Sistemas Operativos I

História dos Sistemas Operativos

Maria João Viamonte / Luis Lino Ferreira

Fevereiro de 2006

Para que serve um Computador ?

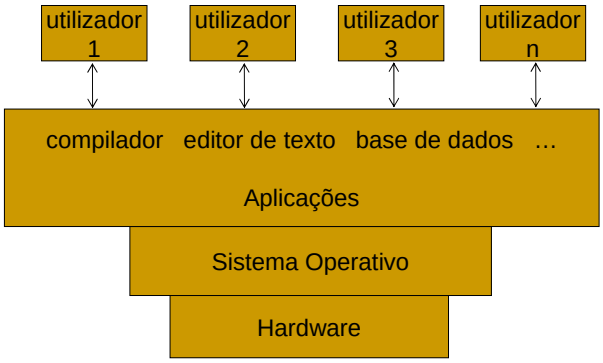
- Para facilitar a vida aos utilizadores
- Para executar programas (aplicações)

Sistema Computacional

- Componentes:
  - Hardware: UCP, memória, dispositivos de I/O
  - Software
    - Sistema Operativo: controla e coordena a utilização do hardware durante a execução de várias aplicações
    - Aplicações: compiladores, sistemas de bases de dados, programas diversos, Jogos, …
  - Utilizadores: pessoas,outros computadores, máquinas

Sistema Computacional

- **SO é um programa que actua como intermediário entre os utilizadores e o hardware**



Sistema Operativo

- Definição
  - Silberschatz “É um programa intermediário entre o utilizador e o hardware”
  - MilenKovic “Colecção organizada de software, consistindo em rotinas de controlo relativas ao hardware do computador que permitem fornecer um ambiente homogéneo para a execução de programas”

Sistema Operativo

- Objectivos
  - Executar comandos e programas do utilizador
  - Facilitar o uso da máquina
  - Utilizar o hardware da máquina de uma forma eficiente (disco, placa gráfica, memória, etc.)
  - Fornecer uma Interface com o SO
  - Gerir recursos
  - Controlar a execução de programas e as respectivas operações de I/O
  - …

Sistema Operativo

- Portanto
  - SO deve colocar o hardware à disposição dos programas e utilizadores, mas de uma forma:
    - Conveniente,
    - Justa,
    - Protegida,
    - Eficiente,
    - ....

Tipos de Computadores

- Super computadores
- *Mainframes*
- Mini computadores
- *Workstations*
- Pc's (microcomputadores)
- Computadores de bolso

Evolução dos Sistemas Operativos

- Processamento Série (*Serial Processing*)
  - Monitor de Controlo
- Processamento em Lotes (*Batch*)
- Multiprogramação

Processamento Série

- Máquinas simples, sem qualquer tipo de sistema operativo
- Os programas eram introduzidos pelo utilizador e depois executados pela máquina
- Hardware baseado em tubos de vácuo
- *Input* através de cartões perfurados
- *Output* através de lâmpadas

Processamento Série

- Baixa produtividade
  - era sempre necessário introduzir tudo à mão
- Todas as operações tinham que ser definidas pelos programas

Evolução ⇒ Primeira aproximação a um SO foi um programa utilitário

Monitor de Controlo

Monitor de Controlo

- Atribuição a cada utilizador de quotas de tempo de utilização da máquina, dispondo da máquina como um todo
- Permitia ao utilizador carregar os seus programas em memória, editá-los e verificar a sua execução
- Execução das operações necessárias através de comandos do monitor

Monitor de Controlo

- No final da sessão guardavam os programas e resultados sob a forma de listagens, fitas de papel perfuradas ou, nos sistemas mais evoluídos, em fita magnética
- Rotinas de I/O reutilizáveis

Monitor de Controlo

- Um monitor típico era composto por um conjunto de rotinas utilitárias que facilitavam a interacção (operação) com máquina:
  - Interpretador de uma linguagem de comando que permite fazer executar os restantes módulos
  - Compilador
  - Tradutor de linguagem simbólica (***Assembler***)
  - Editor de ligações (***Linker***)
  - Carregador de programas em memória (***Loader***)
  - Rotinas utilitárias para o controlo de periféricos: consola; leitor de cartões; leitor/perfurador de fita de papel; bandas magnéticas

Monitor de Controlo

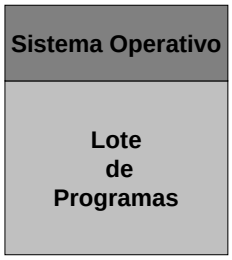
- Ineficiente
  - Durante a maior parte do tempo o processador está inactivo, à espera de um comando ou a efectuar uma operação de I/O
  - O tempo de execução de um programa é gasto essencialmente nas operações de I/O

Processamento em Lotes (*Batch*)

- Automatizavam a sequência de operações que envolvem a execução de um programa:
  - O programa é enviado ao operador do computador
  - O operador junta o programa ao conjunto de programas existentes, criando um lote
  - Cada lote de programas é executado sequencialmente pelo computador
  - Os resultados são fornecidos ao operador à medida que os programas vão acabando

Processamento em Lotes (*Batch*)

- A memória está dividida em duas partes:
  - Sistema Operativo do computador
  - Lote de programas que está a correr



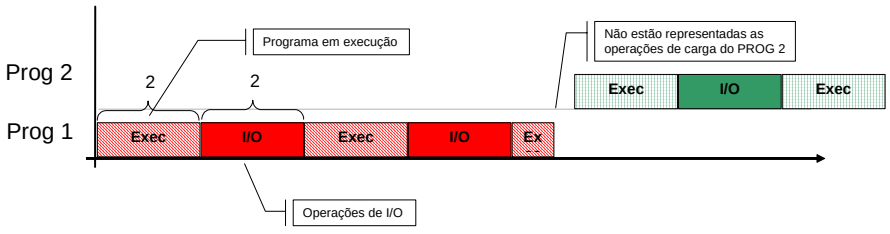
Processamento em Lotes (*Batch*)

- Problemas
  - Não é possível a interacção entre um programa a correr e o utilizador
  - A capacidade de processamento da unidade central de processamento (UCP) evolui exponencialmente
  - **No entanto, os dispositivos de I/O são muito lentos**:
    - Um leitor de cartões lê 20 cartões/s
    - Os primeiros discos rígidos também são muito lentos
  - O tempo de execução de um programa é predominantemente determinado pelas operações de I/O

Processamento em Lotes (*Batch*)

- Solução
  - Para optimizar a utilização da UCP passou a fazer-se a recolha dos dados num computador auxiliar onde eram lidos, para uma banda, os cartões dos diversos trabalhos
  - A banda era colocada no computador central e executados os programas, **produzindo igualmente os ficheiros de saída para outra banda** que, por sua vez, era tratada pelo computador mais pequeno para optimizar o tempo de impressão

Processamento em Lotes (*Batch*)



Utilização do processador:
Prog 1: (2+2+1)/(2+2+2+2+1) = 0,55
Prog 2: (2+2)/(2+2+2) = 0,66

Processamento em Lotes (*Batch*)

- Evolução:
  - Periféricos passaram a poder executar operações autónomas, avisando o processador do fim da sua execução através do mecanismo de interrupções
  - Possibilidade de notificar assincronamente o processador de que uma dada operação terminou
  - As operações de I/O podem prosseguir em paralelo com a execução de um programa que apenas é interrompido para iniciá-las e para tratar a sua terminação
  - Paralelamente, os periféricos de armazenamento de dados sofreram uma evolução significativa, deixando de ser meros dispositivos sequenciais (bandas) para se tornarem verdadeiras memórias secundárias com possibilidade de endereçamento aleatório (tambores e discos)

Multiprogramação

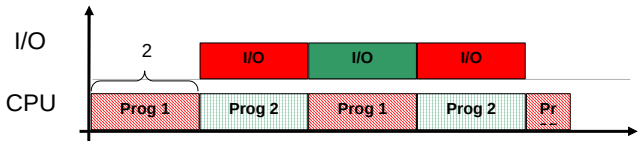
- Vários programas (*jobs*) são carregados para memória central, e o tempo da UCP é repartido por eles:
  - A execução concorrente de vários programas permite optimizar a utilização da UCP
  - Sempre que um programa/processo não necessita da UCP, por exemplo para ler dados de um ficheiro, a sua execução fica bloqueada até que os sectores com os dados sejam lidos e transferidos para a memória, passando a execução para um outro programa/processo

Multiprogramação

- Esta solução torna os sistemas multiprogramados permitindo que diversos programas estejam simultaneamente activos
- Os diversos programas necessitam de estar na memória central para facilmente se mudar de contexto

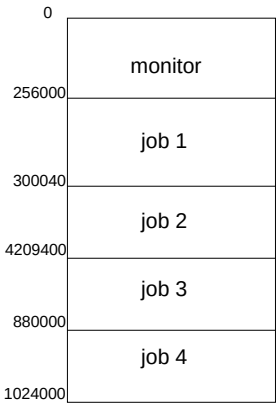
Multiprogramação

- Exemplo



- Utilização da UCP aumenta
- Neste caso particular passa a 100%

Multiprogramação



Multiprogramação

- Funções controladas pelo SO:
  - I/O através de rotinas fornecidas pelo SO
  - Gestão da memória
    - Alocar memória para os vários programas/processos
  - Escalonamento da UCP
    - Decidir que programa/processo vai entrar em funcionamento

Multiprogramação

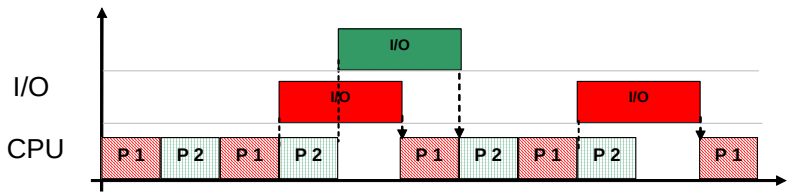
- Problemas:
  - Os primeiros sistemas de multiprogramação não permitiam a interacção com o utilizador

Sistemas de partilha de tempo - Computação Interactiva

- Solução
  - A comutação entre processos passa a ser feita periodicamente ou quando os processos necessitam realizar tarefas de I/O
  - Extensão dos sistemas multiprogramados de modo a permitir a partilha do sistema entre vários utilizadores, assim como, a interacção entre o utilizador e o programa

Sistemas de partilha de tempo - Computação Interactiva

- Exemplo



Sistemas de partilha de tempo - Computação Interactiva

- Os sistemas interactivos obrigaram a uma grande reformulação dos conceitos subjacentes aos SOs, passando a conferir grande importância a aspectos até aí considerados como secundários, tais como:
  - o sistema de ficheiros
  - a protecção dos dados
  - a necessidade de acesso a informação partilhada e actualizada
  - a linguagem de interacção com o sistema
  - etc.

Sistemas de partilha de tempo - Computação Interactiva

- Características
  - Cada programa é visto como um processo pelo SO
  - A comutação entre processos é suficientemente rápida de modo a permitir a interacção em "tempo real" entre um processo e um utilizador
  - Os processos em execução são substituídos em função do seu último período de ocupação da UCP

Sistemas de Secretária (*Desktop*)

- Objectivos
  - Permitir a um utilizador isolado o acesso a um sistema de computação
  - Maximizar a conveniência de utilização
- Primeira geração
  - Mono-utilizador
  - Sem multitarefa
  - Ex: MS-DOS

Sistemas de Secretária (*Desktop*)

- Geração 1.5
  - Mono-utilizador
  - Algumas capacidades de multitarefa, mas o mecanismo de escalonamento ainda não permite a execução concorrente
  - Sem multitarefa
  - Ex: Windows 3.0 e 3.1

Sistemas de Secretária (*Desktop*)

- Segunda Geração
  - Multi-utilizador
    - Sistema de ficheiros multi-utilizador
  - Multi-tarefa
  - Conexão à rede
  - Exemplos
    - Windows 95/NT/XP
    - Linux
    - MAC
    - OS/2

Sistemas Multi-Processador

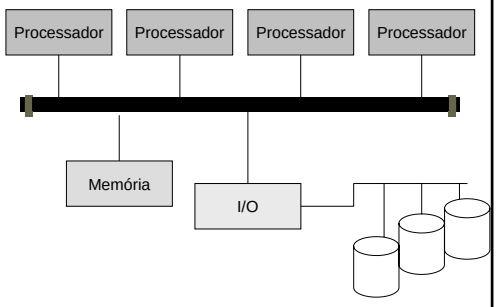
- O computador pode utilizar dois ou mais UCPs, partilhando:
  - o barramento
  - o relógio
  - a memória
  - os periféricos
  - o disco
  - etc.

Sistemas Multi-Processador

- Vantagens:
  - Maior performance
  - Economia de escala
  - Maior fiabilidade

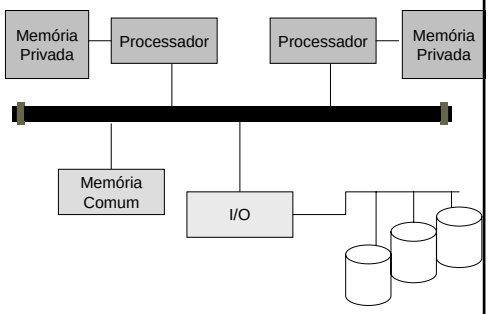
Sistemas Multi-Processador

- *Symmetric multiprocessing* (SMP):
  - Cada processador corre uma cópia idêntica do sistema operativo
  - Podem correr em simultâneo vários processos sem existir degradação no desempenho
  - A maior parte dos sistemas operativos modernos suportam SMP



Sistemas Multi-Processador

- *Asymmetric multiprocessing* (AMP):
  - Processador-mestre corre o sistema operativo, escalona o trabalho dos processadores-escravos
  - Processadores-escravos correm aplicações, sendo atribuída uma tarefa a cada um, é comum em sistemas extremamente grandes



Sistemas Distribuídos

- Sistema em que a comunicação entre máquinas (UCPs) cooperantes é feita através de uma rede de comunicação

Sistemas Distribuídos

- Sistemas Cliente-Servidor
  - O servidor fornece funcionalidades a outras máquinas (clientes)
- Exemplos:
  - Servidor de ficheiros
  - *Web Server*
  - *Grids*

Sistemas Distribuídos

- Sistemas *Peer-to-Peer*
  - Cada máquina tem responsabilidades equivalentes
- Exemplos:
  - *Kazaa*

Clusters

- As máquinas estão ligadas entre si utilizando uma rede local de alto débito
- Um *cluster* de máquinas poderá ser visto pelo utilizador como uma máquina "grande"
- Utilizações:
  - Cálculos intensivos
  - Simulações
  - Servidores Web de alto desempenho, com tolerância a falhas

Sistemas de Tempo-Real

- Cada tarefa do sistema têm associadas restrições temporais (*deadlines*)
- Normalmente utilizados em sistemas dedicados
- Podem ser classificados em sistemas:
  - *Hard-real time*:
    - Onde o não cumprimento de um *deadline* leva à falha do sistema
  - *Soft-real time*:
    - Onde o não cumprimento de um *deadline* leva apenas a uma degradação da saída do sistema

Sistemas de Tempo-Real

- *Hard-real time*:
  - Controlo industrial
  - Sistemas automóveis
  - *Avionics*
  - *Soft-real time*:
    - Sistemas multimédia
    - Dispositivos de rede (*router*)

Sistemas de Bolso

- Características:
  - Memória pequena
  - Processador lento
  - Ecrã pequeno
  - Sistemas de I/O limitados
- Exemplos:
  - PDAs
  - Telemóveis

Sistemas Operativos I

História dos Sistemas Operativos

Maria João Viamonte / Luis Lino Ferreira

Fevereiro de 2006