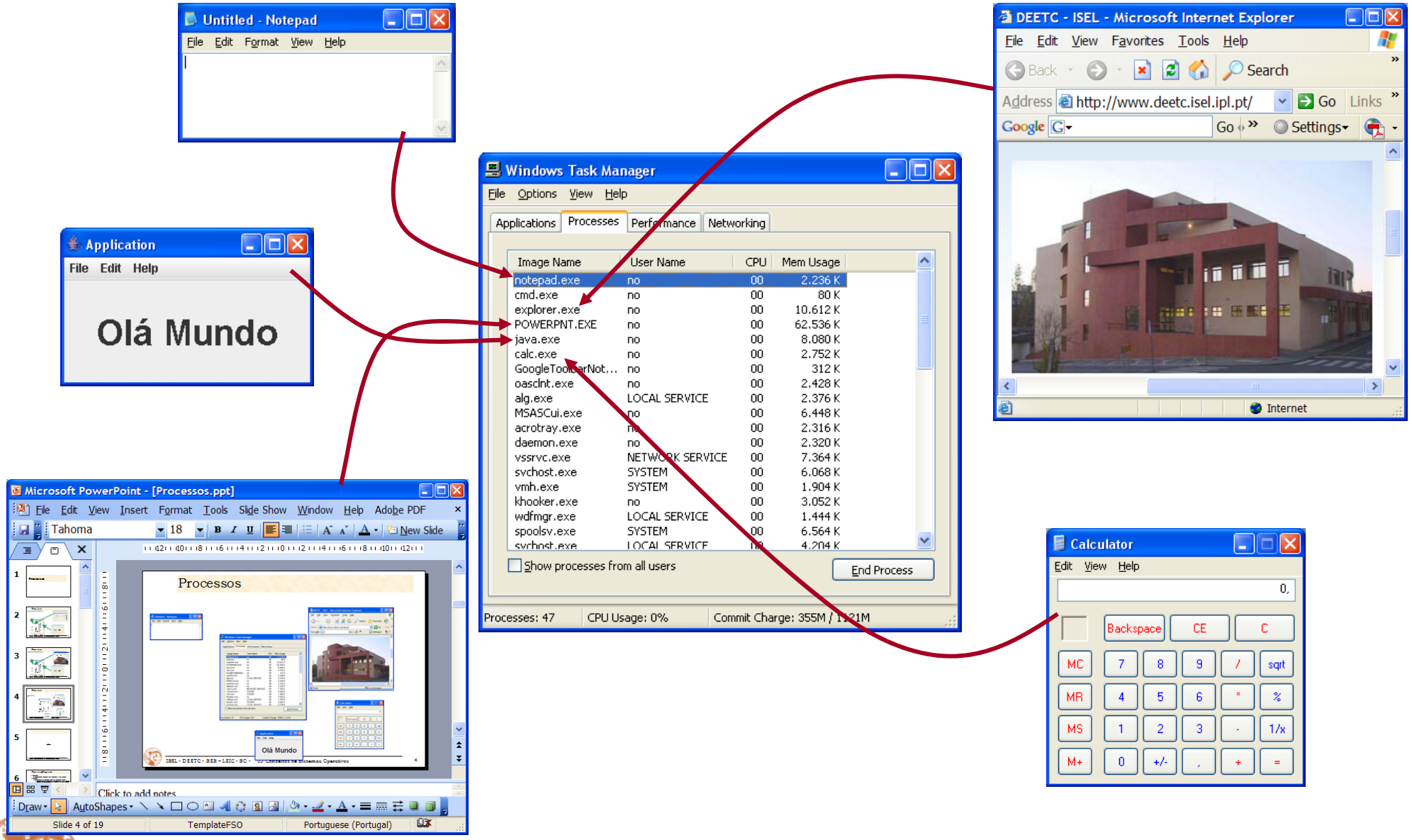
Processos e Tarefas

- Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, ***Operating System Concepts, 6ª Ed.***, John Wiley & Sons Inc., 2002 **[cap. 4, 5 e 6]**

Processos

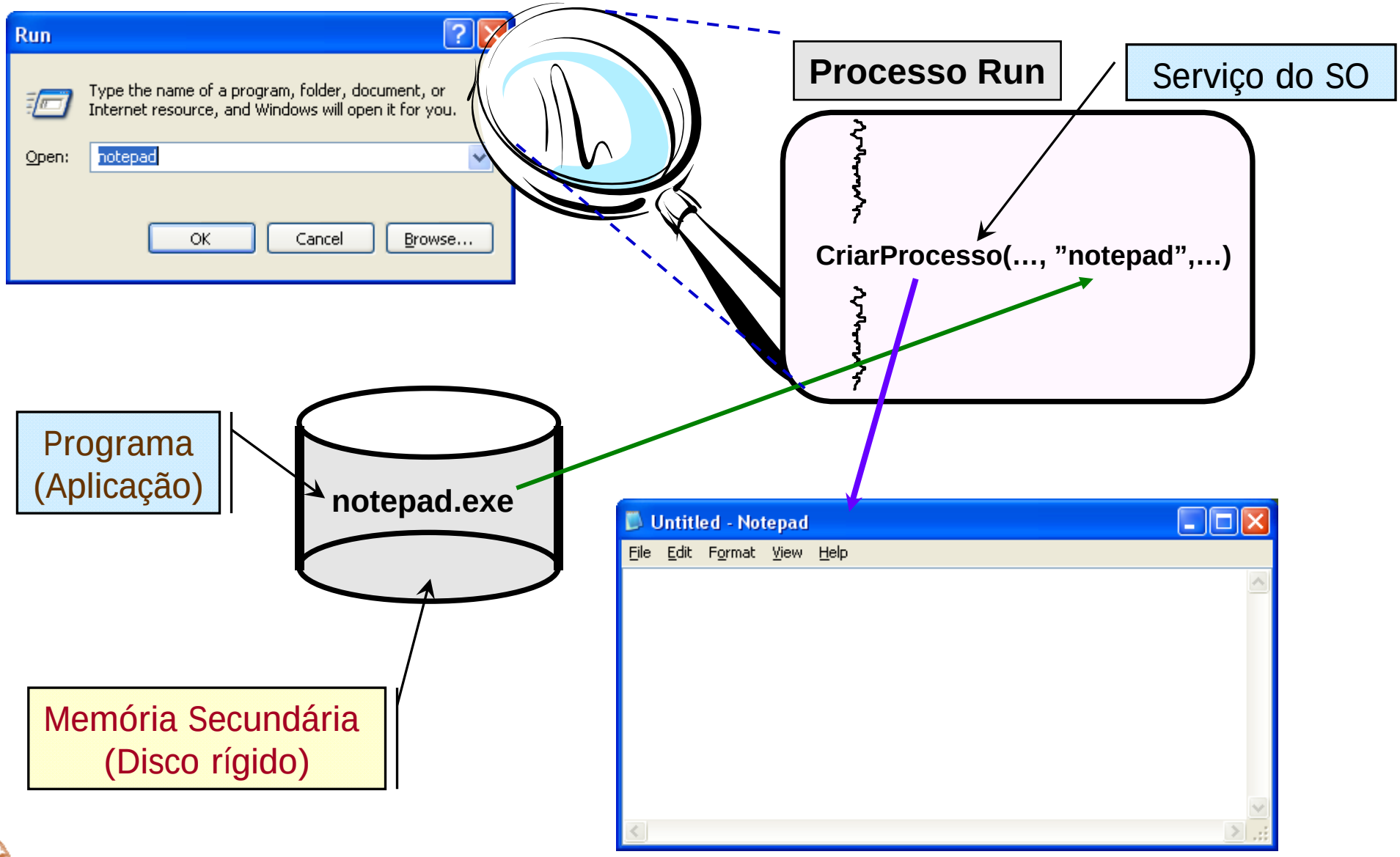
Processos


Untitled - Notepad
File Edit Format View Help
Application
File Edit Help
Olá Mundo
Windows Task Manager
File Options View Help
Applications Processes Performance Networking
Image Name User Name CPU Mem Usage
notepad.exe no 00 2.236 K
cmd.exe no 00 80 K
explorer.exe no 00 10.612 K
POWERPNT.EXE no 00 62.536 K
java.exe no 00 8.080 K
calc.exe no 00 2.752 K
GoogleToolbarNot... no 00 312 K
oasclnt.exe no 00 2.428 K
alg.exe LOCAL SERVICE 00 2.376 K
MSASCui.exe no 00 6.448 K
acrotray.exe no 00 2.316 K
daemon.exe no 00 2.320 K
vssrvc.exe NETWORK SERVICE 00 7.364 K
svchost.exe SYSTEM 00 6.068 K
vmh.exe SYSTEM 00 1.904 K
khooker.exe no 00 3.052 K
wdfmgr.exe LOCAL SERVICE 00 1.444 K
spoolsv.exe SYSTEM 00 6.564 K
Show processes from all users
End Process
Processes: 47 CPU Usage: 0% Commit Charge: 355M / 1121M
DEETC - ISEL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.deetc.isel.ipl.pt/
Internet
Microsoft PowerPoint - [Processos.ppt]
File Edit View Insert Format Tools Slide Show Window Help Adobe PDF
Processos
Olá Mundo
Click to add notes
Slide 4 of 19 TemplateFSO Portuguese (Portugal)
Calculator
Edit View Help
Backspace CE C
MC MR MS M+
sqrt % 1/x

Processo/Programa

- **Programa**
  - Entidade passiva residente num ficheiro contendo uma lista de instruções
- **Processo**
  - Entidade do SO que representa um programa em execução composto por:
    - Espaço de endereçamento contendo o código executável (*text section*), dados (*data section*), stack e espaço para atribuição de memória dinâmica (*heap*)
    - Actividade corrente (*Instruction Pointer* (IP) / *Program Counter* (PC))
    - Estado de execução do processo compostos pelos valores dos registos do processador
    - Identificador do processo (PID);
    - Informação sobre utilização dos recurso: tempo de utilização do CPU, etc (*Accounting*)
    - Estado do processo: Ready, Run, Wait, etc
    - Informação relacionada com o escalonamento do processo, e.g. prioridade (*scheduling*)
    - Informação sobre o estado I/O: lista de dispositivos atribuídos ao processo, lista de ficheiros abertos, etc
    - Credenciais

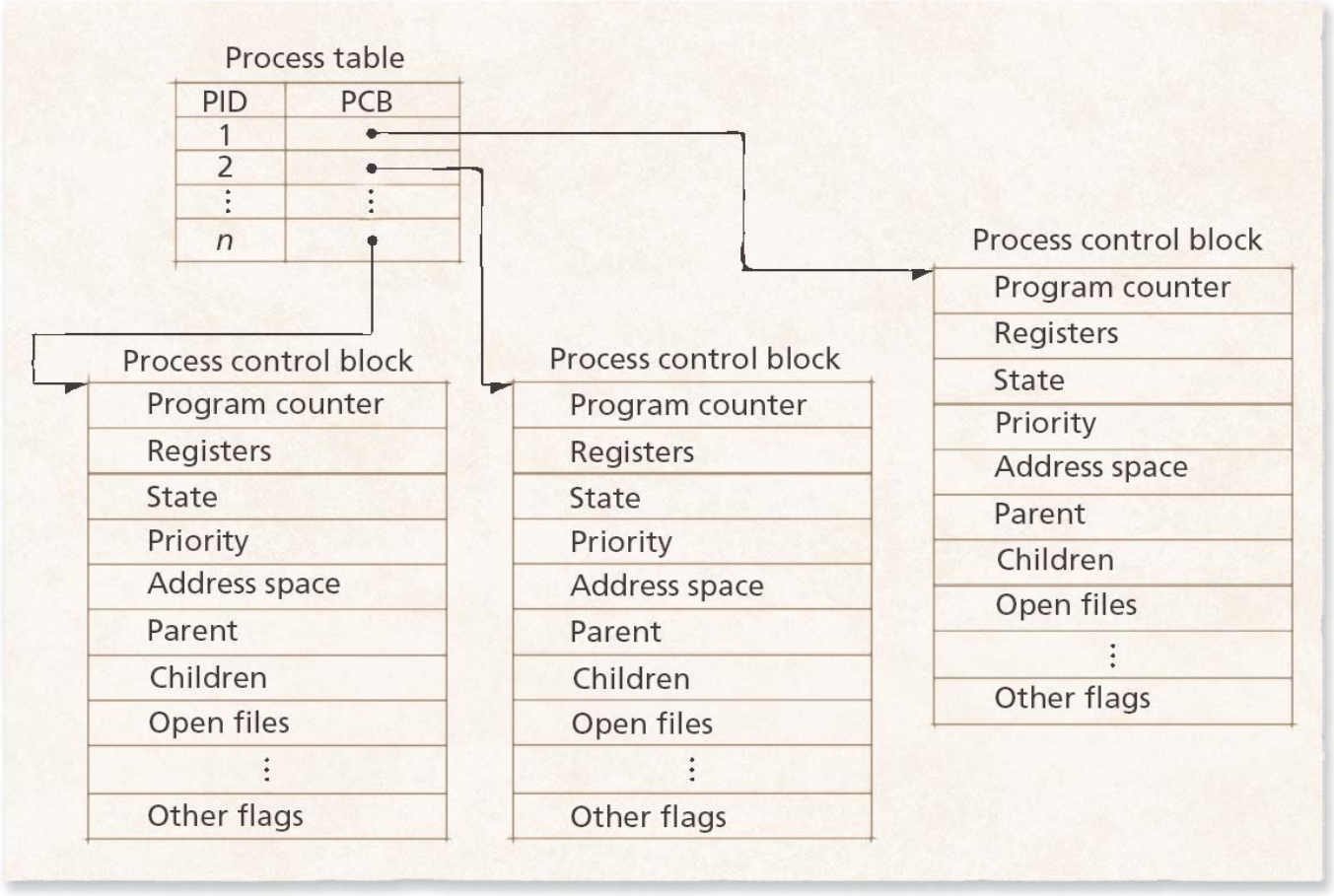
Processo/Programa



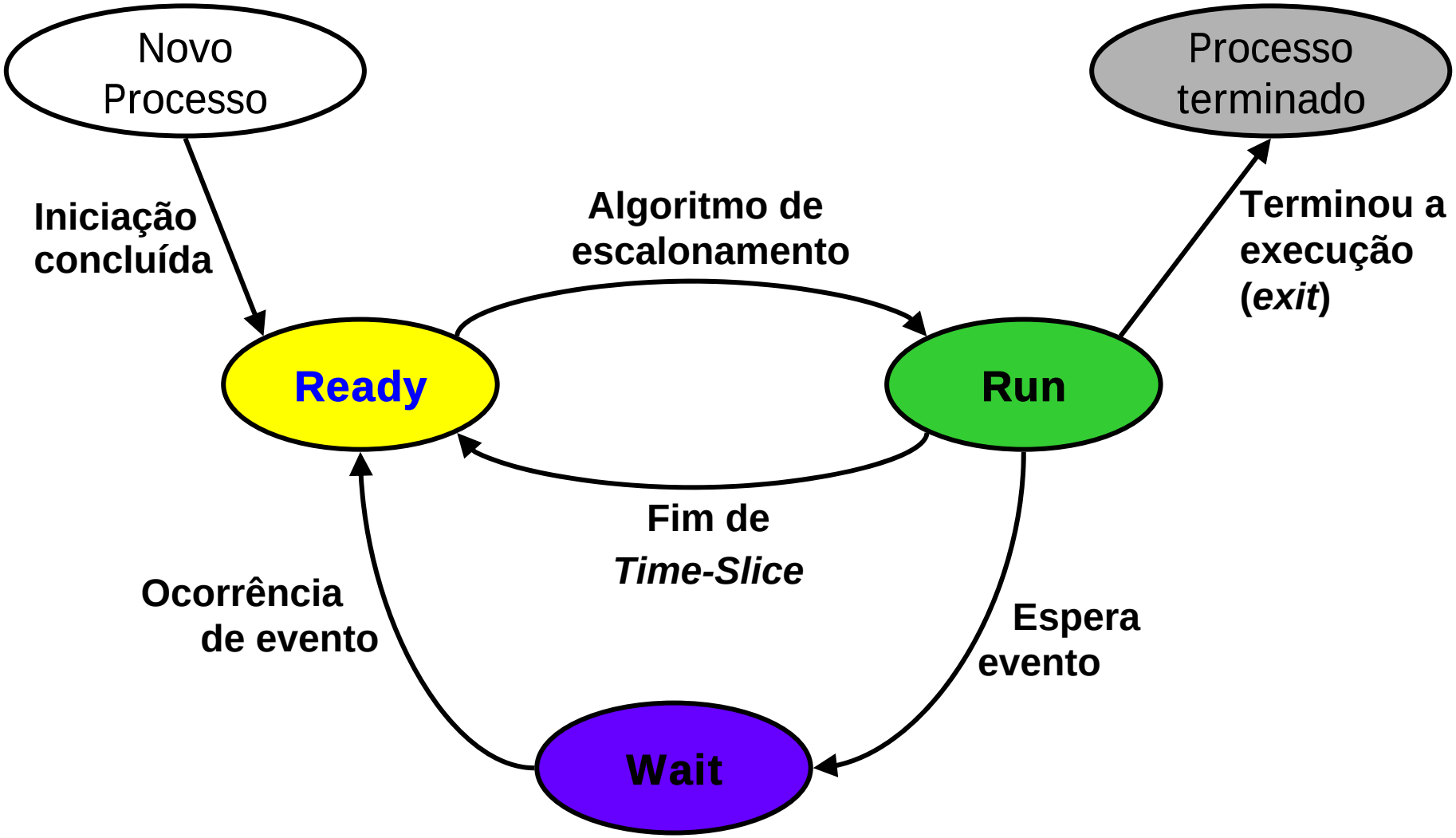
Process Control Block - PCB

- ***Process Control Block* (PCB)**
  - Cada processo é representado no Sistema Operativo por uma estrutura de dados contendo toda a informação relacionada com o processo.
- **Tabela de processos**
  - O sistema operativo mantém uma tabela com referências para todos os processos instanciados no sistema. A tabela facilita o acesso à estrutura PCB de um processo dado um PID e quando um processo termina o SO remove o PCB desta tabela eliminando todos os recursos associados ao processo

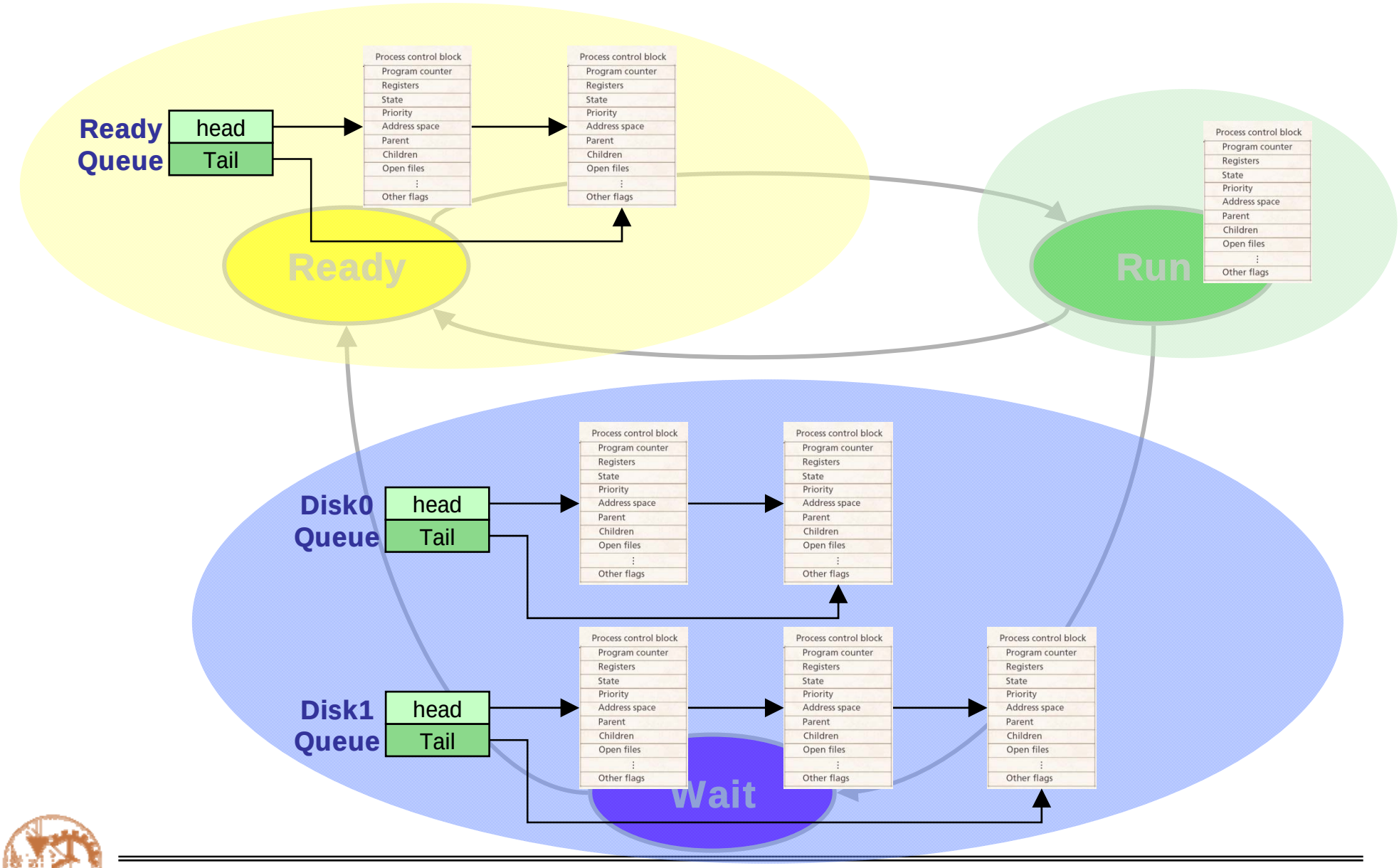
Tabela de processos



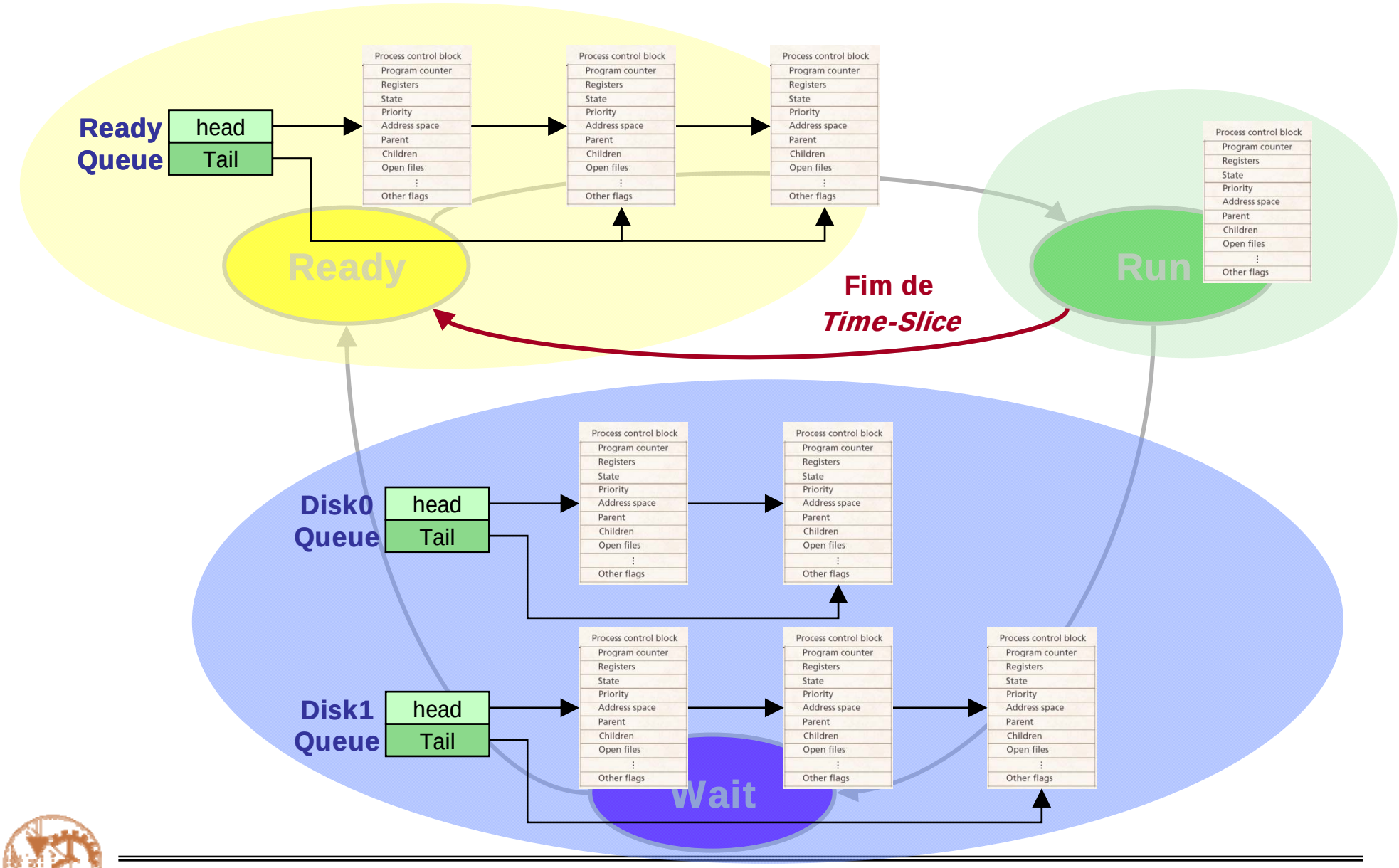
Estados do Processo



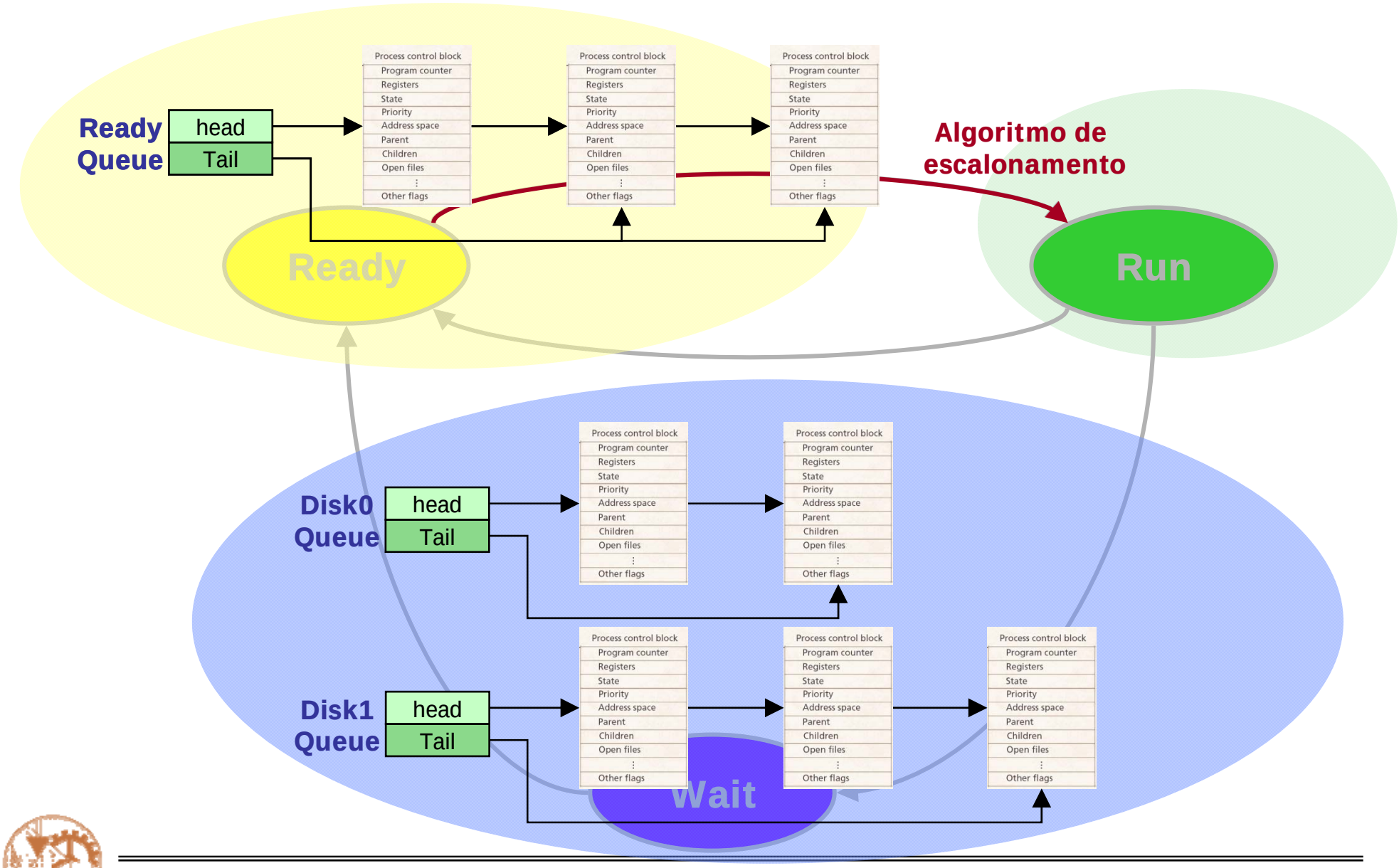
Scheduling queues



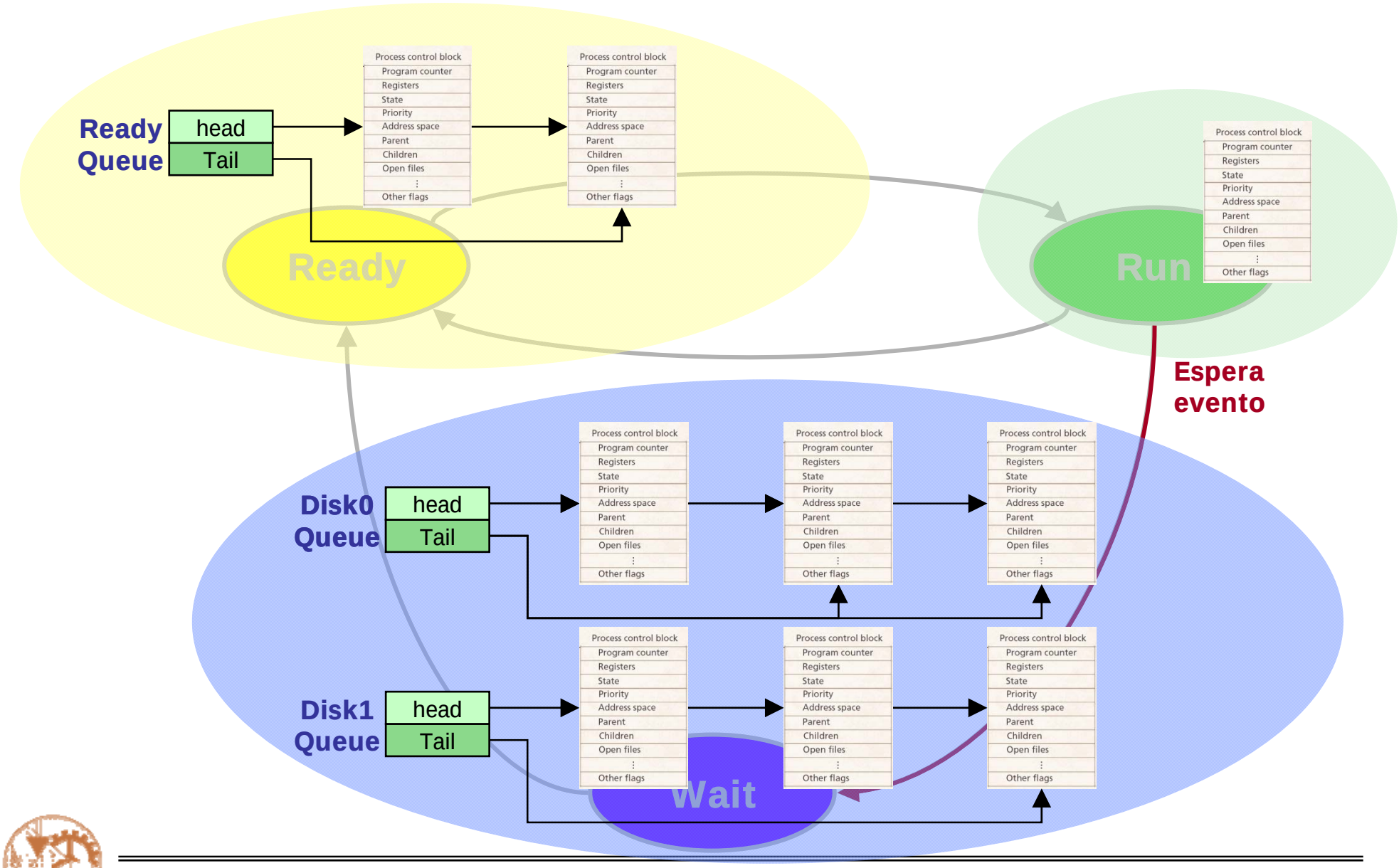
Scheduling queues (Run -> Ready)



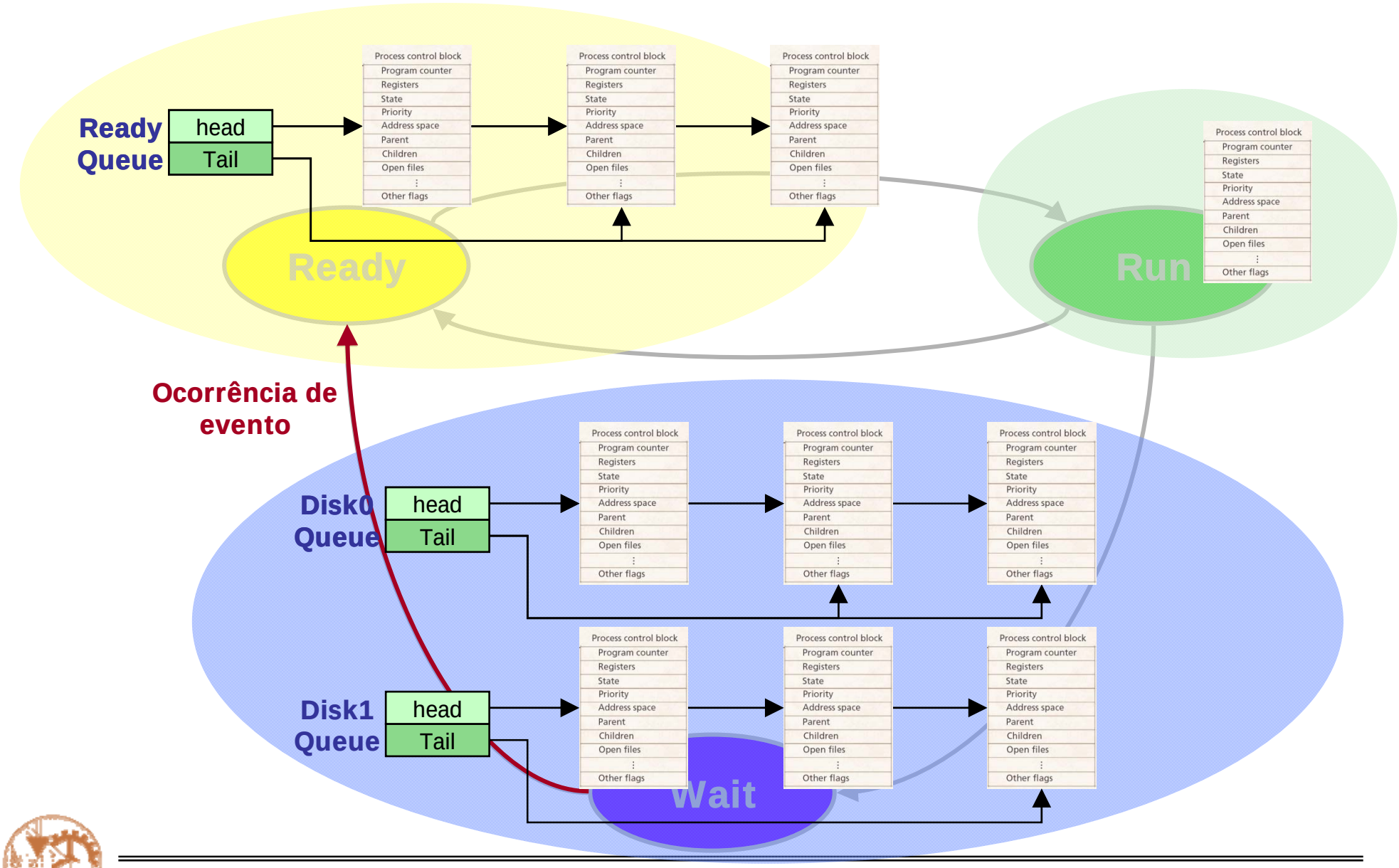
Scheduling queues (Ready -> Run)



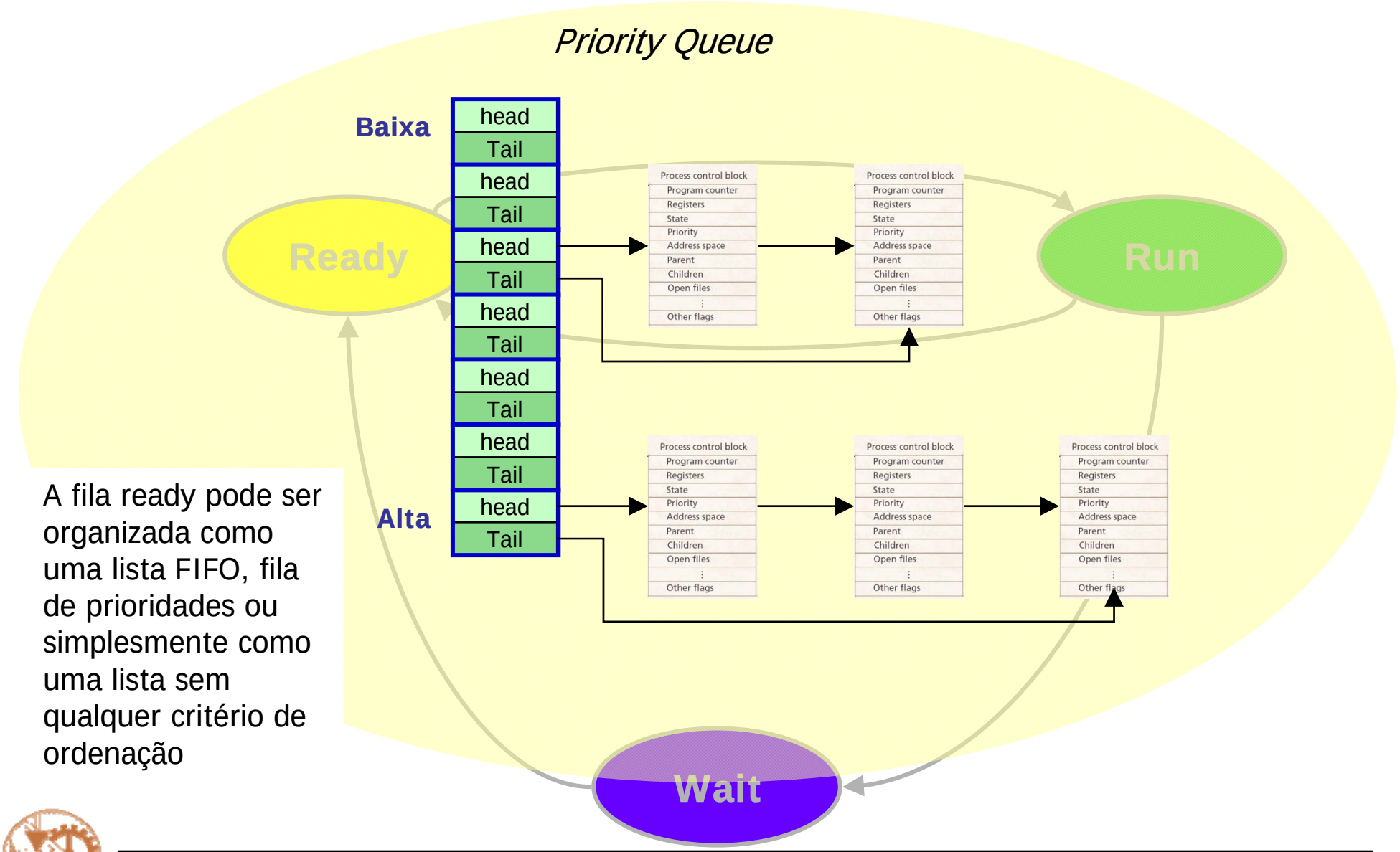
Scheduling queues (Run -> Wait)



Scheduling queues (Wait -> Ready)

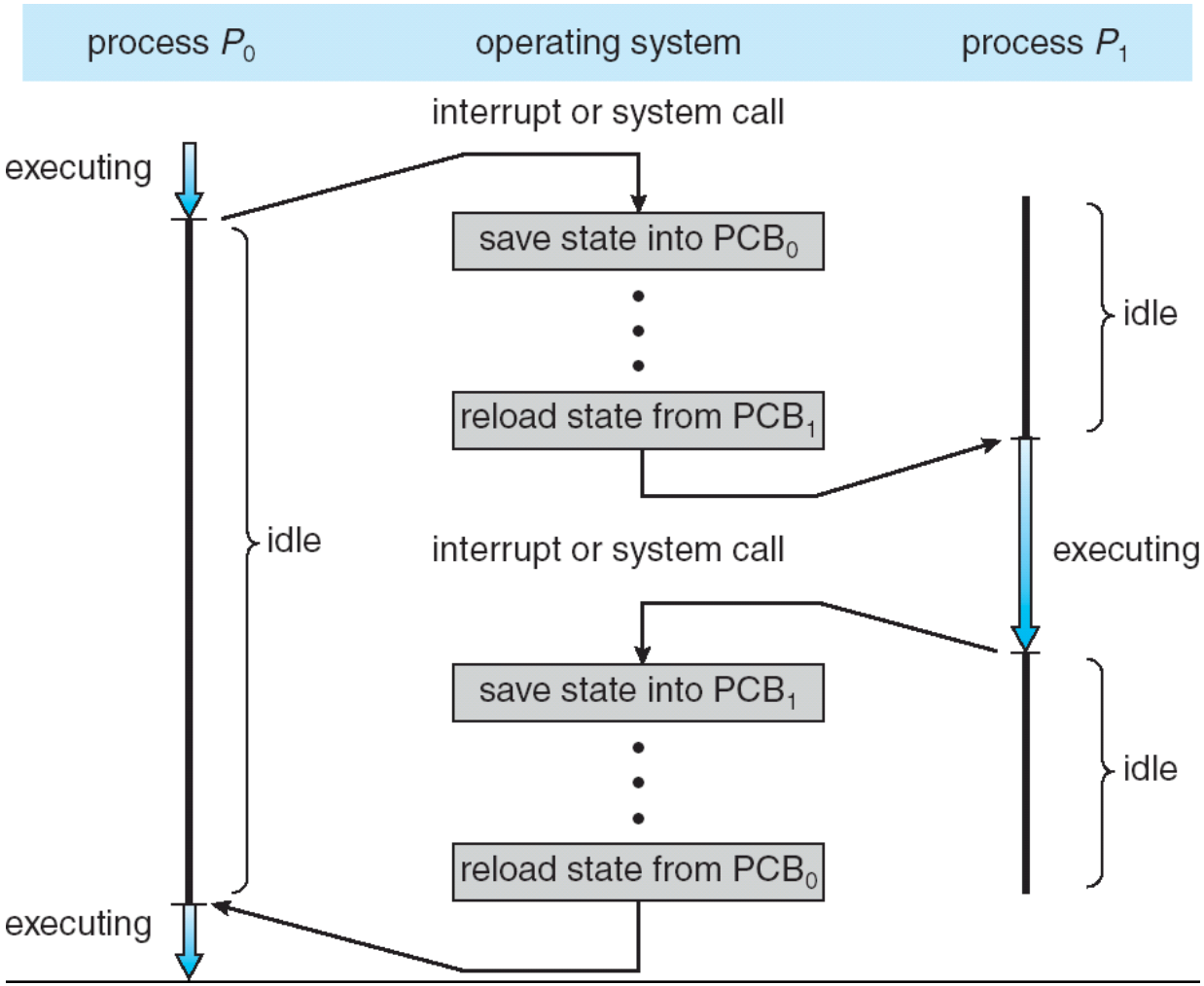


Processos - prioridades

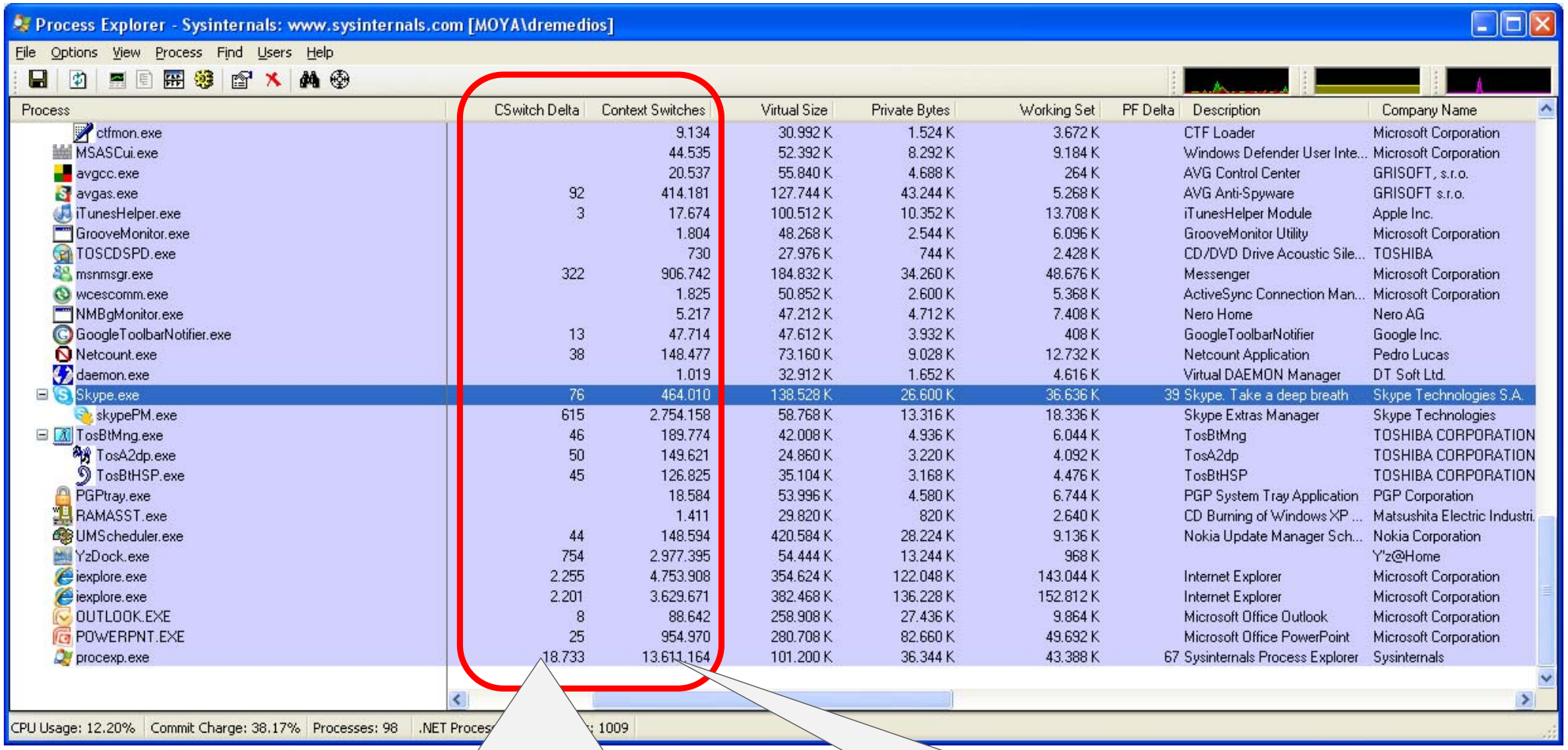


A fila ready pode ser organizada como uma lista FIFO, fila de prioridades ou simplesmente como uma lista sem qualquer critério de ordenação

Context Switch



Context Switch *(cont.)*



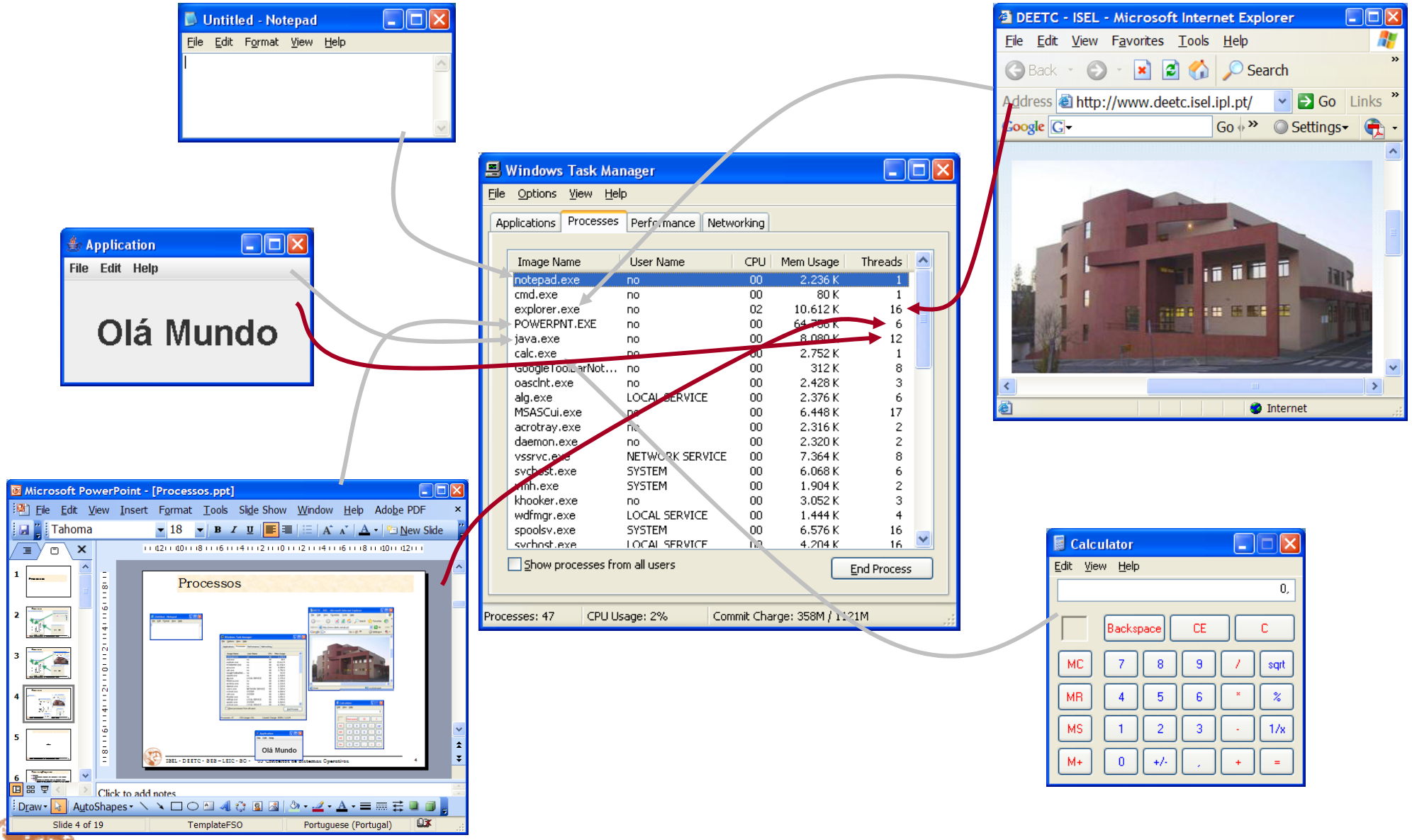
Nº de *Context switches* / Segundo

Nº de *Context switches* total

Tarefas - *Threads*

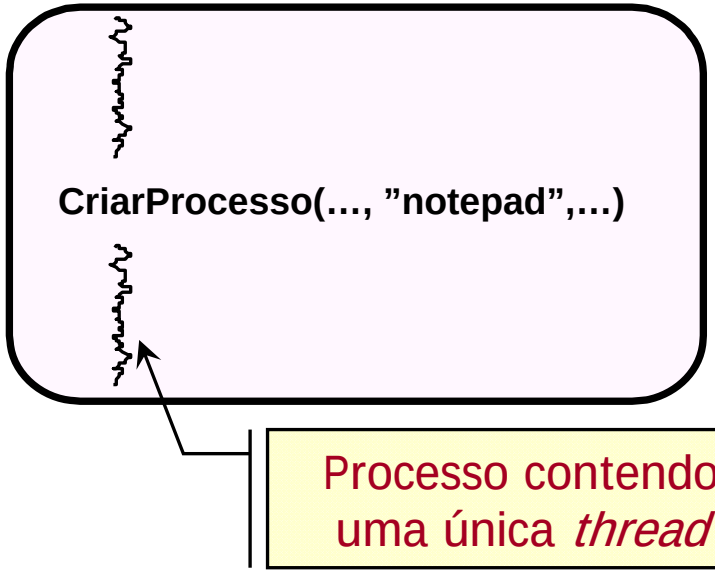
Lightweight Process

Tarefas


Untitled - Notepad
File Edit Format View Help
Application
File Edit Help
Olá Mundo
Windows Task Manager
File Options View Help
Applications Processes Performance Networking
Image Name User Name CPU Mem Usage Threads
notepad.exe no 00 2.236 K 1
cmd.exe no 00 80 K 1
explorer.exe no 02 10.612 K 16
POWERPNT.EXE no 00 6
java.exe no 00 12
calc.exe no 00 2.752 K 1
oasclnt.exe no 00 2.428 K 3
alg.exe LOCAL SERVICE 00 2.376 K 6
MSASCui.exe 00 6.448 K 17
acrotray.exe 00 2.316 K 2
daemon.exe no 00 2.320 K 2
vssrvc.exe NETWORK SERVICE 00 7.364 K 8
sychost.exe SYSTEM 00 6.068 K 6
khooker.exe no 00 3.052 K 3
wdfmgr.exe LOCAL SERVICE 00 1.444 K 4
spoolsv.exe SYSTEM 00 6.576 K 16
Show processes from all users
End Process
Processes: 47 CPU Usage: 2% Commit Charge: 358M / 1121M
DEETC - ISEL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.deetc.isel.ipl.pt/
Internet
Microsoft PowerPoint - [Processos.ppt]
File Edit View Insert Format Tools Slide Show Window Help Adobe PDF
Processos
Olá Mundo
Click to add notes
Slide 4 of 19 TemplateFSO Portuguese (Portugal)
Calculator
Edit View Help
Backspace CE C
MC MR MS M+
sqrt % 1/x

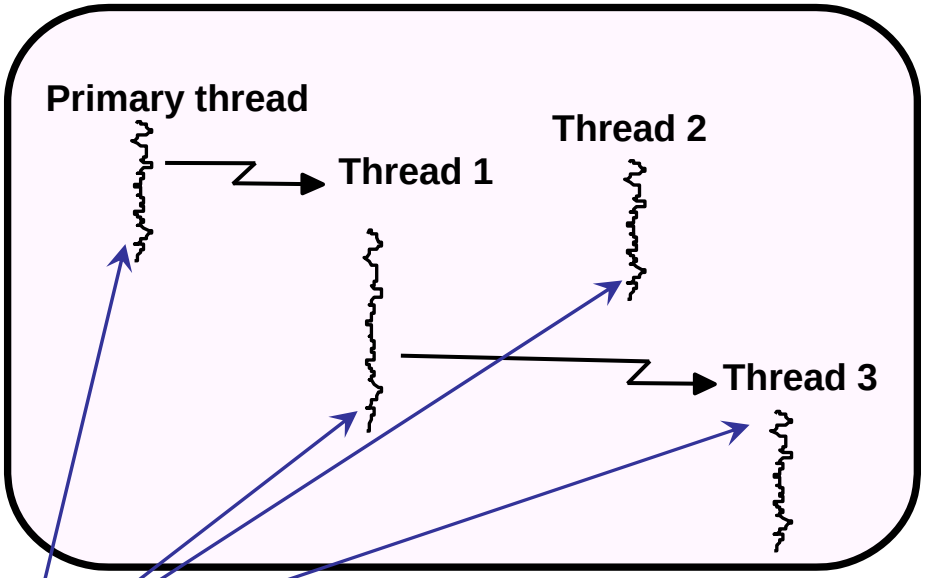
Tarefas

Processo Run



Processo contendo uma única *thread*

Processo Concorrente



Processo contendo várias *threads*

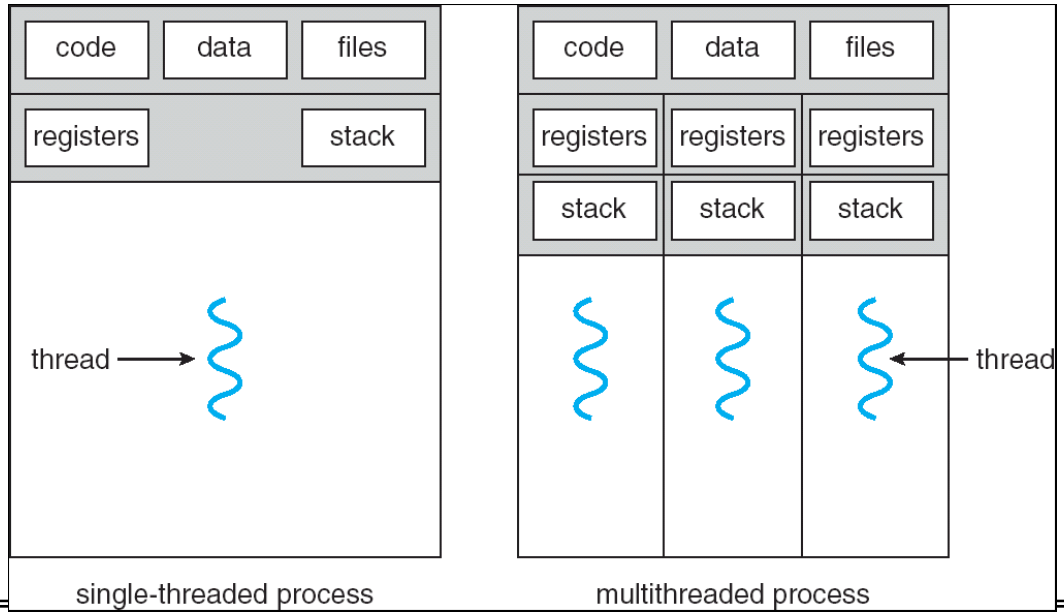
Conceito de Tarefa

- Uma tarefa (thread), é uma entidade que se executa no contexto de um processo. Um processo pode conter várias tarefas. Cada processo tem pelo menos uma tarefa (associada ao entry point da aplicação);
- As tarefas são um fluxo de execução com um início, uma sequência e um fim;
- A utilização de tarefas permite simplificar a criação de programas que lidem com diferentes fontes de dados. Esta utilização permite uma boa utilização de recursos;
- As tarefas de um processo partilham o espaço de endereçamento do processo. Esta partilha leva a que possam surgir problemas de concorrência;
- Cada tarefa tem o seu *stack* próprio;
- Não pode ser assumida qualquer ordem de execução entre as várias tarefas, o programador deve considerar que as tarefas se executam em paralelo;

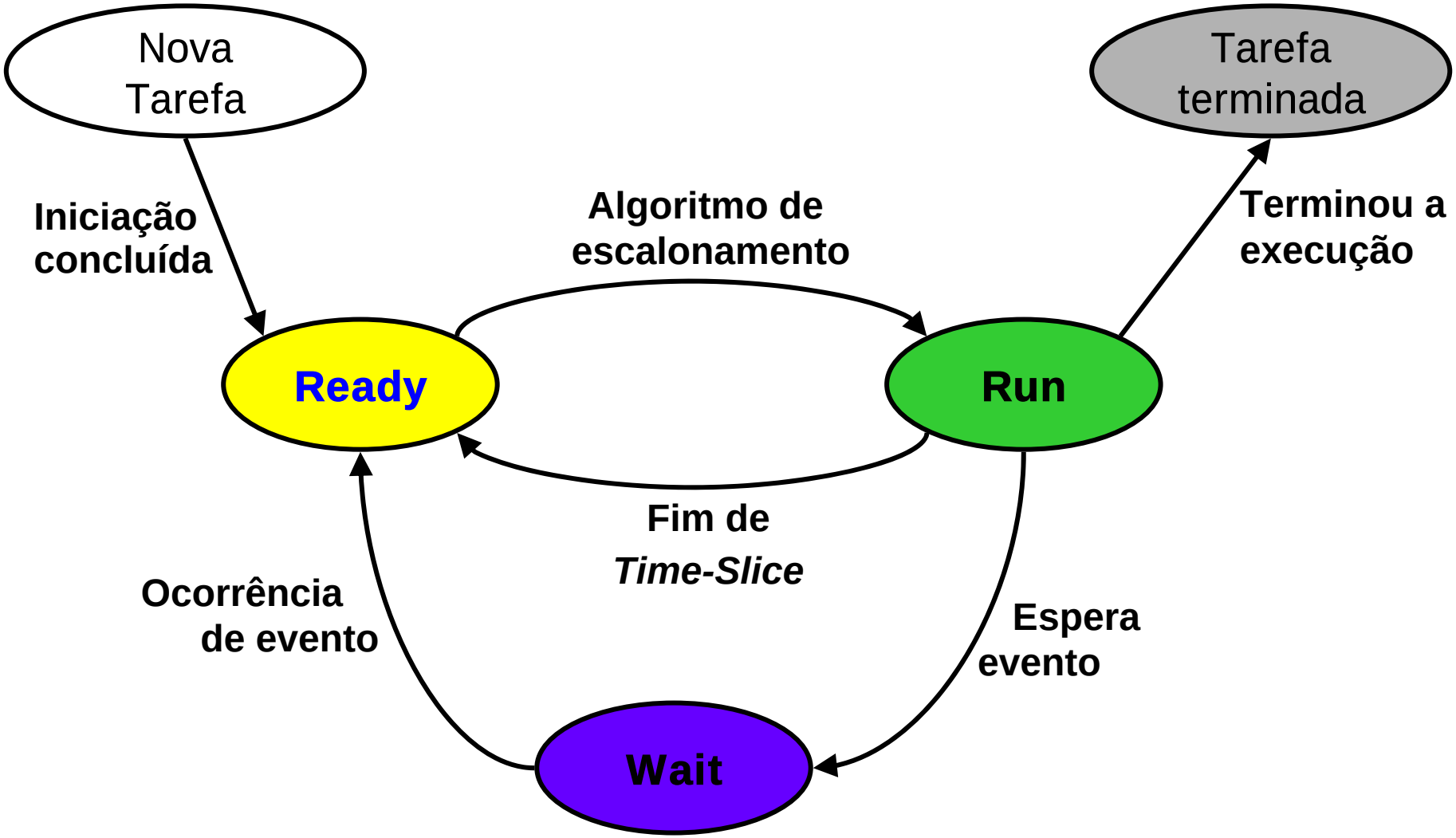
Qual o objectivo de cada *thread* ter o seu *stack*?

Conceito de Tarefa

- **Do mesmo modo que o processo possui uma estrutura PCB, cada tarefa de um processo terá uma estrutura semelhante :**
  - Identificador da tarefa;
  - Instruction Pointer (IP) / Program Counter (PC);
  - Um stack próprio para guardar parâmetros e variáveis locais da tarefa;
  - Registos do CPU;
  - Estado de execução da tarefa (Wait, Ready e Run);



Ciclo de vida de uma tarefa



Processo versus Tarefa

- **Criação:**
  - Computacionalmente é mais pesado criar um processo que uma tarefa;

- **Isolamento de dados:**
  - Um processo tem os seus dados isolados face aos restantes processos que executam no sistema;
  - As tarefas de um dado processo partilham o mesmo espaço de endereçamento;

- **Escalonamento:**
  - Em termos computacionais é mais pesado efectuar a comutação entre processos distintos do que efectuar a comutação entre tarefas do mesmo processo;
  - Efectuar a comutação entre tarefas de processos diferentes tem o mesmo peso computacional que efectuar a comutação de processos.

Scheduling

Scheduling

- Num sistema multiprogramado existem vários processos a competir pelo CPU.
- Esta situação ocorre quando 2 ou mais processos se encontram no estado Ready
- A parte do sistema operativo responsável pela escolha de um dos processos é o ***Scheduler*** e o algoritmo utilizado é designado de **algoritmo de *scheduling***

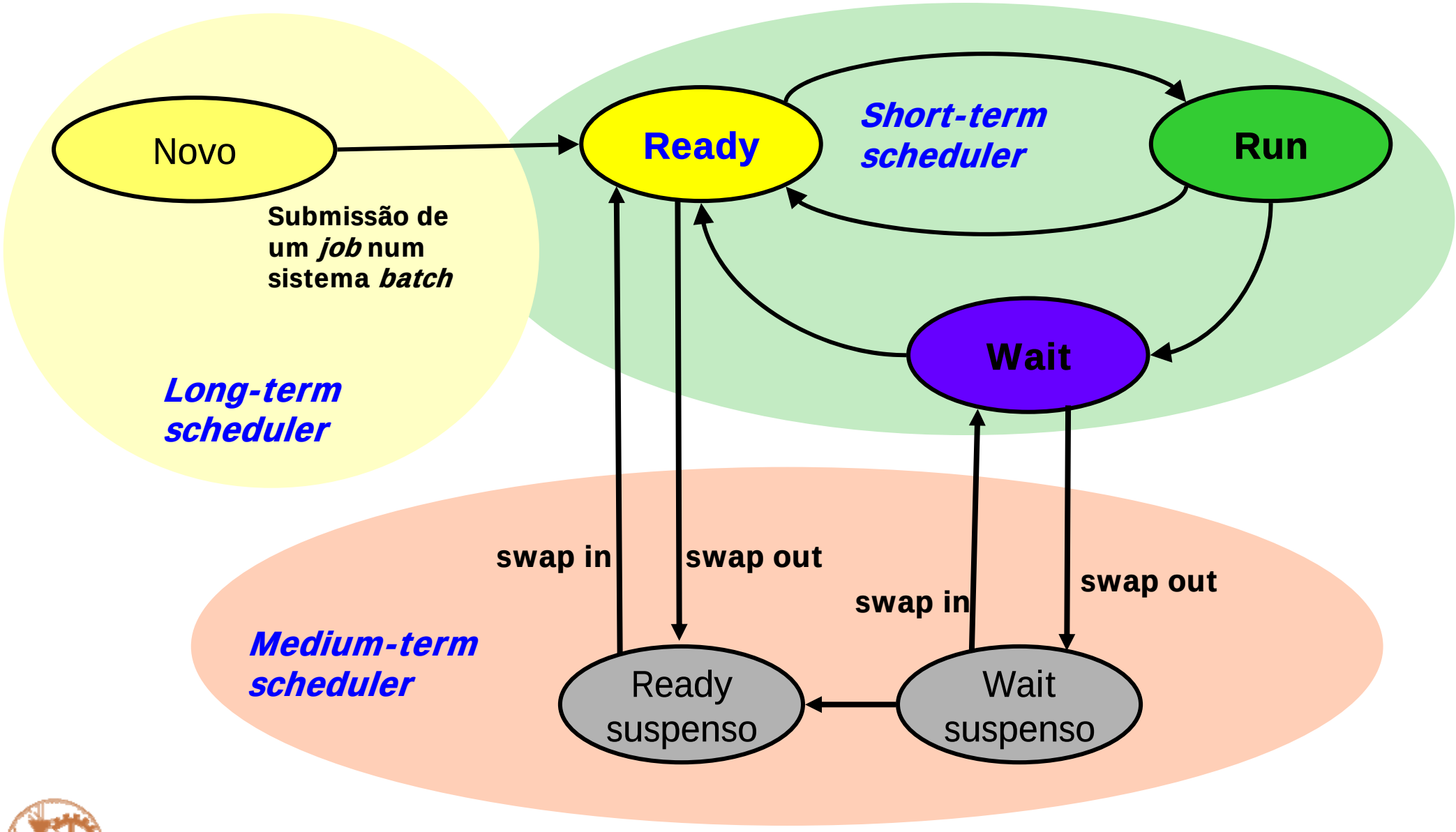
Scheduling

- O escalonador de curto prazo ou de CPU (***short-term scheduler***) selecciona um dos processos da fila *Ready* e atribui-lhe o CPU

- O escalonador de médio prazo (***medium-term scheduler***), utilizado em alguns SO's, tem como objectivo o de seleccionar para memória principal um dos processos parcialmente executados que foram retirados da memória (*swapped out*)

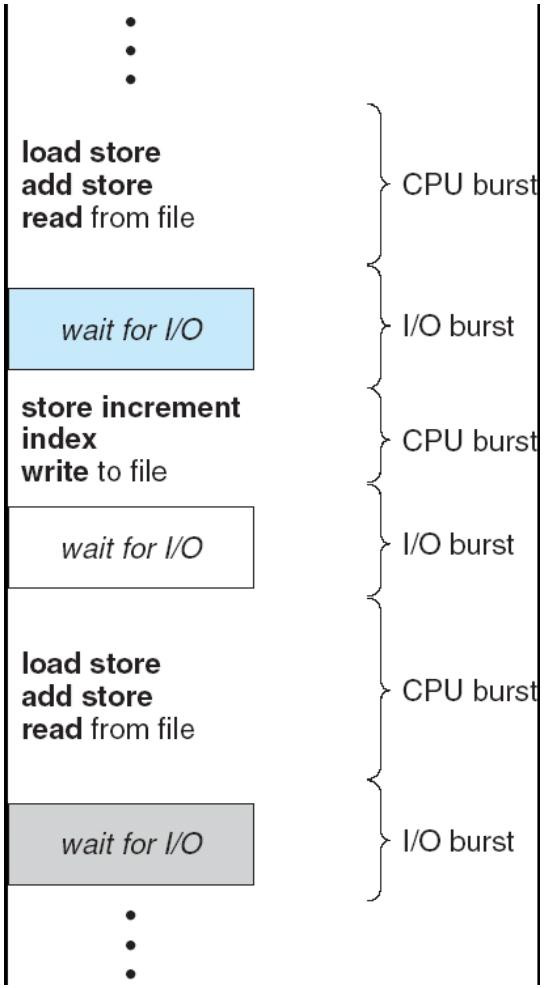
- O escalonador de longo prazo (***long-term scheduler***),
  - Frequentemente, nos sistemas batch são submetidos mais processos do que os que podem ser executados imediatamente
  - Nestes sistemas multiprogramados podem existir processos que têm de esperar em memória secundária pela possibilidade de serem carregados em memória principal
  - A escolha do próximo processo a ser carregado em memória é realizada por este tipo de escalonadores

Scheduling


Novo
Submissão de um *job* num sistema *batch*
Long-term scheduler
Ready
Short-term scheduler
Run
Wait
swap in
swap out
swap in
swap out
Medium-term scheduler
Ready suspenso
Wait suspenso

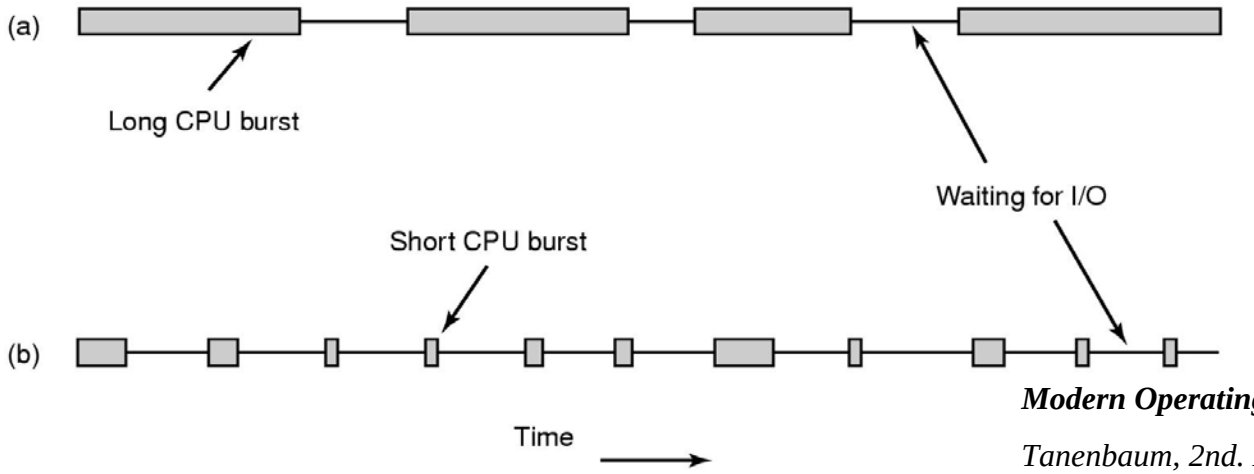
Ciclos CPU e I/O

- **Comportamento dos processos**
  - Os processos alternam entre períodos de computação (**CPU *burst***) e períodos de I/O (**I/O *burst***)



Ciclos CPU e I/O

- **Comportamento dos processos**
  - Os processos alternam entre períodos de computação (**CPU *burst***) e períodos de I/O (**I/O *burst***)
  - Existem processos que tem maiores períodos de computação (***compute bound***) enquanto outros têm maiores períodos de I/O (**I/O *bound***)



***Modern Operating Systems**, Andrew S. Tanenbaum, 2nd. Ed., Prentice Hall, 2001*

No-preemptive scheduling

- **Escalonamento sem preempção (*no-preemptive*) ou cooperativo**
  - Um vez atribuído o CPU atribuído a um processo este só liberta o CPU se transitar para o estado *Wait* ou terminar
  - Único método possível em algumas plataformas hardware (não requer características especiais, e.g. *timer*)
  - O acesso aos recursos partilhados é simplificado (o processo controla o ponto onde pode perder o CPU)
  - Facilita o desenho do Sistema Operativo
  - Utilizado nos sistemas Windows 3.x e nas primeira versões Macintosh

Preemptive scheduling

- **Escalonamento com preempção (*preemptive*)**
  - O algoritmo de escalonamento escolhe um processo e deixa-o executar-se durante um determinado tempo máximo. Se no fim deste intervalo de tempo o processo ainda estiver em execução o *scheduler* suspende-o (preempção) e escolhe outro processo para executar
  - Necessidade da existência de um *interrupt* periódico para garantir a execução do *scheduler*
  - Necessidade de mecanismos para coordenar o acesso a recursos partilhados
  - Maior complexidade no desenho do Sistema Operativo
  - Utilizado em todas versões a partir do Windows 95 e na versão OS X da Macintosh

Critérios de escaloamento

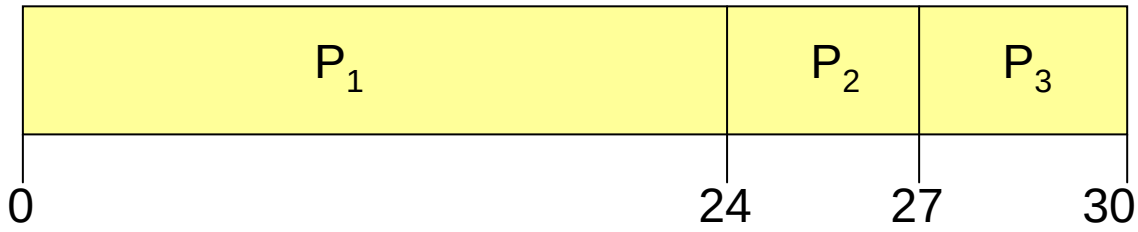
Optimização

- **Utilização do CPU** – mantê-lo o o mais ocupado possível ↗
- ***Throughput*** – nº de processos que terminam a sua execução por unidade de tempo ↗
- ***Turnaround time*** – tempo necessário para executar um processo ↘
- ***Waiting time*** – tempo total que o processo esteve à espera na fila Ready ↘
- ***Response time*** – tempo que decorreu desde a submissão do processo para a fila *Ready* até ser escalonado para execução (produzir a primeira resposta) ↘

First-Come First-Served (FCFS)

Process	Burst Time
P1	**24**
P2	**3**
P3	**3**

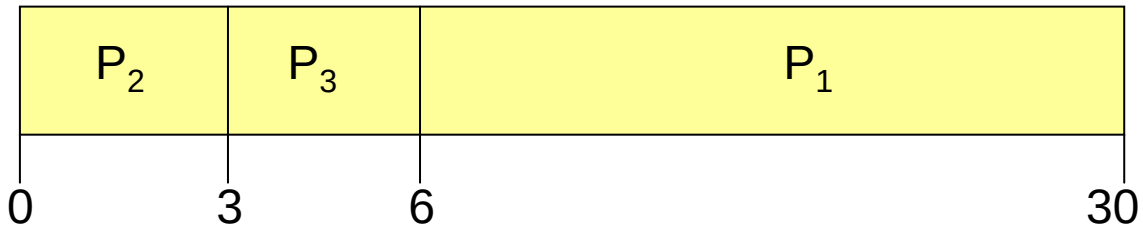
- Se considerarmos a ordem de chegada: ***P1 , P2 , P3***



- Tempo de espera (*waiting time*): *P1* = 0; *P2* = 24; *P3* = 27
- Tempo médio de espera (*average waiting time*): (0 + 24 + 27)/3 = 17

First-Come First-Served (FCFS) (II)

- Consideremos agora a seguinte ordem de chegada dos mesmos processos: ***P*** $_2$ **,** ***P*** $_3$ **,** ***P*** $_1$



- Tempo de espera (*waiting time*): $P_1 = 6; P_2 = 0, P_3 = 3$
- Tempo médio de espera (*average waiting time*):(6 + 0 + 3)/3 = 3
- Melhor que no caso anterior

Conclusão: é vantajoso executar primeiro os trabalhos mais pequenos

Shortest-Job-First (SJF)

- **Associar a cada processo o tempo do próximo período de computação (*CPU burst*). Com base nestes tempos atribuir, em primeiro lugar o CPU, aos processos com menor valor.** (O tempo do próximo CPU *burst* é, geralmente, estimado ver bibliografia)

- **Duas alternativas:**
  - **Non-preemptive** – Depois de atribuir o CPU a um processo este não pode ser interrompido (*preempted*) até terminar o período de computação (CPU *burst*)
  - **preemptive** – Se aparecer um novo processo com um período de computação menor que o tempo restante do processo em execução procede à preensão do processo em execução. Este algoritmo é conhecido por **Shortest-Remaining-Time-First (SRTF)**

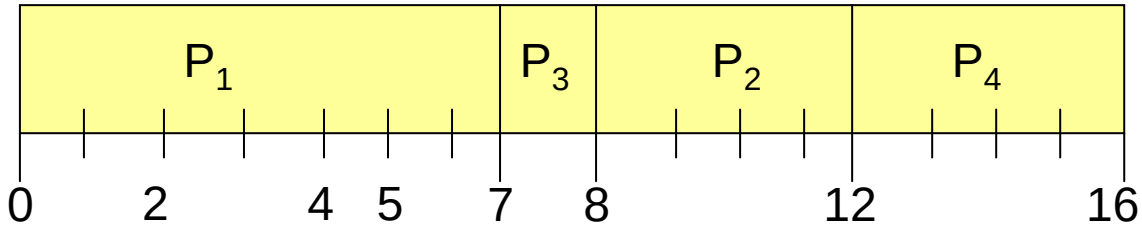
- **Utilizado frequentemente pelos escalonadores de longo prazo**

- **SJF minimiza o tempo médio de espera (*average waiting time*) para um conjunto de processos**

Shortest-Job-First (SJF) (II)

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- **SJF (*non-preemptive*)**

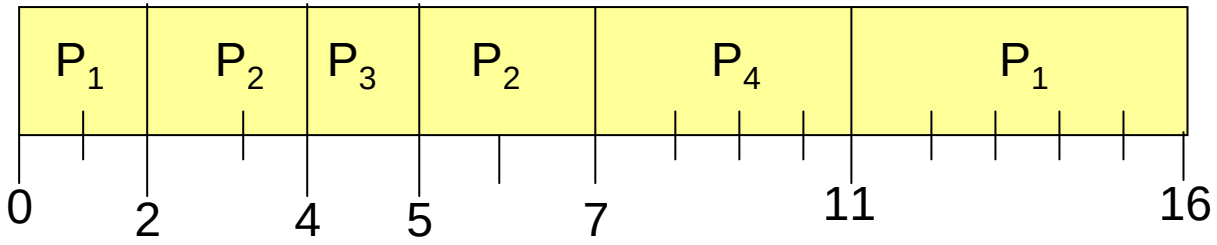


- Tempo médio de espera = (0 + 6 + 3 + 7)/4 = 4

Shortest-Job-First (SJF) (III)

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- **SJF (*preemptive*)**



- Tempo médio de espera = (9 + 1 + 0 +2)/4 = 3

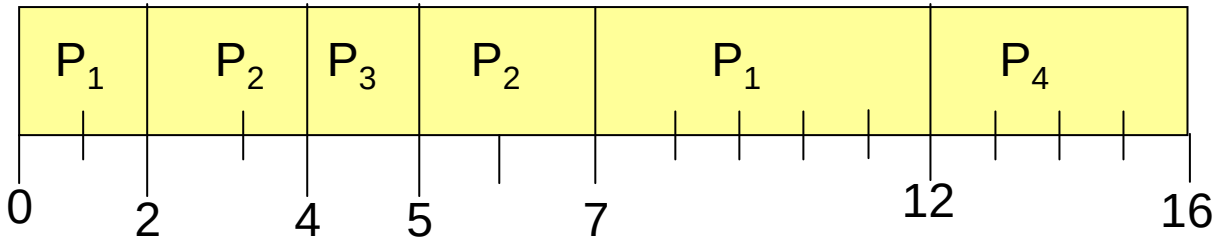
Priority Scheduling

- **A prioridade é um valor inteiro associado a cada processo**
- **O CPU é atribuído ao processo com maior prioridade**
  - *Preemptive*
  - *Non-preemptive*
- **O SJF é uma algoritmo de escalonamento baseado em prioridade onde a prioridade é definida pelo próximo tempo de CPU *burst***
- **Pode ocorrer que os processos de baixa prioridade nunca se executem – *starvation***
- **Para evitar o *starvation* pode-se aumentar a prioridade desses processos em função do tempo decorrido (*Aging*)**

Priority Scheduling

Process	Arrival Time	Burst Time	Priority
P_1	0.0	7	3
P_2	2.0	4	2
P_3	4.0	1	1
P_4	5.0	4	3

- **Prioridade mais alta corresponde ao valor mais baixo**



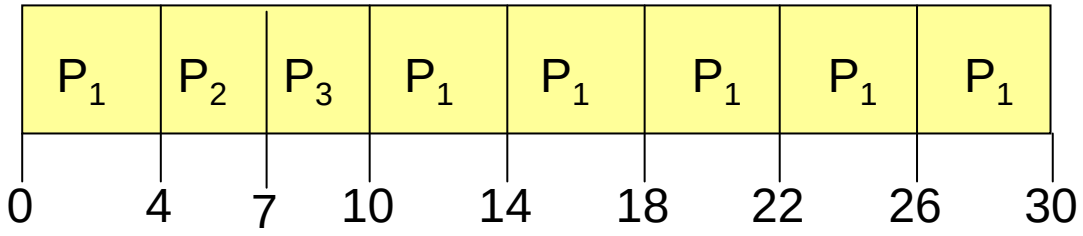
Round Robin (RR)

- **Cada processo obtém um pequena unidade de CPU (*time quantum*) da ordem dos 10-100 ms. Quando o tempo expirar o processo é retirado de execução e adicionado á fila Ready**
- **Se existirem *n* processos na fila Ready e o *time quantum* for *q* então cada processo tem 1/*n* do tempo do CPU em unidades *q* e nenhum processo espera mais que (*n*-1)**q* unidades de tempo**
- **Desempenho**
  - *q* grande ⇒ FCFS
  - *q* pequeno ⇒ *q deve ser grande em relação ao tempo de context switch* senão provoca um *overhead* elevado

Round Robin (RR)

Process	Burst Time
P1	**24**
P2	**3**
P3	**3**

- Time Quantum = 4

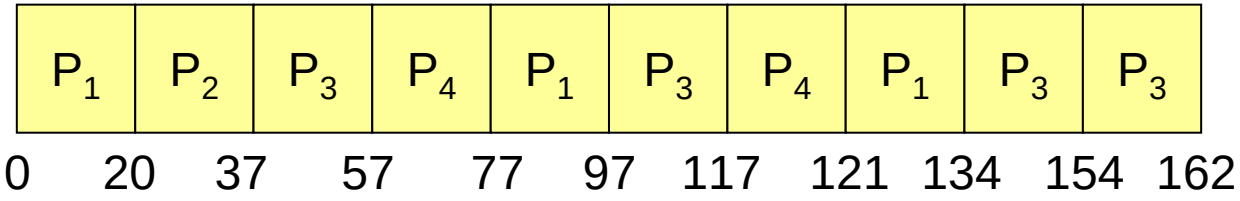


- Tempo de espera (*waiting time*): *P1* = 6; *P2* = 4; *P3* = 7
- Tempo médio de espera (*average waiting time*): (6 + 4 + 7)/3 = 5,66

Round Robin (RR)

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

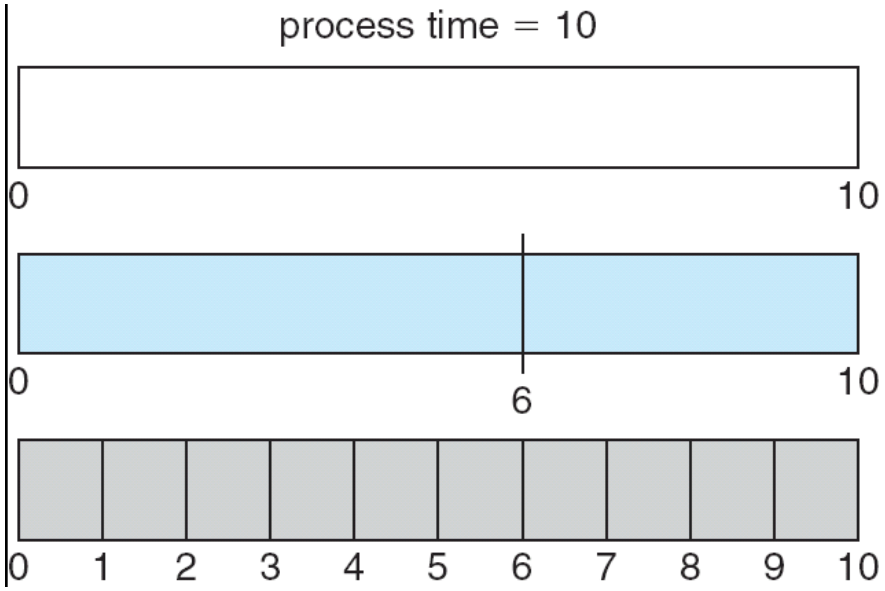
- Time Quantum = 20



RR: Turnaround = 113,5; ResponseTime = 28,5
SJF: Turnaround = 78,5; ResponseTime = 38

- Tipicamente o tempo médio necessário para terminar a execução de um processo (*turnaround*) é maior do que aplicando o **SJF** mas com um melhor tempo de resposta (*Response time*)

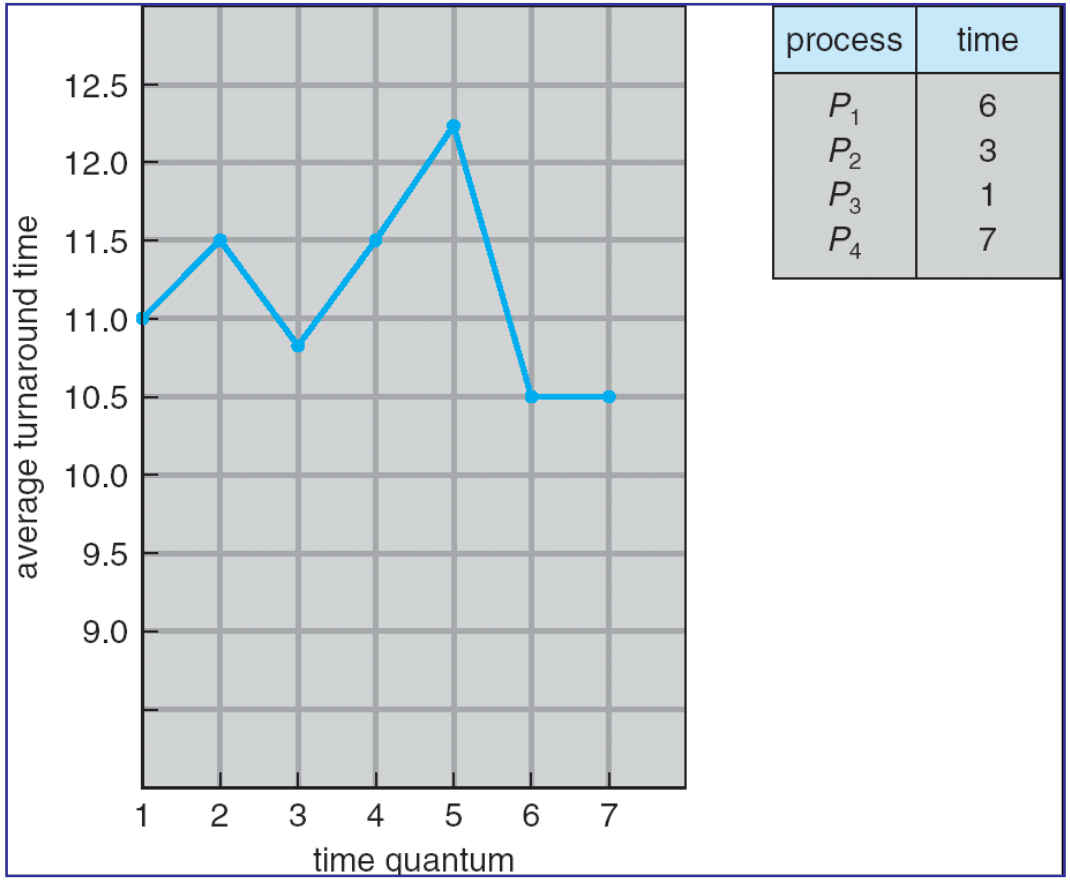
Time Quantum / Context Switch



quantum	context switches
12	0
6	1
1	9

Time Quantum / Turnaround Time

O tempo necessário para executar um processo (*Turnaround time*) depende do *time quantum* e aumentando-o não significa, necessariamente, melhor tempo de *Turnaround* .

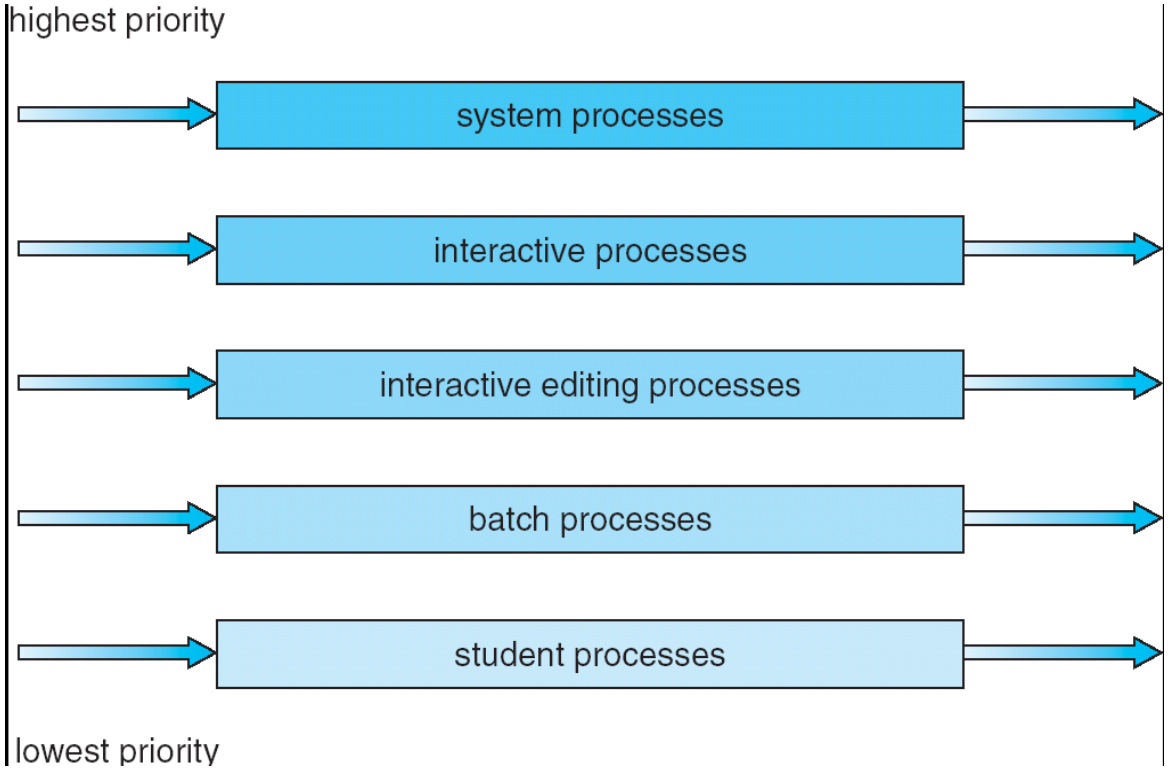


process	time
P_1	6
P_2	3
P_3	1
P_4	7

Multilevel Queue

- **Outra classe de algoritmos foi criada para situações onde os processos podem ser classificados em diferentes grupos em função de alguma característica sua, i.e., dimensão da memória, prioridade, tipo do processo**
- **A fila Ready pode ser dividida:**
  - *foreground* (*interactive*)
  - *background* (*batch*)
- **Cada fila pode ter um algoritmo de escalonamento específico**
  - foreground – RR
  - background – FCFS
- **Escalonamento deve ser realizado entre as filas**
  - Prioridade fixa: primeiros os processos *foreground* e depois os *background* – possibilidade de *starvation*.
  - *Time slice* – cada fila obtém uma percentagem do tempo de CPU
    - 80% to foreground in RR
    - 20% to background in FCFS

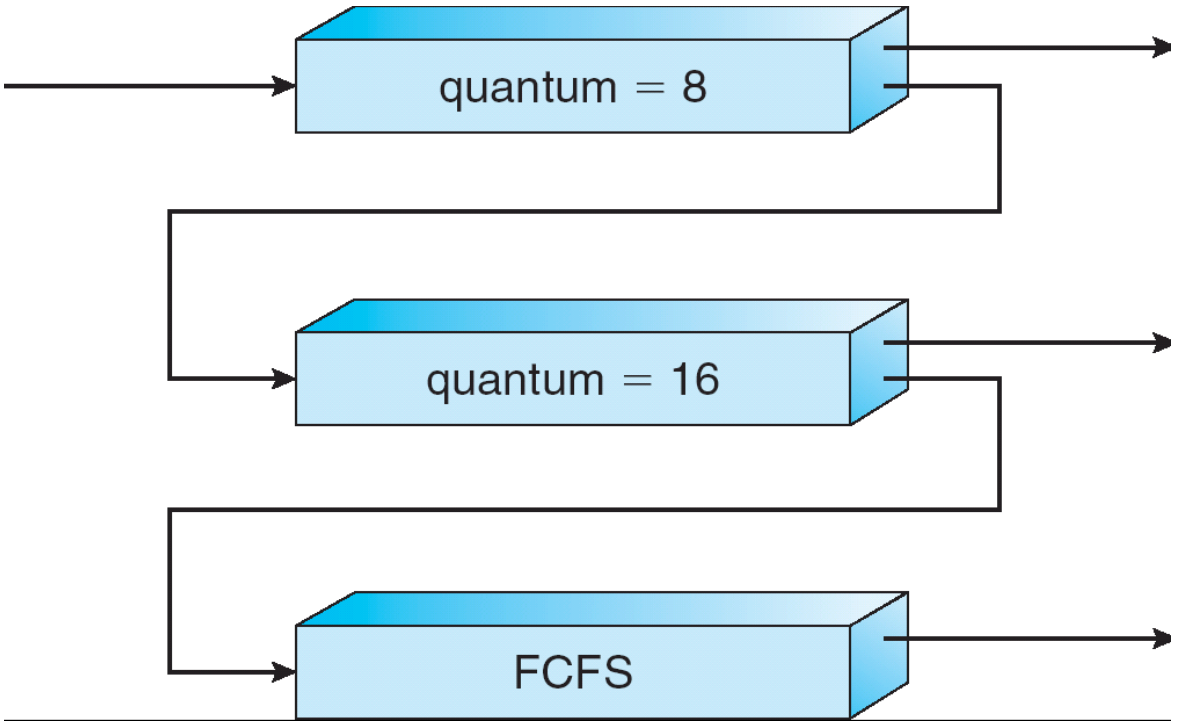
Multilevel Queue



Multilevel Queue

- Separar processos com diferentes características de computação (*CPU-burst*)
- Se um processo utiliza muito tempo de processamento é deslocado para uma fila de baixa prioridade
- Os processos interactivos e com muito I/O (*I/O-bound*) ficam em filas mais prioritárias
- Um processo que esteja há muito tempo numa fila de baixa prioridade pode ser deslocado para uma de maior prioridade evitando o *starvation* (forma de implementar o *aging*)

Multilevel Feedback Queues



Escalonamento no Windows

Escalonamento Windows

- **Escalonamento de tarefas**
  - As tarefas interactivas (rato, janelas, etc) devem ter um bom tempo de resposta
  - Favorecimento de tarefas I/O Bound para manter os periféricos ocupados
  - *Soft real-time*
- **Escalonamento entre múltiplas filas de prioridades diferentes**
  - Iniciando-se pela fila mais prioritária
- **Escalonamento na mesma fila**
  - *Round Robin* com *time slice*

Escalonamento Windows

- **O Windows usa internamente 32 (0..31) níveis de prioridade divididos nas seguintes classes:**
  - 16 níveis para real-time (16..31)
  - 15 níveis variáveis (1..15)
  - 1 nível sistema (0) - reservado para uma tarefa do gestor de memória (*zero page thread*)
- **Os processos são organizados por classes de prioridade:**
  - Idle; Below Normal; Normal; Above Normal; High; Real-Time
- **As threads de um processo têm prioridades relativas à do processo:**
  - Time-critical; Highest; Above-normal; Normal; Below-normal; Lowest; Idle

Escalonamento Windows

O valor de prioridade de uma tarefa resulta da combinação da classe de prioridade do processo e da sua prioridade relativa

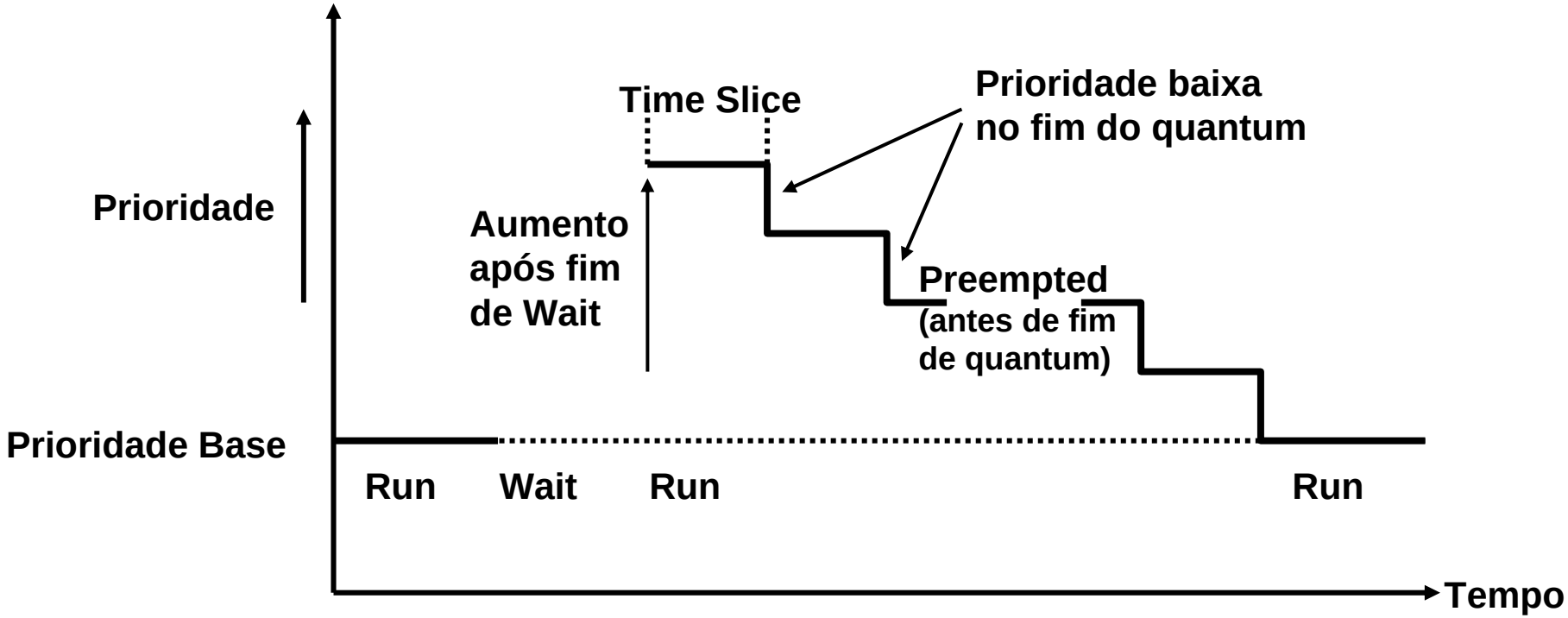
Classes de Prioridade dos Processos

Prioridade relativa das Threads	Idle	Below Normal	Normal	Above Normal	High	Real Time
Time Critical	15	15	15	15	15	31
Highest	6	8	10	12	15	26
Above Normal	5	7	9	11	14	25
Normal	4	6	8	10	13	24
Below Normal	3	5	7	9	12	23
Lowest	2	4	6	8	11	22
Idle	1	1	1	1	1	16

Escalonamento Windows

- **Cada tarefa tem dois valores de prioridade: a corrente e a base**
- **Um processo tem um único valor de prioridade**
- **Uma tarefa com menor prioridade pode ser *preempted* por outra tarefa de maior prioridade que ficou *ready***
- **As tarefas podem ter o seu valor de prioridade ajustada em tempo de execução (prioridade corrente)**
  - As tarefas com prioridade entre 1 e 15 recebem aumentos (*boosts*) quando:
    - Terminam uma operação de I/O
      - Melhorar os tempos de resposta das tarefas interactivas
    - Terminam uma operação de sincronização (semáforo, mutexes)
    - Tarefas que estejam muito tempo na fila *Ready* sem se executarem (prioridade alterada para 15 durante 2 *time slices*)
  - Quando uma tarefa ocupa todo o seu quantum a sua prioridade é reduzida
    - Nunca desce abaixo da prioridade base
  - As tarefas com prioridades real-time (16..31) nunca sofrem alterações no seu valor de prioridade
- **Aumento do *quantum* das tarefas do processo em *foreground (tipico 3x)***

Escalonamento Windows



Exemplos de valores de *quantum*:
– Windows 2000 Professional: 20 ms (favorece a interactividade)
– Windows 2000 Server: 120 ms (favorece a produtividade)

Funções WIN32 sobre prioridade

```
DWORD GetPriorityClass(HANDLE hproc);

BOOL SetPriorityClass(HANDLE hproc, DWORD dwPriorityClass);


BOOL SetThreadPriority(HANDLE hThread, int nPriority);

int GetThreadPriority(HANDLE hThread);
```