MC504/MC514 - Sistemas Operacionais

Introdução

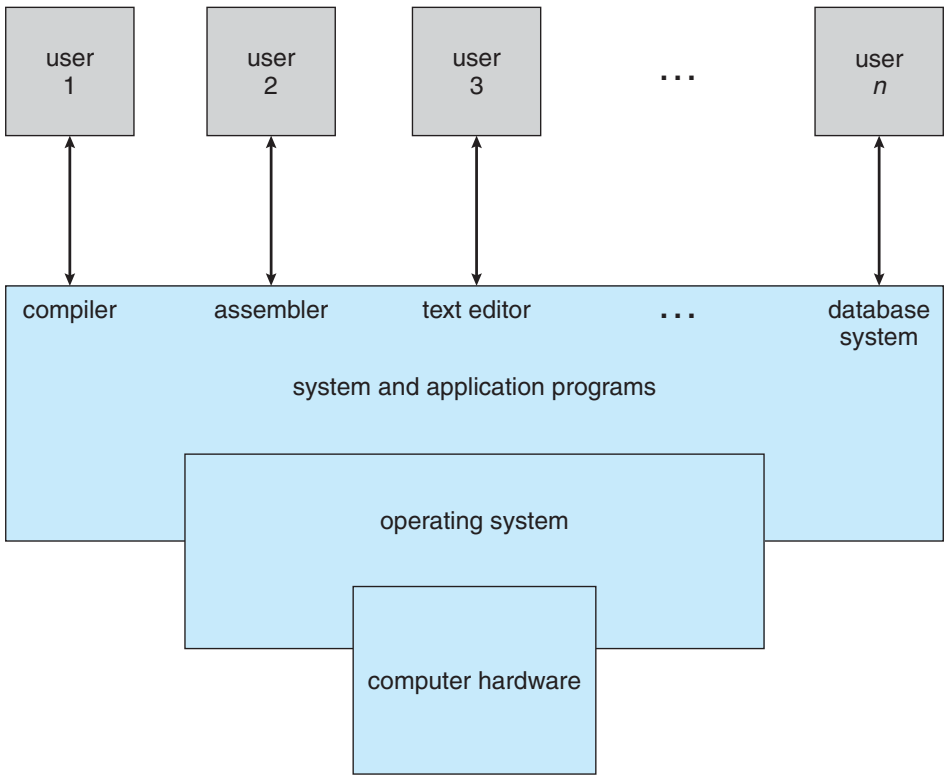
Islene Calciolari Garcia

Instituto de Computação - Unicamp

Segundo Semestre de 2016

Sumário

Sistema operacional

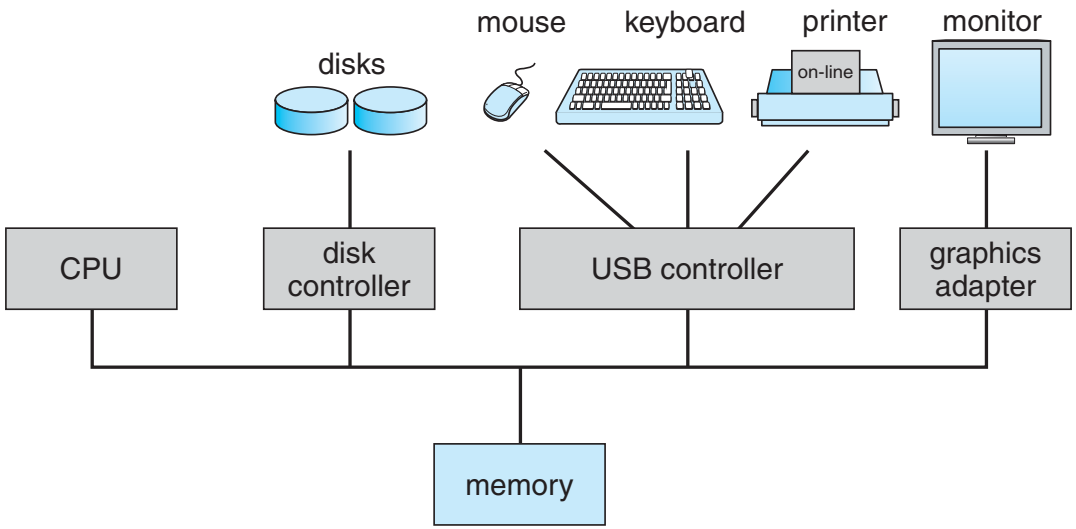


Silberrschatz et al.: Figura 1.01

O sistema operacional isola o hardware das camadas superiores em um sistema computacional

Sistema operacional

- *Gerenciador de recursos*
  fornece uma alocação controlada de processadores, memória e dispositivos de entrada/saída



Silberrschatz et al.: Figura 1.02

Ementa

Conceito de threads e processos: concorrências, regiões críticas, escalonamento. Conceitos de espaços de endereçamento e de gerenciamento de memória virtual, paginação, segmentação. Sistemas de arquivos: hierarquia, proteção, organização, segurança. Gerenciamento de Entrada/Saída. Estudos de caso.

Critério de Avaliação

Provas teóricas

Serão realizadas duas provas escritas P_1 e P_2, sem consulta.

Datas	
P_1:	5 de outubro
P_2:	7 de dezembro

Média

$$M_{prova} = \frac{2 * P_1 + 3 * P_2}{5}$$

Critério de Avaliação
Projetos

Temas:

- Programação Multithread
- Projeto prático com Kernel Linux

Os projetos serão desenvolvidos e apresentados por grupos de no **máximo quatro** alunos.

MC504/MC514: Critério de Avaliação

Média parcial e média final

$$M_{parcial} = (3 * M_{prova} + M_{proj})/4$$

$M_{prova} \geq 5.0$	$M_{parcial} \geq 5.0$	$M_{final} = M_{parcial}$
	$M_{parcial} < 5.0$	$M_{final} = (Exame + M_{parcial})/2$
$M_{prova} < 5.0$	$M_{final} = (Exame + M_{parcial})/2$	

Data do exame: 21 de dezembro.

Em caso de fraude em qualquer atividade avaliativa ao longo da disciplina todos os envolvidos ficarão com média final igual a zero.

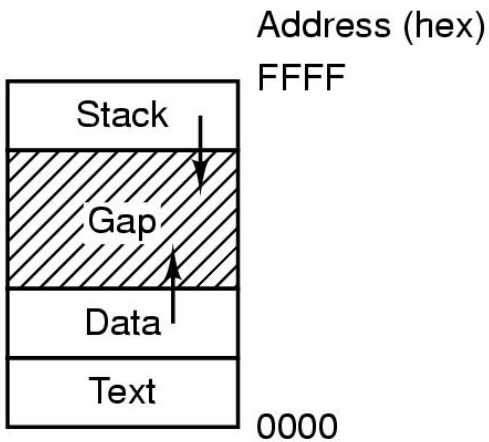
Principais Referências

- Modern Operating Systems, Andrew Tanenbaum
- A. Silberschatz, P Galvin, G. Gagne, Operating Systems Concepts
- W. Stallings, Operating Systems: Internals and Design Principles
- Understanding the Linux Kernel, Daniel P. Bovet e Marco Cesati
- The Litte Book of Semaphores, Allen B. Downey

Processos e espaço de endereçamento

ou porque seu programa de MC202 às vezes terminava em SEGFAULT!

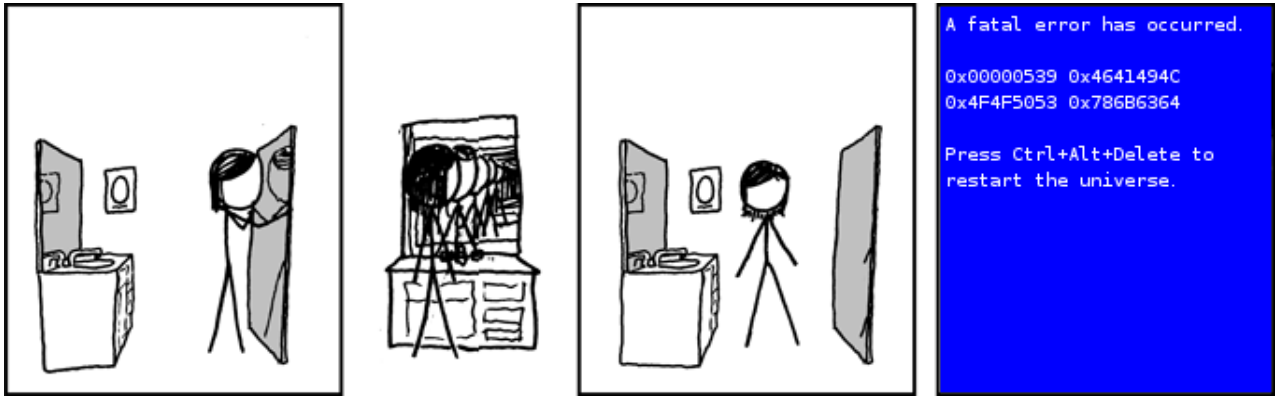
- Programa em execução
- Espaço de endereçamento



Tanenbaum: Figura 1.20

- Veja os códigos: `ender.c` e `ender-malloc.c`
- Endereços reais ou virtuais?

Recursão infinita



http://techtime.getharvest.com/blog/segmentation-faults/

```
void f() {
  printf("Recursão infinita...");
  f();
}
```

- Por que o programa gera um erro de execução?
- Veja o código `rec.c`

Borda da área de dados

Códigos desse tipo podem causar erro de execução?

```
void f() {
  int *v = malloc(4*sizeof(int));
  v[10] = 5;
```

- Veja o código `acesso-pos-n-alocada.c`
- Explore a borda da área de dados com o código `borda.c`
- O que acontece quando um bloco grande de dados é alocado?
- O que acontece quando solicitamos muitas alocações? Veja `loop-malloc`

Pilha de execução

Códigos desse tipo podem causar erro de execução?

```
void f() {
  int v[5];
  int i;

  for (i = 0; i < 10; i++)
     v[i] = i;
}
```

- Veja o código `pilha.c`

Vamos examinar a pilha de execução

- Componentes do frame (não necessariamente nesta ordem):
  - valor de retorno
  - endereço de retorno
  - registradores
  - argumentos para a função
  - variáveis locais da função
- Veja os códigos `pilha2.c`, `pilha3.c`, `pilha4.c` e `pilha5.c`
- No gdb execute comandos do tipo:

```
(gdb) p (int*[10]) *v

(gdb) set var v[0] = ...
```

E na próxima aula...

- Programação multithread (vários fluxos de execução)
- Espaço de endereçamento pode ser compartilhado