Prof. Ricardo Inácio Álvares e Silva

Introdução a Processos

Sistemas Operacionais

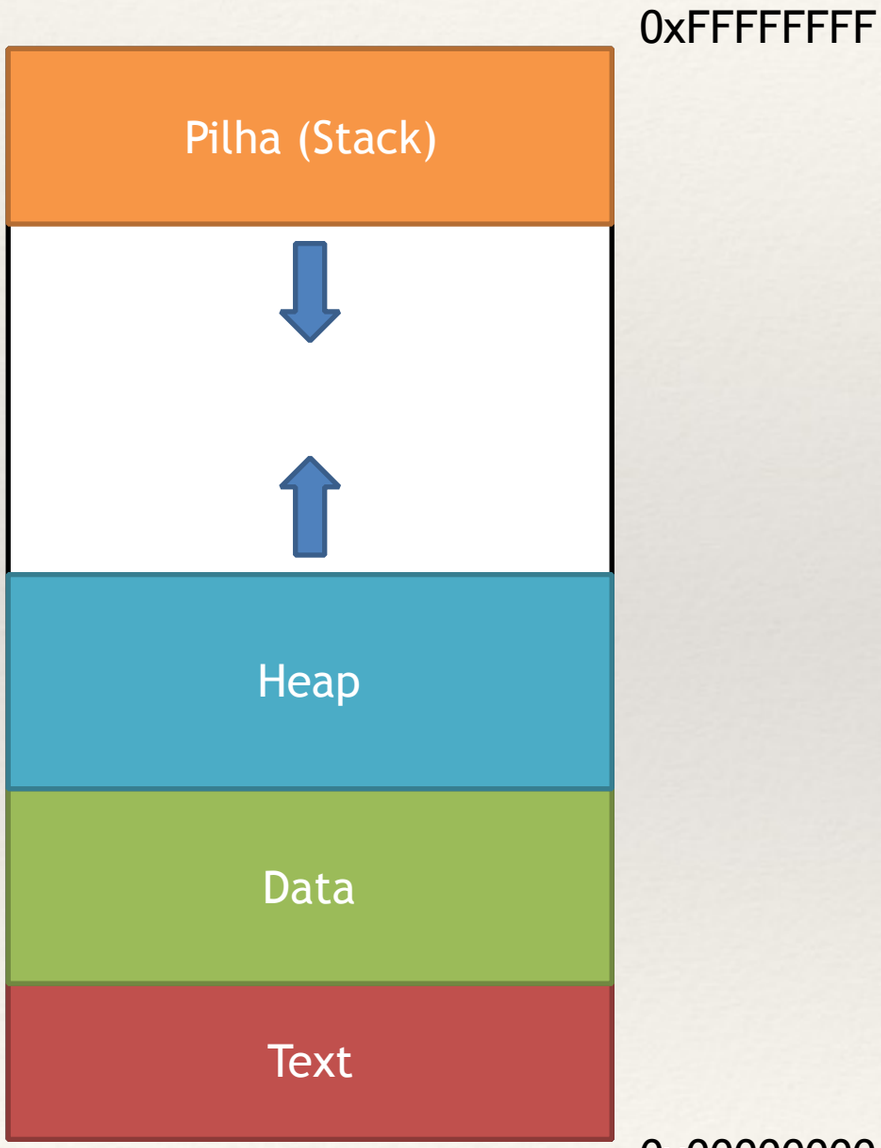
Escopo da aula

- Processos
  - Estrutura
  - Ciclo de vida
  - Pseudoparalelismo
  - Process Control Block
  - Escalonamento
  - Thread

Para relembrar…

- **Sistema multiprogramado**: capaz de carregar na memória e executar vários programas paralelamente, chamados de processos.
- **Processo**: uma instância de um programa em execução, e todos os atributos necessários ao controle dessa execução

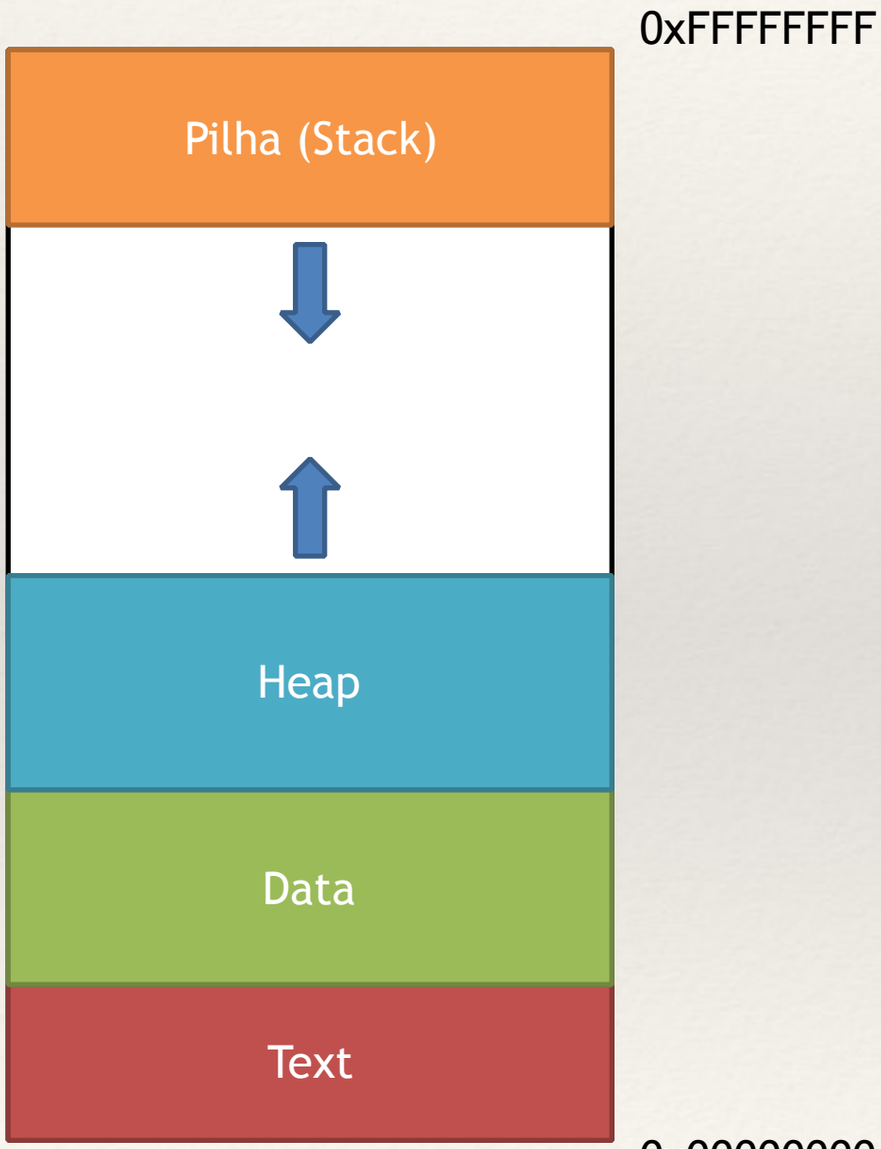
Estrutura de um processo



- **Pilha**: Guarda variáveis do programa
- Exemplo:

```
int main(int argc, char* argv[]) {
    int x, y, z;
    x = 1;
    y = 5;
    z = 8;
}
```

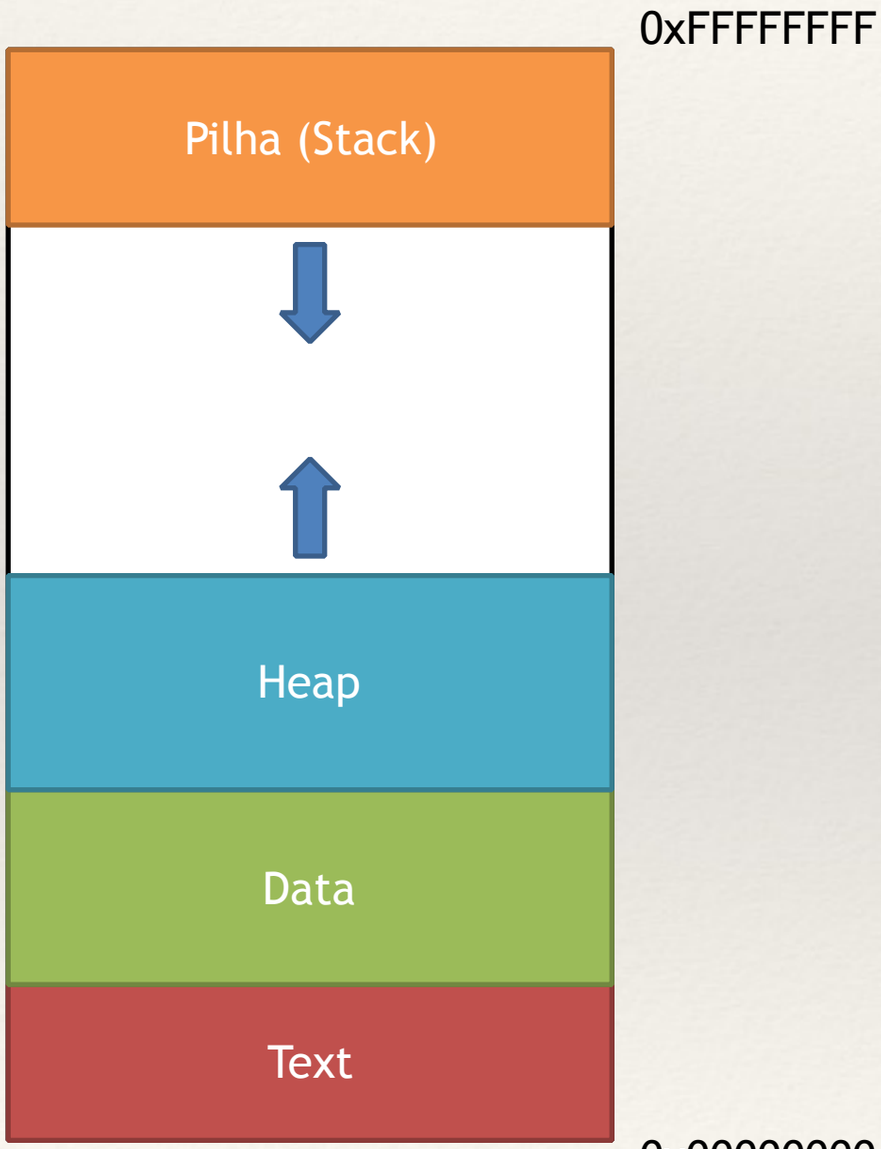
Constituição de um processo



- **Heap**: Guarda memória alocada dinamicamente pelo programa
- Exemplo:

```
int main(int argc, char* argv[]) {
    int x, y, z;
    char nome[] = new String("Aluno");
    x = 1;
    y = 5;
    z = 8;
}
```

Constituição de um processo



- **Data**: Guarda dados constantes e globais utilizados pelo programa
- Exemplo:

```
float altura = 1.79f;
int main(int argc, char* argv[]) {
    int x, y, z;
    char nome[] = "Ricardo";
    x = 1;
    y = 5;
    z = 8;
}
```

Text: são as instruções do programa, sendo executadas pelo processo. Exemplo:

```
int main(int argc, char* argv[])
{
    int x, y;
    x = 1;
    y = 5;
    if (x == y)
      puts(“Sao iguais”);
}
```

Assembly (figurativo), conteúdo do **text**, representa o algoritmo ao lado:

```
.data
STR000001:
.asciiz “Sao iguais”

.text
Main:
add $sp,$sp,8
li  $t0,1
li  $t1,5
jal $t0,$t1,PRINT
sub $sp,$sp,8
li  $v0, 0
syscall

PRINT:
li  $v0,5
li  $a0,STR000001
syscall
jr  $ret
```

Paralelismo

- Exemplo:

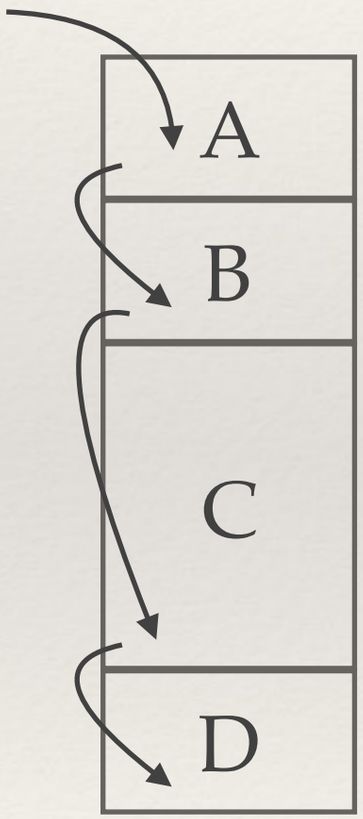
Em um ambiente multiprogramado, um usuário inicia um editor de vídeos. Ele ordena instrui a utilização de um efeito especial cuja aplicação levará pelo menos uma hora, e então abre o navegador Web. Enquanto isso, um processo em segundo plano que desperta periodicamente para verificar chega de emails é ativado. Dessa forma, tem-se três processos ativos. Periodicamente, o sistema operacional decide parar um deles para continuar outro.

- Sistema Operacional deve gerenciar a execução paralela dos diversos processos
- A existência de processos é o que possibilita **multiprogramação**
- Modelo de processo: cada processo é **sequencial**, só possui uma linha de execução de instruções (ou passos do algoritmo)
- Linha de execução: conhecida como **Thread**

- O paralelismo em sistemas multiprogramados na realidade é um **pseudoparalelismo**
  - há vários processos registrados e prontos para executar seus programas
  - o processador só executa um processo por vez
  - o tempo de execução de cada processo é muito curto (dezenas de milisegundos, no máximo)
  - quando termina o tempo de um processo, ele é interrompido e outro é colocado para executar
  - execução dos processos de forma intercalada
- Aparenta estarem todos executando ao mesmo tempo

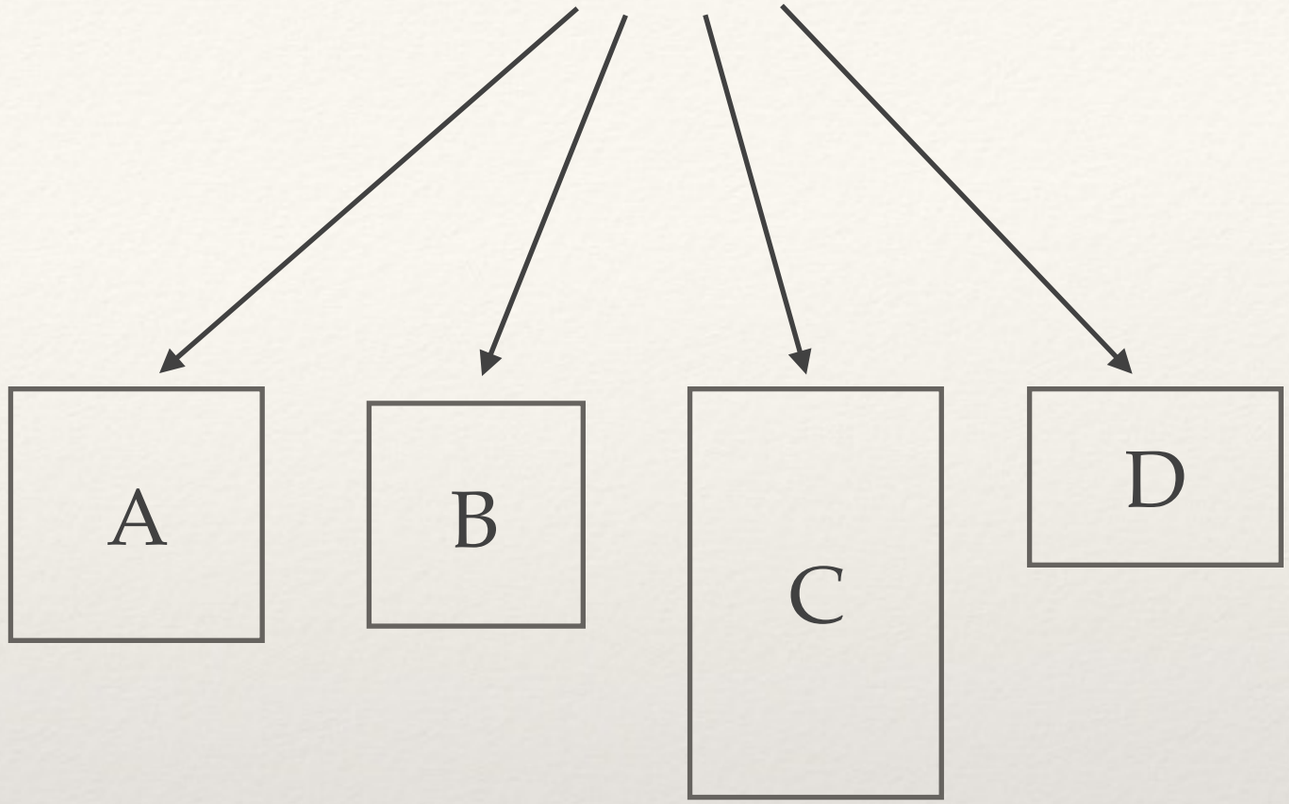
Visão do SO:

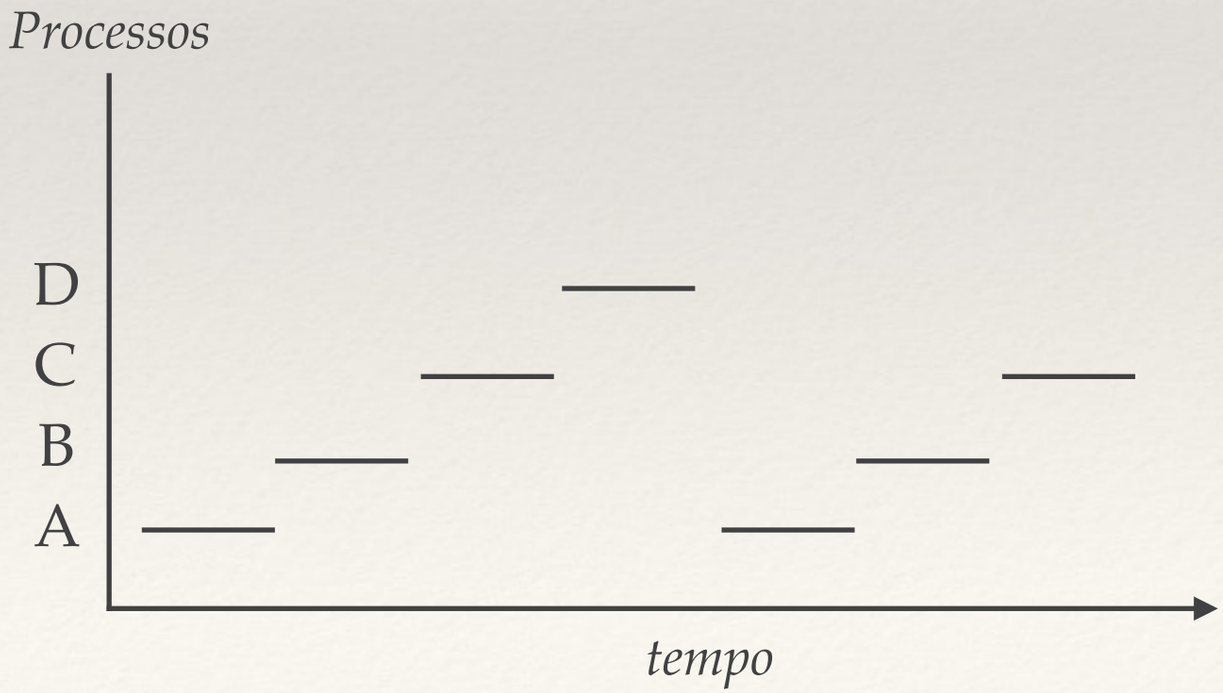
Um único contador de programa (PC)



Visão de cada processo:

Quatro PCs





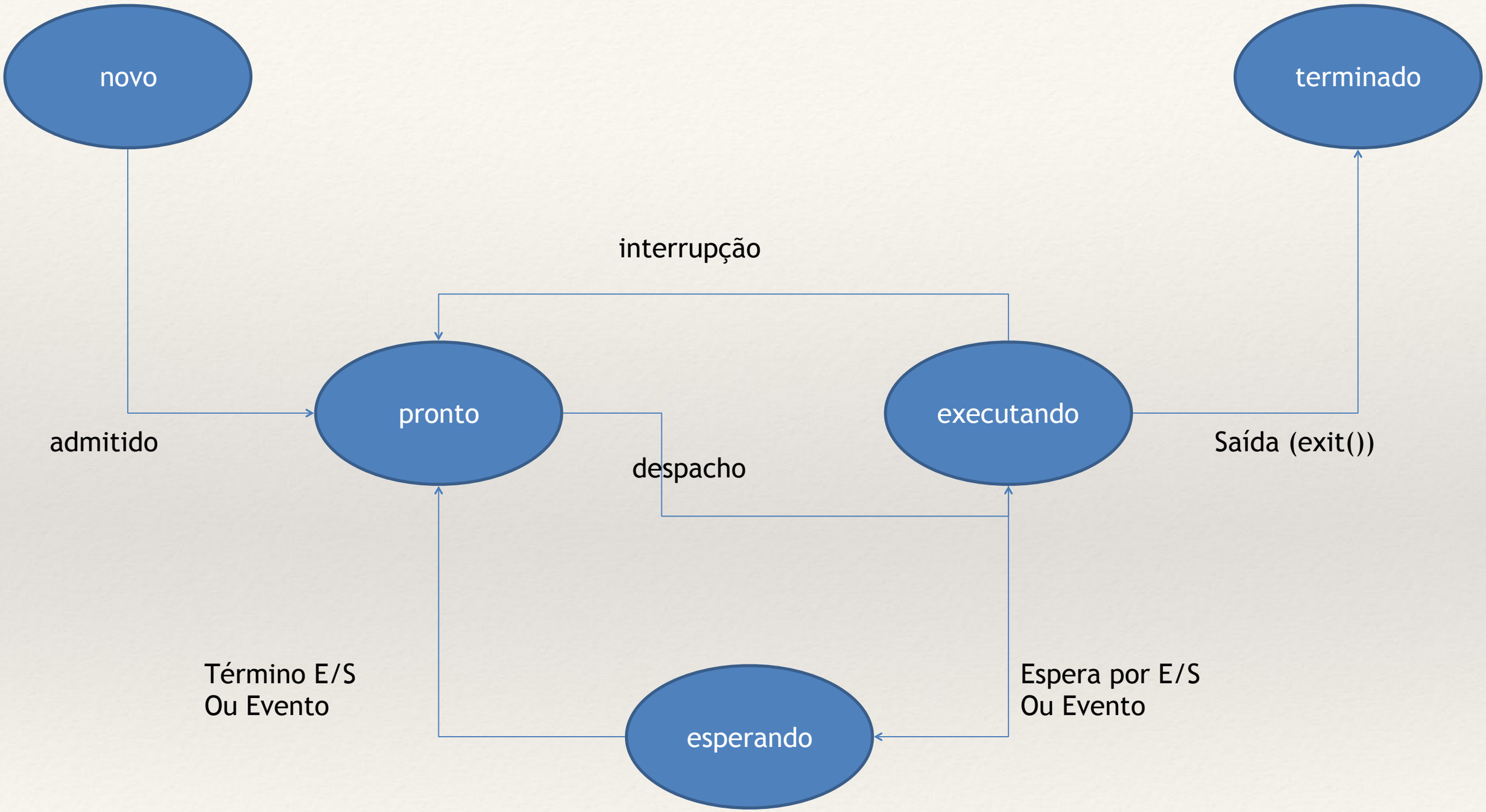
Ciclo de vida de um processo

- Um processo pode ser **criado** nas seguintes situações:
  - Momento de inicialização do sistema: cria pelo menos um processo, que se encarrega de criar os outros
  - Execução de uma chamada ao sistema realizada por um processo existente
  - Requisição de usuário para criar um processo (Windows)

- Um processo é **terminado** nas seguintes situações:
  - Término natural (voluntário): fim de programa ou erro
  - Término fatal (involuntário): erro de programação, bug, ou acesso indevido à recursos protegidos
  - Término a comando de outro processo (involuntário)

Estados do ciclo de vida

- **Novo**: processo está sendo criado
- **Executando**: as instruções estão sendo executadas
- **Esperando**: o processo está esperando algum evento
- **Pronto**: o processo está esperando ser colocado em execução em algum processador (está na fila de processos)
- **Terminado**: o processo terminou a sua execução, pode ser eliminado


novo
terminado
interrupção
admitido
pronto
executando
Saída (exit())
despacho
Término E/S
Ou Evento
esperando
Espera por E/S
Ou Evento

Process Control Block (PCB)

- Para trocar processos em execução pelo processador
  - o sistema operacional tem que que salvar dados essenciais de execução do atual processo
    - mais tarde deverá voltar a executar do ponto de onde estava, como se nunca tivesse sido suspenso
- Esses dados essenciais são conhecidos como **atributos de processos**
- Os atributos são salvos no **PCB** (Silberchatz), também conhecido como **tabela de processos** (Tanenbaum)
- O PCB fica em uma parte da memória destinada ao kernel, e é gerenciada apenas por ele, por motivos de segurança e proteção ao sistema

Atributos de um processo na PCB

- Os atributos que constituem um processo varia entre sistemas operacionais, mas o exemplo abaixo indica os principais, encontrados em praticamente todos:

Gerenciamento do Processo	Gerenciamento da memória	Gerenciamento de arquivos
Registradores *Contador de programa (PC)* *PSW* *Ponteiro de pilha (SP)* *Estado do processo* *Prioridade* *Parâmetros de escalonamento* *ID do processo* *Processo pai* *Grupo do processo* *Sinais* *Tempo de início* *Tempo de CPU usado* *Etc...*	*Ponteiro para text* *Ponteiro para data* *Ponteiro para pilha*	*Diretório raiz* *Diretório de trabalho* *Descritores de arquivos* *ID do usuário* *ID do grupo*

Representação de processo no Linux

```
struct task_struck {
    pid_t pid;
    long state;
    unsigned int time_slice;
    struct files_struct *files;
    struct mm_struct *mm;
}
```

Threads

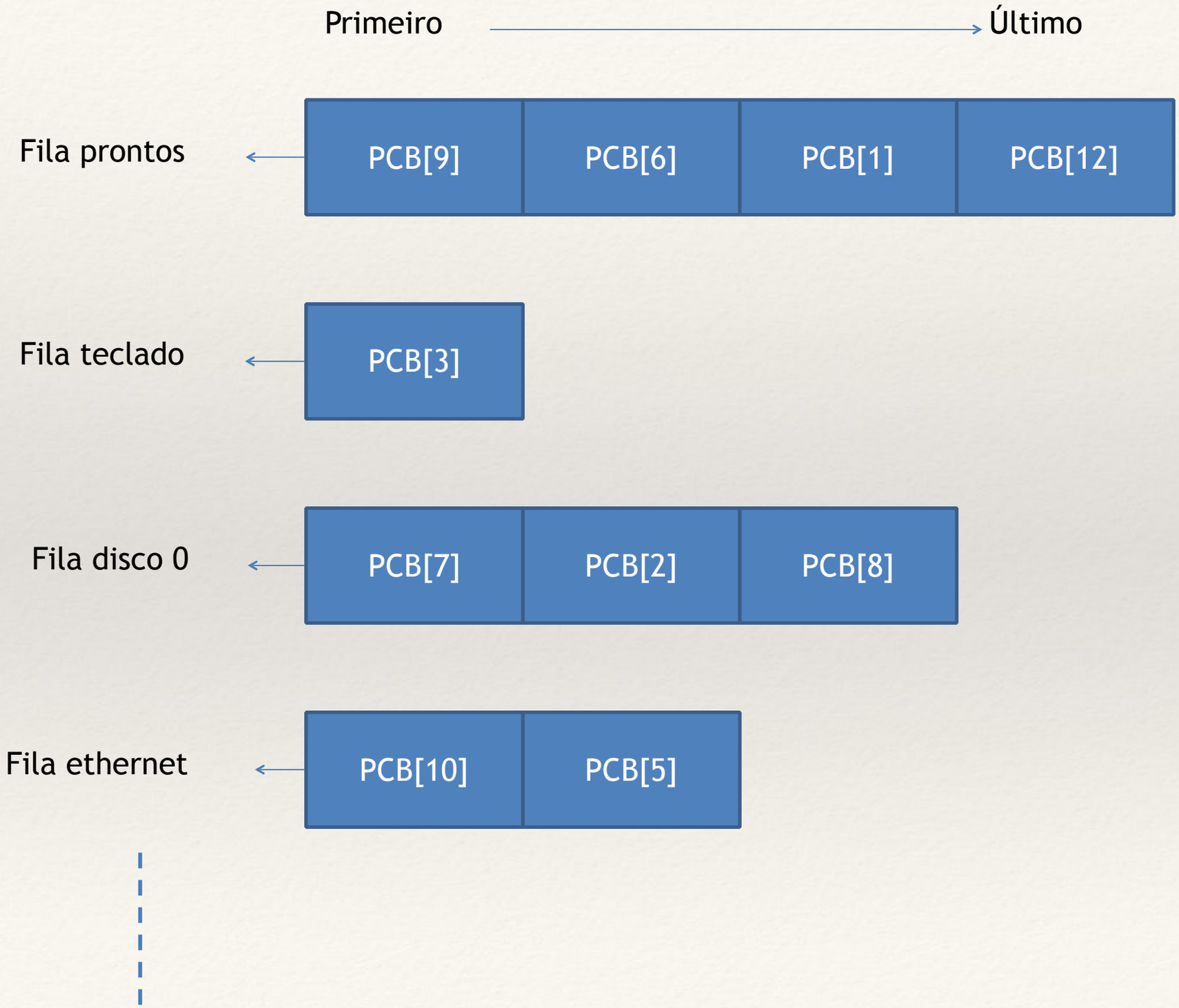
- Todo processo possui ao menos uma **thread**
- Threads são linhas de execução de instruções (ou passos) em sequência
- Cada thread executa apenas uma instrução por vez

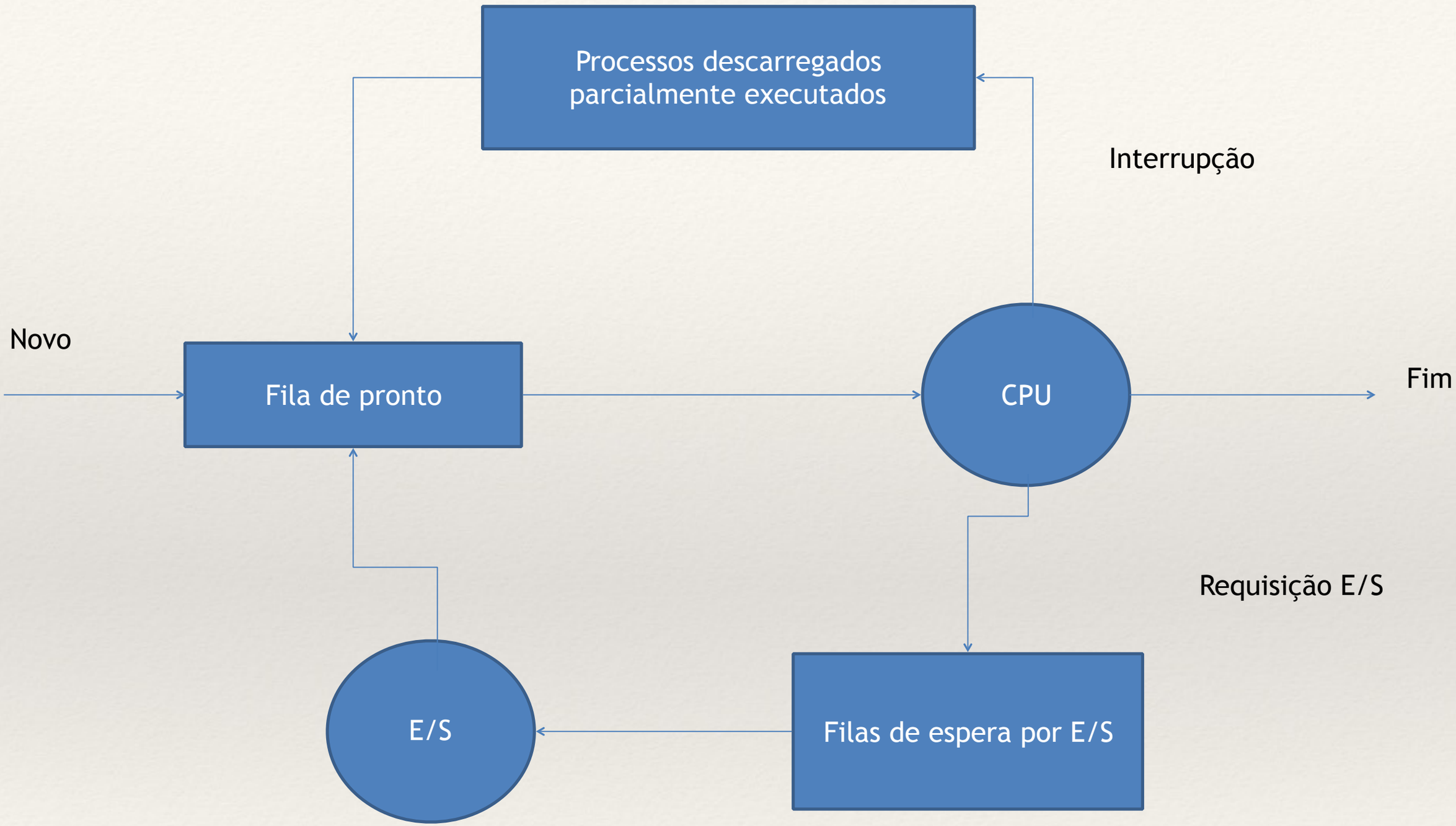
Escalonador de processos

- Um sistema pode ter vários processos carregados na memória
  - tipicamente tem em dezenas ou centenas
- Todos os processos que estão na memória e não estão executando, têm seu estado como “pronto” ou “esperando”
- No momento em que um processo é suspenso
  - sai do estado “executando” para algum outro
  - o SO decide qual o próximo processo a ser executado
- Essa decisão é feita pelo **escalonador de processos**

- Quando criado, um processo é colocado na **fila de tarefas** (*job queue*), onde ficam todos os processos do sistema
- Os processos que estão na memória principal, apenas esperando o momento para serem executados, ficam na **fila de prontos** (*ready queue*)
- Quando um processo faz uma requisição E/S, é colocado na **fila de dispositivo** (*device queue*) para o dispositivo específico ao qual fez a requisição

Filas do escalonador




Processos descarregados parcialmente executados
Interrupção
Novo
Fila de pronto
CPU
Fim
Requisição E/S
E/S
Filas de espera por E/S

Troca de contexto

- Quando um processo saí de execução para entrada de outro, é chamado de **troca de contexto**
- Acontece por interrupção de algum dispositivo ou pelo processo requisitar acesso a algum dispositivo
- Tipicamente a cada 10-100 milissegundos, o relógio do computador interrompe o processo, para que o escalonador passe a vez de execução para outro processo
- O contexto é representado por uma entrada no PCB
  - quando um processo sai de execução, o estado é salvo nele
- Quando um processo vai entrar em execução, ocorre a restauração de estado