***ystem call.*** Quando o *CPU* é interrompido, para o que está a fazer e transfere a sua execução imediatamente para o localização em específico. A interrupção deve transferir o controlo para o serviço de rotina da interrupção. O método directo para tratar este caso é invocar uma rotina genérica que examine a informação da interrupção; a rotina, por sua vez, chama o *handler* específico para tratar a interrupção. Uma vez que as interrupções devem ser tratados o mais rápido possível, existe uma tabela de apontadores para as rotinas de interrupção, fornecendo a velocidade de processamento desejada.

Os tipos de interrupções são os seguintes:

- *Hardware:* Geradas pelos dispositivos de *hardware* para sinalizar o facto de precisarem de 'atenção' do sistema operativo. Podem ter recebido dados (teclas do teclado, dados do cartão de *ethernet, etc*); ou apenas completaram uma tarefa que o sistema operativo tinha pedido anteriormente, como por exemplo, transferir dados entre o disco rígido e a memória.
- *Software:* Geradas pelos programas quando pretendem pedir uma chamada ao sistema para ser executada pelo sistema operativo. As interrupções de *software* também podem ser causadas por erros do programa em execução, como por exemplo, divisão por zero. Este tipo de interrupções são chamadas de *traps* ou *excepciones.*

O fluxo da invocação de uma interrupção é o seguinte:

1. O estado do programa que está atualmente a correr é guardado para posterior retoma de execução
2. O *CPU* muda para modo *kernel*/supervisor.
3. É localizado o código do *kernel* para tratar a interrupção através da tabela de *handlers* e do vetor de interrupção.
4. O código é executado.
5. O *CPU* volta ao modo utilizar e carrega o estado do programa que estava a executar.

Este sistema de interrupções é usado em alternativa ao sistema de *Polling.* Esta técnica é menos eficiente pois, ao contrário de receber uma interrupção, o *CPU* está sempre a perguntar aos dispositivos se precisam de alguma coisa. O problema disto é, se um dispositivo não precisa de nada, foi gasto tempo de *CPU* para nada.

Analogia: A técnica de *polling* pode ser equiparada ao facto de, a cada segundo, estarmos sempre a pegar no telemóvel para ver se estamos a receber uma chamada.

Tempo Virtual

Só conta enquanto executa o processo. Dependendo do sistema operativo pode contar em modo *kernel* ou não. Se houver *threads* é o somatório do tempo de cada uma.

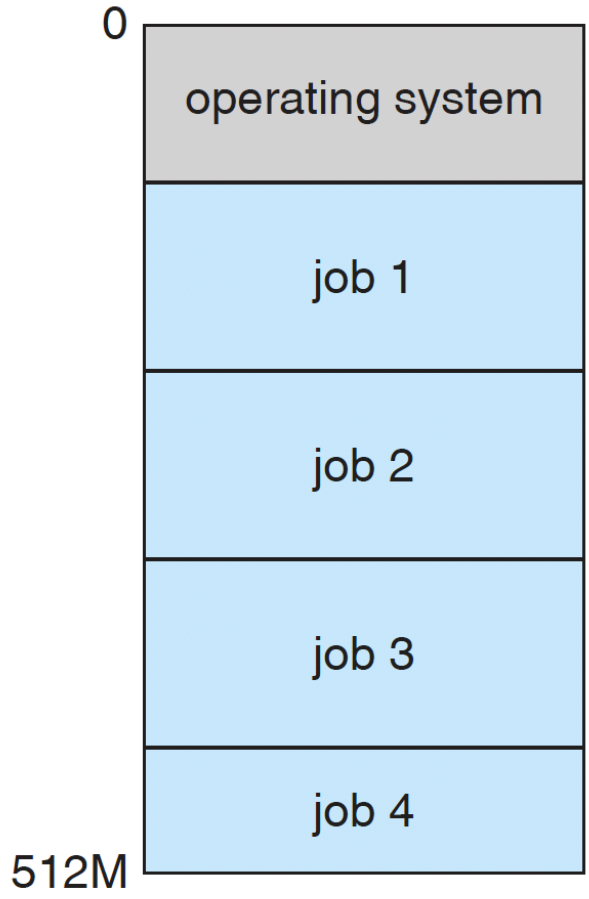
Multiprogramação

Um dos mais importantes aspectos dos sistemas operativos é a habilidade para suportar vários programas em execução ao mesmo tempo. Uma vez que, de um modo geral, apenas um programa não consegue deixar ocupado o tempo inteiro o *CPU* ou os dispositivos

I/O, a multiprogramação permite aumentar a utilização destes recursos, organizando os *jobs* (código + dados).

Uma vez que o tamanho da memória principal é demasiado pequeno para locar espaço para todos os processos, o sistema operativo mantém em disco uma *pool* de processos (j*ob pool*), que contém os processos que aguardam alocação na memória principal.
O conjunto de *jobs* na memória principal pode ser um sub-conjunto daqueles que residem na j*ob pool*. Quando um programa necessita, por exemplo, de uma operação de *I/O,* ao contrário do que aconteceria num sistema de monoprogramação onde o *CPU* não fazia trabalho útil enquanto não fosse feita essa operação, num sistema de multiprogramação, este troca para outro processo que está à espera de ser alocado. Isto permite que, enquanto haja processos para executar, o *CPU* nunca está parado.

Time-sharing é um conceito lógico muito importante na multiprogramação pois os recursos são partilhado por vários processos. A troca de contexto entre os diferentes processos é tão rápida que os utilizadores não se apercebem que estão a partilhar recursos com outros processos.



Analogia: Um advogado nunca trabalha apenas para um cliente. Por exemplo, se estiver à espera de papelada ou à espera de ir a julgamento, ao contrário de estar parado, ele tem outros clientes para tratar.

A multiprogramação, para além de trazer vantagens e eficiência, requer mais cuidados a nível de gestão uma vez que agora existem vários programas a correr em paralelo. É necessário agora ter o seguinte em atenção:

- *Job Scheduling:* a memória principal, dado o seu espaço limitado, pode estar cheia. É necessário que o sistema escolha um dos processos que estejam na *job pool* para o colocar em execução na memória principal.
- *CPU Scheduling:* Escolha de um processo para executar de

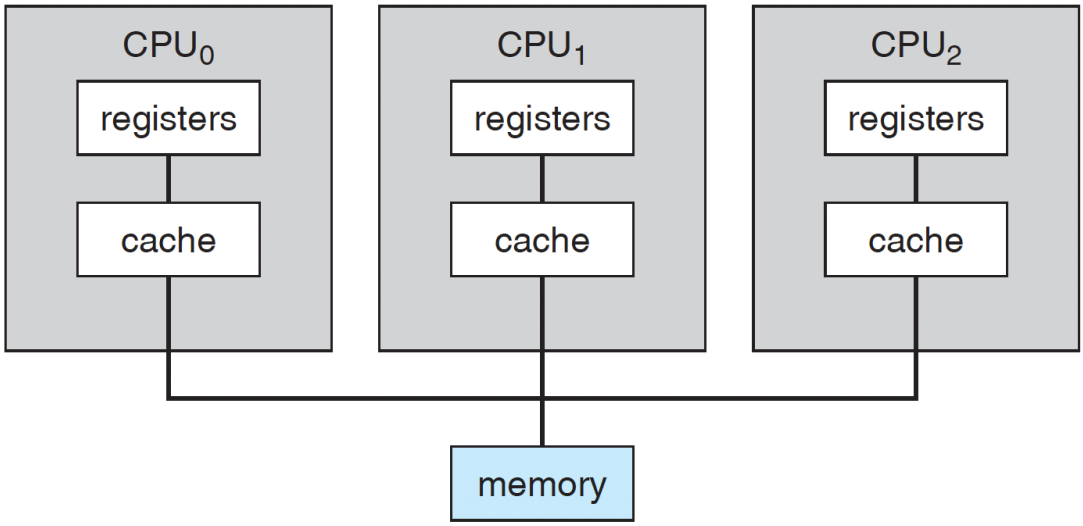
entre os que estão na memória principal à espera de executar.
- Gestão de memória: limitar o espaço de cada processo e não deixar que um possa aceder/afectar o espaço do outro.

Multiprocessamento

Os sistemas com multiprocessadores são sistemas que têm dois ou mais processadores e que partilham o *bus* do computador, assim como memória, periféricos e o relógio.

Estes sistemas têm as seguintes vantagens:

- Maior taxa de processamento: aumentando o número de processadores é expectável que haja mais trabalho feito em menos tempo. No entanto, este aumento não é proporcional ao número de processadores pois existe *overhead* ao fazer com que as diferentes partes trabalhem correctamente.
  Este *overhead* juntamente com a partilha de recursos diminui o ganho proporcional que se esperava ter.
- Economia: os sistemas com multiprocessadores têm um custo mais baixo do que o equivalente a ter múltiplos processadores singulares pois é possível partilhar armazenamento, periféricos, *etc.*
- Maior confiabilidade: se as funcionalidades são distribuídas pelos diferentes processadores, a falha de um não afecta o funcionamento dos outros. Se tivermos 10 processadores e um deles falhar, o resto dos processadores conseguem ter acesso ao trabalho que já foi feito pelo que falhou e então o sistema corre apenas 10% mais lento. A propriedade de continuar a operar proporcionalmente ao nível do *hardware* que está funcional designa-se por *graceful degradation.*



Os sistemas multiprocessamento podem ser divididos em dois tipos:

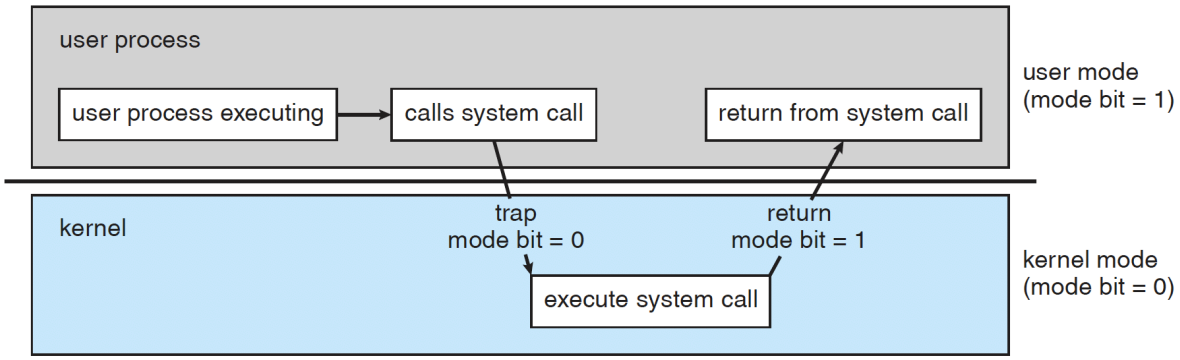
- Assimétricos: um processador principal controla o resto dos processadores. O resto do processadores ou têm uma tarefa em específico ou lhes é atribuída uma tarefa pelo processador principal. Este tipo de multiprocessamento tem uma arquitectura *master-slave* onde o processador principal loca e faz o escalonamento do trabalho para os outros processadores.
- Simétricos: cada processador pode executar qualquer tarefa.

Modos de execução

Para assegurar a correcta execução do sistema operativo, é necessário distinguir entre a execução de código do sistema operativo do código definido pelo utilizador. A forma como isto é feito é através do suporte de *hardware.* Um *bit,* designado por *mode bit,* foi adicionado ao *hardware* para distinguir estes dois modos de execução, *kernel* (0) ou *user* (1). Com este *bit* pode-se distinguir uma tarefa que é executada por parte do sistema operativo de uma executada por parte do utilizador.

Os modos de execução trazem segurança para o sistema pois é garantido que apenas uma parte do sistema, o *kernel*, aceda diretamente ao *hardware.* Doutra maneira não seria possível ter esta garantia. Desta maneira, o código do lado do utilizador está proibido de aceder diretamente ao *hardware* (ficheiros, memória, *etc*).
Em resumo:

- *Kernel mode:* instruções para gerir a memória e como pode ser acedida. Para além disso, também tem privilégios para aceder a periféricos como os discos e cartões de rede. As interrupções também são executadas neste modo.
- *User mode:* modo mais restrito onde apenas é possível aceder a certas zonas da memória e o acesso a periféricos não é possível.



O fluxo de uma chamada ao sistema, como se pode ver na imagem, é o seguinte:

1. O programa em *user mode*, quando invoca a chamada ao sistema, coloca os argumentos da mesma em registos ou numa *stack frame,* indicando que serviço pretende que seja executado pelo sistema operativo.
2. O programa em *user mode* executa a interrupção 'trap'
3. Imediatamente, o *CPU* guarda o estado do programa actual, muda para o modo *kernel* e salta para um sítio fixo da memória que contém as instruções da interrupção.
4. A rotina genérica de interrupção lê o serviço pretendido e os

argumentos e executa a rotina de interrupção correspondente.

5. Quando a rotina acaba de ser executada, o *CPU* volta a colocar o *bit* a 1, indicando que voltou ao *user mode* e carrega o contexto do processo que estava a executar.

Próximos capítulos

Gestão de processos

- Escalonar processos e *threads*
- Criar e remover processos do utilizador e do sistema
- Suspender e resumir processos
- Mecanismos para sincronização de processos
- Mecanismos para comunicação entre processos

Gestão de memória

- Rastrear quais partes da memória estão a ser utilizadas e por quem
- Decidir que processos (ou partes deles) e dados são movidos ou removidos da memória
- Alojar e desalojar espaço na memória