Curso de Engenharia de Computação
Sistemas Operacionais

Processos e Threads – Parte I

Slides da disciplina Sistemas Operacionais
Curso de Engenharia de Computação
Instituto Mauá de Tecnologia – Escola de Engenharia Mauá
Prof. Marco Antonio Furlan de Souza

Onde os processos são executados?

INSTITUTO MAUÁ DE TECNOLOGIA

- **Detalhes sobre a CPU**
  - Quem **executa** os **programas** é a **CPU** (Central Processing Unit), que se localiza dentro de um microprocessador;
  - **Ciclo** de **execução** de **instruções**:
    - **Buscar** na memória principal uma instrução (*fetch*);
    - **Decodificar** a instrução – tipo e operandos (*decode*);
    - **Executar** a instrução (execute);
    - **Repetir** os passos anteriores **até** que o **programa termine**.

Onde os processos são executados?

INSTITUTO MAUÁ DE TECNOLOGIA

- **Detalhes sobre a CPU**
  - **Cada CPU** possui seu **próprio conjunto** de **instruções**: o de um **x86** é **diferente** de um **ARM** (e vice-versa);
  - A **velocidade interna** de uma **CPU** é **muito maior** que da **memória principal –** um conjunto de **registradores internos** da **CPU armazenam valores** provenientes da **memória principal** e também **valores temporários**;
  - Assim, entre outras, o **conjunto** de **instruções** de uma CPU **contempla instruções** para **armazenar** uma **palavra** da **memória** em um **registrador** (*load*) e **armazenar** uma **palavra** de um **registrador** em uma **posição** da **memória**;
  - **Outras instruções** : controle de fluxo, aritméticas, lógicas etc.

Onde os processos são executados?

INSTITUTO MAUÁ DE TECNOLOGIA

- **Detalhes sobre a CPU**
  - **Registradores especiais de uma CPU**
    - **Contador de programa** (*program counter* – **PC**): armazena o **endereço** de **memória** da **próxima instrução** a ser **buscada**. Após ela ter sido buscada, o PC é atualizado com o endereço da instrução sucessora;
    - **Ponteiro de pilha** (*stack pointer* – **SP**): aponta para o **topo** da **pilha** atual **presente** na **memória**. Contém o *frame* (conjunto de **dados**) de um **procedimento** que se encontra **em execução**: **parâmetros** de **entrada**, **variáveis locais** e **variáveis temporárias** que não estão em registradores;
    - **Registrador de estado do programa** (*program status word* – **PSW**): **armazena** os **bits** de **condição** de um **programa** (ex.: resultado de comparação), **prioridade** da **CPU**, **modo** (usuário ou *kernel*), e **outros bits** de **controle**. Normalmente é de **apenas leitura** para **programas** do **usuário** e é **importante** quando se **executa chamadas do sistema** e **E/S**.

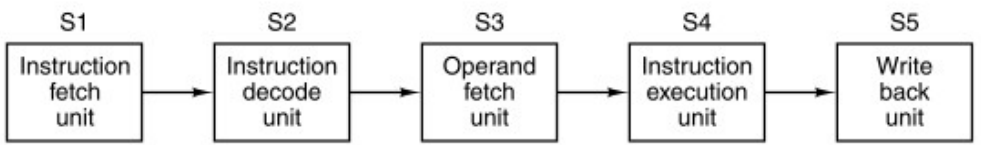
Onde os processos são executados?

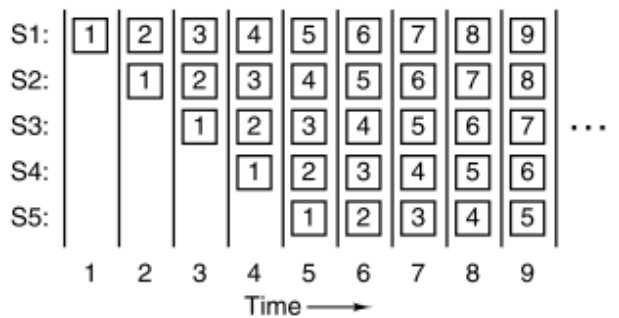
INSTITUTO MAUÁ DE TECNOLOGIA

- **Detalhes sobre a CPU**
  - **Registradores especiais de uma CPU**
    - Um **sistema operacional** precisa **conhecer todos** os **registradores** – quando a **CPU multiplexa** o **tempo**, o **sistema operacional interrompe** um **programa** em **execução** para **(re)inicializar outro** – **precisa salvar** todos os **registradores** que deverão ser **restaurados** mais **tarde.**

Onde os processos são executados?

- ***Pipeline***
  - Para **melhorar** o **desempenho**, **projetistas** de **CPU incorporaram facilidades** para **executar mais** de uma **instrução simultaneamente**;
  - Uma **CPU** tipicamente **contém unidades separadas** de **busca**, **decodificação** e **execução** de forma que, **enquanto** ele está **executando** a **instrução** *n* ela **também** pode **estar decodificando** a **instrução** *n* + *1* e **também buscando** a instrução *n* + *2*. Esta organização é **denominada** de ***pipeline***:

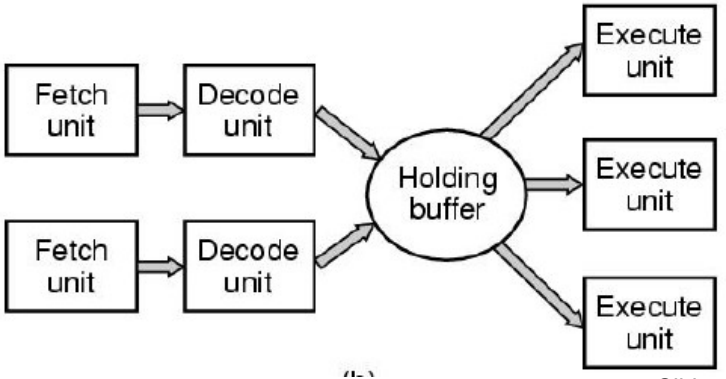




Onde os processos são executados?

INSTITUTO MAUÁ DE TECNOLOGIA

- **CPU superescalar**
  - É uma **evolução** de **pipeline**: existem **várias unidades** de **execução** (ex.: aritmética inteira, aritmética em ponto flutuante, operações booleanas etc.);
  - **Duas ou mais instruções** são **buscadas** de uma **só vez**, **decodificadas** e **depositadas** em um **buffer** (*holding buffer*) **até que** sejam **executadas**;
  - Assim que uma **unidade** de **execução** se **disponibiliza**, ela **procura** no *buffer* se existe uma **instrução** que ela **possa manipular** e, caso positivo, a **remove** do *buffer* e a **executa** (normalmente **fora** de **ordem**).



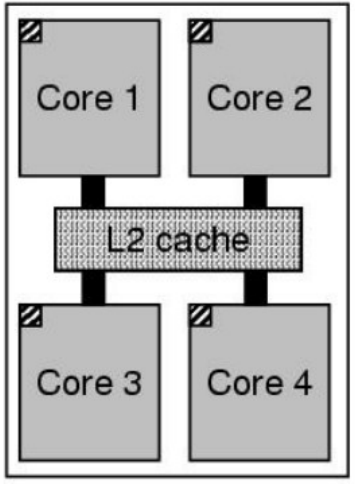
Onde os processos são executados?

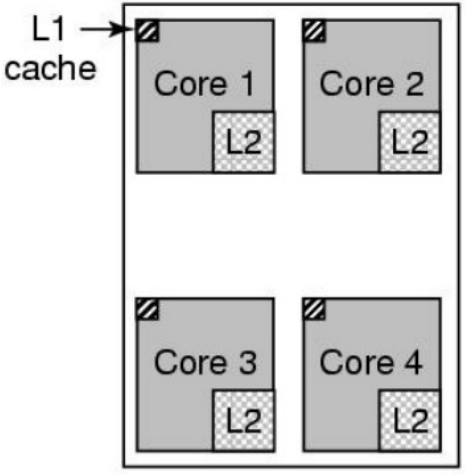
INSTITUTO MAUÁ DE TECNOLOGIA

- **CPU multithreads**
  - Além das técnicas anteriores **CPUs modernas** não apenas **replicam** as unidades funcionais (pipeline, superescalar) como também replicam **partes** da **lógica** de **controle**;
  - Uma forma disso é **denominada** de *multithreading* ou *hyperthreading* (nome da Intel) – **implementa** na CPU (em hardware) a **execução multiplexada** no **tempo** de *threads* (processo mais "leve" – não é paralelismo de verdade, mas o **ciclo** de **chaveamento** entre *threads* aqui é da **escala** de **nanosegundo**;
  - Essas *threads* são **entendidas** como **CPUs distintas** pelo **sistema operacional** – assim, um **computador** com duas **CPUs físicas** que suportam **cada** uma **até duas** *threads* **parecem** como **quatro CPUs** ao **sistema operacional** (que pode alocar muito trabalho a uma CPU deixando a outra "ociosa"…)

Onde os processos são executados?

- **CPU com múltiplos núcleos**
  - **Chips atuais** possuem **quatro**, **oito** ou **mais processadores completos** internamente – **cores** – (aluns podem chegar a 60!);
  - Neste caso, tem-se de fato **CPUs completas** que o sistema operacional pode explorar.





Onde os processos são executados?

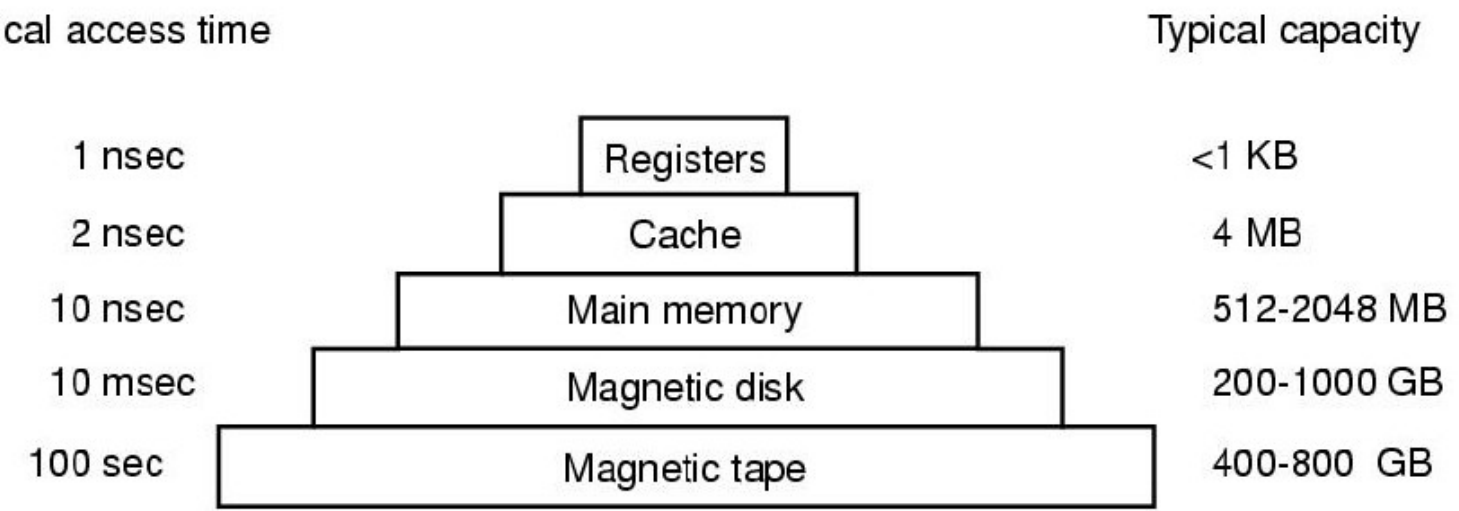
INSTITUTO MAUÁ DE TECNOLOGIA

- **GPUs**
  - Placas gráficas modernas são providas de **GPUs** (*Graphics Processing Unit*);
  - Uma **GPU** é um **processador** com centenas de **minúsculos cores** – excelentes para vários tipos de **pequenas computações realizadas** em **paralelo** (renderização de polígonos, por exemplo);
  - **Não** são **triviais** para **programar**, mas podem ser **úteis** para o sistema operacional em **tarefas especializadas** tais como criptografia, processamento de tráfego de rede.

Onde os processos são executados?

INSTITUTO MAUÁ DE TECNOLOGIA

- **Memória**
  - O **sistema** de **memória** é construído como uma **hierarquia** de **camadas**: camadas **superiores** possuem **maior velocidade**, **menor capacidade** e **maior custo por bit** do que as inferiores.



Onde os processos são executados?

INSTITUTO MAUÁ DE TECNOLOGIA

- **Memória**
  - A camada **mais alta** consiste de **registradores internos** da **CPU** – **mesma velocidade** de **acesso** da CPU. **Tamanho típico** para CPU de 64bits: 64 registradores × 64 bits de armazenamento;
  - **Memória cache**: **controlada** principalmente por **hardware**, é **mapeada** para a **memória principal** – a **memória principal** é dividida em **linhas** de **cache** de **64 bytes**, de modo que a **linha 0 mapeia** para os **endereços 0** a **63**, a **linha 1** para os **endereços 64** a **127** e, assim por diante;
  - Quando um **programa** precisa **ler** uma **palavra** da **memória**, o **hardware** do **cache verifica** se a **linha** já **está** no **cache**. **Se estiver** há um ***cache hit*** e **dado** é **lido** do cache (~2 ciclos de relógio); **senão**, ocorre um ***cache miss*** e é **necessário enviar** uma **requisição** via **bus** à **memória**, com uma **penalidade** substancial de **tempo**.

Onde os processos são executados?

INSTITUTO MAUÁ DE TECNOLOGIA

- **Memória**
  - **Níveis de cache**
    - **L1**: **cache sempre dentro** da **CPU** e usado para **alimentar instruções decodificadas** para a unidade de **execução** da CPU (~16kB);
    - **L2**: **cache** que mantém vários **megabytes** de **palavras recentemente lidas** da **memória**;
    - A **diferença** entre **caches L1** e **L2** é **tempo de acesso**: L1 praticamente não tem atraso, enquanto que L2 demanda um a dois ciclos de relógio.

Onde os processos são executados?

INSTITUTO MAUÁ DE TECNOLOGIA

- **Memória principal**
  - A **memória principal** é normalmente **chamada** de **RAM** (*Random Access Memory*);
  - As **capacidades** deste tipo de memória **variam** de centenas de **megabytes** a vários **gigabytes**;
  - Quando uma **requisição** de acesso a **dado não** pode ser **satisfeita** pelo **cache**, ela é **direcionada** à **memória principal**.

Onde os processos são executados?

INSTITUTO MAUÁ DE TECNOLOGIA

- **Outras memórias**
  - **ROM** (*Read Only Memory*): é **programada** na **fábrica** e seu **conteúdo não** pode ser **modificado**. É **rápida** e **barata** e em alguns computadores ela **armazena** o **código** para dar *boot*. Alguns cartões de **E/S** podem ter uma **ROM** para **gerenciar dispositivos** de **baixo nível**;
  - **EEPROM** (*Electrically Erasable PROM*): são **não voláteis**, mas podem ser **apagadas** e **reescritas**. A **escrita** é **lenta**, então **são utilizadas** como as **ROMs**. **Vantagem**: permitem **corrigir** *bugs*.

Onde os processos são executados?

INSTITUTO MAUÁ DE TECNOLOGIA

- **Outras memórias**
  - **Memória flash**: utilizada como **meio** de **armazenagem** para **dispositivos portáteis**. Meio termo **entre RAM** e **disco**. Se **apagada frequentemente**, pode **desgastar**;.
  - **CMOS**: é uma **memória volátil** que pode ser **utilizada** para **armazenar** a **hora** e a **data** atual do **computador**. É energizada por uma **pequena bateria** que pode **durar** vários **anos**. Pode ser utilizada **também** para **armazenar parâmetros** de **configuração**, a serem utilizados no *boot*.

Processos

INSTITUTO MAUÁ DE TECNOLOGIA

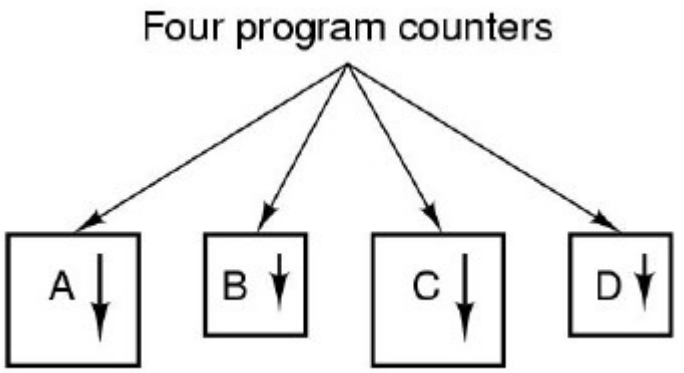
- **Modelo de processo**
  - Todo o software executável no computador (inclui muitas vezes o sistema operacional), é organizado em vários **processos sequenciais**, **ou** apenas **processos**;
  - Um **processo** é apenas uma **instância** de um **programa** em **execução** e **inclui** os **valores atuais** do **contador de programa** (**PC**), **registradores** e **variáveis**. **Conceitualmente**, cada **processo** tem sua **própria CPU virtual**;
  - Na **realidade** a **CPU** real **alterna de processo para processo**, mas é muito **mais fácil pensar** em **uma coleção** de **processos** em **execução** (**pseudo**) **paralelas**;
  - Este chaveamento de processos é denominado de **multiprogramação**.

INSTITUTO MAUÁ DE TECNOLOGIA

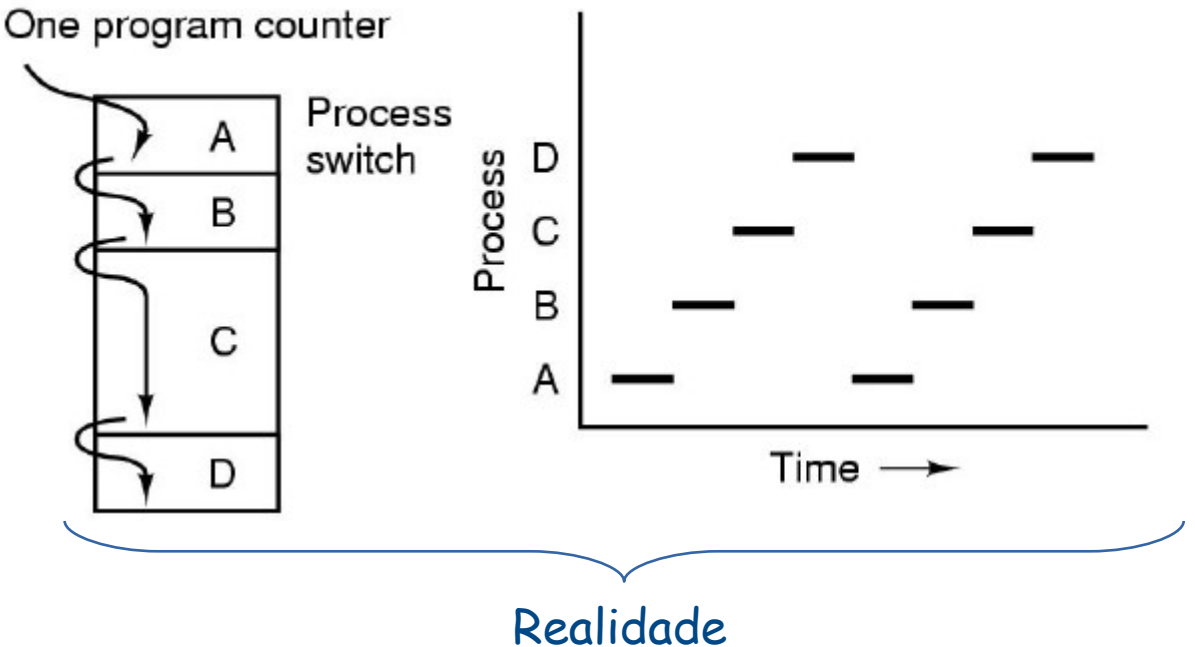
- **Modelo de processo**
  - O **chaveamento** de **processos** ocorre **executando** cada **processo** por **dezenas** ou **centenas** de **milissegundos**. Assim, a qualquer **momento** a **CPU** está **executando apenas um processo**. Após **um** segundo, diversos processos executaram, dando a **ilusão** de **paralelismo**.
  - O **paralelismo real** ocorre em **hardwares** de **sistemas multiprocessadores** (que têm duas ou mais CPUs compartilhando a mesma memória física);
  - Mas os projetistas de **sistemas operacionais** ao longo dos anos desenvolveram um **modelo conceitual** (**processos sequenciais**) que **facilita** o **tratamento** do **paralelismo**.

Processos

- **Modelo de processo**
  - **Conceito de multiprogramação**



Impressão de paralelismo



Realidade

Processos

INSTITUTO MAUÁ DE TECNOLOGIA

- **Modelo de processo**
  - Por **causa** da **alternância** da **CPU** entre os **processos**, a **taxa** na qual um **processo executa** sua **computação não** será **uniforme** e provavelmente **nem** será **reproduzível** se os mesmos processos forem **executados novamente –** não se deve fazer suposições sobre o tempo;
  - **Exemplo**. Um **processo** de **áudio** que **reproduz música** para **acompanhar** um **vídeo** de **alta qualidade** executado por outro dispositivo. Como o **áudio** deve **começar** um **pouco depois** do **vídeo**, ele **sinaliza** ao **servidor** de **vídeo** para **começar** a **reproduzir** e, em **seguida**, executa um **loop inativo 10.000 vezes antes** de **reproduzir** o áudio. **Se** o **loop** for um **temporizador confiável**, **ok**, **mas** se a **CPU** decidir **mudar para** outro **processo durante** o **loop ocioso**, o **processo** de **áudio** pode **não** ser **executado novamente até** que os **quadros** de **vídeo correspondentes** já tenham **passado**, e o **vídeo** e **áudio** ficará irritantemente **fora** de **sincronia**.
  - Esse tipo de problema **não acontece** em **sistemas operacionais de tempo real** – que asseguram que as operações deverão ser realizadas dentro de limites estritos de tempo.

Ciclo de vida de um processo

INSTITUTO MAUÁ DE TECNOLOGIA

- **Criação**
  - Um processo pode ser criado nas seguintes situações:
    - **Inicialização** do sistema (*boot*);
    - Executar uma **chamada de sistema** que **cria processo** por meio de um **outro processo** que já está **em execução**;
    - **Requisição** pelo **usuário** para **criar** um novo **processo** – executa o processo na **linha** de **comando ou** pela **interface gráfica**;
    - Execução de **programas** em **lote** (*mainframe*).

Ciclo de vida de um processo

INSTITUTO MAUÁ DE TECNOLOGIA

- **Criação**
  - Quando um **sistema operacional** é **inicializado**, normalmente **vários processos** são **criados**. **Alguns** destes são processos em **primeiro plano** (*foreground*) e **outros** são **executados** em **segundo plano** (*background*) e **não** estão **associados** a **usuários específicos**;
  - Os **processos** que executam em **segundo plano** são **chamados** de *daemons*. **Exemplo**: o processo que envia um trabalho para impressora.
  - **Após** a **inicialização**, **novos processos** também podem ser **criados**. Um **processo** em **execução** pode executar **chamadas do sistema** para **criar** um ou **mais processos** novos para ajudá-lo a realizar seu trabalho. **Exemplo**: servidor de bancos de dados.
  - **Também** os **usuários** podem **iniciar** um **programa digitando** um **comando** ou **clicando** (duplo) no seu **ícone**. A execução de uma dessas ações inicia um novo processo e executa o programa selecionado nele.

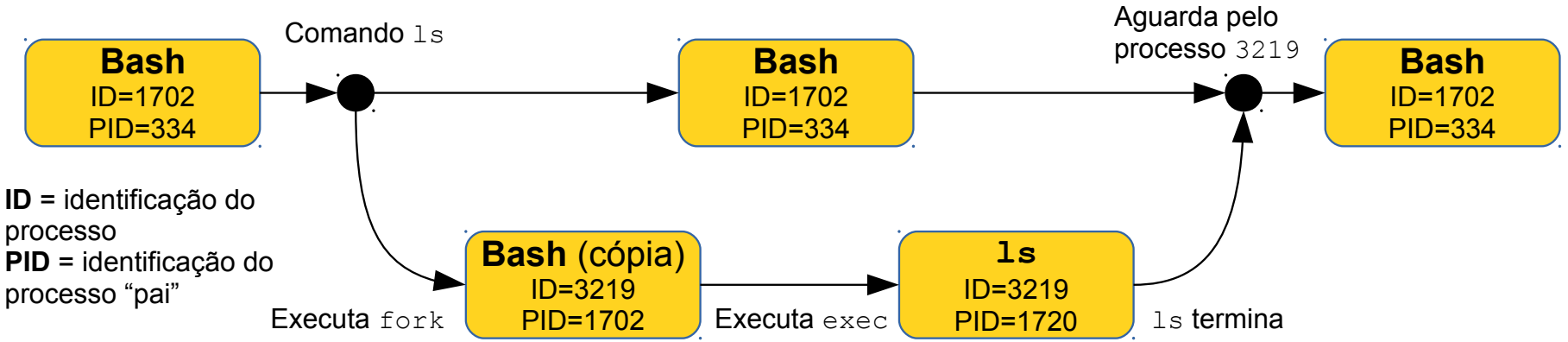
Ciclo de vida de um processo

INSTITUTO MAUÁ DE TECNOLOGIA

- **Criação**
  - No Linux (UNIX), há **apenas uma chamada** de **sistema** para **criar** um **novo** processo: `fork`;
  - Ela **cria** um **clone exato** do **processo** que está em execução. Quando termina, os **dois processos**, **pai** e **filho**, têm a **mesma imagem de memória**, as **mesmas variáveis** de **ambiente** e os **mesmos arquivos abertos**.
  - **Normalmente**, o **processo filho executa** a **chamada de sistema** `execve` ou similar para **alterar** sua **imagem** de **memória** e **executar** um **novo programa**;
  - **Exemplo**: quando um **usuário digita** um **comando**, digamos, `sort`, no **shell**, o shell se **bifurca** em um **processo filho** e o **filho executa** o processo do `sort`;
  - Isso **permite** que o **processo filho manipule** seus **descritores** de **arquivo após** a `fork`, mas antes do `execve`, a fim de realizar o redirecionamento da entrada padrão, da saída padrão e do erro padrão.

Ciclo de vida de um processo

- **Criação**
  - **Outro exemplo no Linux**: quando se digita o comando `ls` na **linha** de **comando**, o **interpretador** de comandos (bash) executa `fork` para criar um **novo processo** e `exec` para **neste processo** executar o **comando** `ls`:



Ciclo de vida de um processo

INSTITUTO MAUÁ DE TECNOLOGIA

- **Criação**
  - **Depois** que um **processo** é **criado**, tanto o **processo pai** quanto o **processo filho** têm seus próprios **espaços de endereço distintos**;
  - Se **um** dos **processos alterar** uma **palavra** em seu **espaço** de **endereço**, a **alteração não** será **visível** para o **outro processo**;
  - No **Linux** (UNIX), o **espaço** de **endereço inicial** do **filho** é uma **cópia** do **pai**, mas são **distintos – nenhuma memória gravável** é **compartilhada**.

Ciclo de vida de um processo

INSTITUTO MAUÁ DE TECNOLOGIA

- **Terminação**
  - Um processo pode ser terminado nas seguintes situações:
    - Término normal pelo processo (voluntário);
    - Término com erro pelo processo (voluntário);
    - Erro fatal (involuntário);
    - Ser terminado (“morto”) por outro processo (involuntário).

Ciclo de vida de um processo

INSTITUTO MAUÁ DE TECNOLOGIA

- **Terminação**
  - A **maioria** dos **processos termina** porque eles **fizeram** o seu **trabalho**. Existe uma **chamada** de **sistema** para informar ao **sistema operacional** que ele está **concluído**. No Linux é `exit`;
  - **Erros fatais** terminam processos. **Exemplo**: se um **usuário** digita o **comando** `gcc foo.c` para **compilar** o **programa** `foo.c` e tal arquivo não existe, o **compilador** simplesmente anuncia esse fato e termina;
  - **Erros** no **programa** podem terminar processo. **Exemplos**: **executar** uma **instrução ilegal**, **referenciar memória inexistente** ou **dividir** por **zero**;
  - Por fim, um processo pode **executar** uma **chamada** de **sistema** dizendo ao **sistema operacional** para **matar** algum **outro processo**. No Linux (UNIX) esta chamada é a `kill`.

Ciclo de vida de um processo

INSTITUTO MAUÁ DE TECNOLOGIA

- **Hierarquias de processos**
  - No Linux (UNIX), um **processo** e **todos** os seus **filhos** e outros **descendentes** juntos **formam** um **grupo de processos**;
  - Quando um **usuário envia** um **sinal do teclado**, o **sinal** é **entregue** a **todos** os **membros** do grupo de processos **atualmente associados** ao **teclado** (geralmente todos os **processos ativos** que foram **criados** na **janela atual**);
  - **Individualmente**, cada **processo** pode **capturar** o **sinal**, **ignorar** o **sinal** ou **executar** a **ação padrão**, que é ser morto pelo sinal.

Ciclo de vida de um processo

INSTITUTO MAUÁ DE TECNOLOGIA

- **Hierarquias de processos**
  - **Outro exemplo**: quando o Linux é inicializado, um **processo especial**, denominado `init`, está **presente** na **imagem** de **inicialização**;
  - **Quando ele** começa a executar, ele **lê** um **arquivo** informando **quantos terminais existem**.
  - Ele bifurca (`fork`) um **novo processo** por **terminal**;
  - **Esses processos esperam** que **alguém efetue login**. Se um **login** for **bem-sucedido**, o **processo** de **login executará** um **shell** para **aceitar comandos**;
  - **Esses comandos** podem **iniciar mais processos** e, assim por diante.
  - Assim, **todos** os **processos** em **todo** o **sistema pertencem** a **uma única árvore**, com `init` na **raiz**.

Referências bibliográficas

TANENBAUM, Andrew S. **Sistemas operacionais modernos**. 3. ed. São Paulo: Pearson, 2013. 653 p.