

# Sistemas Operacionais

## Gestão de tarefas - o conceito de tarefa

Prof. Carlos Maziero

DInf UFPR, Curitiba PR

Julho de 2020

# Conteúdo

- 1 Programas  $\times$  tarefas
- 2 Sistemas monotarefa
- 3 Sistemas multitarefa
- 4 Tempo compartilhado
- 5 Estados e transições

# Programas × tarefas

## Programa

- Código (sequência de instruções) para tratar um problema
- São aplicações ou utilitários
- Conceito **estático**, sem estado interno

Exemplos de programas:

- `C:\Windows\notepad.exe`
- `/usr/bin/firefox`

# Programas × tarefas

## Tarefa:

- Execução das instruções definidas no programa
- Conceito **dinâmico**, com estado interno
- Estado interno evolui a cada instante
- Implementada de várias formas (processos, *threads*, ...)

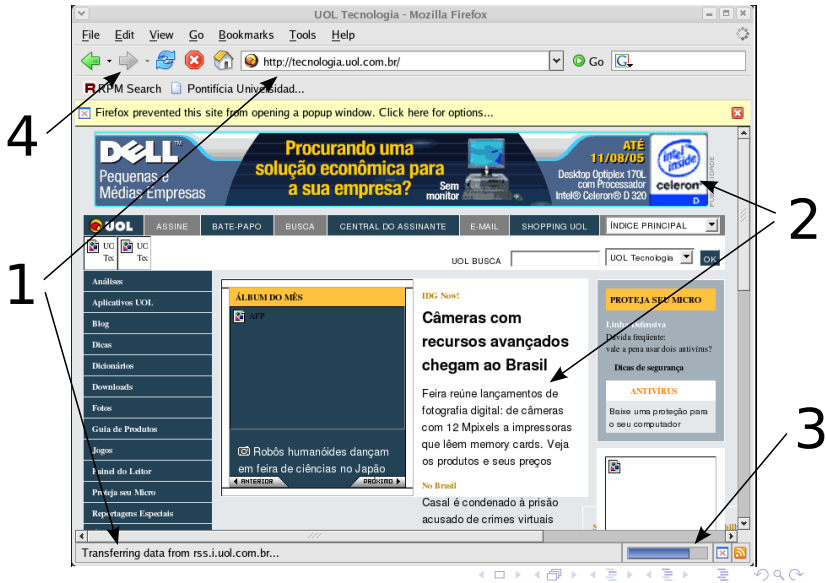
## Exemplos de tarefas:

- O *Notepad* editando um arquivo `readme.txt`
- O visualizador de PDFs mostrando este slide

# Programas × tarefas



# Tarefas de um navegador



# Gerência de tarefas

Como gerenciar as tarefas do sistema?

Histórico:

- 1 Sistemas Monotarefa
- 2 Sistemas Multitarefa
- 3 Sistemas de tempo compartilhado

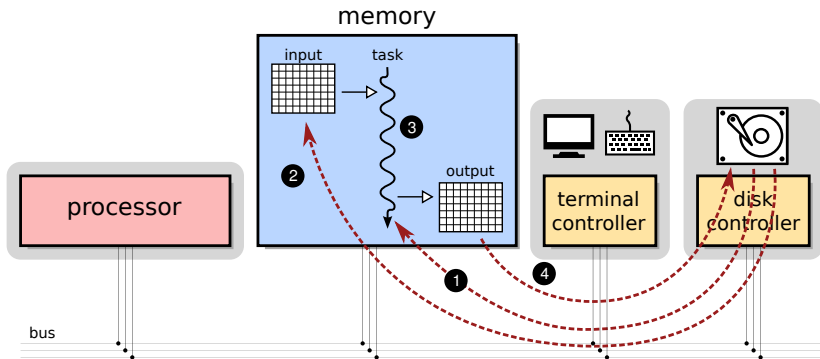
# Sistemas monotarefa

Usado nos primeiros sistemas de computação.

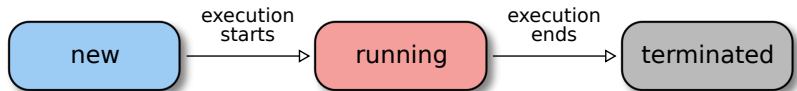
Executam uma única tarefa de cada vez:

- 1 O programa é carregado do disco para a memória
- 2 Os dados do programa são carregado na memória
- 3 O programa executa até sua conclusão
- 4 Os resultados do programa são salvos
- 5 Repete para o próximo programa

# Sistemas monotarefa



# Estados de uma tarefa



# Sistemas multitarefa

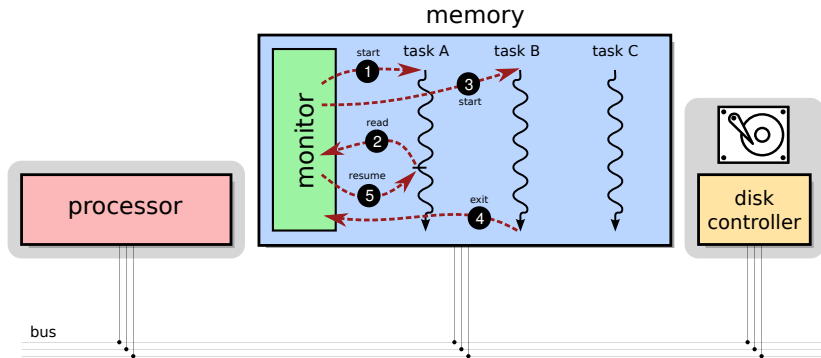
Problemas nos sistemas monotarefa:

- Tarefas esperando por entrada/saída ficam paradas
- Custo de operação do computador era muito elevado
- UNIVAC I (1951): 125 kW, o equivalente a 500 PCs

Solução:

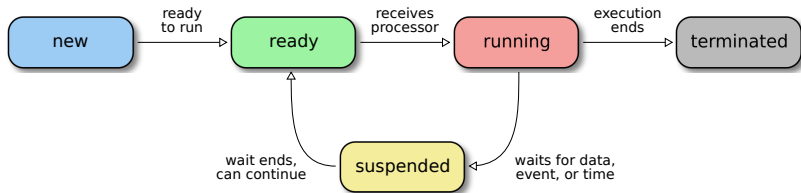
- Carregar várias tarefas na memória
- Usar o processador ocioso para tratar outras tarefas
- Um software *monitor* coordena a troca de tarefas

# Sistemas multitarefa



O monitor é um “embrião” de sistema operacional

# Estados de uma tarefa



Os estados são gerenciados pelo monitor.

# Sistemas multitarefa

Problemas dos sistemas multitarefa simples:

- Aplicações em laço infinito podem bloquear o sistema
- Aplicações interativas não funcionam bem

```
1 void main ()
2 {
3     int i = 0, soma = 0 ;
4
5     while (i < 1000)
6         soma += i ; // erro: i não foi incrementado
7
8     printf ("A soma vale %d\n", soma);
9 }
```

# Sistemas de tempo compartilhado

Solução: *Time Sharing* ou **preempção** por tempo

Conceito introduzido pelo CTSS (MIT, 1965)

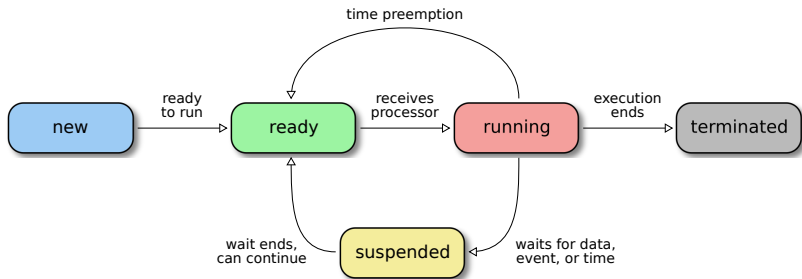
- Cada tarefa recebe uma fatia de tempo (quantum) de CPU
- A tarefa perde a CPU ao acabar seu quantum
- *Quantum* típico vai de 10 ms a 200 ms
- Implementado através de interrupções (*ticks*)

# Sistemas de tempo compartilhado

Funcionamento dos sistemas preemptivos:

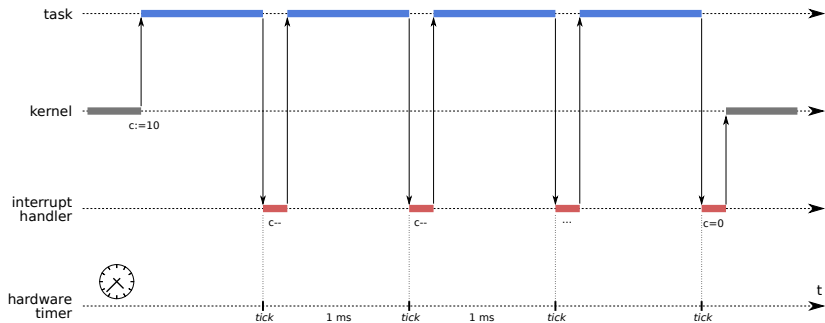
- 1 A tarefa recebe o processador
- 2 A tarefa executa até expirar seu *quantum* ou encerrar
- 3 A tarefa é interrompida pelo hardware
- 4 A tarefa retorna ao estado “pronto”
- 5 Outra tarefa recebe o processador

# Sistemas de tempo compartilhado



# Sistemas de tempo compartilhado

Implementação do tempo compartilhado:



# Estados das tarefas

**Nova** : A tarefa está sendo preparada para executar

**Pronta** : A tarefa está esperando o processador

**Executando** : A tarefa está executando suas instruções

**Suspensa** : A tarefa aguarda algum evento externo

**Terminada** : A tarefa encerrou ou foi abortada

# Transições das tarefas

|         |               |         |                                                                   |
|---------|---------------|---------|-------------------------------------------------------------------|
| $\dots$ | $\rightarrow$ | $N$     | a tarefa ingressa no sistema                                      |
| $N$     | $\rightarrow$ | $P$     | a tarefa está pronta para executar                                |
| $P$     | $\rightarrow$ | $E$     | a tarefa é escolhida para executar                                |
| $E$     | $\rightarrow$ | $P$     | esgota a fatia de tempo da tarefa                                 |
| $E$     | $\rightarrow$ | $T$     | a tarefa encerra sua execução                                     |
| $T$     | $\rightarrow$ | $\dots$ | a tarefa terminada é removida da memória                          |
| $E$     | $\rightarrow$ | $S$     | a tarefa em execução decide aguardar um recurso ou evento externo |
| $S$     | $\rightarrow$ | $P$     | o recurso ou evento aguardado pela tarefa está disponível         |

# Tarefas no Linux

## Comando top:

```

1 top - 16:58:06 up 8:26, 1 user, load average: 6,04, 2,36, 1,08
2 Tarefas: 218 total, 7 executando, 211 dormindo, 0 parado, 0 zumbi
3 %Cpu(s): 49,7 us, 47,0 sy, 0,0 ni, 3,2 id, 0,0 wa, 0,0 hi, 0,1 si, 0,0 st
4 KiB Mem : 16095364 total, 9856576 free, 3134380 used, 3104408 buff/cache
5 KiB Swap: 0 total, 0 free, 0 used. 11858380 avail Mem
6
7 PID USUÁRIO PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
8 32703 maziero 20 0 2132220 432628 139312 S 44,8 2,7 0:53.64 Web Content
9 2192 maziero 20 0 9617080 686444 248996 S 29,8 4,3 20:01.81 firefox
10 11650 maziero 20 0 2003888 327036 129164 R 24,0 2,0 1:16.70 Web Content
11 9844 maziero 20 0 2130164 442520 149508 R 17,9 2,7 1:29.18 Web Content
12 11884 maziero 20 0 25276 7692 3300 S 15,5 0,0 0:37.18 bash
13 20425 maziero 20 0 24808 7144 3212 S 14,4 0,0 0:08.39 bash
14 1782 maziero 20 0 1788328 235200 77268 S 8,7 1,5 24:12.75 gnome-shell
15 ...
  
```

## Comando vmstat (coluna CS – *Context Switch*)