FACULDADE DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Fundamentos de Sistemas de Operação

LEI - 2023/2024

Vitor Duarte
Mª. Cecília Gomes

1

Aula 1

Parte I

- Objetivos e programa
- Funcionamento
- Bibliografia
- Avaliação

Parte II

- Introdução aos SO
  - OSTEP, cap. 2

FACULDADE DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

2

Informações gerais

- NÃO ESQUECER! Tudo está no CLIP.
  - Horário, objetivos, programa, bibliografia, avaliação, datas de testes, etc.....
- Atenção aos avisos
  - Nas aulas, no CLIP, por e-mail
- Pode sempre contactar os docentes
  - Pessoalmente, idealmente no horário de dúvidas
  - e-mail (end. e-mails no CLIP)

FSO no curso (e u.c. relacionadas)

- 1º semestre:
  - Introdução à Programação
- 2º semestre:
  - Arquitetura de Computadores
  - Programação Orientada pelos Objetos
- 3º semestre:
  - **Fundamentos de Sistemas de Operação**
  - Algoritmos e Estruturas de Dados
- Seguintes:
  - Ling. Ambientes de Programação, Redes de Computadores, Sistemas Distribuídos, Concorrência e Paralelismo
  - etc…

De AC já sabem…

- Processador (CPU) e memória
  - Instruções e seus modos utilizador e supervisor
  - Níveis de memória (caches), MV e tabela de páginas
- Controladores de periféricos
  - Entradas/saídas (I/O) com dispositivos (p.e. discos)
  - I/O por espera ativa ou com interrupções

Motivação: da programação à execução

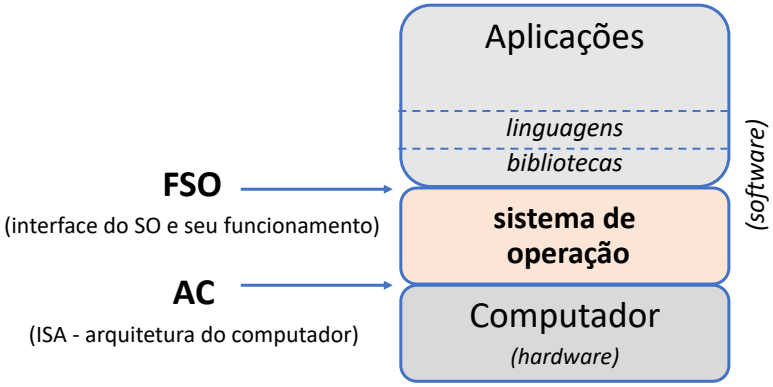
- Exemplo: Java
  - linguagem com um ambiente de execução (*runtime*)

E no meio?
Como transformar, traduzir, interpretar, executar...?
- *Vários níveis de abstração*
- *Cada nível de abstração oferece uma interface*
- *Facilita o nível superior e usa o nível inferior*

- Arquitetura de Computadores
- Sistemas Lógicos/Electrónica

Motivação: níveis/interfaces

- Num sistema de computação (simplificado):



Núcleo do sistema (kernel)

- O núcleo do SO, é o "programa" privilegiado (modo supervisor) que:
  - Gere a memória, o CPU e restantes recursos (periféricos)
  - Único capaz de manipular todo o hardware
- Os programas (modo utilizador) recorrem a este para muitas operações
  - Usando chamadas ao sistema (pedidos ao SO)
- Os periféricos notificam o núcleo quando necessário via interrupções
  - o núcleo inclui os *interrupt handlers* nos respetivos *device drivers*

O SO visto como…

- Gestor de recursos
  - Gere e permite a partilha do hardware pelos programas
  - Gere os programas e suas interações
- Máquina virtual
  - Oferece abstrações de mais fácil utilização
  - Oferece operações (serviços) aos programas

Grandes questões para FSO

- O que é um SO? Como funciona?
- Como suporta e controla a execução de vários programas?
- Como os programas partilham os recursos do computador?
- Que abstrações e interfaces o SO oferece?
- Que consequências têm nas linguagens de programação e suas bibliotecas?
- Como as utilizar (eficientemente)?

Tópicos do programa de FSO

- Funções e organização do sistema operativo
- Gestão do CPU
- Gestão da memória central (RAM)
- Gestão dos dispositivos de entrada / saída
- Virtualização de recursos
- Abstrações de ficheiro e processo
- Sistema de ficheiros
- Introdução à programação concorrente
- Segurança dos sistemas de operação

PRÉ-REQUISITOS

- Assume-se que os alunos dominam as matérias de:
  - Introdução à Programação
  - Arquitetura de Computadores
- Rever matérias!
  - Rever a matéria de AC:
    - Em especial, modos do CPU, endereços virtuais, I/O
  - Rever ambiente laboratorial:
    - Comandos Unix/Linux, linguagem C, compilação e *debug*
    - **Fazer ficha 0 no CLIP!**

Bibliografia

- Referência principal:
  - **Operating Systems: Three Easy Pieces**. *R. Arpacci-Dusseau, A. Arpacci-Dusseau.*

    http://pages.cs.wisc.edu/~remzi/OSTEP/
- Slides das aulas e outra informação complementar, disponibilizados no CLIP.
- Referências complementares:
  - Modern Operating Systems. A.S. Tanenbaum e H. Bos, 4th Ed 2014
  - Operating System Concepts. A. Silberschatz, *et al*, 10th Edition, 2018
  - Tutoriais de C (exemplos):
    - https://www.geeksforgeeks.org/c-programming-language/
    - https://www.w3schools.com/c/
    - https://www.programiz.com/c-programming/

Funcionamento – trabalho semanal

- 9 créditos segundo o sistema ECTS
  - 1 crédito = 28 horas de trabalho, logo aprox. 250 h no semestre...
  - ou, 12 h/semana (se 40 h/semana para 30 ECTS)
- Horas em contacto (5h)
  - Aulas teóricas (3h por semana)
  - Aulas práticas (2h por semana)
  - Esclarecimento de dúvidas (no horário de atendimento)
- Horas em autonomia (7h)
  - Estudo da matéria das aulas teóricas e práticas, preparação para os trabalhos práticos e para os testes
  - Realização dos trabalhos para casa e de grupo

Funcionamento - Aulas

- A presença nas aulas é **muito** recomendada!
- As aulas teóricas já começaram: Hoje!
  - O estudo deve começar esta semana.
- As aulas práticas começam na próxima semana.
  - Deve estar preparado para a aula
- **TPC** – preparar-se sempre para as aulas!

Avalição – Regras Gerais

- A avaliação tem duas componentes:
  - Componente teórica
    - dois testes ou exame (tem de CompT ≥ 9,0)
  - Componente laboratorial ou prática
    - 3 trabalhos individuais e um em grupos de dois alunos
- Noção de frequência obtida pela componente laboratorial
  - tem de CompL >= 9,0
- Qualquer aluno envolvido numa fraude (detetada imediatamente ou *a posteriori*) reprova.

Avalição – Nota Final

- Nota final (tendo frequência):
  - NF = CompT (se CompT < 9,0)
  - NF = 0,3*CompL + 0,7*CompT (se CompT ≥ 9,0 e CompL ≥ 9,0)
- Notas de anos anteriores:
  - CompT do ano passado continua válida
  - CompL dos dois últimos anos continua válida
  - Dispensam de fazer essa componente. Se entregar qualquer elemento de avaliação, anula a nota anterior dessa componente.

Agradecimentos

- Alguns slides e outros materiais são
  - adaptações de elementos de edições anteriores ou de outras u.c.
  - adaptações de materiais disponibilizados na Internet ou associados ao livro de referência.

Parte II: Introdução aos SO

- Como apareceram os SO? Com que objetivos?
- nota: o hardware e o SO evoluíram em conjunto!

Evolução da operação do computador

Resumo:

1. Máquina dedicada a um programa
2. Submissão sequencial de lotes de trabalhos
   - batch (sem ou com SPOOL)
3. Multiprogramação com:
   - Múltiplos programas
   - Múltiplos utilizadores
4. Hoje o hardware é barato (algum), podemos ter uma máquina dedicada a cada utilizador (PC), executando múltiplos programas

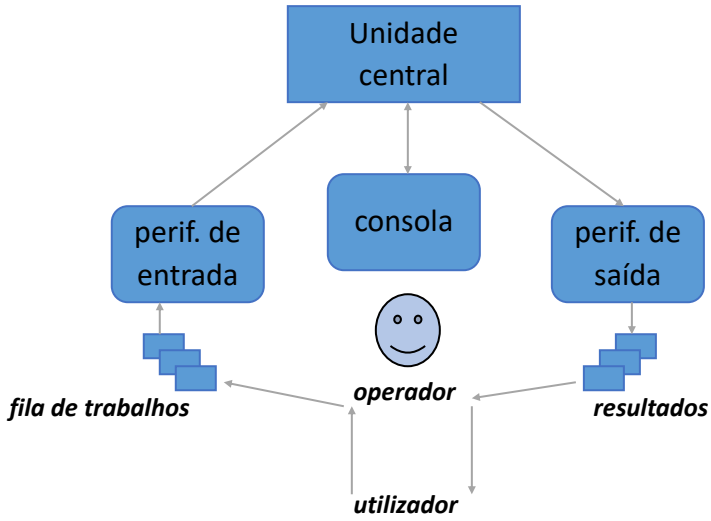
Máquina dedicada

- Um utilizador → um programa
- Hardware, sem mais nada...
  - Código máquina introduzido manualmente
- Auxílio de um programa carregador
  - Uso de fita perfurada ou cartões perfurados
- Aparecem bibliotecas de rotinas para I/O
- Assemblers, compiladores
  - Geram código máquina a partir de texto (código fonte)

Como rentabilizar a máquina?

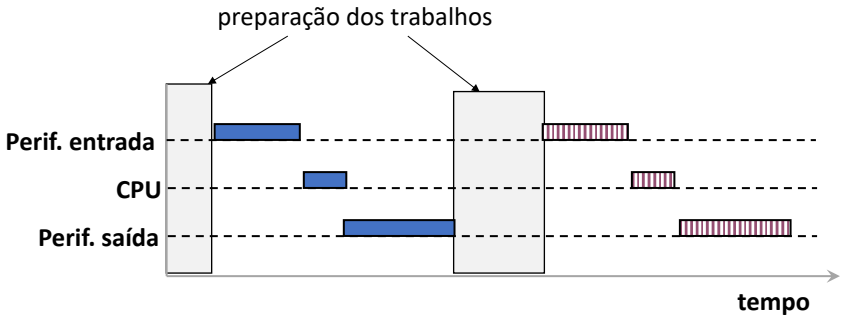
- Aparece o Operador de Sistemas (*sysop*) especialmente treinado para operar o computador e seus periféricos
- O utilizador trabalha off-line, submetendo trabalhos (conjuntos de programas e dados - *batch*)
- O operador define a ordem de execução dos trabalhos (estabelece um escalonamento) que considere justo e rentabilize a máquina
- Programas suspeitos de estar em ciclo infinito são abortados

Processamento Batch



24

Exemplo de utilização



Puramente sequencial

25

Eficiência de um sistema - métricas

- Taxas de utilização dos recursos:
  - De cada periférico e, principalmente, do CPU
  - o ideal é 100%

$$\frac{\textit{tempo utilização}}{\textit{tempo total}}$$

- Débito de trabalhos:
  - quanto mais melhor

$$\frac{\textit{trabalhos executados}}{\textit{tempo usado}}$$

- Tempo de resposta para o utilizador
  - Tempo que decorre desde a submissão do trabalho até obter os resultados
  - Quanto menor melhor para o utilizador

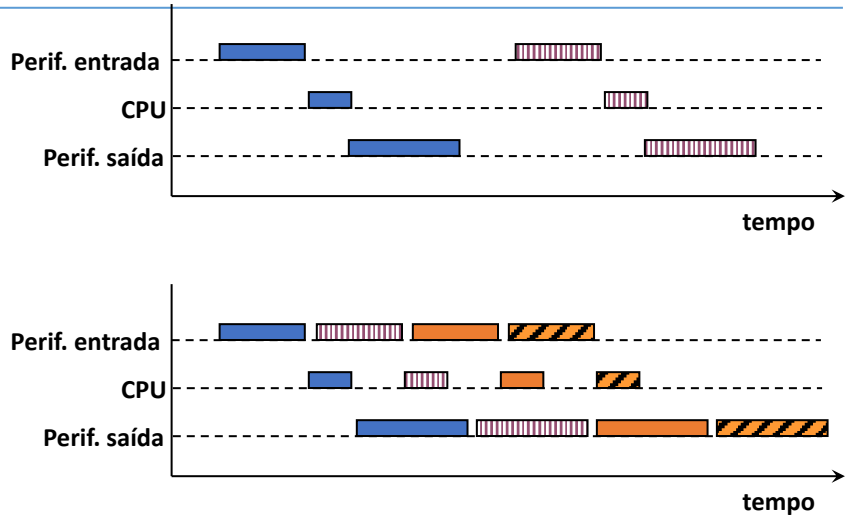
Génese dos SO

- Para auxiliar os programadores/utilizadores
  - Acorda-se numa biblioteca de rotinas para operar com os periféricos (*device drivers*)
    - Todos os programadores a usam e esta pode estar sempre carregada em memória
- Para auxiliar o operador e optimizar o processamento
  - Existe um programa monitor que trata de carregar o próximo trabalho e de o pôr em execução
    - Pode mesmo aceitar comandos, tornando os trabalhos (batch) em verdadeiras sequências automáticas de operações (exemplo: compilar programa, definir os seus dados de entrada, pôr em execução, ...)

Génese dos SO (2)

- E ainda:
- Simultaneidade entre leituras, escritas e a execução de programas
  - Permite *Simultaneous Peripheral Operation On-Line* (SPOOL)
  - Melhora a utilização dos recursos (CPU e periféricos)
  - Necessita de proteção entre utilizadores e do monitor/SO
  - Introduz-se a noção de chamada ao sistema
- É necessário suporte do hardware:
  - Dois modos de execução no CPU
  - O modo utilizador para execução dos programas
    - Estes não podem aceder aos periféricos nem a toda a memória (protege o monitor/spooler)
  - Modo supervisor (ou protegido)
    - Para só o monitor executar todas as instruções do ISA

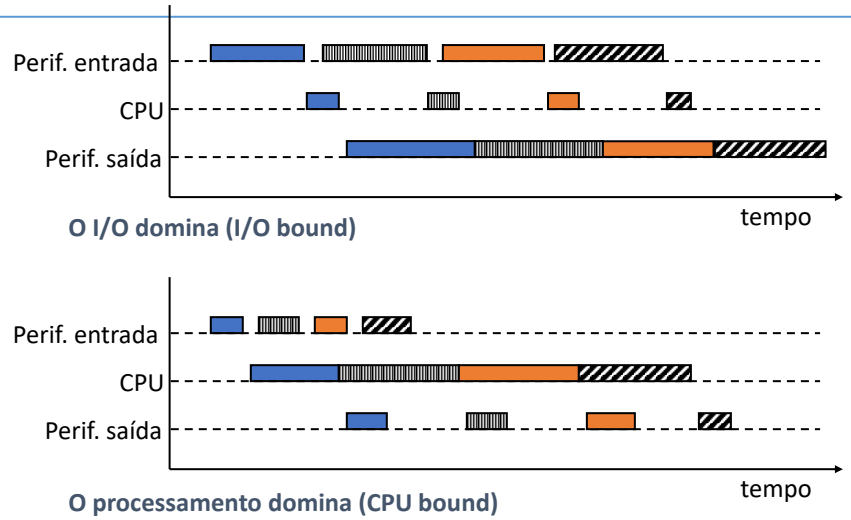
Utilização do CPU e periféricos



Exemplo – IBM 1460 (1963)

30

Perfis das execuções



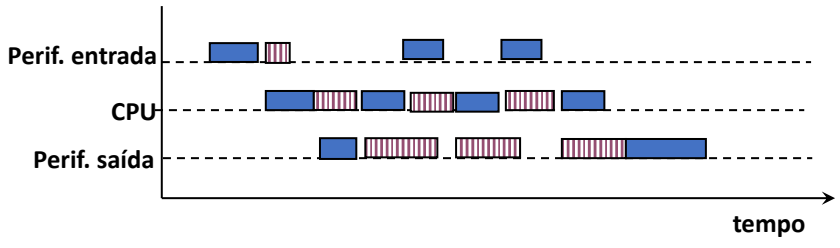
31

Outras melhorias

- Como suportar melhor programas com diferentes perfis?
- Como suportar melhor fases de I/O durante a execução?
  - p.ex. Programas que variam entre I/O e CPU
  - p.ex. Programas interativos
- Como tirar partido do CPU quando decorre I/O?

Múltiplos programas

- Ter vários programas em memória prontos a executar
- Quando um fizer I/O escalonar outro para executar, mantendo o CPU ocupado



Vários programas em execução concorrente

Multiprogramação (multitasking)

- Vários programas em memória, prontos a executar
- Cada programa executa num contexto: **processo**
  - Imagem do programa numa zona de memória
  - Guarda estado do CPU quando não executa
  - Guarda estado do I/O (canais ou *streams* para ficheiros em uso)
- O SO oferece uma "máquina virtual" a cada processo
  - Cada processo "vê" um espaço de memória próprio, recolocado e só pode aceder a essa memória
  - Arquitetura com MMU que permite esta gestão da memória e CPU tem um modo de execução supervisor (os processos têm de pedir certas operações ao SO)
- O SO comuta entre processos, quando necessário

Time-sharing: partilha do tempo

- Um programa pode monopolizar a máquina?
  - Mau para o uso dos periféricos
  - Mau para a interatividade
- Há que partilhar o tempo de CPU
  - por cooperação entre os programas,
  - ou por apreensão do CPU pelo SO (*preemption*).
    - Cada programa usa no máximo uma determinada fatia de tempo (*time-slice* ou *quantum*)
- O processo pode ser interrompido com base no tempo
  - O SO pode comutar de processo

Sistemas interativos multiutilizadores

- Suportado por *multitasking* com *time-sharing*
- Cada utilizador vê uma máquina virtual
  - No disco tem um conjunto de aplicações
  - No disco guarda um conjunto de dados
  - O utilizador interage dando comandos ao sistema via um shell
    - Manda o sistema executar programas (criar processos)
    - Permite programas interativos
  - Normalmente não se apercebe dos outros utilizadores
- Os processos executam em concorrência
  - Não há ordem pré-definida entre os processos ou para as operações de I/O. Até podem ser em simultâneo.
  - Rentabiliza o CPU e restantes recursos
- *ainda se usa SPOOL, p.e. para imprimir*

Sistemas interativos multiutilizadores

- Exemplo: 4 processos concorrem por um CPU
- A quando de I/O ou do fim do *time-slice* o SO atribui o CPU a outro processo