Sistemas Operacionais: Processos, Threads e Escalonamento

1. **Impacto da falha de uma thread:** Se uma thread falha em um processo multi-threaded, geralmente o processo inteiro termina, pois as threads compartilham o mesmo espaço de memória e recursos, e um erro em uma pode corromper o estado das outras.
2. **Descritores de arquivos e isolamento:** Descritores de arquivos são identificadores numéricos que o sistema operacional usa para representar arquivos abertos ou recursos de I/O. Eles são locais a cada processo, garantindo que um processo não acesse diretamente os arquivos de outro, promovendo o isolamento.
3. **Copy-on-write (COW) e eficiência:** COW é uma técnica que evita copiar dados imediatamente ao criar uma cópia de um processo (como no `fork()`). Inicialmente, pai e filho compartilham as mesmas páginas de memória como "somente leitura". A cópia de uma página só é feita (escrevendo nela) se um dos processos tentar modificá-la, economizando memória e acelerando a criação de processos.
4. **Riscos do uso de threads:** Os principais riscos incluem problemas de sincronização (condições de corrida), deadlocks, aumento da complexidade no desenvolvimento e depuração, e o risco de uma thread bloquear todo o processo em certas situações.
5. **Custos ocultos da concorrência (troca de contexto):** A troca de contexto, necessária para alternar entre processos/threads, gera overhead. Custos ocultos incluem tempo de CPU gasto salvando/carregando estados, poluição/invalidação do cache da CPU e do TLB (Translation Lookaside Buffer), e o tempo de acesso à memória para gerenciar esses estados.
6. **Starvation e algoritmos:** Starvation (ou inanição) ocorre quando um processo ou thread é repetidamente impedido de acessar a CPU ou um recurso necessário. Pode acontecer em algoritmos de escalonamento baseados em prioridade (se houver um fluxo contínuo de tarefas de alta prioridade) e no Shortest Job First (se tarefas longas são continuamente preteridas por tarefas curtas).
7. **Memória virtual e isolamento de processos:** Memória virtual é uma técnica que dá a cada processo a ilusão de ter seu próprio espaço de memória grande e contíguo. Ela garante isolamento através do Mapeamento de Endereços Virtuais para Físicos (feito pela MMU e tabelas de páginas exclusivas para cada processo), impedindo que um processo acesse a memória de outro.
8. **Escalonamento de processos e necessidade:** Escalonamento de processos é a atividade do sistema operacional que decide qual processo ou thread deve usar a CPU e por quanto tempo. É essencial para permitir o multi-tarefas, maximizar a utilização da CPU, garantir justiça e responsividade, e otimizar o throughput.
9. **Algoritmo Shortest Job First (SJF):** SJF escolhe o processo com o menor tempo de execução estimado para rodar em seguida.
    - **Vantagens:** Minimiza o tempo médio de espera e de turnaround.
    - **Desvantagens:** Pode causar starvation para processos longos e é difícil de implementar na prática, pois exige conhecer o tempo de execução futuro das tarefas.
10. **Tipos de sistemas de arquivos:** São métodos para organizar e armazenar dados.
    - **FAT (FAT16, FAT32):** Simples, compatível, mas limitado em tamanho e sem journaling (recuperação de erros).

    - **NTFS:** Usado no Windows, robusto, com journaling, segurança (permissões), compressão.
    - **ext (ext2, ext3, ext4):** Usado no Linux, com ou sem journaling (ext3/4), robusto e eficiente.
    - **HFS+/APFS:** Usados na Apple, com journaling (HFS+) e recursos avançados para SSDs (APFS).
    - **ZFS:** Avançado, com integridade de dados, snapshots, copy-on-write, alta escalabilidade.
11. **Múltiplas filas com realimentação (Multilevel Feedback Queue - MLFQ):** Um algoritmo de escalonamento complexo que usa múltiplas filas com diferentes prioridades e quanta de tempo. Processos que usam seu quantum inteiro são rebaixados para filas de menor prioridade (realimentação), e processos em filas de baixa prioridade podem ser promovidos (envelhecimento) para evitar starvation.
12. **Vantagens do uso de threads vs. processos:**
    - **Menor overhead:** Criação e troca de contexto são mais rápidas, pois threads são "mais leves" e compartilham recursos.
    - **Compartilhamento de recursos:** Compartilham o mesmo espaço de memória, facilitando a comunicação e a troca de dados.
    - **Responsividade:** Se uma thread bloqueia, outras podem continuar executando.
    - **Paralelismo:** Permitem verdadeira execução paralela em CPUs multi-core.
13. **Thread no nível de usuário e kernel:**
    - **Nível de Usuário:** Gerenciadas por uma biblioteca no espaço do usuário, sem intervenção direta do kernel. Rápidas, mas se uma bloqueia, todo o processo bloqueia, e não há paralelismo real em multi-core.
    - **Nível do Kernel:** Gerenciadas diretamente pelo kernel. Mais lentas de criar/trocar contexto (exigem syscalls), mas permitem paralelismo real e não bloqueiam o processo inteiro em chamadas de sistema.
14. **Diferença entre processo e thread:**
    - **Processo:** Unidade de execução independente, com seu próprio espaço de memória isolado e recursos. Mais "pesado".
    - **Thread:** Unidade de execução "leve" dentro de um processo, compartilhando o espaço de memória e recursos do processo pai, mas com sua própria pilha e registradores.
15. **Critérios desejáveis em um bom algoritmo de escalonamento:** Otimização da utilização da CPU, alto throughput (número de tarefas concluídas), baixo tempo de turnaround, baixo tempo de espera, bom tempo de resposta (para interatividade), justiça (evitar starvation) e baixa sobrecarga do próprio escalonador.
16. **Quando preferir múltiplos processos a múltiplas threads:** Quando a isolamento e segurança são críticos (falha de um processo não afeta outros), para rodar programas ou serviços totalmente independentes, para aplicar limites de recursos estritos, para integrar com código legado não thread-safe, e em ambientes multi-usuário.
17. **Escalonador preemptivo e não preemptivo:**
    - **Preemptivo:** Pode interromper um processo em execução para ceder a CPU a outro (ex: por expiração de quantum ou chegada de um processo de maior prioridade). Essencial para sistemas interativos.

    - **Não preemptivo:** Uma vez que um processo começa a executar, ele só libera a CPU voluntariamente (ao terminar ou bloquear por I/O). Mais simples, mas pode levar a longos tempos de espera.
18. **Diferença entre escalonador e despachante:**
    - **Escalonador:** Decide *qual* processo vai rodar em seguida, baseando-se no algoritmo de escalonamento.
    - **Despachante:** É o módulo que *executa* a decisão do escalonador, realizando a troca de contexto (salvar o estado do processo antigo e carregar o estado do novo) e transferindo o controle da CPU.
19. **Algoritmo Round Robin (RR) e quantum:** RR é um algoritmo preemptivo para sistemas de tempo compartilhado. Cada processo recebe uma fatia de tempo de CPU, chamada **quantum**. Se o processo não termina dentro do quantum, ele é preemptado e colocado no final da fila. O quantum é o tempo fixo que cada processo pode usar a CPU antes de ser interrompido.
20. **O que é salvo durante uma mudança de contexto entre processos:** O estado completo do processo, incluindo: registradores da CPU (PC, SP, etc.), palavra de status do processador (PSW), informações de gerenciamento de memória (ponteiro para a tabela de páginas), informações sobre arquivos abertos e estado de I/O, e dados de contabilidade do processo. Tudo isso é armazenado no Bloco de Controle de Processo (PCB).
21. **Diferença da troca de contexto entre threads e processos:** A troca de contexto entre threads é mais rápida e leve porque threads do mesmo processo compartilham o mesmo espaço de memória e a maioria dos recursos, então menos informações precisam ser salvas/restauradas (principalmente os registradores e a pilha da thread). Na troca de processo, o sistema precisa salvar/carregar o *contexto completo* do processo, incluindo informações de gerenciamento de memória e tabelas de arquivos.
22. **Threads compartilham o mesmo espaço de memória:** Significa que todas as threads de um processo operam dentro da mesma região de memória virtual. Elas acessam as mesmas variáveis globais, o mesmo heap e o mesmo código executável. Cada thread, no entanto, tem sua própria pilha de execução.
23. **Vantagens do Lottery Scheduling:** Útil para garantir compartilhamento proporcional da CPU (mais tickets = mais chances de rodar) e para evitar starvation, pois todo processo com pelo menos um ticket tem chance de ser escolhido.
24. **Cache de arquivos para desempenho:** O sistema operacional usa uma parte da RAM como cache de arquivos. Ao ler, verifica o cache primeiro (se hit, é rápido). Se não está no cache, lê do disco (lento) e armazena no cache, muitas vezes fazendo "read-ahead". Ao escrever, os dados são primeiro escritos no cache e depois, assincronamente, para o disco (write-back), melhorando a responsividade.
25. **Algoritmos de escalonamento em sistemas de tempo real:**
    - **Rate Monotonic Scheduling (RMS):** Prioridade estática baseada na frequência da tarefa (períodos mais curtos = maior prioridade). Preemptivo.
    - **Earliest Deadline First (EDF):** Prioridade dinâmica baseada na data limite (deadline) mais próxima. Preemptivo e ótimo em utilização.
    - **Least Laxity First (LLF):** Prioridade dinâmica baseada na "folga" (laxity) da tarefa (menor folga = maior prioridade). Preemptivo.
26. **Modelos de multithreading (N:1, 1:1, N:M):**

    - **N:1 (Muitos para Um):** Mapeia várias threads de usuário para uma única thread de kernel. Rápido, mas se uma thread bloqueia, todo o processo bloqueia.
    - **1:1 (Um para Um):** Cada thread de usuário mapeia para uma thread de kernel. Permite paralelismo real em multi-core, mas tem maior sobrecarga.
    - **N:M (Muitos para Muitos):** Um pool de threads de kernel suporta várias threads de usuário. Busca equilibrar o paralelismo com a sobrecarga, combinando vantagens dos outros modelos.
27. **Escalonamento por prioridade:** Atribui um número de prioridade a cada processo. A CPU é dada ao processo com a maior prioridade. Pode ser preemptivo ou não. O principal problema é a starvation, combatida com técnicas como o "envelhecimento" (aging), que aumenta gradualmente a prioridade de processos antigos.
28. **Paginação e utilidade:** Paginação é uma técnica de gerenciamento de memória que divide o espaço de endereço lógico (do processo) em "páginas" e a memória física em "frames" de mesmo tamanho. Uma tabela de páginas mapeia páginas lógicas para frames físicos, permitindo que a memória de um processo seja não contígua na RAM. Útil para: implementar memória virtual, garantir isolamento entre processos e permitir compartilhamento de memória.
29. **Turnaround time e avaliação de escalonadores:** Turnaround time é o tempo total desde a submissão de um processo até sua conclusão. É uma métrica chave para avaliar escalonadores, pois algoritmos que minimizam o tempo médio de turnaround são considerados mais eficientes em termos de throughput.
30. **Mudança de contexto e overhead:** A mudança de contexto é o processo de salvar o estado de um processo/thread e carregar o estado de outro para alternar a execução. Gera overhead (tempo não produtivo) devido ao tempo gasto salvando/carregando registradores, trocando informações de gerenciamento de memória (tabelas de páginas) e invalidando caches da CPU, além do custo da própria transição de modo (usuário/kernel).
31. **Poluição de cache em servidores DNS e exploração por atacantes:** É a injeção de registros DNS falsificados no cache de um servidor DNS recursivo. Atacantes exploram isso para redirecionar usuários de sites legítimos para sites maliciosos (phishing, distribuição de malware) ou para causar negação de serviço, pois o servidor DNS passará a retornar IPs incorretos.
32. **Como DHCP evita conflitos de endereçamento:** O DHCP centraliza a gestão de IPs, atribuindo endereços de um pool de forma dinâmica via "leases" (concessões), por um tempo limitado. Permite exclusões de faixas de IP e reservas para MACs específicos, e algumas implementações verificam a disponibilidade do IP antes de atribuí-lo, prevenindo colisões.
33. **O que é DNS e sua principal função:** O DNS (Domain Name System) é um sistema hierárquico que atua como a "agenda telefônica da internet". Sua principal função é **traduzir nomes de domínio legíveis por humanos (ex: `www.google.com`) em endereços IP numéricos (ex: `172.217.160.142`)**, que são compreendidos por computadores.
34. **Principal função do protocolo DHCP:** A principal função do DHCP (Dynamic Host Configuration Protocol) é **automatizar a configuração de parâmetros de rede IP** para dispositivos cliente, principalmente a atribuição dinâmica de endereços IP.

35. **Por que DHCP usa UDP e não TCP:** DHCP usa UDP porque os clientes que estão buscando um IP ainda não possuem um, o que impede o estabelecimento de uma conexão TCP (que exige IPs de origem e destino). Além disso, o UDP suporta broadcast (necessário para o cliente encontrar o servidor) e é mais leve e rápido, ideal para as mensagens curtas e pontuais do DHCP.
36. **Reserva de IP no DHCP e uso:** Uma reserva de IP é uma configuração no servidor DHCP que garante que um dispositivo específico (identificado por seu endereço MAC) sempre receba o mesmo endereço IP. É usada para dispositivos que precisam de um IP consistente, como servidores, impressoras de rede, roteadores e dispositivos IoT específicos, sem a necessidade de configuração manual estática.
37. **Servidor DNS autoritativo vs. recursivo:**
    - **Autoritativo:** Detém os registros originais e definitivos para um domínio específico (é a fonte da verdade). Ele responde apenas por suas próprias zonas.
    - **Recursivo:** Atua como um intermediário para os clientes, realizando todo o processo de consulta (perguntando aos servidores raiz, TLD e autoritativos) em nome do cliente para encontrar o IP, e então armazena o resultado em cache.
38. **Função do agente relay em redes segmentadas com DHCP:** Em redes com múltiplos segmentos (sub-redes) separados por roteadores (que não encaminham broadcasts DHCP), o agente relay (geralmente o próprio roteador) intercepta as mensagens DHCP broadcast do cliente e as converte em mensagens unicast, encaminhando-as ao servidor DHCP localizado em outra sub-rede. Ele também retransmite as respostas do servidor de volta para o cliente.
39. **Zona reversa no DNS e utilidade:** Uma zona reversa mapeia endereços IP de volta para nomes de domínio (o inverso de uma zona normal), usando registros PTR. É utilizada principalmente para **anti-spam em servidores de e-mail** (verificam se o IP do remetente corresponde ao seu domínio), para leitura de logs e para verificações de segurança/confiança.
40. **Quatro passos do processo de atribuição de IP via DHCP (DORA):**
    1. **DHCPDISCOVER:** Cliente (sem IP) envia um broadcast para encontrar servidores.
    2. **DHCPOFFER:** Servidor DHCP responde com uma oferta de IP e configurações.
    3. **DHCPREQUEST:** Cliente solicita o IP oferecido (ou para renovar uma concessão).
    4. **DHCPACK:** Servidor confirma a atribuição do IP e configurações.
41. **Parâmetros além do IP configurados pelo DHCP:** Além do IP e do tempo de concessão, o DHCP pode configurar a máscara de sub-rede, gateway padrão, servidores DNS, nome de domínio, servidores WINS, servidores NTP, rotas estáticas, servidores TFTP e nomes de arquivos de boot, entre outros.
42. **Importância do tempo de leasing em redes com dispositivos móveis:** O tempo de leasing (concessão) é crucial para redes com dispositivos móveis (smartphones, notebooks) porque permite a **reutilização eficiente de endereços IP**. Dispositivos móveis se conectam e desconectam frequentemente; um tempo de leasing curto libera rapidamente IPs de dispositivos inativos para serem reatribuídos a outros, otimizando o pool de endereços e a escalabilidade da rede.
43. **O que é DNSSEC e como contribui para a segurança do DNS:** DNSSEC (Domain Name System Security Extensions) são extensões que adicionam

autenticação criptográfica aos dados DNS. Ele contribui para a segurança protegendo contra poluição de cache e manipulações de dados, verificando a origem e a integridade das respostas DNS através de assinaturas digitais e uma cadeia de confiança.

44. **Diferença entre modo automático, dinâmico e manual no DHCP:**
    - **Automático:** O servidor atribui um IP permanente ao cliente de um pool, que não é reutilizado.
    - **Dinâmico:** O servidor atribui um IP por um tempo limitado (lease time); o IP pode ser reutilizado após a expiração. (Mais comum).
    - **Manual (Reserva):** O administrador configura o servidor para sempre atribuir um IP fixo e específico a um cliente com um dado endereço MAC.
45. **Processo de resolução de nomes pelo DNS:**
    1. O cliente solicita um nome de domínio ao seu resolvedor DNS local (cache).
    2. Se não estiver no cache, o resolvedor local consulta um servidor raiz.
    3. O servidor raiz indica o servidor TLD (`.com`, `.org`, etc.).
    4. O resolvedor local consulta o servidor TLD, que indica o servidor DNS autoritativo para o domínio específico.
    5. O resolvedor local consulta o servidor autoritativo, que finalmente retorna o endereço IP.
    6. O resolvedor local armazena o IP em cache e o envia de volta ao cliente.
46. **O que acontece se o tempo de concessão expira e o cliente ainda está na rede:** O cliente tenta renovar sua concessão em 50% e 87.5% do tempo de lease. Se não conseguir renovar até 100% do tempo, a concessão expira, o cliente perde seu endereço IP e não consegue mais se comunicar na rede. Ele então tenta iniciar o processo DHCP novamente (DHCPDISCOVER) para obter um novo IP.
47. **Três tipos de registros de recursos (RR) no DNS e funções:**
    - **A Record (Address):** Mapeia um nome de domínio para um endereço IPv4.
    - **MX Record (Mail Exchange):** Especifica o(s) servidor(es) de e-mail responsável(is) por um domínio.
    - **NS Record (Name Server):** Identifica os servidores DNS autoritativos para um domínio.
48. **Concessão (lease) de IP no contexto do DHCP:** É a atribuição temporária de um endereço IP e outras configurações de rede de um servidor DHCP para um dispositivo cliente por um período de tempo determinado.
49. **Mensagens DHCPDISCOVER, DHCPOFFER, DHCPREQUEST e DHCPACK:**
    - **DHCPDISCOVER:** O cliente envia para encontrar servidores DHCP.
    - **DHCPOFFER:** O servidor oferece um IP e configurações ao cliente.
    - **DHCPREQUEST:** O cliente solicita o IP oferecido (ou para renovar).
    - **DHCPACK:** O servidor confirma a atribuição do IP e a concessão.
50. **Consulta DNS iterativa e recursiva:**
    - **Recursiva:** O resolvedor DNS local se compromete a fornecer a resposta completa ao cliente, consultando outros servidores se necessário. (Cliente -> Resolvedor Local)
    - **Iterativa:** Um servidor DNS responde com a melhor informação que tem, indicando outro servidor para o qual o solicitante deve direcionar a

próxima consulta, sem resolver a requisição por completo. (Resolvedor Local -> Outros Servidores DNS)

Espero que este resumo detalhado ajude em seus estudos!