

2.1 Processos

Processo

- É um programa em execução
 - O conceito pode ir além desta definição
- CPU alterna a execução rapidamente dando a ilusão de paralelismo
- Pseudoparalelismo: ilusão que todos os programas estão em execução ao mesmo tempo
 - Após um intervalo o primeiro processo em execução é suspenso
 - Segundo processo passa a ser executado
 - Após um intervalo, é suspenso e a execução volta para o primeiro

Processo

- Programa em execução
 - Contador de programa atual
 - Registradores do processador
 - Espaço de memória
 - Outros...
- A alternância é conhecida como multiprogramação
- O Processo contém
 - Programa
 - Entrada/saída
 - Estado

SOs podem usar processos em

- Auditoria e segurança do sistema
- A contabilização do uso de recursos
- A contabilização de erros
- Gerência de impressão
- Comunicação de eventos
- Serviços de redes
- Interface de comandos (Shell), entre outros

Criação de processos

- Início do sistema

- Quando o sistema operacional é inicializado, são criados vários processos
- Existem os de primeiro plano, que interagem com os usuários e suas aplicações
- E os de segundo plano, que possuem uma função específica, como um processo para atualizar e-mails quando alguma mensagem é recebida na caixa de entrada

Criação de processos

- Execução de uma chamada ao sistema de criação por um processo em execução
 - P. ex., quando um processo está fazendo download, ele aciona um outro processo para ajudá-lo
 - Enquanto um faz o download, o outro está armazenando os dados em disco

Criação de processos

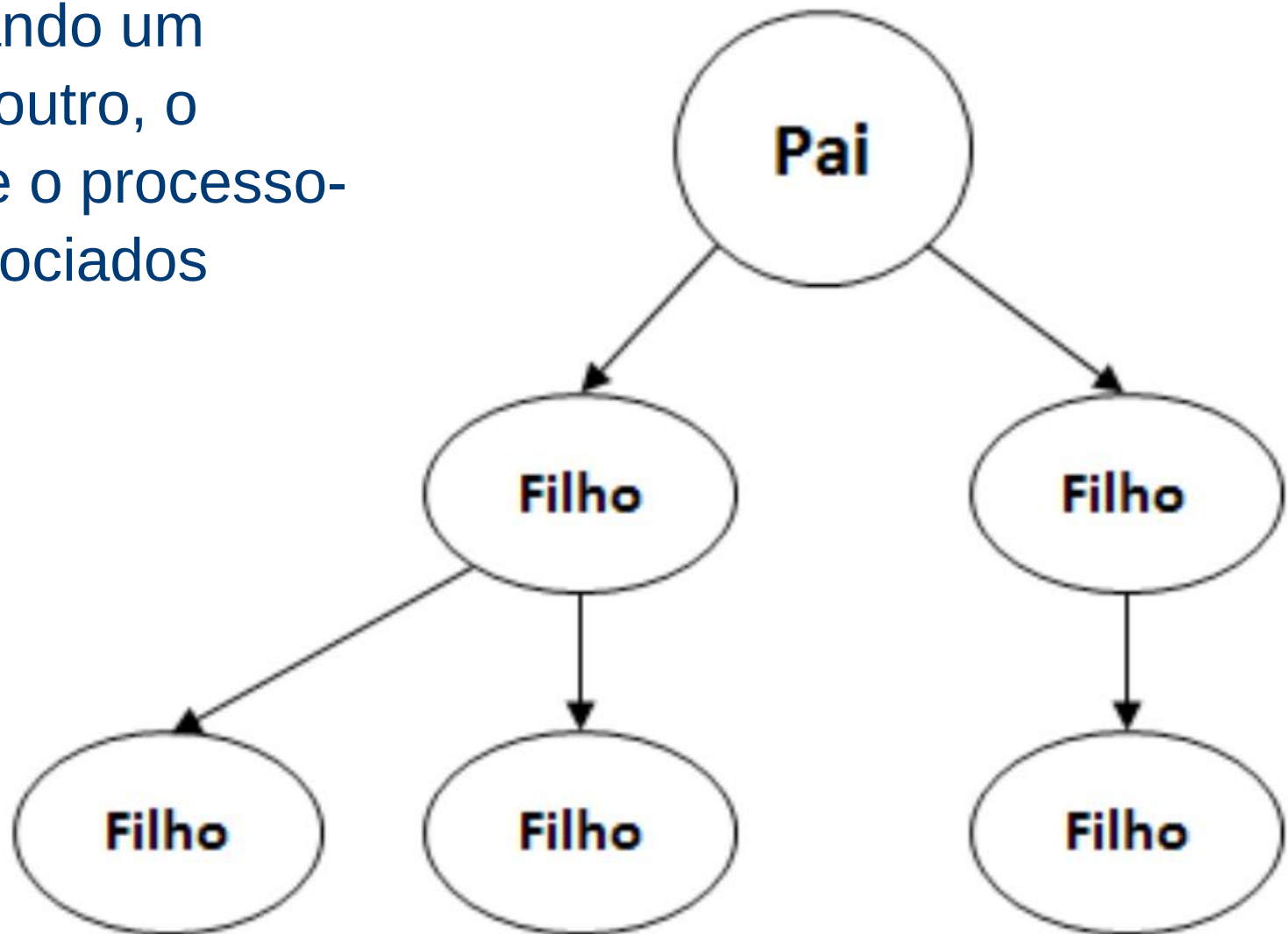
- Uma requisição do usuário para criar um novo processo
 - Quando o usuário digita um comando ou solicita a abertura de um ícone para a abertura de um aplicativo
- Início de um job em lote
 - Esses processos são criados em computadores de grande porte, os mainframes

Término de processos

- Saída normal (voluntária)
 - Acontece quando o processo acaba de executar por ter concluído seu trabalho
- Saída por erro (voluntária)
 - Acontece quando o processo tenta acessar um arquivo que não existe e é emitida uma chamada de saída do sistema
 - Em alguns casos, uma caixa de diálogo é aberta perguntando ao usuário se ele quer tentar novamente

Hierarquia de Processos

- Em alguns sistemas (ex.: Unix-like), quando um processo cria outro, o processo-pai e o processo-filho ficam associados



- P. ex., quando um usuário envia um sinal do teclado (como CTRL + ALT + DEL), este sinal é entregue para todos os processos que compõem o grupo de processos do teclado
- Quando um processo-pai é “morto”, todos os filhos vinculados a ele são “mortos” também
- O Windows não possui uma hierarquia de processos
 - Cada um possui um identificador próprio e quando um processo cria outro, existe uma ligação entre eles, mas ela é quebrada quando o processo-pai passa seu identificador para outro processo
 - Quando um processo-pai é “morto”, os processos vinculados a ele não são mortos

Estados do processo

- Em execução

- Um processo está em execução quando está sendo processado pela CPU
- Os processos são alternados para a utilização do processador

- Pronto

- Um processo está no estado de pronto quando possui todas as condições necessárias para executar e está aguardando
- O sistema operacional é quem define a ordem e os critérios para execução dos processos

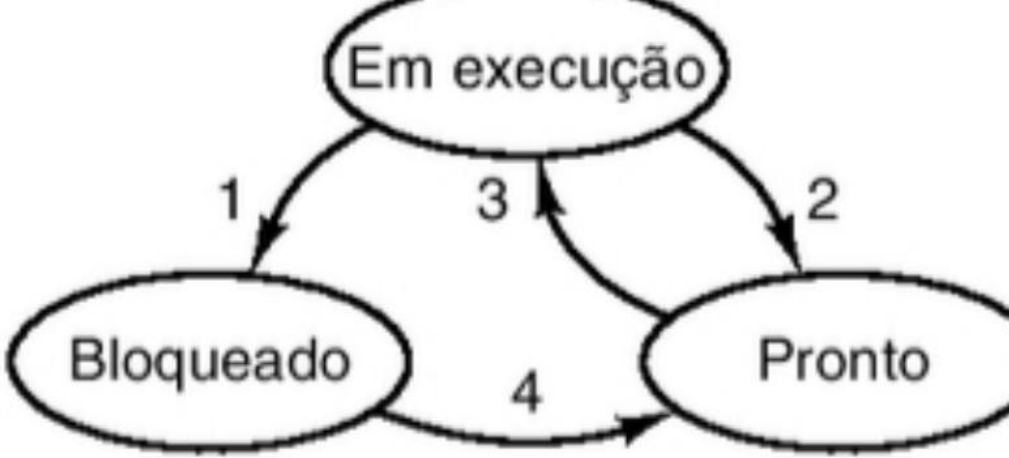
- Espera ou Bloqueado

- Um processo está no estado de espera quando aguarda por um evento externo (um comando do usuário, p. ex.) ou por um recurso (uma informação de um dispositivo de entrada/saída, p. ex.) para executar

Término de processos

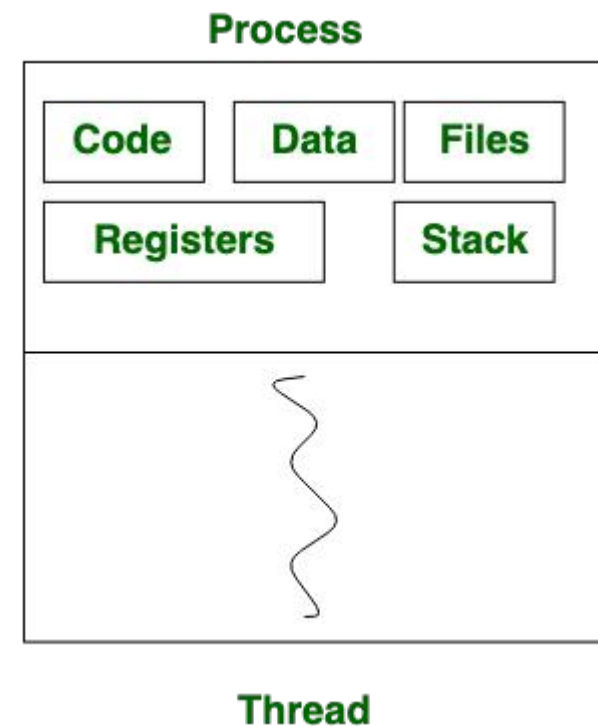
- Erro fatal (involuntário)
 - Acontece quando ocorre um erro de programa, por exemplo, a execução ilegal de uma instrução ou a divisão de um número por zero
 - Neste caso, existe um processo com prioridade máxima que supervisiona os demais processos e impede a continuação do processo em situação ilegal
- Cancelamento por um outro processo
 - Acontece quando um processo que possui permissão emite uma chamada ao sistema para cancelar outro processo

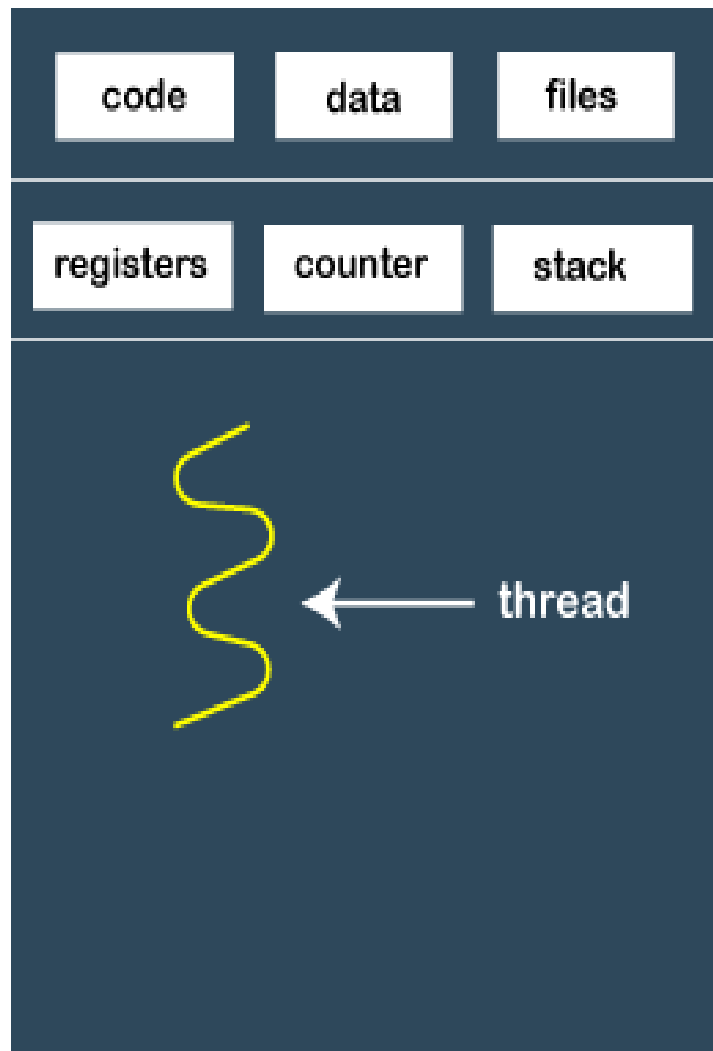
4 mudanças podem acontecer entre os estados

- 1: quando um processo aguarda um evento externo ou uma operação de entrada/saída e não consegue continuar o processamento
- 
- ```
graph TD; E([Em execução]) -- 1 --> B([Bloqueado]); E -- 2 --> P([Pronto]); P -- 3 --> E; B -- 4 --> P;
```
- 2 e 3: realizadas pelo escalonador sem que o processo saiba
    - 2: quando o escalonador decide que o processo já teve tempo suficiente em execução e escolhe outro processo para executar
    - 3: demais processos já utilizaram o seu tempo de CPU e o escalonador executa o processo que estava aguardando
  - 4: quando a operação de entrada/saída ou o evento externo que o processo estava esperando ocorre

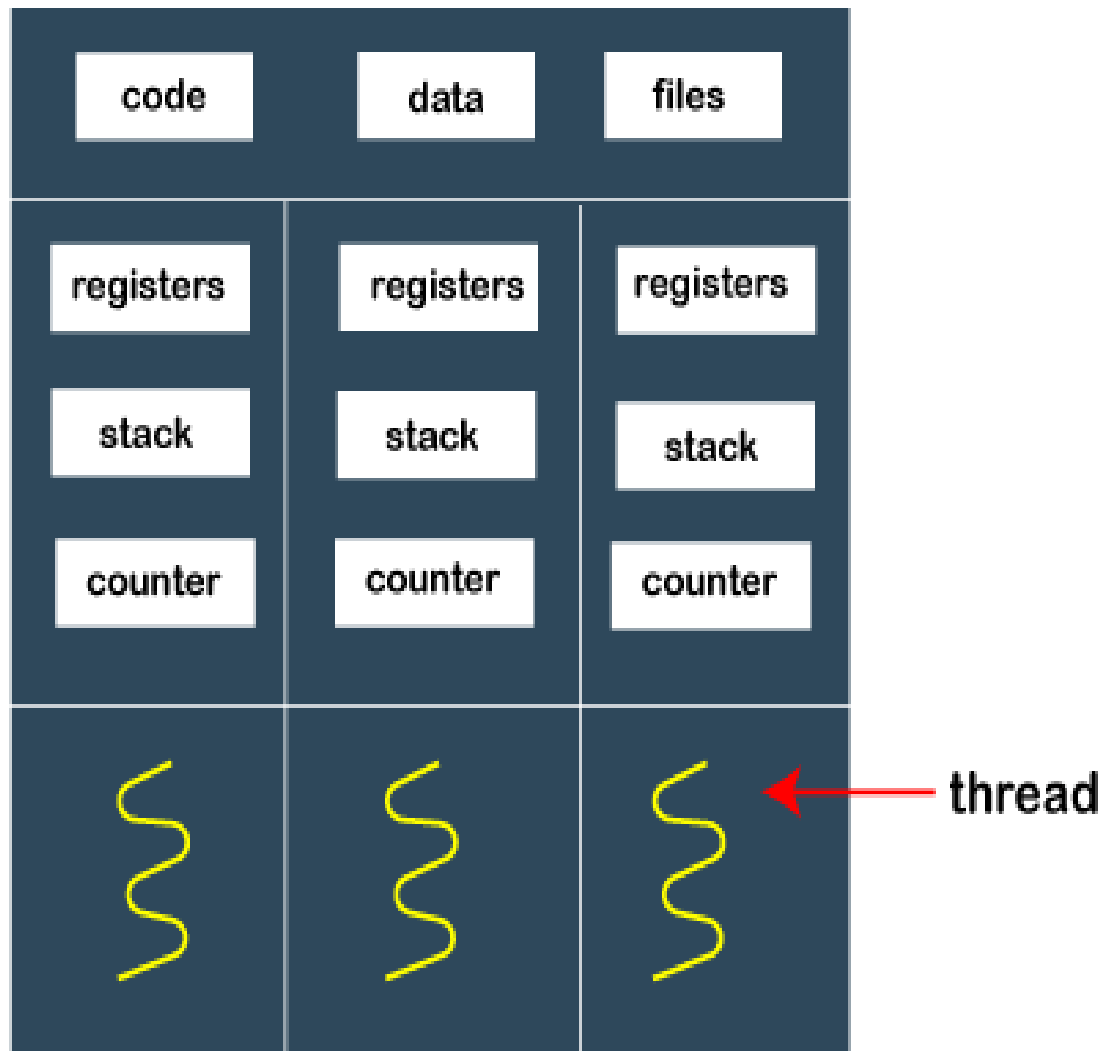
# Thread

- Pequenas unidades dentro de um processo
- Compartilham a mesma memória (processo não)
- Facilita a comunicação porém requer preocupação com sincronização
- Menor tempo criação, término, etc.
- Não precisa chamar o SO





Single-threaded process



Multi-threaded process

# Thread

- Reduz o tempo gasto na criação, eliminação e troca de contexto de processos nas aplicações concorrentes
  - Economiza recursos do sistema como um todo
- Fluxo de controle (execução) dentro do processo
- Chamado também de processo leve
- Um processo pode conter um ou vários threads que compartilham os recursos do processo
- Threads podem ser executados em paralelo
- São mais fáceis de criar e destruir (usa poucos recursos)



# Exemplos

- Um processo aguarda a leitura do teclado, enquanto outro escreve na tela
- Em um jogo cada personagem usa um thread
- Em aplicações científicas cálculos numéricos repetitivos são distribuídos entre threads executadas em várias CPUs
- Em um navegador web (ex.: Chrome) cada aba pode ser executada em um thread
- Java implementa programação concorrente em threads

# Implementação de Processos

- O SO mantém um quadro de processos contendo informações sobre o estado do processo, seu contador de programa, o ponteiro da pilha, a alocação de memória, o status dos arquivos abertos, entre outros, que permitem que o processo reinicie do ponto em que parou

| Gerenciamento de processos    | Gerenciamento de memória           | Gerenciamento de arquivos     |
|-------------------------------|------------------------------------|-------------------------------|
| Registradores                 | Ponteiro para o segmento de código | Diretório-raiz                |
| Contador do programa          | Ponteiro para o segmento de dados  | Diretório de trabalho         |
| Palavra de estado do programa | Ponteiro para o segmento de pilha  | Descritores de arquivos       |
| Ponteiro de pilha             |                                    | Identificador (ID) do usuário |
| Estado do processo            |                                    | Identificador (ID) do grupo   |
| Prioridade                    |                                    |                               |

(etc.)

# Implementação de Threads

- Pode ocorrer no espaço do usuário, no núcleo do SO, e em uma implementação híbrida
- Thread de usuário
  - São implementados pela aplicação do usuário e o SO “não sabe de sua existência”
  - A vantagem é que não é necessária nenhuma mudança entre os modos de usuário e núcleo, tornando-se rápido e eficiente
- Thread do núcleo
  - São implementados e gerenciados pelo núcleo do SO
  - A desvantagem é que todo o gerenciamento dos threads é feito por chamadas ao sistema, o que compromete a performance

- Threads híbridos

- São implementados tanto no espaço do usuário, quanto no núcleo do sistema operacional
- O sistema operacional sabe dos threads do usuário e faz o seu gerenciamento
- A vantagem desta implementação é a flexibilidade em função das duas implementações

# Video

- Thread (entenda como sua aplicação funciona) // Dicionário do Programador [ Código Fonte T ]
  - <https://youtu.be/xNBMNKjpJzM>