Sistemas Operacionais

Processos e threads

Guilherme Meira

Agenda

1. Processos

2. Threads

Processos

- **Processos** são o conceito mais central de qualquer sistema operacional
- Um processo representa um programa em execução
- Processos permitem que um computador realize varias tarefas ao mesmo tempo, mesmo quando apenas uma CPU está disponível

Processos

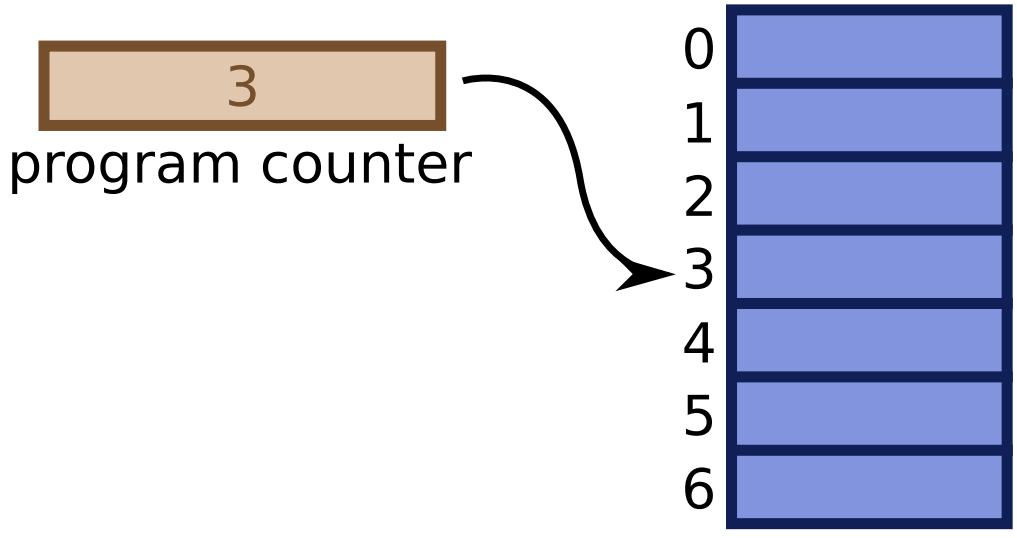
- Computadores modernos estão sempre realizando diversas tarefas ao mesmo tempo:
  - Navegando na web
  - Tocando música
  - Checando novos e-mails
  - Buscando por vírus
  - Copiando arquivos
  - Dentre várias outras
- O sistema operacional faz com que a CPU troque muito rapidamente entre processos, dando a impressão de que estão executando em paralelo (**pseudoparalelismo**)
- Sistemas com vários processadores serão estudados mais adiante

Processos

- Computadores modernos estão sempre realizando diversas tarefas ao mesmo tempo:
  - Navegando na web
  - Tocando música
  - Checando novos e-mails
  - Buscando por vírus
  - Copiando arquivos
  - Dentre várias outras
- O sistema operacional faz com que a CPU troque muito rapidamente entre processos, dando a impressão de que estão executando em paralelo (**pseudoparalelismo**)
- Sistemas com vários processadores serão estudados mais adiante

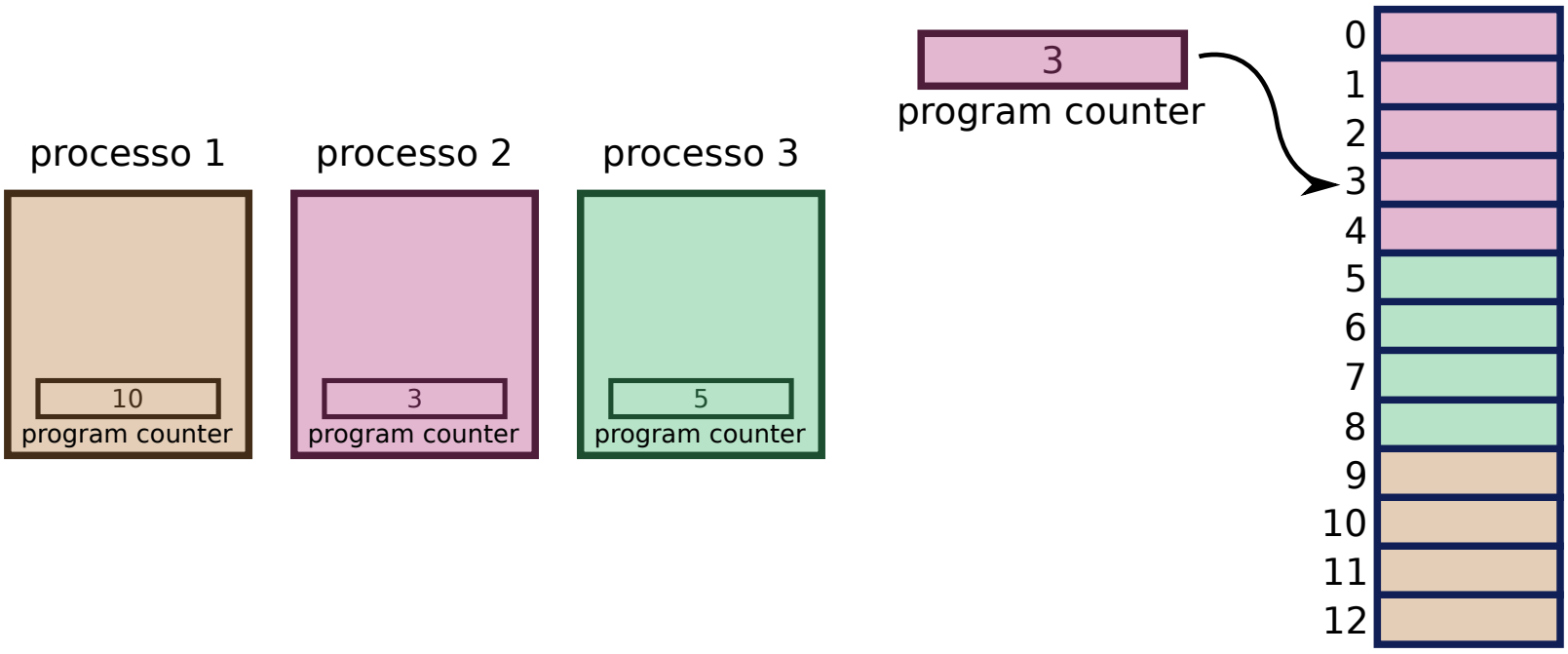
Processos

- O **program counter (PC)** é um registrador da CPU que aponta para a posição de memória que contém a próxima instrução a ser executada



Processos

- Uma CPU tem apenas um program counter
- O sistema operacional precisa armazenar o program counter de cada processo na memória
- Quando um processo vai receber um tempo da CPU, seu program counter é carregado no program counter real da CPU



Processos

- O sistema operacional determina quando um processo vai executar e por quanto tempo ele vai ter a posse da CPU
- Para a maioria dos processos isso não é relevante
- Isso pode afetar processos com requerimentos de tempo real (ex: tocar áudio e video em sincronia)

Processos

- Qual a diferença entre um **processo** e um **programa**?

Processos

- Qual a diferença entre um **processo** e um **programa**?
- Uma analogia:
  - Uma pessoa está fazendo um bolo seguindo uma receita
  - A pessoa é o **processador**, a receita é o **programa**, a ação de fazer o bolo é o **processo**
  - A pessoao é interrompida pelo filho que foi picado por uma abelha
  - Ele anota onde parou na receita e pega um livro de primeiros socorros para ajudar o filho
  - Aqui, trocamos de um processo para o outro
  - Quando ele termina de cuidar do filho, continua o processo anterior de onde parou

Processos

- Um **programa** é uma sequência de passos que pode ficar guardada em disco sem fazer nada
- Um **processo** é a atividade de executar as instruções de um programa
- O mesmo programa pode ser executado mais de uma vez ao mesmo tempo por processos diferentes

Processos

Criação de processos

- Durante a execução de um sistema operacional, processos podem ser criados por diversos motivos:
  - Inicialização do sistema
  - Execução de uma chamada de sistema por um processo que já esteja rodando
  - O usuário requisita a criação de um processo
  - Inicio de um lote

Processos

Criação de processos

- Quando o sistema operacional é iniciado, vários processos são criados
- Alguns são processos de primeiro plano, que interagem com o usuário
- Outros rodam em segundo plano, e não estão associados a um usuário em específico, mas a uma tarefa
- Processos que ficam rodando em plano de fundo são chamados de **daemons**

Processos

Criação de processos

- Em UNIX, processos são criados pela chamada de sistema `fork`
- Essa chamada cria um clone do processo que a chamou
- O novo processo, então, pode executar a chamada `execve` para carregar um novo programa
- No Windows, processos são criados pela chamada `CreateProcess`
- Tanto em Windows como em UNIX, cada processo tem seu próprio espaço de endereçamento (não é possível acessar a memória de outro processo)

Processos

Término de processos

- Após serem criados, processos eventualmente são terminados, por diversos motivos:
  - Saída normal (voluntário)
  - Saída com erro (voluntário)
  - Erro fatal (involuntário)
  - Morto por outro processo (involuntário)

Processos

Hierarquia de processos

- Em UNIX, processos possuem uma hierarquia
- Um processo e seus filhos formam um **grupo**. Quando o usuário envia um sinal a um processo, todos os processos do grupo recebem o sinal (veremos sinais com mais detalhes adiante)
- Quando o UNIX inicializa, um processo especial chamado `init` cria todos os demais processos. O `init` é o pai de todos os processos no UNIX
- O Windows não possui o conceito de hierarquia de processos

Processos

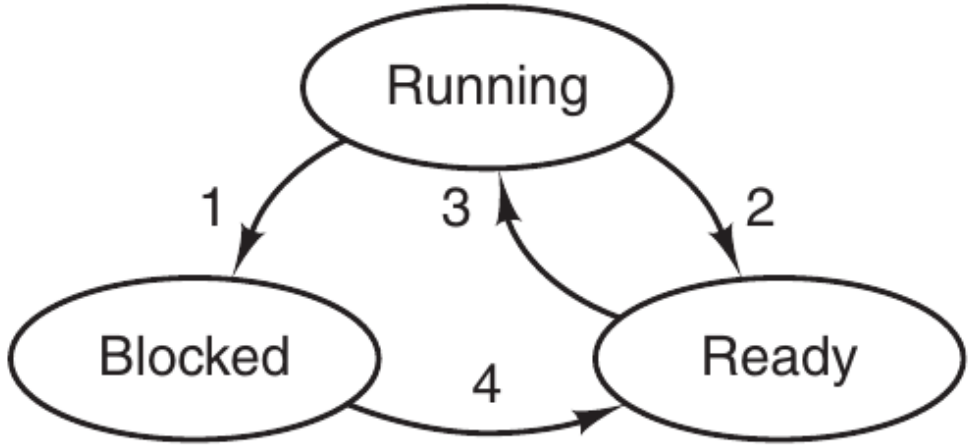
Estados de um processo

- Um processo pode estar em um de três estados:

Executando o processo está utilizando a CPU no momento

Pronto o processo está pronto para rodar, mas o sistema operacional não o deu a posse da CPU

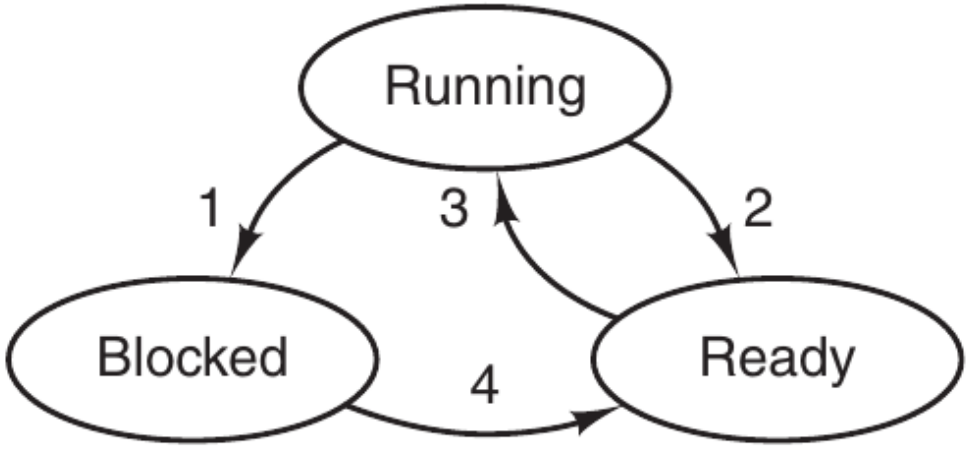
Bloqueado o processo está esperando algum evento externo para poder rodar



Processos

Estados de um processo

- As transições no diagrama representam:
  1. Processo bloqueia para esperar por algum evento
  2. Escalonador do sistema escolhe outro processo para colocar na CPU
  3. Escalonador escolhe este processo
  4. O evento que estava sendo aguardado acontece



Processos

Estados de um processo

- O estado **bloqueado** é fundamentalmente diferente dos demais
- Enquanto nos estados **executando** e **pronto** o processo deseja ser executado, no estado **bloqueado**, mesmo com a CPU disponível o processo não pode rodar
- Um processo pode entrar nesse estado, por exemplo, para esperar uma entrada vinda do teclado ou de outro processo
- As transições entre os estados **executando** e **pronto** são geradas por uma parte do sistema operacional chamada de escalonador (veremos mais detalhes adiante)

Processos

Implementação de processos

- O sistema operacional mantém uma tabela de processos contendo todas as informações sobre o estado do processo
- Essas informações são salvas quando o processo vai de **executando** para **pronto** ou **bloqueado** e são restauradas quando o processo for voltar a rodar
- As informações armazenadas na tabela de processos variam de sistema para sistema, mas em geral possuem informações sobre o próprio processo, sobre gerência de memória e sobre gerência de arquivos

Processos

Modelando multiprogramação

- **Multiprogramação** é o uso da CPU por vários processos
- Ela permite que aumentemos a utilização da CPU
  - Se um processo faz computação 20% do tempo, 5 processos executando em multiprogramação manteriam a CPU ocupada 100% do tempo
  - Modelo pouco realista! Assume que os processos nunca estarão aguardando por I/O ao mesmo tempo

Processos

Modelando multiprogramação

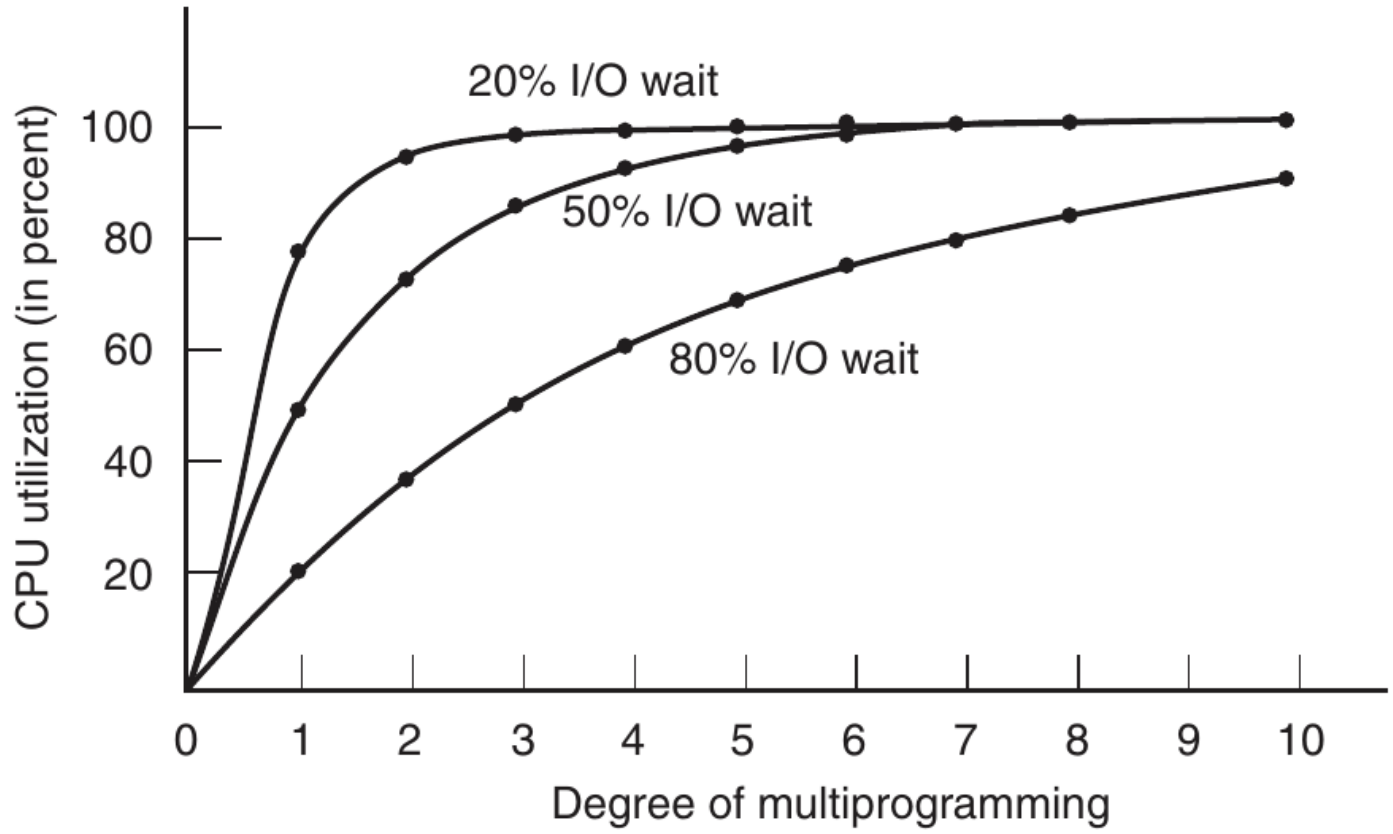
- Podemos utilizar probabilidade para encontrarmos um modelo melhor:
  - Um processo passa uma fração p do tempo esperando por I/O
  - Com n processos na memória, a probabilidade de que todos os processos estejam esperando por I/O (isto é, a CPU está ociosa) é p^n
  - A utilização da CPU é, então, calculada por:

$$utilizacao_{CPU} = 1 - p^n$$

Processos

Modelando multiprogramação

- Assim, podemos plotar a utilização estimada da CPU em função do número de processos em execução:



Processos

Modelando multiprogramação

- Este modelo também é uma aproximação, pois assume que os processos são independentes entre sí
- Quando dois processos estão no estado **pronto**, apenas um pode utilizar a CPU e o outro terá que esperar, portanto, existe uma dependência entre os processos
- Poderíamos chegar a modelos mais complexos utilizando Teoria das Filas, mas esse modelo é suficiente para fazermos algumas predições

Processos

Modelando multiprogramação

Suponha que um computador tenha 8GB de memória. O sistema operacional ocupa 2GB e cada processo executando na máquina também ocupa 2GB.

- Quantos processos podem executar simultaneamente nesta máquina?

Processos

Modelando multiprogramação

Suponha que um computador tenha 8GB de memória. O sistema operacional ocupa 2GB e cada processo executando na máquina também ocupa 2GB.

- Quantos processos podem executar simultaneamente nesta máquina?
  - Três
- Se o tempo de espera por I/O é de 80%, qual é a utilização da CPU?

Processos

Modelando multiprogramação

Suponha que um computador tenha 8GB de memória. O sistema operacional ocupa 2GB e cada processo executando na máquina também ocupa 2GB.

- Quantos processos podem executar simultaneamente nesta máquina?
  - Três
- Se o tempo de espera por I/O é de 80%, qual é a utilização da CPU?
  - $1 - 0.8^3 \approx 49\%$
- Se adicionarmos mais 8GB de memória, qual será a nova utilização da CPU?
  - Poderemos ter 7 processos em paralelo: $1 - 0.8^7 \approx 79\%$

Processos

Modelando multiprogramação

Ou seja, adicionar 8GB de memória aumentou a ocupação da CPU em 30%. E se adicionássemos mais 8GB?

Processos

Modelando multiprogramação

Ou seja, adicionar 8GB de memória aumentou a ocupação da CPU em 30%. E se adicionássemos mais 8GB?

- Podemos ter até 11 processos: $1 - 0.8^{11} \approx 91\%$
- Ganhamos apenas mais 12% de ocupação da CPU

Agenda

1. Processos

2. Threads

Threads

Modelando multiprogramação

Ou seja, adicionar 8GB de memória aumentou a ocupação da CPU em 30%. E se adicionássemos mais 8GB?

Threads

Modelando multiprogramação

Ou seja, adicionar 8GB de memória aumentou a ocupação da CPU em 30%. E se adicionássemos mais 8GB?

- Podemos ter até 11 processos: $1 - 0.8^{11} \approx 91\%$
- Ganhamos apenas mais 12% de ocupação da CPU