Aluna: Ingrid Camilly Costa Padilha

Exercícios Teóricos – Processos

1. Qual a diferença entre programa e processo?

- **Programa** é um conjunto de instruções armazenadas em um arquivo (entidade estática).
- **Processo** é a execução de um programa em andamento, incluindo contexto, recursos e estado (entidade dinâmica).

2. Quais são os estados de um processo e quando ocorrem as transições?

- **Novo (New):** processo está sendo criado.
- **Pronto (Ready):** aguardando CPU.
- **Executando (Running):** em uso pelo processador.
- **Bloqueado (Waiting):** esperando um evento ou recurso.
- **Finalizado (Terminated):** terminou a execução.

Transições:

- *New → Ready:* processo inicializado.
- *Ready → Running:* escalonado pelo SO.
- *Running → Waiting:* aguarda evento/entrada/saída.
- *Running → Ready:* perda da CPU (preempção).
- *Running → Terminated:* execução concluída.

---

3. O que contém um Process Control Block (PCB)?

- Identificação do processo (PID).
- Estado atual.
- Contador de programa.
- Registros da CPU.
- Informações de memória.

- Arquivos abertos.
- Informações de escalonamento.

4. O que acontece com os recursos de um processo quando ele termina?

- São liberados pelo sistema operacional (CPU, memória, arquivos e dispositivos).

5. Qual a diferença entre fork() e exec() no UNIX?

- fork(): cria um novo processo duplicando o processo pai.
- exec(): substitui a imagem do processo em execução por um novo programa.

6. Como funciona a hierarquia de processos em UNIX?

- Cada processo tem um pai (criador).
- O init/systemd é o processo raiz.
- Processos filhos podem criar seus próprios filhos, formando uma árvore de processos.

7. Compare memória compartilhada e troca de mensagens (IPC).

- **Memória compartilhada:** vários processos acessam a mesma área de memória.
    - Rápido, mas exige sincronização.
- **Troca de mensagens:** processos se comunicam via envio/recebimento de mensagens.
    - Mais seguro, mas pode ser mais lento.

8. Cite exemplos de chamadas de sistema usadas em IPC.

- Pipes (pipe(), mkfifo()).
- Filas de mensagens (msgget(), msgsnd(), msgrcv()).
- Memória compartilhada (shmget(), shmat(), shmdt()).
- Sinais (kill(), signal()).

9. Por que é importante que o sistema operacional faça gerenciamento de processos?

- Garante **uso eficiente da CPU**.
- Evita **deadlocks e conflitos**.
- Coordena a **execução concorrente**.
- Assegura **justiça** e **estabilidade** no sistema.

10. Explique a diferença entre processos independentes e processos cooperativos.

- **Independentes:** não compartilham dados ou recursos, execução isolada.
- **Cooperativos:** interagem e compartilham informações, podendo afetar uns aos outros.

11. O que é um processo zumbi em UNIX/Linux?

- Processo que terminou, mas cujo PCB permanece porque o pai ainda não leu seu status (via wait()).

12. Explique a diferença entre chamadas bloqueantes e não bloqueantes em IPC.

- **Bloqueantes:** processo fica parado até que a operação termine.
- **Não bloqueantes:** processo continua a execução mesmo sem resposta imediata.

13. Qual a diferença entre processo pesado (process) e thread (processo leve)?

- **Processo pesado:** unidade de execução completa, com espaço de memória independente.
- **Thread:** compartilha memória e recursos com outras threads do mesmo processo, mais leve.

14. Por que sistemas operacionais multiprogramados precisam de troca de contexto (context switch)?

- Para **alternar entre processos** de forma eficiente.
- Permite **multiprogramação** e **uso justo da CPU**.
- Garante **responsividade** em sistemas multitarefa.

15. Cite vantagens e desvantagens da comunicação via memória compartilhada.

Vantagens:

- Alta performance.
- Ideal para grande volume de dados.

Desvantagens:

- Necessidade de sincronização.
- Risco de condições de corrida.
- Mais complexa de implementar corretamente.