INTRODUÇÃO

- O que é um Sistema Operativo (SO) ?
- Objectivos de um SO.
- Alguns conceitos básicos sobre SO's.
- Evolução dos SO's. Tipos de SO's.

O que é um Sistema Operativo ?

Um programa grande e complexo (/ conjunto de programas) que controla a execução dos programas do utilizador e actua como intermediário entre o utilizador de um computador e o *hardware*.

Objectivos principais de um SO:

- **fornecer uma gestão eficiente e segura dos recursos computacionais (gestão + controlo)**
- **fornecer ao utilizador uma máquina virtual mais fácil de programar do que o *hardware* subjacente (conveniência + eficiência)**

O que seria dos programadores sem um sistema operativo ?

A vida de um programador sem um sistema operativo

Cada programador

- **teria de conhecer profundamente o *hardware***
- **teria de saber controlar adequadamente o *hardware***

Todos os programas

- **teriam código para fazer as mesmas coisas**
- **fariam, provavelmente, coisas erradas**

Componentes de um sistema de computação

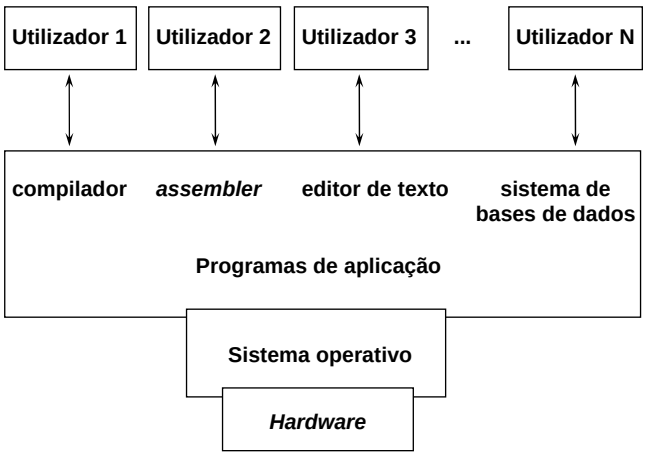
Hardware **- fornece os recursos computacionais básicos (*CPU*, memória, dispositivos de *I/O*, ...)**

Sistema operativo - controla e coordena o uso do *hardware* entre os vários programas de aplicação dos vários utilizadores

Programas de aplicação - definem o modo como os recursos do sistema são usados para resolver os problemas computacionais dos utilizadores (compiladores, sistemas de bases de dados, programas de cálculo comercial, jogos, …)

Utilizadores - (pessoas, máquinas, outros computadores)

Onde *encaixa* o SO ?



Factos importantes

O SO fornece

- **uma interface simples e poderosa**
- **serviços de mais alto nível.**

Os serviços do SO só podem ser acedidos através de chamadas ao sistema.

Os utilizadores e os programas não podem aceder directamente ao *hardware.*

O conjunto de chamadas ao sistema (*API - Application Programming Interface*) é o que os programas "pensam" que é o SO.

Alguns conceitos sobre SO's

Núcleo (*Kernel*)

- O principal programa do SO.
  Contém código para os serviços fundamentais.
  Está sempre em memória principal.

Device Drivers

- Programas que fornecem uma interface
  simples e consistente com os dispositivos de *I/O*
- Podem fazer parte do *kernel* ou não.

Programa

- Um ficheiro do disco contendo código numa linguagem de alto nível ou código-máquina (programa executável).

Processo

- Um programa em execução.
- A colecção de estruturas de dados e recursos do SO detidos por um programa enquanto está a ser executado.

Ficheiro

- Colecção de informação relacionada entre si.
- Unidade lógica de armazenamento.
- O SO mapeia os ficheiros em dispositivos físicos
  onde a informação é gravada de forma permanente (memória secundária).
- Para muitos utilizadores, o sistema de ficheiros é o aspecto mais visível de um SO.

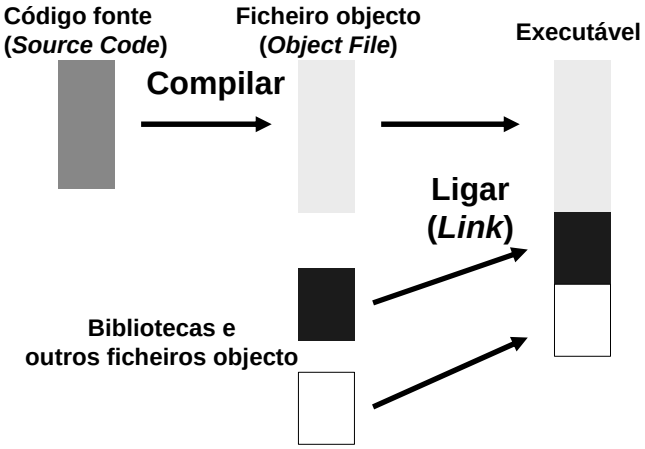
Chamadas ao sistema

- Os programas do utilizador comunicam com o SO e pedem-lhe serviços
  fazendo chamadas ao sistema.
- A cada chamada corresponde uma rotina da biblioteca de sistema.
- Esta rotina coloca os parâmetros da chamada ao sistema
  em locais especificados (ex.: registos do processador) e
  executa uma instrução de *trap* para passar o controlo ao sistema operativo.

Shell

- Interpretador de comandos dados ao sistema operativo.
- Não faz parte do sistema operativo.

Producão de um executável

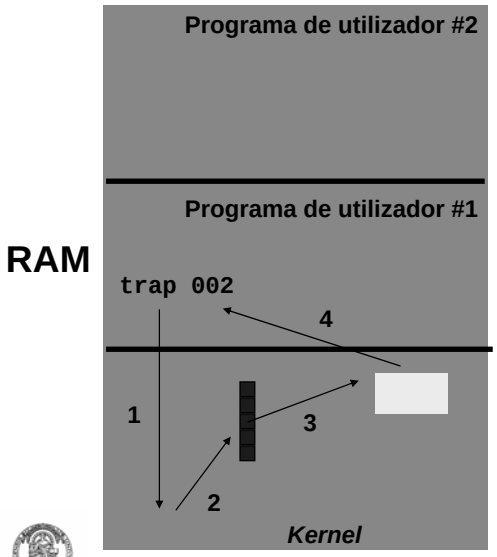



```
#include <sys/types.h>
#include <dirent.h>
#include "ourhdr.h"

int main(int argc, char *argv[])
{
    DIR   *dp;
    struct dirent   *dirp;

    if (argc != 2)
      err_quit("a single argument (the directory name)
                is required");
    if ( (dp = opendir(argv[1])) == NULL)
      err_sys("can't open %s", argv[1]);

    while ( (dirp = readdir(dp)) != NULL)
      printf("%s\n", dirp->d_name);

    closedir(dp);
    exit(0);
}
```

Funções da biblioteca de sistema.

Estas funcões contêm
uma instrução de *trap*.

Execução



Modo utilizador

1. Programa faz *trap* (interrupção gerada por *software*)
2. O SO determina o nº do serviço
3. O serviço é localizado e executado.
4. O controlo retorna ao programa do utilizador

Modo supervisor (*l*sistema *lkernel*)

Evolução dos sistemas operativos

- Processamento em série
- Processamento em lote *(batch)*
- Multiprogramação
- Tempo partilhado (t*ime-sharing*)
- Multiprocessamento
- Sistemas distribuídos
- Sistemas de tempo-real

Primeiros sistemas

(fim dos anos 40, início dos anos 50)

Estrutura

- grandes máquinas (volumosas) controladas a partir de uma consola
- programador/utilizador é o operador
- fita de papel ou cartões perfurados

Software

- interpretador de comandos
- *assemblers*
- *loader* (carregador de programas em memória)
- *linker*
- bibliotecas de rotinas comuns
- compiladores
- *device drivers*

Segurança

- boa

Eficiência

- baixa utilzação da CPU
- tempo de preparação significativo para a execução de um programa

Sistemas *batch* simples

(*batch* - em lote)

- Contratar um operador
- O utilizador deixou de ser o operador
- Juntar leitor de cartões perfurados
- Reduzir o tempo de computação pondo tarefas (*jobs*) semelhantes no mesmo lote (ex.: todos os programas em FORTRAN a serem compilados)
- Sequenciamento automático das tarefas - transfere automaticamente o controlo de uma tarefa para a seguinte (primeiro sistema operativo rudimentar)

- **Monitor residente**
  - » controlo inicial pertence ao monitor;
  - » transfere o controlo para a tarefa;
  - » quando a tarefa termina o controlo regressa ao monitor.
- **Problemas**
  - » Como é que o monitor reconhece a natureza da tarefa ? (ex. Fortran versus Assembly) ou que programa executar ?
  - » Como é que o monitor distingue
    - uma tarefa de outra ?
    - o programa dos dados ?

Solução: introduzir cartões de controlo

Sistemas *batch* simples (cont.)

Cartões de controlo

- Cartões especiais que indicam ao monitor residente que operações executar
  - $JOB - 1º cartão do *job*
  - $FTN - executar o compilador de Fortran
  - $RUN - executar o programa
  - $END - último cartão do *job*
- Caracteres especiais (ex.:$ na coluna 1) distinguem os cartões de controlo dos cartões de programas ou de dados

Partes do monitor residente

- Interpretador dos cartões de controlo responsável por ler e executar as instruções dos cartões.
- Carregador (*loader*) carrega os programas de sistema e os de aplicação na memória.
- *Device drivers* conhecem as características especiais e propriedades de cada um dos dispositivos de *I/O* do sistema.

Sistemas *batch* simples (cont.)

Problema:

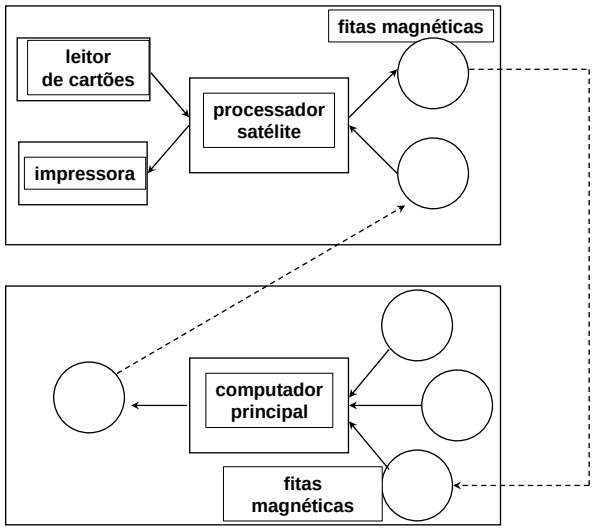
- Baixo desempenho dado que as entradas/saídas não podem ser feitas em sobreposição com o cálculo, e o leitor de cartões é muito lento.

Solução:

- Operação *off-line* (não-ligada ao computador principal). Acelerar a computação carregando as tarefas em memória a partir de uma fita magnética sendo a leitura de cartões e a impressão feita *off-line.*

Vantagens da operação *off-line*

- O computador principal não fica limitado pela velocidade dos leitores de cartões e impressoras, mas apenas pela velocidade (maior) de leitura das fitas magnéticas.
- Possibilidade de usar vários sistemas de transferência leitor→fita e fita→impressora para o mesmo processador.

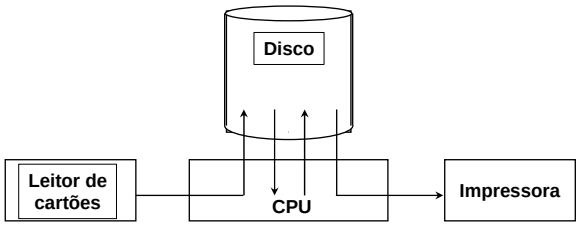


Outros melhoramentos:

- **Sobreposição das operações de entrada e de saída (I/O)**
  (coincide c/ a introdução de canais de DMA, controladores de periféricos, ...)
- **Sobreposição do cálculo com operações de I/O**
  (utilização de *buffers* de I/O em memória principal)

Spooling (Simultaneous Peripheral Operations On-Line)

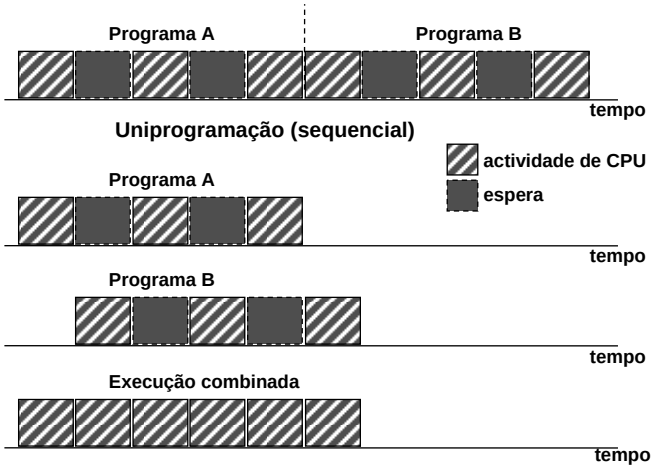
- **Forma mais sofistacada de *buffering*: usar discos p/guardar temporariamente as I/O's.**
- **Permite sobrepor a fase de cálculo de um processo c/ a fase de I/O de outro.**



Sistemas *batch* c/multiprogramação

Várias tarefas são mantidas em memória simultâneamente, e a *CPU* é partilhada entre elas

Quando o programa actual fica à espera que uma operação de *I/O* se complete, o processador pode executar outro programa



Multiprogramação

Multiprogramação: execução interlaçada de processos.

Algumas características do SO necessárias para multiprogramação

- **Escalonamento da *CPU***
  o sistema deve escolher entre os várias processos prontos a executar, aquele que vai ser executado
- **Gestão de memória**
  o sistema deve alocar a memória aos diferentes processos
- **Gestão de *I/O***
- **Protecção**
  não deve haver possibilidade de os processos se afectarem mutuamente em todos os níveis: escalonamento de CPU, memória acessível, *I/O*, ...

Sistemas de tempo partilhado (*time-sharing*) - Computação interactiva

- Vários utilizadores simultâneos,
  cada um com a impressão de que tem o computador só para si.
- A *CPU* é partilhada entre diversas tarefas
  que são mantidas em memória e em disco
  (a *CPU* só é alocada a uma tarefa se ela estiver em memória).
- A comutação entre tarefas
  ocorre com uma elevada frequência.
- Para se obter tempos de resposta razoáveis
  as tarefas podem ser transferidas de e para o disco
  (*swapping* e memória virtual).
- É possível a comunicação *on-line* entre o utilizador e o sistema;
  quando o sistema operativo termina a execução de um comando,
  procura a próxima "instrução de controlo" vinda do teclado.
- Deve existir um sistema de ficheiros *on-line*
  para que os utilizadores possam aceder aos programas e aos dados.

Sistemas de computação pessoal

(anos 80)

- *Computador pessoal* -
  sistema de computação dedicado a um único utilizador
- Dispositivos de *I/O*
  - » teclado, rato, écran, impressora
- O sistema operativo não era, em geral, multitarefa.
- Possível adoptar tecnologia
  desenvolvida para sistemas operativos maiores;

  frequentemente existe um único utilizador
  e não são necessários mecanismos avançados
  de gestão da *CPU* ou de protecção.

Sistemas multiprocessador

Sistemas com vários processadores em comunicação entre si, na mesma máquina, sob o controlo integrado de um sistema operativo

Sistema *tightly coupled*

- **Os processadores partilham a memória e o relógio.**
- **A comunicação é vulgarmente feita através de memória partilhada.**

Vantagens dos sistemas multiprocessador

- **Maior desempenho e poder computacional.**
- **Maior fiabilidade:**
  se um processador falha, os outros podem tomar conta do s/trabalho (*gracefull degradation,* sistemas *fail-soft*).
- **Flexibilidade (possibilidade de reconfiguração).**
- **Crescimento modular.**
- **Especialização funcional.**
- **Melhor relação custo/desempenho**
  (comparado com máquinas uniprocessador equivalentes).

Sistemas multiprocessador (cont.)

Multiprocessamento simétrico

- **Todos os processadores são funcionalmente idênticos.**
- **Os recursos (memória, dispositivos de *I/O*, …) estão disponíveis para todos os processadores.**
- **O SO também é simétrico no sentido em que qualquer processador pode executá-lo.**
- **Muitos processos podem ser executados em simultâneo sem degradação do desempenho.**

Multiprocessamento assimétrico

- **A cada processador é atribuído uma tarefa (*task*) específica; a gestão é feita por um "processador-mestre" que atribui trabalho a "processadores-escravo".**
- **Mais comum em sistemas de muito grande porte.**

Sistemas de rede

- Os utilizadores estão conscientes da existência de múltiplos computadores ligados por uma rede (o que pode não acontecer num sistema distribuído).
- Cada máquina tem o seu sistema operativo local.
- Os utilizadores podem aceder a outras máquinas.
- Os sistemas operativos de rede não diferem significativamente dos sistemas operativos para uniprocessadores.

  Precisam de um controlador de rede, de *device drivers* e de programas para *login* remoto e acesso a ficheiros remoto.

Sistemas distribuídos

Distribuir a computação entre diversos processadores fisicamente separados (máquinas diferentes).

Sistemas *loosely coupled*

- Cada processador tem a sua memória local.
- Os processadores comunicam entre si através de linhas de comunicação (ex.: barramentos de alta velocidade, …).

Vantagens dos sistemas distribuídos

- Partilha de recursos (impressoras, ficheiros,...).
- Aumento da velocidade de computação - partilha de carga.
- Fiabilidade.
- Comunicação.

Sistema operativo distribuído

- Sistema operativo comum (partilhado por todas as máquinas).
- O utilizador acede a recursos remotos como se fossem locais (podendo não ter consciência disso).
- O sistema aparece perante os utilizadores como se se tratasse de um sistema uniprocessador.

Sistemas de tempo-real (*real-time*)

- Frequentemente usados em aplicações dedicadas
  como o controlo de experiências científicas, controlo industrial, controlo de aviões, …
- Restrições de tempos de execução muito bem definidas
  Objectivo principal: resposta rápida aos acontecimentos
- Sistemas *hard real-time*
  - » Garantem que as tarefas críticas são completadas num tempo especificado.
  - » Pouca ou nenhuma memória secundária;
    dados armazenados em RAM ou ROM.
  - » A sua funcionalidade não é suportada por sistemas operativos de uso geral.
- Sistemas *soft real-time*
  - » As tarefas críticas têm prioridade sobre as outras
    e mantêm essa prioridade até se completarem.
  - » As tarefas têm um tempo especificado
    mas podem ser completadas para além dele.
  - » A sua funcionalidade pode ser suportada
    por sistemas operativos de uso geral.
  - » Úteis em aplicações multimedia, de realidade virtual, … .