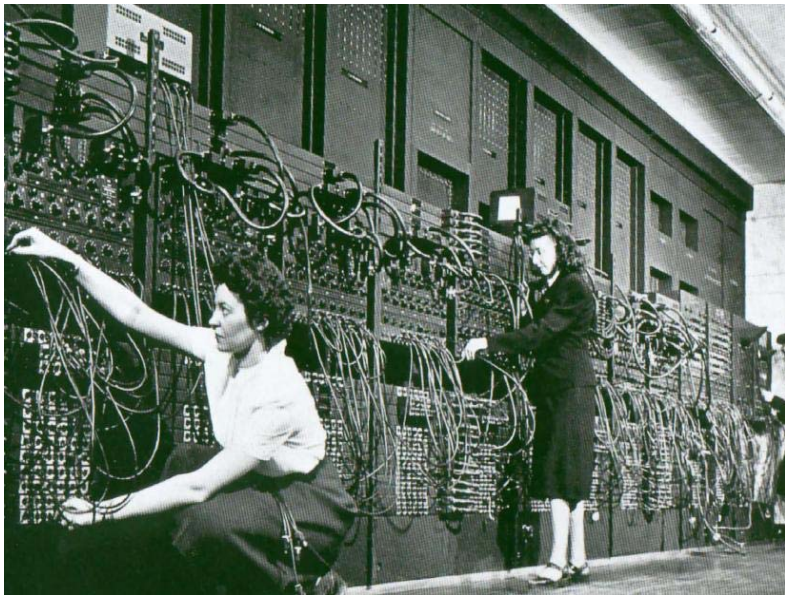
Sistemas Operativos

Apresentação

2006/2007

Paulo Marques
Departamento de Eng. Informática
Universidade de Coimbra
pmarques@dei.uc.pt

Motivação

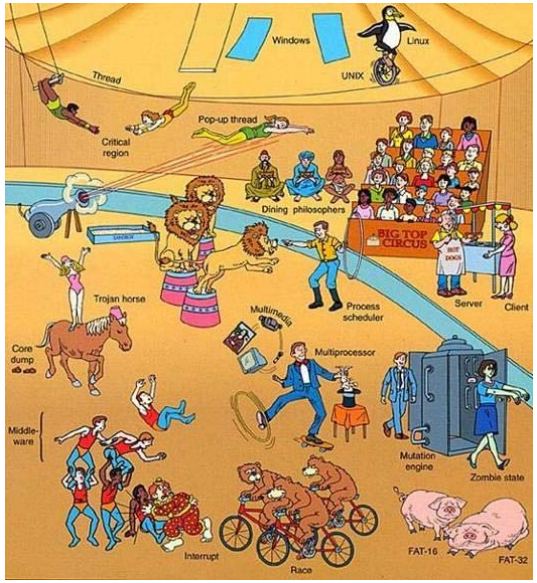


Motivação…

Motivação…

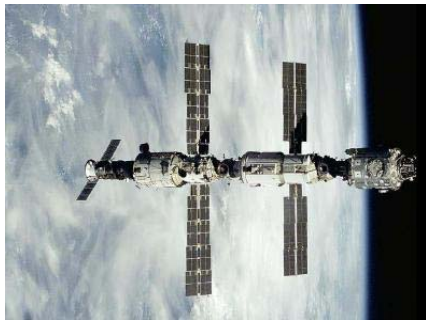
Motivação…

Sistemas Operativos segundo Tanenbaum



© Prentice Hall, “Modern Operating Systems”, Tanenbaum & Woodhull

Motivação…

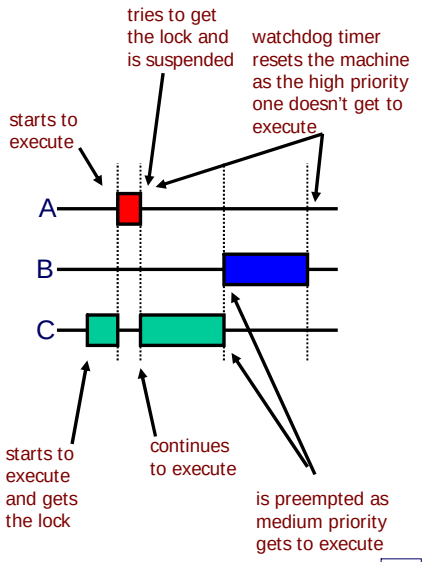


Motivação…

Mars Path Finder

The Mars PathFinder Problem

- **Priority Inversion Problem** (in Mars Path Finder):
  - Low priority thread locks a semaphore.
  - High priority thread starts to execute and tries to lock the same semaphore. It's suspended since it cannot violate the other thread lock.
  - Medium priority threads comes to execute and preempts the low priority one. Since it doesn't need the semaphore, it continues to execute.
  - Meanwhile the high priority one is starving. After a while, a watchdog timer detects that the high priority one is not executing and resets the machine.



Sobre o que é que vamos falar?

- O **Sistema Operativo** é um "programa especial" que permite isolar o *hardware* dos programas que executam neste
  - Gera a memória
  - Gere os discos
  - Gere os periféricos (teclado, rato, placa gráfica)
  - Gere os utilizadores e programas, protegendo todo o sistema

    ... tudo para que o programador não tenha de o fazer

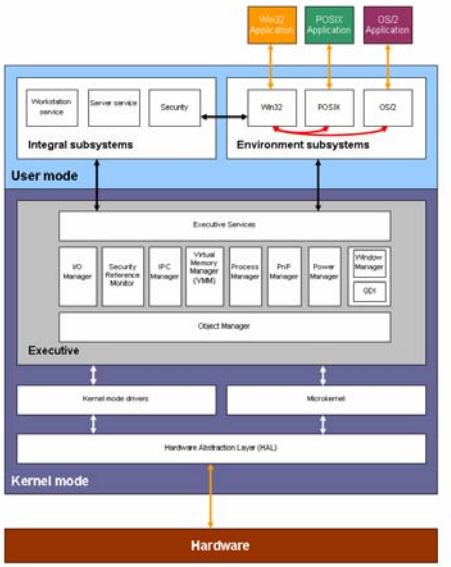
- **OBJECTIVOS**
  - Saber como funciona um sistema operativo "por dentro"
  - Saber utilizar e programar utilizando as funções fornecidas pelos sistemas operativos, tirando partido das suas potencialidades
  - Aprender programação concorrente
  - Aprender programação em "C"

The Mars PathFinder Problem

imer
machine
priority
get to

ed as
iority
cute

Windows 2000 "on the inside"



Plano das Aulas Teóricas

1. Breve introdução à linguagem "C"
2. Funções de um Sistema Operativo
3. Gestão de Processos
4. Multithreading
5. Programação Concorrente
6. Exclusão mútua, sincronização e dead-locks
7. Gestão de memória / Memória virtual
8. Escalonamento de processos
9. Entrada/Saída e escalonamento de disco
10. Sistemas de Ficheiros
11. Segurança
12. Introdução aos sistemas operativos de tempo real

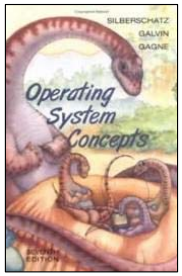
Plano das Aulas Práticas

1. Introdução à programação em C e Unix
2. Gestão de Processos
3. Sinais (Interrupções assíncronas)
4. Comunicação entre processos usando *pipes*
5. Multiplexação de Entrada/Saída
6. Memória Partilhada + Programação Concorrente
7. Threads + Programação Concorrente

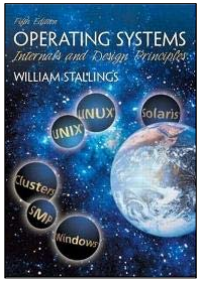
Docentes

- **Aulas Teóricas**
  - Paulo Marques
    pmarques@dei.uc.pt
    Gabinete D2.5
    Atendimento: Seg/Qui. 10h-12h
- **Aulas Práticas**
  - Bruno Cabral
    bcabral@dei.uc.pt
    Gabinete D1.9
  - Nuno Pimenta
    nuno@dei.uc.pt
    Gabinete D2.16
- **Aulas Práticas Laboratoriais**
  - TBD (2 assistentes)

Bibliografia – Parte Teórica



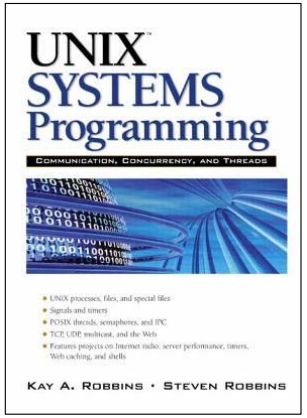
- **Operating System Concepts 7th Edition**
  by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne
  John Wiley & Sons, ISBN 0471694665



- **Operating Systems: Internals and Design Principles**
  5th Edition, by William Stallings
  Prentice Hall, ISBN 0131479547

Bibliografia – Parte Prática

- **Unix Systems Programming: Communication, Concurrency and Threads 2nd Edition**
  by Kay Robbins, Steve Robbins
  Prentice Hall, ISBN 0130424110



Bolonha

Modelo de Aulas de ECTS

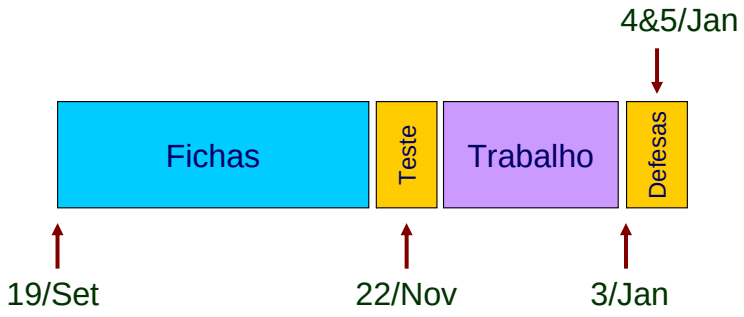
Actividades	HT(*)	HC(*)	ECTS total	ECTS / semana
T	30	30	1,11	0,07
P ou TP	15	15	0,56	0,03
PL	45	30	1,67	0,11
O	75	2	2,78	0,14
TOTAIS	165	77	6,11	0,36

- 1 ECTS = 27h de trabalho!
- **Semanalmente**:
  - 2h de aulas presenciais teóricas → Conhecimentos Teóricos
  - 1h de aulas presenciais práticas → Explicação Trabalhos/Matéria
  - 6h de trabalho extra aula → Resolução de fichas/trabalhos/estudo
    [c/ Aulas Práticas Laboratoriais de Apoio, se necessário]

Avaliação

- Conhecimentos
  - Um exame global → 10 valores
  - Um teste prático → 2 valores
  - *Mínimo de 35% na média de ambos*
- Competências
  - 7 Fichas semanais → 3 valores
    [nem a primeira nem a pior ficha são contabilizadas]
  - 1 Trabalho global c/ Defesa → 5 valores
  - *Mínimo de 35% na média de ambos*
- Exame de Recurso
  - Perguntas extra sobre a prática
    → Aluno pode optar por responder ou não

Planeamento da componente prática



Componente Prática

- Programação concorrente em "C" em ambiente Unix/Linux
  - A linguagem C é considerada pré-requisito. É assumido que muito rapidamente (duas semanas a contar a partir de agora) se conseguem ambientar à linguagem.
  - O ambiente UNIX é considerado pré-requisito. É assumido que muito rapidamente (duas semanas a contar a partir de agora) se conseguem ambientar à linguagem.
  - Existem apontamentos sobre LINUX no material de apoio da cadeira.
  - A primeira aula de Sistemas Operativos é um mini *crash-course* sobre "C". Mas, não é suficiente → É necessário ler um livro e praticar.
- Desde já é importante instalarem uma distribuição Linux
  - Recomendo Ubuntu (www.ubuntu.com), mas qualquer uma serve
  - Podem utilizar uma máquina virtual (e.g. VMWARE), ou um LiveCD, caso não queiram alterar as partições do vosso disco

Conduta Ética e Fraude

Casos detectados de fraude conduzirão à reprovação imediata do(s) aluno(s).

- É expressamente proibida a partilha de código fonte, por mais pequena que seja, entre alunos.
- É expressamente proibida a utilização directa de código fonte recolhido da *web*.
  - Caso tenha dúvidas se uma determinada utilização de código é ou não admissível, consulte o seu professor das TP.

Para além de fomentar a aprendizagem, os trabalhos visam também testar as suas competências no final da aprendizagem. Submeter e/ou utilizar código escrito por outrem, mesmo quando entendido, não demonstra que possui as competências que deverá adquirir em SO. Existe uma grande diferença entre saber ler código e entender código e conseguir escrevê-lo. Parte integrante da aprendizagem nesta disciplina é ter a capacidade de o escrever autonomamente. Tal só se consegue tentando (e tentando!) programar.