

# Sistemas Operacionais

## Apresentação da Disciplina

Francisco José da Silva e Silva

Laboratório de Sistemas Distribuídos (LSD)  
Departamento de Informática / UFMA  
<http://www.lsd.ufma.br>

17 de agosto de 2010



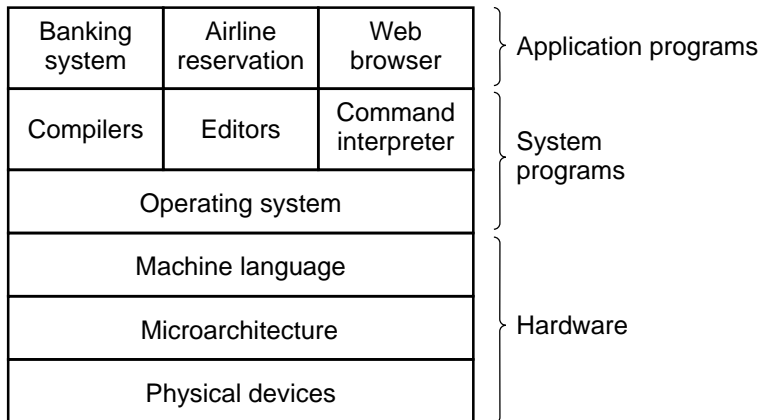
- 1 A Disciplina de Sistemas Operacionais
- 2 Introdução aos Sistemas Operacionais



# A Disciplina de Sistemas Operacionais



# Arquitetura em camadas de um sistema de computação



# O Microprograma

- Usualmente localizado em memória ROM;
- É um interpretador que traduz as instruções de linguagem de máquina como ADD, MOVE e JUMP em uma série de pequenos passos;
- O conjunto de instruções interpretadas definem a linguagem de máquina.



# Tópicos da Disciplina

- 1 Introdução aos sistemas operacionais
- 2 Gerenciamento de processos
- 3 Sincronização de processos concorrentes e cooperantes
- 4 Bloqueios perpétuos
- 5 Gerenciamento de memória
- 6 Gerenciamento de E/S
- 7 Gerenciamento de informações
- 8 Proteção e segurança



# Bibliografia

- 1 Andrew S. Tanenbaum, Sistemas Operacionais Modernos 3ª Ed, Pearson Prentice Hall, 2010;
- 2 Abraham Silberschatz et. al., Fundamentos de Sistemas Operacionais, LTC, 2010;
- 3 William Stallings, Operating Systems internals and design principles, Prentice Hall, 2008.
- 4 Gary Nutt, Operating Systems: a modern perspective, Addison Wesley, 2002;



# Página do Curso

A página do curso pode ser acessada através do seguinte apontador:  
<http://www.deinf.ufma.br/~fssilva/graduacao/soi/index.html>



# As aulas

- Na disciplina de sistemas operacionais I apresentamos as funções desempenhadas pelos sistemas operacionais em sistemas de computação e como eles realizam estas tarefas;
- Estamos, portanto, interessados na organização interna de sistemas operacionais e nos algoritmos utilizados na sua construção;
- Em sistemas operacionais II utiliza-se os conceitos apresentados nesta disciplina para detalhar como os sistemas operacionais mais utilizados atualmente (como o Unix, Linux e Windows) foram construídos;
- Em sistemas operacionais II são também realizadas diversas atividades práticas que exercitarão a programação com as chamadas ao sistema dos sistemas operacionais apresentados.



# Alguns recursos na Web

- Universidade da Califórnia, Berkeley:  
<http://inst.eecs.berkeley.edu/~cs162/>
- Operating Systems Design, University of Illinois:  
<http://www.cs.uiuc.edu/education/courses/cs323.html>
- Instituto de Matemática e Estatística da Universidade de São Paulo:  
<http://www.ime.usp.br/dcc/grad/catalogo2004/disciplinas/MAC0422.html>
- Departamento de Informática da PUC Rio:  
<http://www.puc-rio.br/ferramentas/ementas/ementa.aspx?cd=INF1019>



# Sistemas Operacionais: Por que Estudar?

- Infraestrutura: analista de suporte, de rede, de segurança
  - Conhecer como o SO funciona é essencial para o gerenciamento de sistemas computacionais
- Desenvolvimento de Sistemas
  - É fundamental entender como programas são executados em um sistema de computação
  - Concorrência e sincronização são essências na programação de muitos sistemas
  - Para entender as limitações das diversas plataformas ou para acesso direto a suas funcionalidades
  - No projeto de sistemas embarcados
- Sistemas operacionais é um dos fundamentos da computação e é conteúdo central nos currículos de referência da SBC e ACM/IEEE.



# Introdução aos Sistemas Operacionais



# Classificação de Software

Um software pode ser classificado de acordo com seu propósito:

- Software aplicativo: projetado para resolver um problema específico;
- Software de sistema: objetiva disponibilizar um ambiente de programação geral na qual aplicações possam ser desenvolvidas.



# O que é um SO?

Um SO é um conjunto de programas, implementados tanto em software quanto em firmware, que torna o hardware possível de ser utilizado.



# Funções do SO em um Sistema de Computação

- Apresenta ao usuário uma máquina mais flexível e adequada para se programar do que aquela que o hardware nu apresenta;
- Possibilita o uso eficiente e controlado dos vários componentes de hardware e software do sistema. Realiza gerenciamento de processos, memória, dispositivos de E/S e informações;



# Exemplo de Abstração de Recursos

Considere as seguintes operações para escrita de um bloco de informação em um dispositivo de disco:

```
load(block, length, device); // copia bloco da mem principal
                               // para buffer do disp
seek(device, track); // movimenta cabeçote de leitura e gravação
out(device, sector); // escreve do buffer para o dispositivo
```

A seqüência de comandos abaixo realizariam a escrita de um bloco específico:

```
load(block, length, device);
seek(device, 236);
out(device, 9)
```



# Exemplo de Abstração de Recursos I

Uma abstração simples poderia empacotar estes comandos em um único procedimento:

```
write(char *block, int len, int device,  
      int track, int sector) {  
    ...  
    load(block, length, device);  
    seek(device, 236);  
    out(device, 9);  
    ...  
}
```



# Exemplo de Abstração de Recursos II

Uma abstração de mais alto nível poderia traduzir o endereçamento físico de trilha e setor por um endereçamento lógico do bloco. Desta forma, a operação:

```
write(block, 100, device, 236, 9);
```

poderia ser escrita assim:

```
write(block, 100, device, 3788);
```



## Exemplo de Abstração de Recursos III

Uma abstração ainda maior poderia tratar o disco como um espaço de armazenamento de arquivos:

```
fprintf(fileID, "%d", datum);
```

Escreve um inteiro armazenado em `datum` em um dispositivo com um offset implícito do início do arquivo

