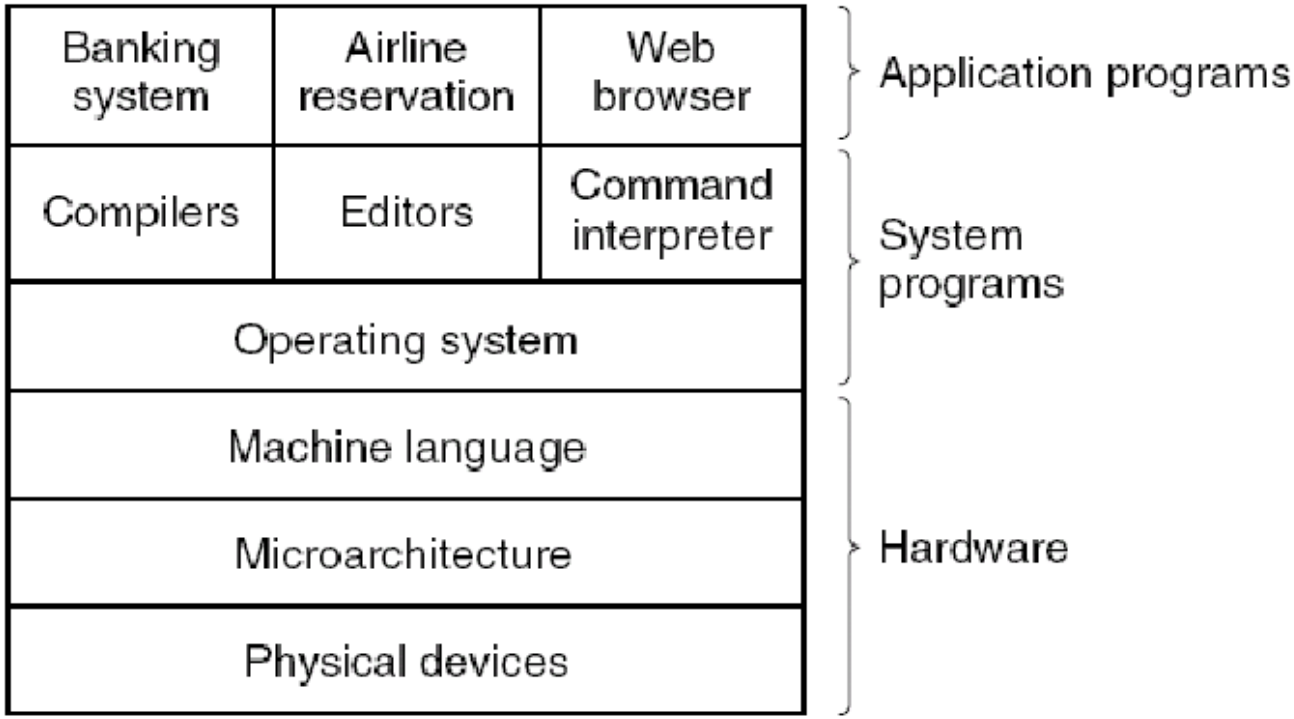
Componentes de um sistema computacional moderno



O que é um Sistema de Operação?

- Para responder a esta questão:
  - vamos dividir a turma em 4 partes
    - CPU
    - memória
    - programas (utilizador(es))
    - dispositivos de entrada e saída (impressoras, discos, monitores de vídeo etc)

O que é um Sistema de Operação?

- Vamos tentar responder às questões colocadas nas seguintes situações:
  - vários programas querem utilizar a CPU e a memória
    - posso executar todos os programas simultaneamente?
    - qual programa deve executar primeiro?
    - Se não houver memória para todos o que fazer?
  - vários programas querem utilizar a impressora
    - posso ter vários utilizadores imprimindo? Ou escrevendo no ecrã? Ou escrevendo em um ficheiro em disco?
    - Qual deve ser a ordem para imprimir?
  - seu programa é maior do que a memória RAM
    - ele pode executar?

O que é um Sistema de Operação?

- Uma máquina estendida (ou virtual), mais fácil de programar do que o hardware.
- Desta forma, um SO é um programa (ou conjunto de programas) que atua como intermediário entre o utilizador e o hardware do computador.
- Atua como um gerenciador de recursos de hardware:
  - gestor de memória
  - gestor de I/O
  - escalonador de CPU
  - gestor de tarefas (multi-tarefa/multi-programação)
  - gestor de ficheiros
  - gestor de comunicações

O que é um Sistema de Operação?

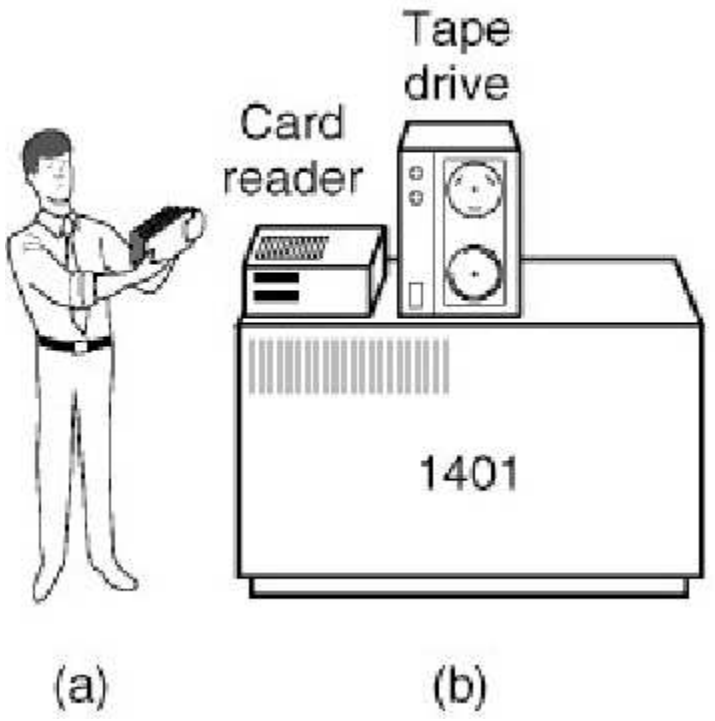
Alguns sistemas consideram também as funções abaixo como fazendo parte de um SO, porém depende da estrutura e implementação do sistema.

- Suporte multimédia
- Interface com o utilizador (window manager)
- Internet browser

Gerações de Sistemas de Operação

- 1ª geração (1945 - 55)
  Tubos à vácuo e “plugboards”
- 2ª geração (1955 - 65)
  Transistores e sistemas em “batch”
- 3ª geração (1965 - 80)
  circuitos integrados e multiprogramção
- 4ª geração (1980 - Presente)
  computadores pessoais

Sistema em “Batch” (1)

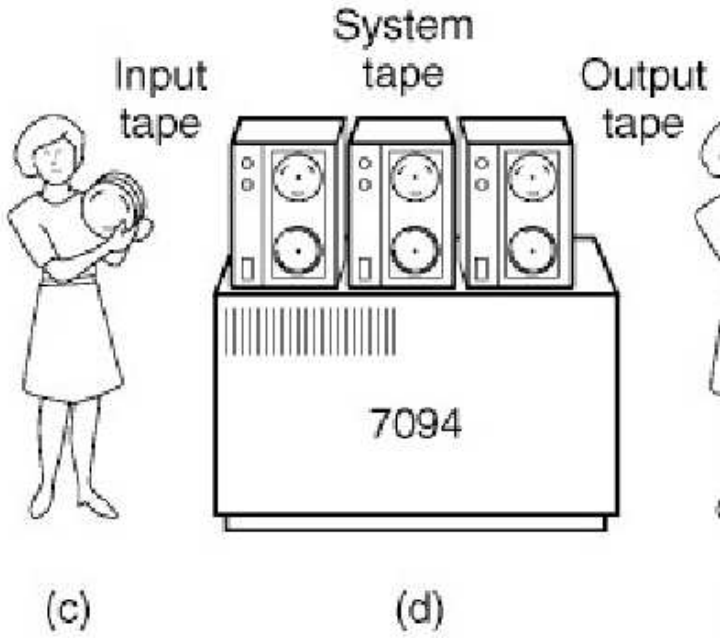


(a) Utilizador leva cartões para leitura

(b) Sistema lê e armazena “batch jobs” numa unidade de fita

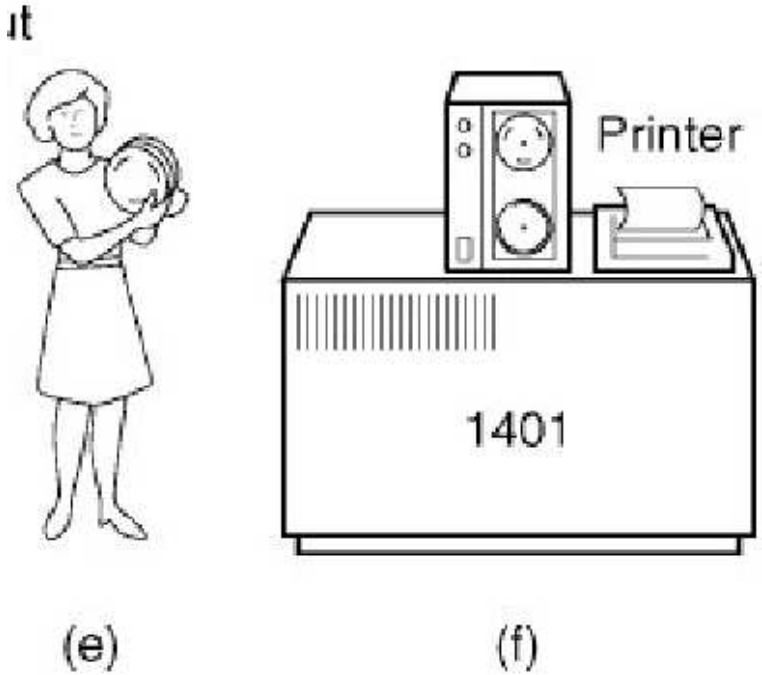
“job”: designação dada a um programa que deve executar em “batch”.

Sistema em “Batch” (2)



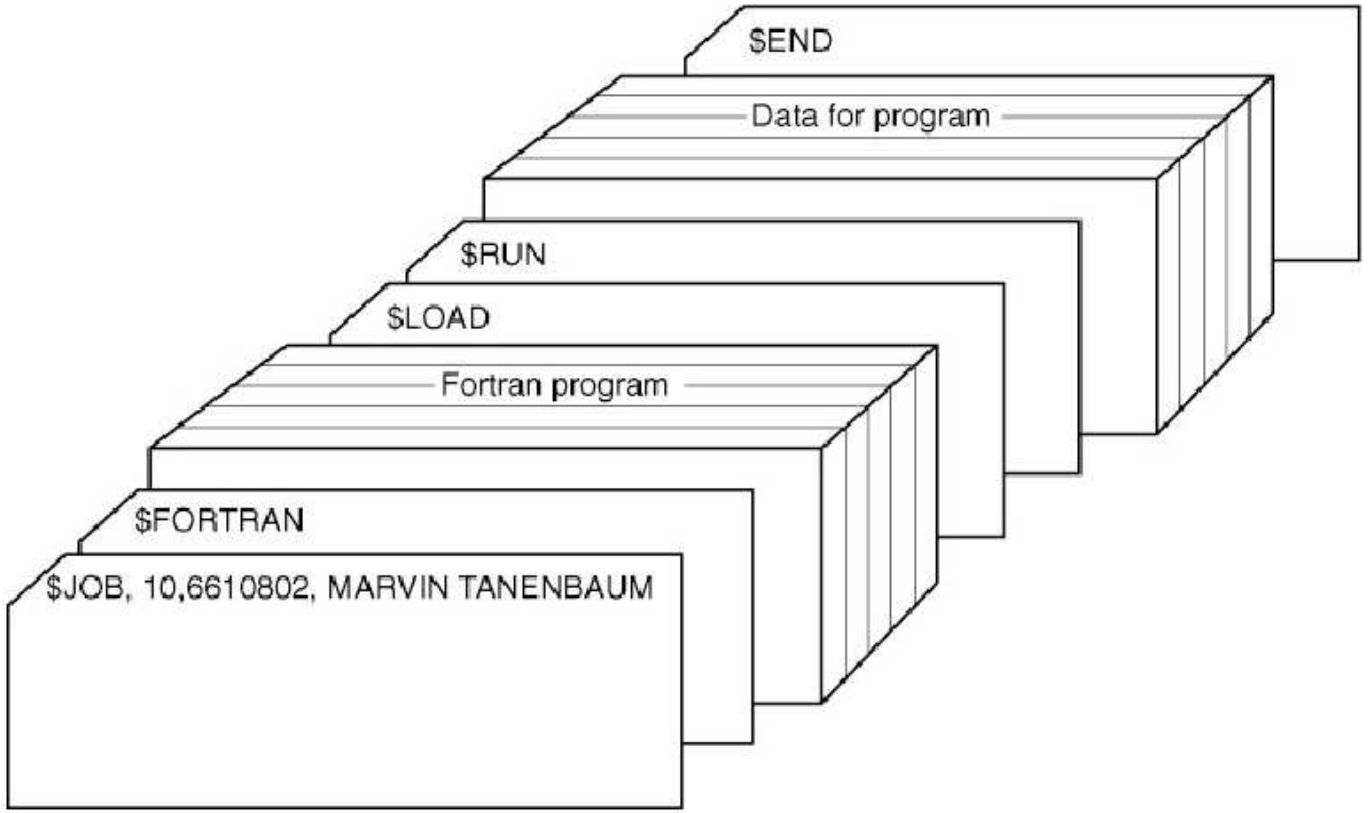
(c) Utilizador leva fita para correr jobs
(d) Computador executa

Sistema em “Batch” (3)



(e) Utilizador leva resultados do programa para imprimir
(f) Computador imprime

Sistema em “Batch” (4)



Estrutura de um job típico FMS

Funções de um SO

Um SO cumpre em geral duas funções:

- Coordenador:
  - administra a partilha dos recursos de um computador, e.g. controla o acesso concorrente à memória, ao sistema de ficheiros e à rede.
  - permite que vários utilizadores/aplicações usem o computador “em simultâneo”, de forma eficiente e justa.
  - resolve pedidos conflituosos de programas e utilizadores.
- Facilitador:
  - torna a tarefa de programação de aplicações, mais simples, rápida e menos sujeita a erros.
  - esconde a complexidade do hardware dos utilizadores, fornecendo-lhes interfaces de acesso mais convenientes de usar.

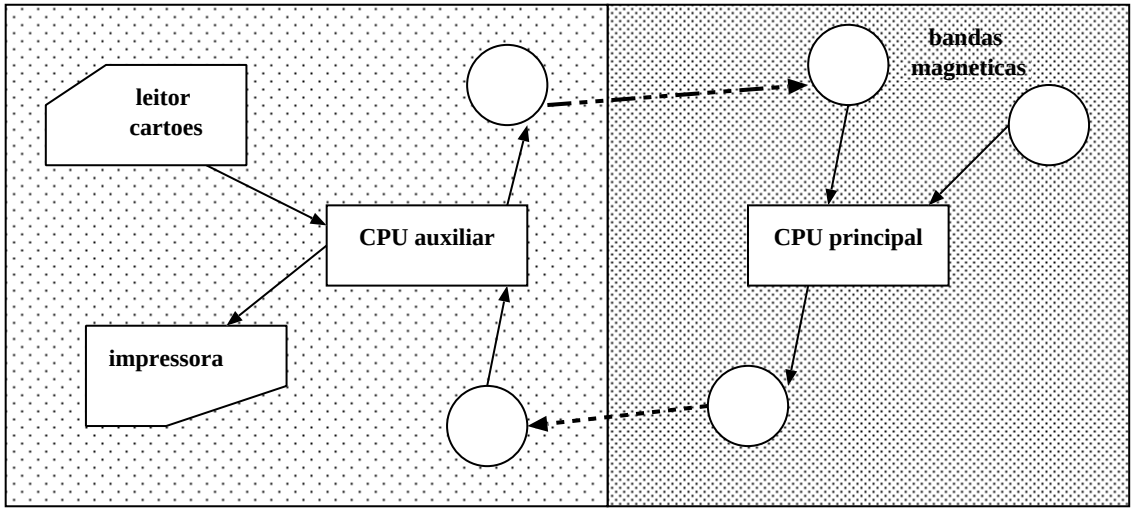
Os primeiros sistemas (início anos 50)

- Estrutura:
  - máquinas enormes acessíveis por uma consola
  - um utilizador e um programa em execução de cada vez
  - programador/utilizador era o operador da máquina
  - fitas perfuradas e cartões
- Software:
  - assembladores, compiladores
  - carregadores (loaders), ligadores (linkers)
  - bibliotecas de procedimentos mais comuns (libraries)
  - controladores de periféricos (device-drivers)
- Seguros
- Uso ineficiente de recursos:
  - baixo aproveitamento da CPU
  - tempo de arranque muito significativo

Sistemas Batch Simples

- adotar um operador mais experiente
  → utilizador deixa de ter acesso ao hardware.
- agrupar programas dos utilizadores e execução gerida pelo operador.
  → melhora eficiência se os programas tiverem requisitos semelhantes
- e se um dos programas parar/falhar na sua execução?
  → operador tem de intervir.
- monitor residente: primeiro programa de um *SO rudimentar* para controlar a execução de programas:
  → (1) o monitor reside em memória e inicia execução
  → (2) transfere o controlo para outro programa
  → (3) quando o programa termina, transfere o controlo novamente para o monitor, que volta a (2).
- Dificuldades:
  1. Como é que o monitor sabe qual a natureza de uma tarefa ou qual o programa a executar?
  2. Como é que um monitor distingue: uma tarefa de outra? os dados do programa?
- Solução: através de cartões de controlo (como visto nos slides anteriores)

Operação off-line: sobreposição de execução com I/O

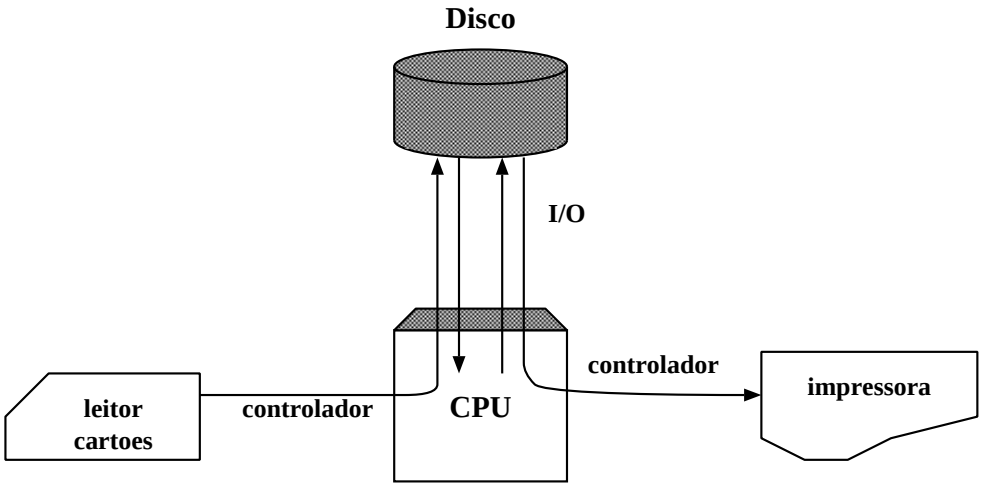


- Vantagens:
  - sobreposição da execução do CPU e operações I/O, embora com 2 máquinas independentes.
  - o computador não fica condicionado pela menor velocidade dos leitores de cartões e impressoras de linhas, depende da velocidade das unidades de bandas magnéticas.
  - torna o sistema independente dos periféricos.

Operação off-line (cont.)

- Como sobrepor execução do CPU e operações I/O numa só máquina?
  - buffering – interpondo um buffer entre periférico e CPU.
  - Após o CPU ler do buffer os dados e começar a operar sobre estes, ambos CPU e controlador do periférico podem operar em simultâneo, estando ambos ocupados.
  - na prática, CPU é muito mais rápida e tem de esperar!

Spooling

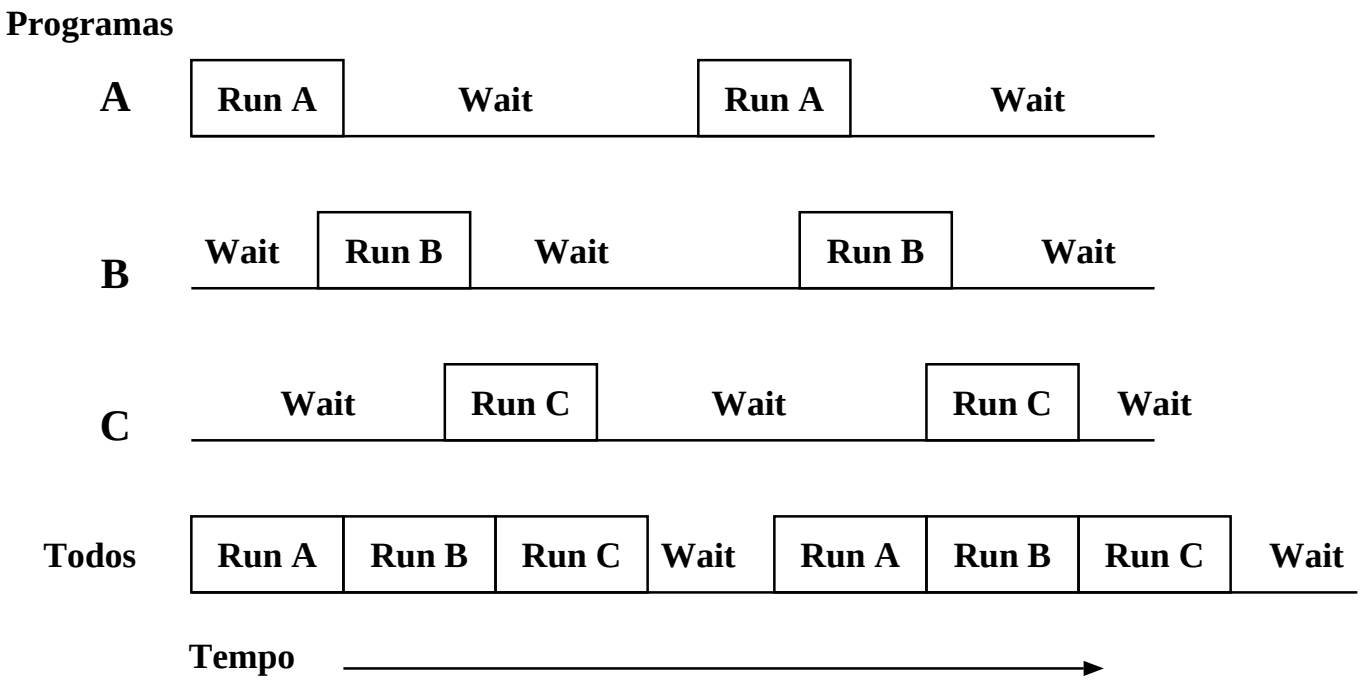


- usa o disco como um buffer de grandes dimensões.
- sobrepõe operações de I/O de uma tarefa com a execução de outras tarefas. Enquanto executa uma tarefa, o SO:
  - → lê a próxima tarefa do leitor de cartões para o disco – fila de tarefas para execução.
  - → O SO seleciona desta fila a próxima tarefa para execução – scheduling de tarefas.
  - → escreve os resultados da tarefa anterior para a impressora.
- Um exemplo de spooling em SOs atuais é o envio de tarefas para a impressora.

Multiprogramação de Sistemas Batch

- **Sistemas uni-programados:**
  → uma tarefa era executada de princípio a fim.
  → ordem de execução das tarefas correspondia à ordem de leitura.
  → o CPU não podia passar para uma nova tarefa se a actual estivesse parada à espera de I/O.
- **Sistemas multiprogramados:**
  - os discos permitem acesso directo (não sequencial) e rápido, pelo que é possível pensar numa estratégia de escalonamento de tarefas (scheduling).
  - **a ideia é aumentar a utilização do CPU**, para isso:
    → ter em memória várias tarefas (programas) prontas a serem executadas.
    Requer capacidade de gestão de memória pelo SO.
    → cada tarefa é executada durante um certo tempo, caso seja interrompida o SO pega automaticamente numa nova. A tarefa interrompida será retomada logo que possa continuar a sua execução.
    Que tarefa escolher? CPU scheduling.
    → a execução concorrente de tarefas requer do SO a capacidade de limitar a possibilidade de interferência entre as tarefas.

Exemplo de multiprogramação



Multi-tarefa (Multitasking) ou Time-Sharing

- **Sistemas Multi-tarefa:**
  - estende o conceito de multiprogramação.
  - o CPU é partilhado por várias tarefas prontas a executar.
  - cada tarefa executa durante uma fração (quantum) de tempo, Δt.
  - o SO comuta frequentemente entre várias tarefas em execução de forma que cria a ilusão de estar a executar todas em simultâneo.
  - vai permitir a interação dos utilizadores com os respectivos programas em execução.
- **Sistemas time-sharing:**
  - uso interativo de um computador
  - muitos utilizadores a partilharem o computador.
- **Multiprogramação em Batch versus Time Sharing**
  - Mutiprogramação em Batch: *objetivo principal é maximizar uso de CPU*.
  - Time-sharing: *objetivo principal é minimizar tempo de resposta*.

Sistemas de Computador Pessoal

- computadores pessoais – sistema dedicado a um utilizador.
- periféricos I/O: teclado, rato, monitores, impressoras.
- até recentemente, não possuíam os requisitos necessários para suportar um sistema multi-tarefa.
- Sistemas operativos: MSDOS, MacOS, Windows, Windows-NT,...Windows2000, Linux.

Sistemas Paralelos (multiprocessadores)

- usar vários CPUs para executar mais tarefas por unidade de tempo.
- **sistemas de memória partilhada** – os CPUs partilham uma memória global e é através dela que comunicam.
- **sistems de memória distribuída** – os CPUs têm a sua memória local, não há partilha. Os CPUs comunicam entre si por troca de mensagens.
- **Vantagens:**
  - aumenta o *throughput* – número de tarefas executadas por unidade de tempo.
  - melhora a fiabilidade.
  - multiprocessamento simétrico:
    cada processador corre uma cópia idêntica do SO.

Sistemas tempo-real

- Outra forma de um SO, normalmente usado como controlador de periféricos dedicado a uma tarefa.
- Restrições de tempo, bem definidas. Ou se realiza dentro do tempo estabelecido ou o sistema falhará.
- Exemplo: controlo de um robot submarino.

Outras classificações de sistemas de operação

- monolítico (antigos sistemas: p.e. DOS, FreeBSD)
- micro-kernel: conjunto básico, pequeno de serviços (Amoeba)
- exo-kernel: exporta micro-kernels como máquinas virtuais para outras máquinas (Aegis)
- distribuído: faz migração automática de tarefas e co-escalonamento (p.e. Glunix, Mosix)