UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
DISCIPLINA: SISTEMAS OPERACIONAIS
PROFESSORA: DEBORAH MAGALHÃES
ALUNO: MAURICIO BENJAMIN DA ROCHA

LISTA DE REVISÃO

1. Quais as principais funções do sistema operacional?

- Gerenciamento de recursos: O sistema operacional gerencia os recursos de hardware do computador, como processadores, memória, dispositivos de armazenamento e periféricos. Ele coordena e aloca esses recursos entre os processos e aplicativos em execução (Oculta a complexidade).

- Gerenciamento de processos: O sistema operacional é responsável pelo gerenciamento de processos, que são as instâncias em execução de um programa. Ele cria, interrompe, agenda e controla os processos para garantir um uso eficiente dos recursos do sistema.

- Gerenciamento de memória: O sistema operacional aloca e gerencia a memória do computador. Ele controla a alocação de memória para os processos em execução, gerencia a memória virtual (se disponível) e realiza a troca de dados entre a memória principal e o disco rígido.

- Gerenciamento de sistemas de arquivos: O sistema operacional fornece uma interface para acessar, organizar e manipular arquivos no sistema de armazenamento. Ele gerencia a estrutura de diretórios, controla permissões de acesso e realiza operações de leitura, gravação e exclusão de arquivos.

- Gerenciamento de dispositivos: O sistema operacional controla e coordena a comunicação com os dispositivos de entrada e saída, como teclado, mouse, impressoras, discos rígidos, placas de rede, entre outros. Ele fornece drivers e interfaces para que os aplicativos possam interagir com esses dispositivos.

- Interface com o usuário: O sistema operacional oferece uma interface para que os usuários possam interagir com o computador. Isso pode ser por meio de uma interface gráfica do usuário (GUI), linha de comando ou outros métodos de interação, dependendo do sistema operacional em questão.

- Gerenciamento de segurança: O sistema operacional implementa mecanismos de segurança para proteger o sistema contra acesso não autorizado, ataques de malware e outras ameaças. Isso inclui autenticação

de usuários, controle de acesso a recursos e proteção contra vírus e outras formas de software malicioso (Linux é um com referência em segurança).

2. Quais os principais elementos que compõem um sistema de informação?
Obs: suponho que seja sistema operacional ao invés de sistema de informação

- Kernel: É o núcleo do sistema operacional e representa a parte central responsável pelo gerenciamento de recursos do sistema. Ele controla a alocação de recursos, gerencia processos e threads, lida com interrupções, gerencia memória e realiza operações de E/S (entrada/saída). O kernel é a camada mais próxima do hardware e fornece interfaces para que os programas e os componentes do sistema operacional interajam entre si.

- Gerenciador de processos: É responsável pelo gerenciamento e escalonamento de processos. Ele atribui tempo de processador, controla a criação, interrupção e término de processos, gerencia as trocas de contexto entre os processos e garante que o processador seja utilizado de forma eficiente.

- Gerenciador de memória: É responsável pela alocação e desalocação de memória para os processos em execução. Ele gerencia a memória física disponível, coordena a alocação de memória para os processos, realiza a memória virtual (se disponível) e lida com a troca de dados entre a memória principal e o armazenamento secundário.

- Sistema de arquivos: É responsável pelo gerenciamento dos arquivos e diretórios presentes no sistema. Ele fornece métodos para criar, abrir, fechar, ler, gravar e excluir arquivos, além de controlar permissões de acesso e garantir a integridade dos dados armazenados.

- Gerenciador de dispositivos: É responsável por controlar a comunicação entre o sistema operacional e os dispositivos de hardware. Ele fornece interfaces e drivers para que o sistema operacional possa interagir com dispositivos como teclado, mouse, impressoras, discos rígidos, placas de rede e outros periféricos.

- Interface com o usuário: É responsável por permitir que os usuários interajam com o sistema operacional. Pode ser uma interface gráfica do usuário (GUI) que inclui elementos como janelas, menus, ícones e botões, ou uma interface de linha de comando (CLI) que permite aos usuários inserir comandos e receber feedback textual.

- Serviços de rede: Alguns sistemas operacionais possuem serviços de rede embutidos que permitem a comunicação e o compartilhamento de recursos

entre computadores conectados em rede. Isso pode incluir serviços como compartilhamento de arquivos, impressão em rede, servidor web, servidor de e-mail, entre outros.

3. Quais as gerações dos sistemas operacionais e qual a principal característica de
cada 1 delas?

1. Primeira Geração - Sistemas Operacionais de Válvulas (1944~1955):
- Característica: Utilizavam válvulas eletrônicas para implementar as funcionalidades do sistema operacional.
- Exemplo: ENIAC, UNIVAC I.

2. Segunda Geração - Sistemas Operacionais de Transistores (1955~1965):
- Característica: Utilizavam transistores para substituir as válvulas, tornando os sistemas mais confiáveis, compactos e eficientes em termos de consumo de energia.
- Exemplo: IBM 1401, DEC PDP-8.

3. Terceira Geração - Sistemas Operacionais de Circuitos Integrados e uso da multiprogramação (1965~1980):
- Característica: Utilizavam circuitos integrados e multiprogramação para evitar “perda” de tempo no intervalo entre processos, possibilitando a criação de sistemas ainda mais compactos, poderosos e eficientes.
- Exemplo: IBM System/360, DEC PDP-11.

4. Quarta Geração - Sistemas Operacionais de Chips e Microprocessadores (1980~1990):
- Característica: Introdução dos microprocessadores, que permitiram a criação de sistemas operacionais mais versáteis e de baixo custo, levando ao surgimento dos computadores pessoais.
- Exemplo: Microsoft Disk Operating System (MS-DOS), Apple DOS.

5. Quinta Geração - Sistemas Operacionais Multitarefa e Multimídia, Sistemas Operacionais de Código Aberto e Dispositivos Móveis (1990~Até dias atuais):

- Característica: Suporte a multitarefa, permitindo a execução simultânea de vários aplicativos, e recursos avançados de multimídia, crescimento do software de código aberto, com sistemas operacionais como o Linux, e o surgimento de dispositivos móveis, como smartphones e tablets.

- Exemplo: Windows 95/98/Me, MacOS, Linux (distribuições como Ubuntu, Fedora), Android, iOS.

4. O que é Multiprogramação?

A multiprogramação é uma técnica usada em sistemas operacionais para aumentar a eficiência do processador, permitindo que vários programas ou processos sejam executados simultaneamente. Na multiprogramação, vários programas são carregados na memória ao mesmo tempo e o processador alterna rapidamente entre eles, dando a impressão de que estão sendo executados simultaneamente.

A ideia básica por trás da multiprogramação é aproveitar melhor o tempo ocioso do processador. Enquanto um programa está aguardando a conclusão de uma operação de entrada/saída ou aguardando por dados da memória, o processador pode passar para outro programa que está pronto para ser executado. Isso permite que o processador seja utilizado de forma mais eficiente, aumentando a produtividade do sistema.

O sistema operacional deve controlar a execução e a interrupção dos programas, fornecendo mecanismos para alternar entre eles de forma transparente. Isso envolve o escalonamento de processos, a alocação de tempo de processador e o gerenciamento da memória.

5. Para acompanhar as evoluções na indústria do hardware, diferentes tipos de sistemas operacionais foram desenvolvidos. Entre eles, destaque 3 tipos utilizados atualmente e comente brevemente sobre eles.

- Sistemas Operacionais de Uso Geral:

  Exemplos: Windows, macOS, Linux.

  Características: Esses sistemas operacionais são projetados para atender às necessidades gerais dos usuários, oferecendo uma ampla gama de recursos e aplicativos. Eles fornecem uma interface gráfica do usuário (GUI) amigável, suporte a multitarefa, gerenciamento de arquivos, conectividade de rede e uma variedade de aplicativos, como navegadores da web, suítes de produtividade e reprodutores de mídia. Esses sistemas operacionais são usados em desktops, laptops e servidores.

- Sistemas Operacionais Móveis:
  Exemplos: Android, iOS.

  Características: Esses sistemas operacionais são projetados especificamente para dispositivos móveis, como smartphones e tablets. Eles são otimizados

para eficiência de energia, oferecem interfaces de toque intuitivas e fornecem acesso a uma ampla variedade de aplicativos móveis por meio de lojas de aplicativos. Os sistemas operacionais móveis oferecem recursos como chamadas telefônicas, mensagens de texto, navegação na web, aplicativos de mídia, jogos e muito mais. Além disso, eles suportam recursos avançados, como reconhecimento de voz, assistentes virtuais e integração com dispositivos vestíveis.

- Sistemas Operacionais Embarcados:

  Exemplos: FreeRTOS, VxWorks, Embedded Linux.

  Características: Esses sistemas operacionais são projetados para serem executados em dispositivos embarcados, como sistemas de controle industrial, dispositivos médicos, roteadores, sistemas de automação residencial e outros dispositivos eletrônicos. Eles são otimizados para requisitos específicos, como tamanho reduzido, baixo consumo de energia e tempo de resposta rápido. Os sistemas operacionais embarcados fornecem um ambiente de execução confiável e estável, gerenciando tarefas, comunicação entre dispositivos, interação com periféricos e controle de dispositivos específicos.

6. Apesar da gama de sistemas operacionais, focados em nichos específicos, como, por exemplo: sistemas operacionais servidores, sistemas operacionais de redes de sensores sem fio, entre outros, tais sistemas possuem conceitos em comuns. Defina processo, espaço de endereçamento e arquivo.

- Processo: Em um sistema operacional, um processo é uma instância em execução de um programa. Ele representa uma entidade ativa que possui seu próprio estado, incluindo o contador de programa, registradores, pilha e outras informações relevantes. Um processo é responsável por executar as instruções do programa e realizar operações no sistema, como acesso a arquivos, comunicação com outros processos e uso dos recursos do sistema, como memória e processador.

- Espaço de endereçamento: O espaço de endereçamento é o intervalo de endereços de memória disponíveis para um processo em um sistema operacional. Cada processo tem seu próprio espaço de endereçamento virtual, que é a percepção do processo de como a memória é organizada. O espaço de endereçamento é dividido em várias seções, como código, dados, pilha e heap, e é usado para armazenar instruções, dados, pilhas de chamadas e estruturas de dados dinâmicas do programa em execução. O espaço de endereçamento permite que o processo acesse e manipule sua

memória de forma isolada, protegendo-o de interferências de outros processos.

- Arquivo: Em sistemas operacionais, um arquivo é uma unidade lógica de armazenamento de informações ou dados. Ele pode representar diferentes tipos de informações, como documentos, imagens, programas executáveis ou até mesmo dispositivos físicos, como impressoras ou portas seriais. Um arquivo possui um nome, um formato e um conjunto de atributos que especificam suas propriedades, como permissões de acesso, tamanho e data de modificação. Os sistemas operacionais fornecem interfaces e chamadas de sistema para criar, abrir, fechar, ler, gravar e manipular arquivos, permitindo que os programas interajam com o sistema de arquivos e acessem os dados armazenados neles.

7. Em muitas situações, a CPU é compartilhada entre 2 ou mais processos. No momento em a CPU é atribuída à outro processo, o processo corrente precisa guardar seu estado atual para que seja dada continuidade à sua execução em um instante posterior. Quais informações o processo armazena e onde elas são armazenadas?

1. Registradores do Processador: Os valores dos registradores do processador, como o contador de programa (PC), registradores de propósito geral, registrador de status e registradores de ponto flutuante, precisam ser salvos. Esses registradores contêm informações cruciais sobre o estado da execução do processo, incluindo a próxima instrução a ser executada e os valores dos dados processados.

2. Ponteiro de Pilha: O ponteiro de pilha (stack pointer) indica a posição atual na pilha do processo. A pilha é usada para armazenar informações temporárias, como variáveis locais e valores de retorno de chamadas de função. O ponteiro de pilha precisa ser salvo para que o processo possa retomar sua execução a partir do ponto em que parou.

3. Informações de Estado do Processo: Além dos registradores e do ponteiro de pilha, outras informações de estado do processo também podem ser armazenadas. Isso pode incluir o estado dos sinais pendentes, informações de gerenciamento de memória, contexto de E/S (entrada/saída) e outros detalhes específicos do processo que são relevantes para sua execução correta.

Essas informações do estado do processo são armazenadas em uma estrutura de dados chamada de contexto do processo ou PCB (Process Control Block). O PCB é uma estrutura de dados mantida pelo sistema operacional para cada processo em execução. Ele contém todas as informações necessárias para o sistema operacional gerenciar e controlar o processo.

Quando um processo é interrompido e sua CPU é atribuída a outro processo, o sistema operacional salva o estado atual do processo em seu PCB correspondente. Em seguida, o sistema operacional carrega o estado do próximo processo a ser executado a partir do PCB correspondente e continua a execução a partir desse ponto. Esse processo de troca de contexto é realizado pelo sistema operacional em uma operação conhecida como escalonamento de processos, que é responsável por decidir qual processo deve ter acesso à CPU em um determinado momento.

8. Explique os principais eventos que levam a criação de um processo e diferencie o fluxo de criação no Unix e no Windows.

Os principais eventos que levam à criação de um processo em um sistema operacional são:

- Inicialização do Sistema Operacional: Quando um sistema operacional é inicializado, um processo especial chamado de processo "init" (no Unix/Linux) ou "System" (no Windows) é criado. Esse processo é o primeiro a ser executado e é responsável por iniciar e gerenciar outros processos durante o funcionamento do sistema.

- Solicitação de um Usuário: Um usuário pode iniciar a criação de um processo ao solicitar a execução de um programa ou aplicativo específico. Isso pode ser feito por meio de uma linha de comando, interface gráfica ou outro meio de interação com o sistema operacional. Quando o usuário inicia um programa, o sistema operacional cria um novo processo para executar esse programa.

- Eventos de Interação do Sistema: Diversos eventos podem ocorrer no sistema operacional que desencadeiam a criação de processos. Isso pode incluir eventos como chegada de mensagens ou interrupções de dispositivos externos, como solicitações de entrada/saída (E/S) de dispositivos de armazenamento ou redes. Quando um evento desse tipo ocorre, o sistema operacional pode criar um processo específico para lidar com essa solicitação ou evento.

Quanto à diferença no fluxo de criação de processos no Unix e no Windows, existem algumas distinções:

No Unix/Linux:

- A criação de processos é feita por meio do comando "fork()", que cria uma cópia exata do processo pai, conhecida como processo filho. Após a criação do processo filho, ele pode executar um programa diferente usando a chamada do sistema "exec()". O processo filho herda a maioria dos recursos

e propriedades do processo pai, como o espaço de endereçamento e o ambiente de execução.

No Windows:

- A criação de processos é realizada por meio da função "CreateProcess()" que cria um novo processo. No Windows, a criação de processos envolve a criação de uma nova estrutura de dados conhecida como processo de usuário (user process), que contém informações sobre o novo processo, como identificação, atributos de segurança e informações sobre o ambiente de execução. O processo de usuário é, então, associado a um processo do kernel (kernel process), que representa o espaço de endereçamento e os recursos do sistema associados a esse processo.

Em resumo, embora tanto o Unix quanto o Windows sigam o conceito de criação de processos, existem diferenças na implementação e nos detalhes do fluxo de criação de processos em cada sistema operacional.

9. Explique os principais fluxos que levam ao término de um processo.

- Conclusão Normal: Um processo pode terminar sua execução normalmente quando conclui todas as instruções do programa e atinge seu ponto final. Nesse caso, o processo é encerrado de forma adequada, liberando todos os recursos alocados e notificando o sistema operacional de sua conclusão.

- Saída do Programa: Um processo pode ser encerrado por meio de uma instrução de saída explicitamente chamada dentro do programa em execução. Essa instrução pode ser usada para indicar que o programa alcançou um estado específico e precisa ser encerrado. Quando essa instrução é executada, o processo é finalizado.

- Erro ou Exceção: Um processo pode ser terminado devido à ocorrência de um erro grave ou exceção durante a sua execução. Isso pode acontecer quando ocorrem erros de acesso à memória, violações de segurança, falhas de divisão por zero, entre outros problemas. Nessas situações, o sistema operacional pode capturar a exceção, registrar o erro e encerrar o processo afetado para garantir a estabilidade e segurança do sistema.

- Interrupção Externa: Um processo pode ser interrompido ou terminado por eventos externos, como um sinal enviado por outro processo ou pelo sistema operacional. Isso pode ocorrer quando um processo recebe um sinal de encerramento, sinalizando que ele precisa ser finalizado. Esses sinais podem ser gerados por solicitação do usuário, por decisões de gerenciamento de recursos ou por eventos específicos do sistema.

- Encerramento por Solicitação do Usuário: Um processo pode ser finalizado por solicitação direta do usuário. Isso pode ocorrer quando o usuário decide fechar um programa ou encerrar uma tarefa específica. Através de interações com a interface do usuário ou comandos específicos do sistema operacional, o usuário pode encerrar um processo em andamento.

10. Defina hierarquia de processos e explique a diferença entre a implementação dessa hierarquia no Unix e Windows.

A hierarquia de processos é uma estrutura em que os processos são organizados em um sistema operacional. Ela estabelece relações de dependência entre os processos, criando uma estrutura em forma de árvore, onde cada processo tem um processo pai, exceto o processo raiz.

No Unix (incluindo o Linux), a hierarquia de processos segue um modelo simples em que cada processo pode ter vários processos filhos, mas apenas um processo pai. Quando um processo é criado, ele herda o ID do processo pai (PID) e se torna um processo filho desse processo pai. Isso cria uma estrutura hierárquica em que os processos podem ser organizados em várias camadas.

No Unix, a hierarquia de processos é implementada por meio do mecanismo de chamada de sistema "fork()". Quando um processo cria um novo processo usando a chamada "fork()", o processo filho herda o PID do processo pai e se torna um processo filho. O processo pai pode ter controle sobre seus processos filhos, enviando sinais, aguardando sua conclusão ou até mesmo terminando-os.

No Windows, a hierarquia de processos também é baseada em uma estrutura de árvore, onde cada processo pode ter vários processos filhos, mas a diferença é que um processo pode ter vários processos pais. Isso é conhecido como modelo de herança múltipla de processos no Windows.

No Windows, a criação de processos é feita por meio da função "CreateProcess()", que permite criar um novo processo e definir os processos pais para o novo processo. Isso significa que um processo no Windows pode ser criado com um ou mais processos pais, o que resulta em uma estrutura hierárquica com uma relação de herança múltipla.

A implementação da hierarquia de processos no Windows permite uma maior flexibilidade na organização dos processos e na comunicação entre eles. No entanto, também pode resultar em uma estrutura mais complexa em comparação com o modelo de hierarquia de processos mais simples do Unix.

Em resumo, a hierarquia de processos é uma estrutura organizacional que estabelece relações de dependência entre os processos em um sistema

operacional. No Unix, cada processo tem um processo pai e pode ter vários processos filhos, enquanto no Windows um processo pode ter vários processos pais e processos filhos. Essas diferenças na implementação refletem nas relações entre os processos e nas capacidades de comunicação e controle entre eles.

11. Explique o ciclo de vida de um processo, citando todos os seus estados e transições.

O ciclo de vida de um processo descreve os diferentes estados pelos quais um processo passa durante sua execução em um sistema operacional. Existem diferentes modelos de ciclo de vida de processo, mas aqui está uma descrição geral dos principais estados e transições comumente encontrados:

1. Novo: Nesse estado, o processo é criado pelo sistema operacional. Ele está pronto para ser atribuído aos recursos necessários e iniciar sua execução. O sistema operacional realiza as tarefas de inicialização, alocação de recursos e criação de estruturas de dados, como o PCB (Process Control Block).

Transição: Quando um processo é criado e passa a existir, ele faz a transição do estado "Novo" para o próximo estado.

2. Pronto: No estado pronto, o processo está aguardando para ser executado pela CPU. Ele já possui todos os recursos necessários para sua execução, mas está aguardando sua vez na fila de processos prontos.

Transição: Quando o processo recebe a CPU do sistema operacional para execução, ele faz a transição do estado "Pronto" para o estado "Executando".

3. Executando: Nesse estado, o processo está sendo executado pela CPU. Ele está realizando suas instruções e processando seus dados.

Transições:
- Interrupção: O processo pode ser interrompido por um evento externo, como uma interrupção de hardware ou um sinal enviado por outro processo. Nesse caso, ele faz a transição do estado "Executando" para o estado "Bloqueado" ou "Pronto", dependendo do tipo de interrupção e do tratamento definido.
- Fim de Quanto: O processo pode terminar sua fatia de tempo de execução, e o sistema operacional decide interrompê-lo e selecionar outro processo para executar. Nesse caso, o processo faz a transição do estado "Executando" para o estado "Pronto".

4. Bloqueado (ou Espera): Quando um processo está aguardando por algum evento externo, como entrada/saída (E/S) de um dispositivo ou uma operação de

leitura/gravação em disco, ele entra no estado bloqueado. O processo não pode prosseguir até que o evento ocorra e ele seja notificado.

Transição: Quando o evento pelo qual o processo estava esperando ocorre, ele faz a transição do estado "Bloqueado" para o estado "Pronto" e fica pronto para ser novamente escalonado para execução.

5. Terminado: Nesse estado, o processo conclui sua execução e libera todos os recursos alocados. Ele não está mais em execução e não será programado novamente.

Transição: O processo faz a transição do estado atual (qualquer estado) para o estado "Terminado" quando conclui sua execução.

É importante ressaltar que o ciclo de vida de um processo pode variar dependendo do sistema operacional e de suas políticas de escalonamento e gerenciamento de processos. Além disso, existem estados adicionais, como "Suspendido" ou "Zumbi", que podem ser encontrados em algumas implementações específicas.

12. Qual o papel do escalonador de processos?

O escalonador de processos é uma parte fundamental do sistema operacional responsável por decidir qual processo será executado pela CPU em determinado momento. Ele é responsável por fazer o gerenciamento e a distribuição adequada dos recursos de processamento disponíveis, a fim de garantir uma utilização eficiente da CPU e um bom desempenho do sistema como um todo.

O papel do escalonador de processos pode ser resumido nas seguintes funções:

- Decidir a ordem de execução: O escalonador determina qual processo será selecionado para executar em seguida, com base em uma política de escalonamento definida. Essa política pode levar em consideração vários fatores, como prioridades de processo, tempo de chegada, tempo de execução e outros critérios.

- Distribuir recursos de processamento: O escalonador assegura que todos os processos tenham acesso justo à CPU, evitando a monopolização por um único processo. Ele distribui as fatias de tempo de execução (quantum) de forma equitativa entre os processos, permitindo que cada um tenha sua chance de utilizar a CPU.

- Gerenciar a prioridade dos processos: O escalonador pode atribuir prioridades aos processos com base em políticas predefinidas. Processos de

alta prioridade podem receber mais tempo de execução ou serem escolhidos com mais frequência pelo escalonador, enquanto processos de baixa prioridade podem ter menos tempo de execução ou serem escolhidos com menor frequência.

- Lidar com eventos de interrupção e bloqueio: O escalonador gerencia eventos de interrupção, como sinais de hardware ou solicitações de entrada/saída (E/S), garantindo que o processo adequado seja interrompido e colocado no estado correto (por exemplo, bloqueado ou pronto) para lidar com a interrupção. Ele também lida com processos bloqueados, movendo-os para o estado pronto quando o evento de bloqueio é resolvido.

- Otimizar o desempenho do sistema: O escalonador visa maximizar a eficiência e o desempenho geral do sistema, selecionando processos de forma a minimizar o tempo de espera, melhorar a utilização da CPU, reduzir o tempo de resposta e garantir uma execução justa e equitativa para todos os processos.

13. O que é chamada de sistema? Cite pelo menos 1 exemplo de cada categoria de chamadas.

Uma chamada de sistema, também conhecida como syscall (system call), é uma interface entre um programa de aplicação e o kernel (núcleo) de um sistema operacional. Ela permite que o programa solicite serviços e recursos do sistema operacional para realizar operações que estão fora do seu escopo normal de execução. Existem diferentes categorias de chamadas de sistema, sendo as principais:

1. Gerenciamento de arquivos:
   - open(): Abre um arquivo existente ou cria um novo arquivo.
   - read(): Lê dados de um arquivo.
   - write(): Escreve dados em um arquivo.
   - close(): Fecha um arquivo aberto.

2. Gerenciamento de processos:
   - fork(): Cria um novo processo a partir do processo atual.
   - exec(): Substitui a imagem de um processo por outro programa.
   - exit(): Termina a execução de um processo.
   - wait(): Suspensa a execução de um processo até que um processo filho seja encerrado.

3. Gerenciamento de memória:
   - malloc(): Aloca um bloco de memória dinamicamente.
   - free(): Libera um bloco de memória alocado anteriormente.

4. Gerenciamento de dispositivos de E/S:
   - read(): Lê dados de um dispositivo de E/S, como teclado ou mouse.
   - write(): Escreve dados em um dispositivo de E/S, como monitor ou impressora.
   - ioctl(): Controla e configura dispositivos de E/S.

5. Gerenciamento de rede:
   - socket(): Cria um ponto de extremidade para comunicação em rede.
   - connect(): Estabelece uma conexão com um servidor em uma rede.
   - send(): Envio de dados para um destino em uma rede.
   - recv(): Recebe dados de uma conexão de rede.

Esses são apenas alguns exemplos de chamadas de sistema em cada categoria. É importante notar que as chamadas de sistema podem variar de acordo com o sistema operacional e a linguagem de programação utilizada, mas o conceito geral de fornecer uma interface para serviços do sistema operacional permanece o mesmo.

14. Explique, com suas palavras, o que o seguinte comando está fazendo:
#ps aux | grep aluno

O comando lista todos os processos em execução no sistema e, em seguida, filtra apenas as linhas que contêm a palavra "aluno".

"ps aux": O comando "ps" é usado para exibir informações sobre os processos em execução no sistema. A opção "aux" é usada para exibir todos os processos em um formato detalhado.
"|": O caractere "|" é conhecido como "pipe" ou tubulação. Ele permite redirecionar a saída de um comando para a entrada de outro comando. Logo ele redireciona a saída do comando "ps aux" para o comando "grep aluno".

"grep aluno": O comando "grep" é usado para pesquisar padrões em um texto. Neste caso, estamos procurando o termo "aluno". Quando combinado com o pipe, o "grep" recebe a saída do comando anterior e filtra as linhas que contêm o padrão especificado.