Estratégia Concursos

Aula 00

TRT-ES 17ª Região (Analista Judiciário - Tecnologia da Informação) Arquitetura de Computadores

Autor:
Evandro Dalla Vecchia Pereira

15 de Fevereiro de 2023

Índice

SISTEMAS OPERACIONAIS

Quando falamos em sistema operacional logo pensamos em Windows, Linux, Android etc. Esses são apenas alguns exemplos dos existentes na atualidade, mas o que é um sistema operacional (S.O.)?

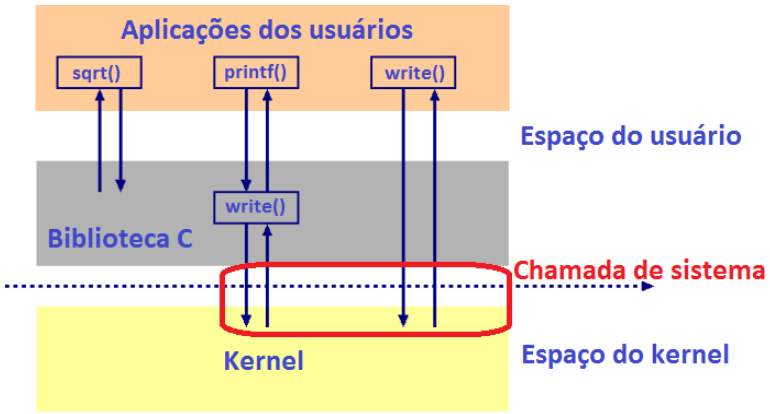
Podemos dizer que basicamente um S.O. possui duas funções:

- Apresentar ao usuário uma máquina estendida ou máquina virtual, afinal de contas "alguém" tem que "conversar" com o hardware;
- Gerenciar um sistema complexo: processadores, memórias, discos, dispositivos de E/S, arquivos etc.

De uma forma mais ampla, algumas funções do S.O. são:

- Permitir aos programas o armazenamento e a obtenção de informações;
- Controlar o fluxo de dados entre os componentes do computador;
- Responder a erros e a pedidos do usuário.
- Impor o escalonamento entre programas que solicitam recursos (memória, disco, entre outros);
- Etc.

Ok, se o S.O. faz o "meio de campo" entre o hardware e os programas do usuário, como um programador faria um acesso a um disco, por exemplo, para ler ou escrever em um arquivo? Para isso existem as **chamadas de sistema** (*system calls*) que são "instruções estendidas", abstraindo do programador os detalhes de "baixo nível" e garantindo que o programador não faça alguma "bobagem". Vejamos um exemplo para a programação em C:



Primeiro vamos ver o conceito de **kernel**: é o **núcleo** do sistema operacional, com um controle total de tudo relacionado ao sistema. O *kernel* é um dos primeiros programas a ser carregado durante a inicialização e assim que começa a ser executado inicia um processo de detecção de todo o hardware necessário para que ocorra um bom funcionamento do computador.

Uma simples alteração da versão do *kernel* pode ser suficiente para resolver problemas de hardware, além de compatibilidade no computador. Além disso, o *kernel* opera solicitações de entrada/ saída de software e gerência memória, aparelhos periféricos, entre outros.

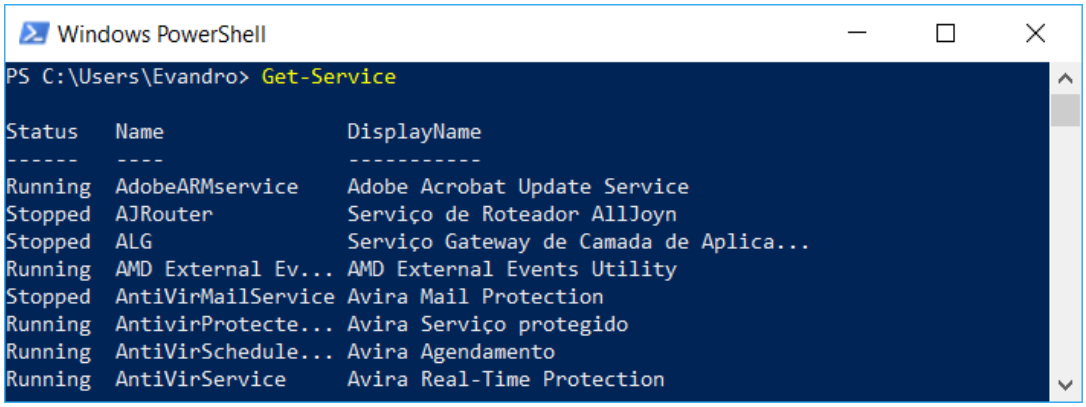
Sabendo disso tudo, vamos voltar à figura. Mesmo para quem não programa em C, os comandos mostrados são intuitivos: sqrt() – *square root* (raiz quadrada) – não precisa realizar uma chamada de sistema, pois recebe um valor e retorna sua raiz quadrada. Mas para escrever em um arquivo – comando write() – é necessário acessar alguma mídia (HD, SSD, pen drive, entre outros) e, para isso, é necessário que o S.O. entre em modo *kernel*. Mas o que é isso? Veremos...

Modo kernel: uma aplicação pode executar instruções não privilegiadas e privilegiadas, ou seja: instruções que oferecem risco ao sistema, ex.: instruções que acessam dados no disco.

Modo usuário: uma aplicação só pode executar instruções não privilegiadas (instruções que não oferecem riscos ao sistema).

Já falamos do núcleo, agora vamos para a "beirada"...a interface do usuário com o sistema operacional. Na atualidade é muito comum a utilização de GUI (*Graphical User Interface*), ou seja, o usuário apenas clica em janelas, ícones, entre outros elementos para interagir com S.O. Mas ainda existe o **shell**, um processo que lê o teclado e espera por comandos, interpreta-os e passa seus parâmetros ao S.O. Por isso também é conhecido como **interpretador de comandos**.

Antigamente os sistemas operacionais tinham como interface única o shell, mas com o tempo a interface gráfica dominou o mercado (mas ainda existem sistemas operacionais em que predomina o shell). Abaixo um exemplo de shell, o PowerShell:



De uma forma bem específica, encontramos na literatura especializada quatro tipos de gerenciamento realizados por um S.O.:

- Gerência de processos (unidade básica de trabalho do sistema operacional), o que inclui a sua criação, sua exclusão e o fornecimento de mecanismos para a sua comunicação e sincronização;

- Gerência de memória, controlando que partes estão sendo usadas e por quem. Além disso, é responsável pela alocação e liberação dinâmica de seu espaço;
- Gerência de dispositivos de entrada/saída (E/S) ligados ao computador, o que inclui o envio de sinais que informam as ações que o usuário espera que o dispositivo realize, o tratamento das interrupções e erros gerados pelos dispositivos, entre outros;

- Gerência de armazenamento, que inclui o fornecimento do sistema de arquivos para a representação de arquivos e diretórios e o gerenciamento do espaço em dispositivos de armazenamento de dados (HD, SSD, pen drive, entre outros).

Tipos de Kernel e outras classificações

Em relação à arquitetura do *kernel*, o sistema operacional pode ser classificado como monolítico, *microkernel* ou híbrido, conforme veremos a seguir.

Monolítico: os controladores de dispositivos e as extensões de núcleo são executadas no espaço de núcleo, com acesso completo ao hardware. Como todos os módulos são executados em um mesmo espaço de endereçamento, se houver ocorrência de erro em um desses espaços, todo o sistema pode ser afetado. Há um único arquivo objeto, sendo que toda rotina fica visível às demais. Há uma chamada de núcleo (chamada de supervisor) para trocar o modo usuário/núcleo. Alguns exemplos: Linux, BSD e MS-DOS.

Microkernel (micronúcleo): conforme o nome já indica, é um núcleo de tamanho bastante reduzido e, por esse motivo, ele executa o mínimo de processos possível no espaço do *Kernel*. Alguns desses processos são executados no espaço do usuário. Com um *kernel* micronúcleo, se ocorrer um erro, basta reiniciar o serviço que apresentou o problema. Com isso, evita-se que todo o sistema seja derrubado (como ocorre com o *Kernel* monolítico). Alguns exemplos: AIX, Minix e GNU Hurd.

Híbrido: funciona como um meio-termo, se comparado a sistemas monolíticos e de micronúcleos. O híbrido combina a estabilidade e a segurança do *microkernel* com o desempenho do monolítico. O *kernel* híbrido é semelhante a um micronúcleo, mas tem um código (“não essencial”) no espaço do núcleo para que as operações executadas sejam mais rápidas. Alguns exemplos: AmigaOS, Android, Macintosh e Windows.

Sistemas exonúcleos: fornecem um clone do computador real para cada usuário, mas com um subconjunto dos recursos. Por exemplo: uma VM recebe os blocos do disco 0 a 2047 e outra do 2048 a 4095. Na camada inferior existe um programa chamado exonúcleo (*exokernel*).

A ideia é permitir que o desenvolvedor tome todas as decisões relativas ao rendimento do hardware. Os exonúcleos são extremamente pequenos, já que sua função se limita à proteção e à multiplexação dos recursos. Os desenvolvimentos de núcleos clássicos (monolítico ou micronúcleo) abstraem o hardware, deixando esses detalhes “de baixo nível” aos controladores do dispositivo. Nos sistemas clássicos, usando a memória física, ninguém poderá afirmar qual é sua real localização, por exemplo.

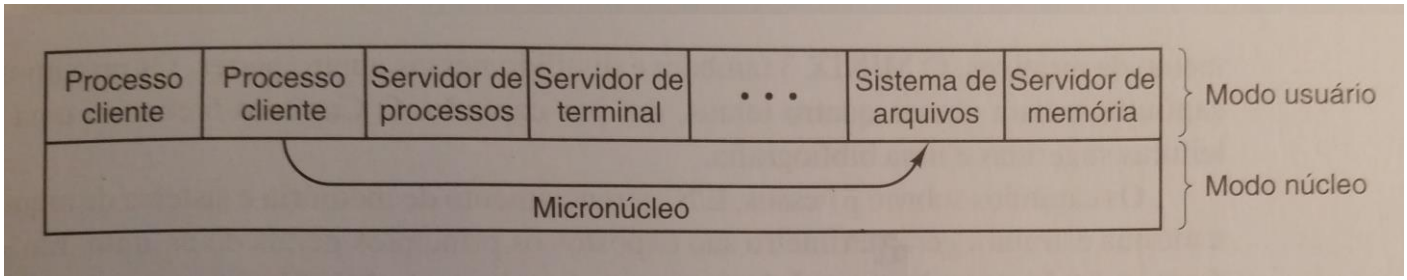
A finalidade de um exonúcleo é permitir uma aplicação que solicite uma região específica da memória, simplesmente assegurar que os recursos pedidos estão disponíveis e que o programa tem direito a acessá-los. Pelo fato do **exonúcleo proporcionar uma interface de baixo nível ao hardware**, carecendo de todas as funções de alto nível dos outros sistemas operacionais, ele é complementado por uma biblioteca de sistema operacional. Esta biblioteca se comunica com o exonúcleo subjacente e facilita aos programadores de aplicativos com funções que são comuns em outros sistemas operacionais.

Sistemas em camadas: como o nome sugere, é construído sobre uma hierarquia de camadas. O primeiro sistema desenvolvido dessa maneira foi o sistema criado no *Technische Hogeschool Eindhoven* (THE), na Holanda. Tratava-se de um sistema de lote simples para um computador holandês (o Electrologica X8). O S.O. possuía seis camadas:

Camada	Função
5	Operador
4	Programas de usuário
3	Gerenciamento de E/S
2	Comunicação operador-processo
1	Gerenciamento de memória e tambor
0	Alocação do processador e multiprogramação

Máquinas virtuais: Idênticas ao hardware verdadeiro, podendo executar qualquer sistema operacional. É um assunto denso que merece uma aula específica.

Modelo cliente-servidor: possui um núcleo mínimo (*microkernel*), sendo que a maior parte das funções do S.O. ficam em processos de usuário. O cliente obtém o serviço através de mensagens para os processos servidores:



Fonte: Tanenbaum; Woodhull.

QUESTÕES COMENTADAS

1. (FCC/DPE-SP - 2010) NÃO é uma função do sistema operacional:

A) Permitir aos programas armazenar e obter informações.

B) Controlar o fluxo de dados entre os componentes do computador.

C) Responder a erros e a pedidos do usuário.

D) Impor escalonamento entre programas que solicitam recursos.

E) Gerenciar apenas a base de dados.

Comentários:

De uma forma mais ampla, algumas funções do S.O. são:

- Permitir aos programas o armazenamento e a obtenção de informações;
- Controlar o fluxo de dados entre os componentes do computador;
- Responder a erros e a pedidos do usuário.
- Impor o escalonamento entre programas que solicitam recursos (memória, disco, entre outros).

A alternativa E está bem longe de ser uma função do S.O., ainda mais que expressa "APENAS" e ainda uma "BASE DE DADOS" genérica. Portanto, a **alternativa E** está correta e é o gabarito da questão.

2. (FCC/TRT16 - 2014) Um Sistema Operacional (SO) realiza o gerenciamento

..I.. , que inclui o fornecimento do sistema de arquivos para a representação de arquivos e diretórios e o gerenciamento do espaço em dispositivos com grande capacidade de armazenamento de dados.

..II.. , que são a unidade básica de trabalho do SO. Isso inclui a sua criação, sua exclusão e o fornecimento de mecanismos para a sua comunicação e sincronização.

..III.. , controlando que partes estão sendo usadas e por quem. Além disso, é responsável pela alocação e liberação dinâmica de seu espaço.

As lacunas I, II e III são, correta e respectivamente, preenchidas por:

A) de armazenamento - de processos - de memória

B) em memória secundária - de serviços - em memória principal

C) de arquivos - de barramentos - de discos

D) de discos - de threads - de cache

E) de I/O - de tempos de CPU - de RAM

Comentários:

De uma forma bem específica, encontramos na literatura especializada quatro tipos de gerenciamento realizados por um S.O. (que serão abordados em tópicos específicos nesta aula):

- Gerência de processos (unidade básica de trabalho do sistema operacional), o que inclui a sua criação, sua exclusão e o fornecimento de mecanismos para a sua comunicação e sincronização;

- Gerência de memória, controlando que partes estão sendo usadas e por quem. Além disso, é responsável pela alocação e liberação dinâmica de seu espaço;

- Gerência de dispositivos de entrada/saída (E/S) ligados ao computador, o que inclui o envio de sinais que informam as ações que o usuário espera que o dispositivo realize, o tratamento das interrupções e erros gerados pelos dispositivos, entre outros;

- Gerência de armazenamento, que inclui o fornecimento do sistema de arquivos para a representação de arquivos e diretórios e o gerenciamento do espaço em dispositivos de armazenamento de dados (HD, SSD, pen drive, entre outros).

Portanto, a **alternativa A** está correta e é o gabarito da questão.

3. (FCC/TRF3 - 2016) Um Técnico Judiciário de TI do TRF3, ao estudar os princípios dos sistemas operacionais, teve sua atenção voltada ao processo que perfaz a interface do usuário com o sistema operacional. Observou que este processo lê o teclado a espera de comandos, interpreta-os e passa seus parâmetros ao sistema operacional. Entendeu, com isto, que serviços como login/logout, manipulação de arquivos e execução de programas são, portanto, solicitados por meio do interpretador de comandos ou

A) Kernel.

B) System Calls.

C) Shell.

D) Cache.

E) Host.

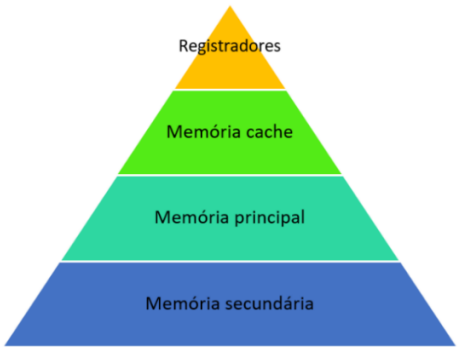
Comentários:

Na atualidade é muito comum a utilização de GUI (Graphical User Interface), ou seja, o usuário apenas clica em janelas, ícones, entre outros elementos para interagir com S.O. Mas ainda existe o shell, um processo que lê o teclado e espera por comandos, interpreta-os e passa seus parâmetros ao S.O. Por isso também é conhecido como interpretador de comandos. Portanto, a **alternativa C** está correta e é o gabarito da questão.

4. (Quadrix/COFECI - 2017) O gerenciador de memória é a parte do sistema operacional que gerencia, parcialmente, a hierarquia de memórias.

Comentários:

Vamos ver uma figura simples sobre a hierarquia de memória:



O gerenciador de memória gerencia parcialmente porque não gerencia a hierarquia completa, como por exemplo a memória secundária. Para gravar/ler de um HD é necessário que sinais sejam enviados (gerência de E/S) e que o sistema de arquivos entre em ação, para definir onde está um arquivo (quais blocos do HD), o tamanho do bloco etc. Portanto, a questão está **correta**.

5. (UPENET-IAUPE/UPE - 2017) O software responsável pelo gerenciamento dos recursos do hardware para o usuário, a fim de que os softwares aplicativos não tenham que interagir diretamente com os dispositivos periféricos, é definido como

A) compilador.

B) driver.

C) sistema operacional.

D) drive.

E) controlador.

Comentários:

Seria muito complicado exigir que todo programador tivesse conhecimento do hardware a ser utilizado. Também seria uma programação muito mais complexa e demorada! Podemos dizer que basicamente um S.O. possui duas funções:

- Apresentar ao usuário uma máquina estendida ou máquina virtual, afinal de contas “alguém” tem que “conversar” com o hardware;
- Gerenciar um sistema complexo: processadores, memórias, discos, dispositivos de E/S, arquivos etc.

Portanto, a **alternativa C** está correta e é o gabarito da questão.

6. (IESES/IGP-SC - 2017) Considere as afirmativas abaixo referentes as funções que são de responsabilidade de um Sistema Operacional Moderno:

I. Controlar os dispositivos de entrada/saída.

II. Efetuar o gerenciamento de programas em execução.

III. Oferecer mecanismos de proteção aos recursos básicos do computador.

Estão corretas as afirmativas:

A) I e III

B) II e III

C) I, II e III

D) I e II

Comentários:

(I) Faz parte da gerência de E/S; (II) Gerência de processos; (III) Como a questão fala em sistema operacional moderno, o oferecimento de mecanismos de proteção poderia ser considerado certo. Aí depende de qual a fonte consultada para elaborar a questão. De qualquer forma a **questão foi anulada (eu marcaria a alternativa C)**, com a justificativa de que esse assunto não estava no edital.

7. (COSEAC/UFF - 2019) Os sistemas operacionais normalmente possuem uma casca, que é a parte visível com a qual o usuário entra em contato, e outra parte interna. Essas duas partes são conhecidas, respectivamente, por:

A) API e shell.

B) GUI e cluster.

C) shell e kernel.

D) kernel e CPU.

E) buffers e spooling.

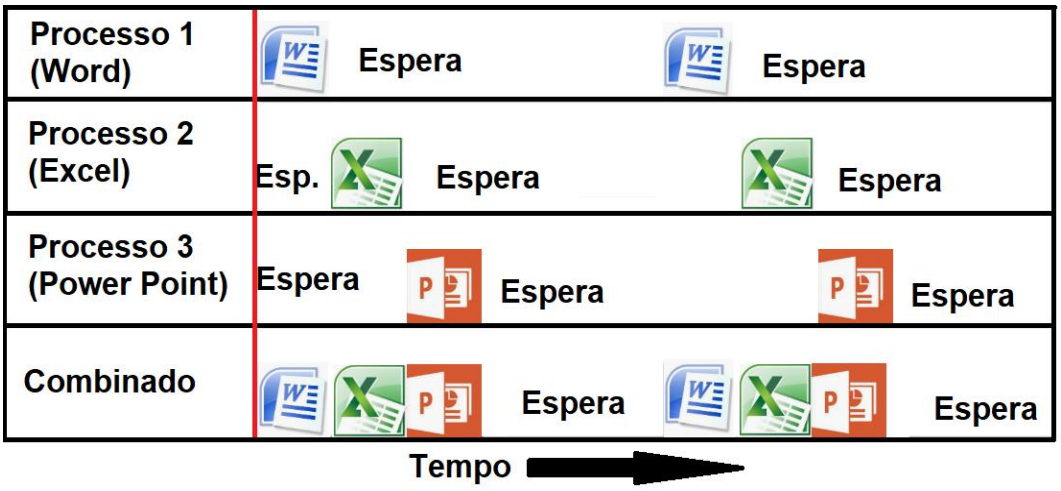
Comentários:

"Casca" poderíamos interpretar como quem faz a interface de quem está "fora" com o núcleo e isso é papel do shell (ou alguma interface gráfica, claro). A parte interna podemos interpretar como o núcleo, ou seja, o kernel. Portanto, a **alternativa C** está correta e é o gabarito da questão.

GERÊNCIA DE PROCESSOS

Um computador e sistema operacional modernos permitem que diversas atividades sejam realizadas ao mesmo tempo, mesmo que a máquina possua apenas um processador! Mas como é possível fazer com que 2 ou mais programas sejam executados ao mesmo tempo com apenas 1 processador? Aí lembramos da ideia de pseudoparalelismo que existe com o escalonamento de uso do processador, sendo que cada processo recebe algum tempo, de acordo com alguma política ou algoritmo.

Mas o que é **processo**? É simplesmente um programa em execução, incluindo os valores correntes dos registradores (PC, IR, entre outros) e das variáveis (ex.: soma, total, em um programa que realiza cálculos). Cada processo pensa que está "sozinho no mundo" e executa em uma CPU virtual, mas sabemos que na prática a CPU alterna de um processo para outro. Essa alternância é conhecida como **multiprogramação**. Veja na figura abaixo, onde é mostrada a execução de 3 processos em uma única CPU:



Na figura vemos apenas um processo de cada programa, mas pode haver N processos do Word, do Excel, do Power Point, ou qualquer outro. O limite se resume a recursos (memória, limitação do sistema operacional, entre outros).

Quando a CPU muda o processo, ocorre uma **troca de contexto (chaveamento ou mudança de contexto)**. Trata-se de um processo computacional de armazenar e restaurar o estado (contexto) de uma CPU de forma que múltiplos processos possam compartilhar essa CPU (multiprogramação). É garantido que quando o contexto anterior armazenado for restaurado, o ponto de execução volte ao mesmo estado que foi deixado durante o armazenamento.

Imagine que você tenha carregado um documento do Word, em seguida uma planilha do Excel e uma apresentação do Power Point, todos os arquivos muito grandes. Suponha que tenham sido carregadas 50 páginas do Word, 50 do Excel, 100 slides do Power Point, depois a CPU volta para o processo do Word e carrega as restantes, e assim por diante. Geralmente tudo ocorre muito rápido e temos a impressão de um paralelismo de verdade, a não ser que seu processador seja muito antigo e você queira carregar dezenas ou centenas de processos que exijam um certo desempenho.

Criação, Término e Hierarquia de Processos

Existem quatro eventos principais que acarretam a **criação de processos**, a saber:

- Inicialização do sistema;
- Realização de uma chamada de sistema por um processo em execução para a criação de um processo, ex.: fork() no Linux;
- Um pedido de um usuário para a criação de um novo processo, ex.: clicar duas vezes para abrir um documento do Word;
- Início de uma tarefa em lote (computadores de grande porte).

Quando o S.O. é inicializado, geralmente vários processos são criados, alguns de primeiro plano (**foreground**), que interagem com os usuários; e outros em segundo plano (**background**), que não estão associados a algum usuário, mas possuem alguma função específica (ex.: processo que aguarda a solicitação por impressão). Processos em background também são chamados de **daemons**.

"Dentro" de um processo, além do código do programa em si, podemos encontrar:

- Contexto de software: informações como nome do processo, identificador (PID), proprietário (*owner* - UID), prioridade de execução, entre outros;
- Contexto de hardware: valores de registradores;
- Espaço de endereçamento: espaço reservado para os dados do processo (ex.: texto editado através do Word).

Na figura abaixo podemos ver um esquema de um processo:

Existe uma estrutura de dados no núcleo do sistema operacional que serve para armazenar a informação necessária para tratar um determinado processo. Trata-se do **Bloco de Controle do Processo** (PCB - *Proccess Control Block*). Como o PCB possui informações críticas do processo ele deve ficar armazenado em uma área da memória protegida do acesso de usuários. Geralmente as informações contidas em um PCB incluem:

- Identificador do processo (PID);

- Registradores da CPU;
- O espaço de endereçamento do processo;
- Prioridade do processo;
- Entre outras.

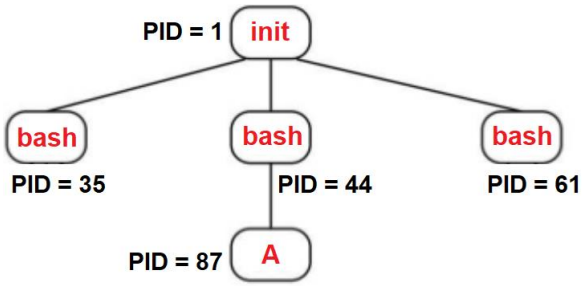
E a estrutura de dados responsável por habilitar o sistema operacional a localizar e acessar rapidamente o bloco de controle de processo (PCB) de um processo é denominada **Tabela de Processo**.

Nada dura para sempre, em algum momento ocorre o **término do processo**, normalmente devido a alguma das seguintes situações:

- Término normal (voluntário);
- Término por erro (voluntário), ex.: divisão por zero;
- Erro fatal (involuntário), ex.: programa recebe como parâmetro o nome de um arquivo que não existe;
- Eliminado por outro processo (involuntário), ex.: comando kill (Linux).

A maioria dos processos termina porque já cumpriu sua tarefa. Se você executa um programa que recebe 10 valores, realiza uma bateria de cálculos, mostra um resultado e finaliza, neste momento ele envia uma chamada de sistema para avisar ao S.O. que terminou (ex.: exit). Os programas aceitam término voluntário, geralmente com algum menu ou combinação de teclas (ex.: ALT + F4, no Windows).

Em relação à **hierarquia de processos** existe apenas 1 pai e 0 ou mais filhos. Por exemplo, no Linux o processo "init" (PID = 1) é o primeiro a ser executado, logo após o carregamento do Kernel. A função dele é controlar todos os outros processos que são executados no computador. Digamos que a partir dele sejam abertos 3 *shells* (*bash*) e a partir de um deles seja executado um programa "A". Abstraindo a existência de outros processos, a hierarquia descrita ficaria assim (PIDs inventados, com exceção do init):

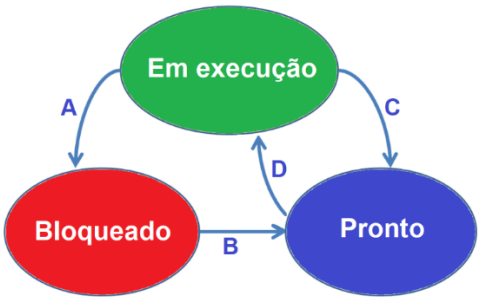


Estados de um Processo

De uma forma resumida, um processo pode estar em um dos seguintes estados:

- **Executando**: realmente utilizando o processador;
- **Pronto**: temporariamente parado, mas pronto (aguardando) para utilizar o processador;
- **Bloqueado** (ou espera): incapaz de executar até que algum evento ocorra (ex.: terminar de receber os dados de um arquivo que está armazenado em um HD).

As quatro trocas possíveis entre os estados são apresentadas na figura abaixo:



Quando um processo está em execução (no processador) e é necessário realizar alguma atividade que não dependa do processador (ex.: aguardar os dados do arquivo "teste.txt" solicitados ao HD), não tem o porquê ele ainda utilizar o processador! Seria um desperdício! Então o processo é bloqueado **(A)** e um outro processo que estiver pronto (e for a sua vez) deverá ocupar a CPU **(D)**. Depois que o HD realizar a leitura do arquivo e colocar os dados em um buffer, é enviado um aviso para dizer que está tudo pronto, é só buscar no buffer! Nesse momento o processo deixa de estar bloqueado e fica pronto (entra na fila) para utilizar a CPU novamente **(B)**.

Os processos que estão prontos ficam no aguardo de sua vez para utilizar a CPU **(D)** e os que estão em execução, não solicitam nenhuma atividade que ensejaria um bloqueio, mas que utilizam a CPU até um determinado limite de tempo, perdem o uso da CPU e voltam para o "grupo" dos prontos **(C)** . Dessa forma os processos ficam mudando seus estados até finalizarem.

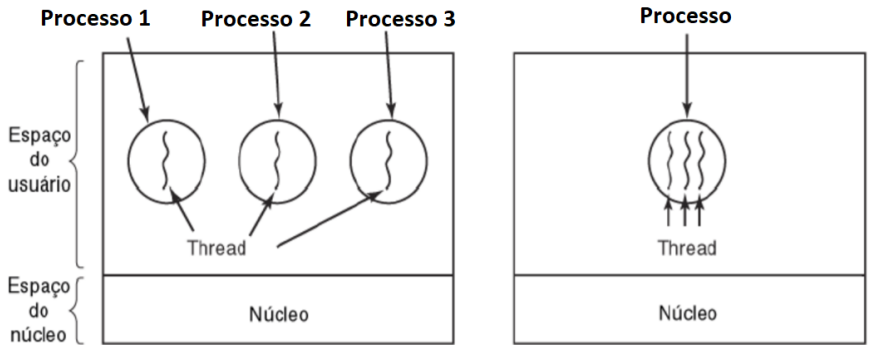
Dependendo do **comportamento do processo**, ele ficará mais tempo bloqueado ou pronto. Processos que passam a maior parte computando (**CPU-bound**) tendem a ir poucas vezes para o estado bloqueado (ou talvez nunca!), pois utilizam muito o processador e quando a fatia de tempo termina vão para o estado "pronto". Já os processos que esperam muito por E/S (**I/O-bound**) tendem a ficarem bloqueados seguidamente, indo depois para o estado "pronto" e somente depois poderem utilizar a CPU. Um exemplo de um processo *CPU-bound* é a execução de um filme (fica mais de hora rodando). Um exemplo de processo *I/O-bound* é um chat, pois aguarda a digitação do teclado a todo momento.

Threads

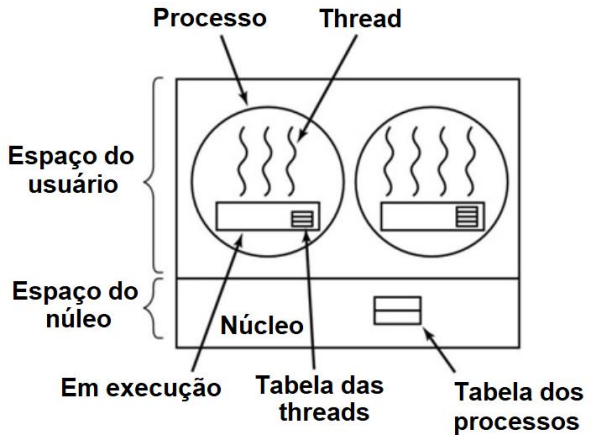
Um processo "tradicional" possui um espaço de endereçamento e um fluxo de controle (execução do código). Porém, há situações em que se deseja ter mais de um fluxo de controle e execução no mesmo processo, executando quase em paralelo. Esses fluxos são chamados **threads** (ou processos leves). Resumindo: *threads* de um mesmo processo compartilham a mesma seção de código na memória.

Imagine um editor de texto, que possui inúmeras funcionalidades: contador de palavras, contador de páginas, correção ortográfica instantânea, entre outras. Cada uma delas geralmente é implementada em uma *thread*, então a cada digitação elas verificam e a quantidade de palavras aumentou/diminuiu (e atualiza essa informação na tela), se a quantidade de páginas foi alterada, se a palavra digitada está correta (após consultar um arquivo de dicionário), e assim por diante.

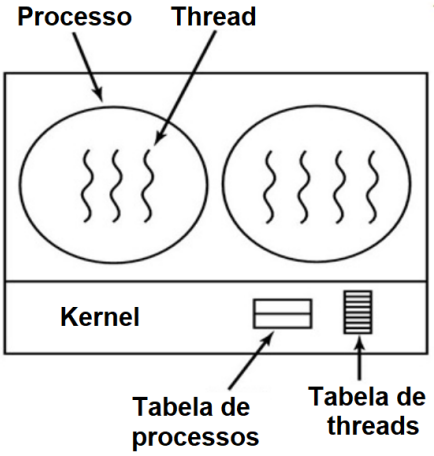
Se não fosse assim, vários processos deveriam ser carregados (o que seria mais "pesado") ou a execução de todo o código na sequência para verificar todas as funcionalidades. Abaixo uma figura para mostrar 3 processos contendo uma única *thread* (esquerda) e um único processo com 3 *threads* (direita).



Geralmente as *threads* são divididas em duas categorias: thread ao nível do usuário e thread ao nível do *kernel* (núcleo). Quando falamos em *thread* ao nível de usuário queremos dizer que o controle das *threads*, o escalonamento entre elas, fica a cargo do espaço do usuário. Na figura a seguir podemos ver que a tabelas dos processos fica no *kernel*, mas as tabelas das *threads* de cada processo ficam nos próprios processos.



Quando falamos em *thread* ao nível do *kernel* queremos dizer que o controle das *threads* fica a cargo do *kernel*, ou seja, tanto a tabela de processos quanto a tabela de *threads* ficam no *kernel*:



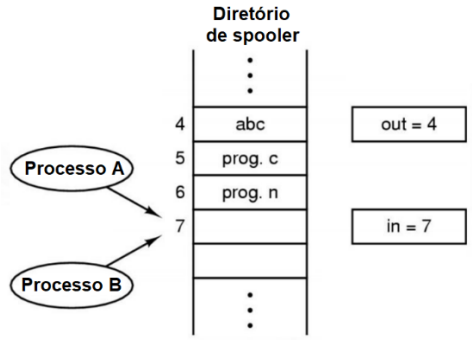
Comunicação entre Processos

Os processos frequentemente necessitam se comunicar. Um exemplo simples de entender é quando um usuário utiliza uma sequência de comandos no *shell* com o *pipe* separando cada um deles. Isso significa que a saída de um é a entrada do seguinte:

$ ls | grep x | sort -r | tee arquivo_saida

No exemplo acima, o resultado do “ls” é a entrada para o comando “grep x”. O resultado é a entrada para “sort -r” e o resultado deste último é a entrada para “tee arquivo_saida”.

Condições de corrida: alguns processos que trabalham juntos podem **compartilhar algum armazenamento comum**, ou seja, cada processo pode ler e escrever nesse local. Esse armazenamento compartilhado pode estar na memória principal ou pode ser um arquivo. Vamos utilizar como exemplo o diretório de *spooler*, onde os arquivos são colocados para que o *daemon* de impressora verifique de tempos em tempos o que deve ser impresso e mover a fila. Imagine a situação em que dois processos (A e B) quase que simultaneamente decidem enviar para o diretório de *spooler* o arquivo a ser impresso:



Seções críticas: para evitar as condições de corrida ou outras situações que envolvem memória compartilhada ou arquivo compartilhado deve-se encontrar uma maneira de proibir que mais de um processo leia e modifique dados compartilhados ao mesmo tempo. Ou seja, é necessária uma **exclusão mútua**. O problema mostrado na figura anterior ocorreu porque o processo B começou a utilizar uma das variáveis compartilhadas antes que o processo A tivesse terminado de trabalhar com ela.

Nem sempre o sistema acessa alguma região compartilhada, mas quando isso ocorre ele está acessando uma **região crítica** ou **seção crítica**. São necessárias quatro condições válidas para termos uma boa solução para esse problema:

1. Dois processos não podem estar ao mesmo tempo dentro de uma seção crítica;
2. Nenhuma suposição pode ser realizada sobre as velocidades ou sobre o número de processadores;
3. Nenhum processo que está sendo executado fora de uma seção crítica pode bloquear outros processos;
4. Nenhum processo deve ter que esperar eternamente para entrar em uma seção crítica.

Semáforo: é um tipo de variável que pode ser verificada e alterada em instruções atômicas, ou seja, sem possibilidades de interrupções. Esse tipo de variável é empregado em tarefas como o compartilhamento de recursos entre processos. É uma variável do tipo inteiro que possui o valor 0 quando não tem nenhum sinal a despertar, ou um valor positivo quando um ou mais sinais para despertar estiverem pendentes (usamos o termo “despertar”, pois um fluxo de execução é “colocado para dormir” quando tenta entrar em uma região crítica que já está ocupada).

Existem as operações **down** e **up** (ou **sleep** e **wakeup**). A operação *down* verifica se o valor é maior que 0. Se for, ele decrementa um e continua. Se o valor for 0, o processo (ou a *thread*) é “colocado para dormir” (bloqueado) sem completar a operação *down*. É garantido que iniciada uma operação de semáforo, nenhum

outro processo pode acessar o semáforo até que a operação tenha terminado ou sido bloqueada (ação atômica). Isso evita as condições de corrida.

Vamos imaginar a seguinte situação: há um semáforo que está com o valor 0, ou seja, já tem processo(s) ocupando a região crítica. Enquanto isso, outros 3 chegam a essa região e "tentam entrar", ficando bloqueados. Ao sair da região crítica, é aplicada a operação *up* (valor do semáforo passa para 1) e um dos três processos pode entrar (aleatório, fila ou outra maneira de escolher). Quando tal processo entrar é aplicada a operação *down* (semáforo passa a ser 0). Resumindo: o valor do semáforo diz quantos processos podem entrar!

Mutex: versão simplificada do semáforo, quando não for necessário "contar", utilizando-se apenas os estados "livre" ou "ocupado". Consequentemente é necessário apenas um bit para representá-lo, mas na prática geralmente é utilizado um valor inteiro. Quando um processo precisa entrar na região crítica, ele chama *mutex_lock*, que será bem-sucedida se a região estiver livre. Caso contrário, o processo ficará bloqueado até que o processo que estiver na região crítica saia (*mutex_unlock*).

Escalonamento de Processos

O escalonamento de tarefas é uma atividade de processamento realizada pela CPU de um computador. Essa atividade permite executar de forma mais eficiente os processos considerados prioritários para o sistema operacional. Existem situações em que pode ocorrer o escalonamento, mas existem duas em que ele DEVE ocorrer:

- Quando um processo termina: não tem motivo para ocupar tempo de processamento após concluída sua execução;
- Quando um processo é bloqueado em uma operação de E/S ou em um semáforo: não tem motivo para ficar aguardando "dentro" da CPU por um retorno do dispositivo ou a saída do semáforo.

Relacionado ao assunto, uma palavra que muitas vezes aparece em sua prova é a **preempção**, que é o ato do S.O. utilizar as interrupções do relógio para **retirar a CPU do processo em execução**. Ou seja, o processo não pode monopolizar o processador! Quando o sistema não for preemptivo pode ocorrer uma situação denominada **starvation** (inanição). Traduzindo, *starvation* quer dizer "morrer de fome", ou seja, aquele processo que nunca consegue chegar ao processador, fica eternamente aguardando, sempre tem alguém que "fura a fila".

As **categorias de algoritmos de escalonamento** são:

- **Lote**: geralmente utilizado em computadores de grande porte, sem usuários esperando uma resposta rápida. São aceitáveis algoritmos não-preemptivos ou algoritmos preemptivos com longos períodos de tempo para cada processo. Isso **reduz as trocas de processo**, o que melhora o desempenho;

- **Interativo**: em um ambiente em que os usuários interagem a **preempção é fundamental**! Assim não ocorre uma monopolização da CPU por um processo, negando serviço a outros. Uso típico em computadores de propósito geral, ex.: PC para jogos, edição de textos, navegação na Internet etc.;

- **Tempo real**: por incrível que pareça a preempção pode ser desnecessária, pois os processos sabem que não podem ser executados por longos períodos de tempo. Normalmente os processos realizam

suas atividades e são rapidamente bloqueados. Um sistema de tempo real executa apenas o que é necessário, ex.: radar que registra a velocidade do veículo e fotografa se ultrapassar um limite.

Escalonamento em Sistemas de Lote

Alguns algoritmos são convenientes tanto para sistemas de lote como nos interativos, mas vamos estudar agora apenas aqueles mais utilizados apenas para sistemas de lote.

First-Come First-Served (FCFS): o nome do algoritmo já diz tudo ("O primeiro a chegar é o primeiro a ser atendido"), ou seja, os processos recebem tempo de CPU na ordem em que solicitam. Basicamente há uma fila única e os processos vão "entrando" nela, os processos prontos são executados na ordem da fila e quando um processo é bloqueado (aguardando uma entrada, por exemplo), ele retorna para o fim da fila quando estiver pronto novamente. Ok, mas e se um processo começar a ser executado, nunca for bloqueado e tiver uma estimativa de execução de 4h, ele monopolizará o processador esse tempo todo? Isso mesmo! Esse algoritmo é do tipo **não-preemptivo**, uma característica de sistemas de lote.

Shortest-Job First (SJF) ("Tarefa mais curta primeiro"): mais um não-preemptivo, mas esse algoritmo presume que os tempos de execução são conhecidos previamente. Imagine uma situação em que os *jobs* (tarefas) de uma empresa são executados há muitos anos e já se sabe que os *jobs* do tipo A levam 2 minutos, do tipo B 6 minutos e do tipo C 4 minutos. Os *jobs* são agendados à tarde para serem executados às 8h do dia seguinte. Vamos ver como ficaria o escalonamento para a execução no dia seguinte dos *jobs* que foram adicionados na ordem B1, A1, C1, A2 e C2:

Shortest Remaining Time Next (SRT) ("Menor tempo de execução restante"): versão preemptiva do algoritmo SJF. Nesse algoritmo o escalonador sempre escolhe o *job* com tempo de execução mais curto (obviamente o tempo precisa ser conhecido previamente). Quando um novo *job* chega, seu tempo é comparado com o que falta para concluir o *job* corrente. Se o novo precisar de menos tempo, o processo corrente é suspenso e o novo é iniciado. Esse esquema permite que *jobs* novos curtos tenham uma boa prioridade.

Escalonamento em Sistemas Interativos

Round-robin: é realizado um rodízio entre os processos, sendo que a cada processo é atribuído um intervalo de tempo (**quantum**), durante o qual ele pode ser executado. Se ao final do *quantum* o processo ainda estiver em execução é realizada a **preempção** da CPU e esta é alocada a um outro processo. Obviamente que se o processo tiver terminado antes do *quantum* ter expirado ou se tiver sido bloqueado, a troca da CPU é realizada neste momento.

Um ponto interessante é a definição da duração do *quantum*, pois trocar de um processo para outro exige uma quantidade de tempo para salvar e carregar registradores e mapas de memória, atualizar tabelas e listas etc. Vamos supor que esse **chaveamento de contexto** demore 1ms e que o *quantum* seja de 9ms. Nesse caso 10% da CPU são desperdiçados em sobrecarga administrativa.

Escalonamento por prioridade: cada processo recebe uma prioridade e o processo pronto, com a maior prioridade, tem a permissão para executar. Por exemplo, um *daemon* que envia um e-mail em segundo plano deveria ter uma prioridade mais baixa do que a de um processo responsável por uma videoconferência ao vivo.

Para evitar que processos com alta prioridade monopolizem o uso da CPU, o escalonador pode diminuir a prioridade do processo em execução em cada interrupção de relógio. Se sua prioridade ficar abaixo da do próximo processo com maior prioridade, deve ocorrer um chaveamento de processo. Uma alternativa é ar um *quantum* a cada processo e, quando esse *quantum* esgotar, é dada a chance de execução ao próximo processo com maior prioridade.

Escalonamento por múltiplas filas: basicamente os processos são agrupados em classes e cada classe possui uma prioridade: 1 *quantum*, 2 *quanta* (esse é o plural de *quantum*), 4, 8, e assim por diante. Na medida em que um processo utiliza todos os *quanta* destinados a ele, em seguida ele é movido para a classe seguinte (a que recebe mais quanta). Por exemplo, se um processo precisa de 50 *quanta*, primeiro ele recebe 1, depois 2, 4, 8, 16, 32. No total o processo teria recebido 63 *quanta*, ou seja, na última classe em que esteve (recebendo 32 *quanta*), na verdade só utilizou 19 e parou a execução, liberando o processador.

Escalonamento garantido: a ideia é “fazer promessas realistas aos usuários sobre o desempenho, e cumpri-las!”. Como assim? Se houver N usuários trabalhando em uma CPU, cada um recebe cerca de 1/N do poder dessa CPU! Essa é uma promessa realista simples de cumprir. Para que essa promessa realmente seja cumprida, o sistema deve monitorar quanto da CPU cada processo de cada usuário recebeu e dar mais poder de CPU para quem teve menos. Assim ocorre uma compensação e aos poucos todos usuários terão um uso da CPU similar.

Escalonamento por sorteio: o escalonamento garantido parece uma boa, mas é difícil implementar! Uma solução mais simples é o algoritmo de escalonamento por sorteio, o qual fornece “bilhetes de loteria” para os vários recursos do sistema (ex.: tempo de CPU). Quando uma decisão de escalonamento tiver que ser tomada, um “bilhete” é sorteado e o ganhador recebe o recurso. Os processos mais importantes podem receber “bilhetes” extras, aumentando as chances de serem sorteados.

QUESTÕES COMENTADAS

1. **(ESAF/CGU - 2008) Analise as seguintes afirmações, levando em conta as chamadas de sistemas usadas com semáforos, e assinale a opção verdadeira.**

   I. A chamada de sistema UP adiciona uma unidade ao valor corrente de um semáforo.

   II. Se o valor do semáforo é zero, uma chamada de sistema DOWN não será completada e o processo será suspenso.

   III. Quando um processo inicia a execução de uma chamada de sistema UP ou DOWN, nenhum outro processo terá acesso ao semáforo até que o processo complete a execução ou seja suspenso.

A) Apenas I e II são verdadeiras.

B) Apenas I e III são verdadeiras.

C) Apenas II e III são verdadeiras.

D) I, II e III são verdadeiras.

E) I, II e III são falsas.

Comentários:

Em um semáforo existem as operações down e up (ou sleep e wakeup). A operação down verifica se o valor é maior que 0. Se for, ele decrementa um e continua. Se o valor for 0, o processo (ou a thread) é "colocado para dormir" (bloqueado) sem completar a operação down. É garantido que iniciada uma operação de semáforo, nenhum outro processo pode acessar o semáforo até que a operação tenha terminado ou sido bloqueada (ação atômica). Isso evita as condições de corrida. Portanto, a **alternativa D** está correta e é o gabarito da questão.

2. **(CESPE/ABIN - 2010) No contexto de sistemas operacionais, semáforos são tipos de variáveis que podem ser verificadas e alteradas em instruções atômicas, ou seja, sem possibilidades de interrupções. Esse tipo de variável é empregado em tarefas como o compartilhamento de recursos entre processos.**

Comentários:

Como vimos há pouco, um semáforo possui as operações atômicas up e down. Esse tipo de variável (semáforo) é ideal para controlar o acesso a regiões críticas, evitando que mais processos (ou threads) que determinado limite consigam acessar tal região ao mesmo tempo. Isso é o ideal no compartilhamento de recursos, evitando problemas como, por exemplo, o compartilhamento de uma impressora (imagine cada processo imprimindo uma página de seu documento ao mesmo tempo!). Portanto, a questão está **correta**.

3. **(IADES/PG-DF - 2011) O escalonamento de tarefas é uma atividade de processamento realizada pela CPU de um computador. Esta atividade permite executar de forma mais eficiente os processos**

considerados prioritários para o sistema operacional. Assinale a alternativa que apresenta o escalonamento de tarefas em um computador, utilizado como servidor de arquivos de uma rede.

A) O escalonamento garantido busca atender a demanda da rede, priorizando ações de leitura

B) O algoritmo de escalonamento FIFO (First In, First Out) atua na gravação de arquivos em disco, implementando o conceito de pilha de escalonamento.

C) Os algoritmos de escalonamento preemptivos devem permitir que um processo seja interrompido durante sua execução.

D) O algoritmo de escalonamento de múltiplas filas permite o acesso simultâneo a arquivos e banco de dados disponibilizados na rede.

E) O escalonador de longo prazo seleciona os processos na interface de rede, dando prioridade às ações de I/O (Input/Output).

Comentários:

(A) O escalonamento garantido é aquele que busca fazer promessas realistas aos usuários sobre o desempenho; (B) O algoritmo de escalonamento FIFO (First In, First Out) é uma fila, o primeiro a entrar é o primeiro a sair; (C) Exato! Os preemptivos permitem que um processo seja interrompido durante sua execução, evitando uma monopolização do uso da CPU; (D) O algoritmo de escalonamento de múltiplas filas atua com classes de prioridade; (E) Não chegamos a ver na aula. Trata-se de um escalonador que seleciona os processos que estão na memória secundária e que serão levados para a memória principal. Portanto, a **alternativa C** está correta e é o gabarito da questão.

4. (CESPE/STF - 2013) Em um algoritmo de escalonamento FIFO, os processos são executados na mesma ordem que chegam à fila. Quando um processo do tipo cpu-bound está na frente da fila, todos os processos devem esperá-lo terminar seu ciclo de processador.

Comentários:

Um processo CPU-bound é aquele que usa muito o processador, ou seja, raramente é bloqueado. Por isso os demais devem esperar acabar seu ciclo de uso do processador (quantum). Portanto, a questão está **correta**.

5. (FCC/Câmara Municipal-SP - 2014) No escalonamento usando o algoritmo Round-Robin,

A) o escalonador seleciona o processo à espera com o menor tempo de execução estimado até a conclusão, reduzindo o tempo médio de espera, mas aumentando a variância dos tempos de resposta.

B) processos são despachados na ordem FIFO (First-in-First-Out), mas recebem uma quantidade limitada de tempo de processador denominada quantum.

C) a prioridade de cada processo é uma função não apenas do seu tempo de serviço, mas também do tempo que passou esperando pelo serviço.

D) o escalonador ajusta dinamicamente o comportamento do processo, de tal forma que o próximo processo a obter o processador seja aquele que chegar à frente da fila de nível mais alto, que não estiver vazia, na rede de filas.

E) o processo que tem o prazo de execução mais curto é favorecido, medindo a diferença entre o tempo que um processo requer para finalizar e o tempo restante até atingir o seu prazo final.

Comentários:

Algoritmo Round-robin: é realizado um rodízio entre os processos, sendo que a cada processo é atribuído um intervalo de tempo (quantum), durante o qual ele pode ser executado. Se ao final do quantum o processo ainda estiver em execução é realizada a preempção da CPU e esta é alocada a um outro processo. Obviamente que se o processo tiver terminado antes do quantum ter expirado ou se tiver sido bloqueado, a troca da CPU é realizada neste momento. Portanto, a **alternativa B** está correta e é o gabarito da questão.

6. (CESPE/TRE-PI - 2016) A respeito das características do algoritmo de escalonamento SPF (shortest process first), assinale a opção correta.

A) Os processos são executados na ordem em que chegam à fila de espera e executados até o final, sem nenhum evento preemptivo.

B) No SPF, um processo recém-chegado e em espera, cujo tempo estimado de execução completa seja menor, provoca a preempção de um processo em execução que apresente tempo estimado de execução completa maior.

C) O SPF favorece processos longos em detrimento dos mais curtos. Estes, ao chegarem à fila de espera, são obrigados a aguardar a conclusão dos processos longos que já estiverem em andamento, para, então, entrar em execução.

D) Os processos são despachados na ordem em que são colocados em espera e recebem uma quantidade limitada de tempo do processador para execução; além disso, são interrompidos caso sua execução não se conclua dentro do intervalo de tempo delimitado.

E) O escalonador seleciona o processo que estiver à espera e possuir o menor tempo de execução estimado e o coloca em execução até a sua conclusão.

Comentários:

O nome mudou um pouquinho, mas a ideia é a mesma:

Shortest-Job First (SJF) (“Tarefa mais curta primeiro”): algoritmo não-preemptivo, presume que os tempos de execução sejam conhecidos previamente. Imagine uma situação em que os jobs (tarefas) de uma empresa são executados há muitos anos e já se sabe que os jobs do tipo A levam 2 minutos, do tipo B 6 minutos e do tipo C 4 minutos. Os jobs são agendados à tarde para serem executados às 8h do dia seguinte. Vamos ver como ficaria o escalonamento para a execução no dia seguinte dos jobs (tarefas) que foram adicionados na ordem B1, A1, C1, A2 e C2:

Portanto, a **alternativa E** está correta e é o gabarito da questão.

7. (IESES/IGP-SC - 2017) Acerca da gerência de processos dos sistemas operacionais, assinale a alternativa correta:

A) Um conjunto de processos está em estado de deadlock quando todos os processos no conjunto estão esperando por um evento que só pode ser causado por outro processo do conjunto.

B) Em um escalonamento preemptivo, um processo só perde o processador se terminar ou entrar em estado de espera.

C) No algoritmo de escalonamento de processos Round Robin, o escalonador sempre escolhe para execução o processo com menor expectativa de tempo de processamento. Esse algoritmo baseia-se no fato de que privilegiando processos pequenos o tempo médio de espera decresce.

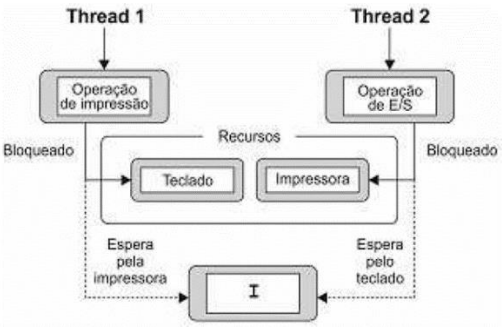
D) Starvation é uma situação que não pode ocorrer quando um sistema operacional provê prioridades a processos.

Comentários:

(A) Isso aí... fica um aguardando o outro liberar um recurso, que por sua vez aguarda um terceiro e por aí vai... (B) Em um escalonamento preemptivo um processo perde o processador se terminar, ficar bloqueado ou se terminar seu quantum; (C) No algoritmo de escalonamento de processos Round Robin ocorre um "rodízio", sendo que cada processo recebe a mesma quantidade de quantum em uma ordem FIFO; (D) Starvation justamente ocorre quando um sistema operacional provê prioridades a processos. Imagine o uso do recurso processador, sendo que um processo tenha a prioridade máxima e um outro tenha a mínima. O primeiro leve 4 dias para terminar a execução e antes disso outros processos "surgiram" com uma prioridade maior do que aquele que tem a mínima...quando ele terá acesso ao processador?

Portanto, a **alternativa A** está correta e é o gabarito da questão.

8. (FCC/DPE-RS - 2017) Considere a figura abaixo.



Do ponto de vista do sistema operacional, a situação indica que a caixa I deve ser preenchida com?

A) starvation.

B) multithreading.

C) superthreading.

D) deadlock.

E) hyperthreading.

Comentários:

Note que a thread 1 está realizando uma operação de impressão, bloqueando o teclado e está à espera da impressora. A thread 2 está realizando uma operação de E/S, bloqueando a impressora e à espera do teclado. Ou seja, nenhuma thread libera o recurso que está usando e cada uma quer um recurso que a outra possui. As duas ficarão “trancadas”, esperando... isso é um deadlock! Portanto, a **alternativa D** está correta e é o gabarito da questão.

9. (CONSULPLAN/TRE-RJ - 2017) Quando um processo aguarda por um recurso que nunca estará disponível ou mesmo um evento que não ocorrerá, acontece uma situação denominada deadlock (ou como alguns autores denominam: impasse ou adiamento indefinido). Para que um deadlock ocorra, quatro condições são necessárias. Uma delas tem a seguinte definição: "cada recurso só pode estar alocado a um único processo em um determinado instante". Assinale a alternativa que apresenta tal condição.

A) Espera circular.

B) Exclusão mútua.

C) Não-preempção.

D) Espera por recurso.

Comentários:

Ou um processo ou outro pode utilizar determinado recurso. O nome é bem intuitivo: exclusão mútua. Ou seja, se um processo A está utilizando um recurso R, um processo B não pode utilizar o recurso R ao mesmo tempo! Portanto, a **alternativa B** está correta e é o gabarito da questão.

10. (FCC/DPE-RS - 2017) Dentre as políticas de escalonamento de processos a seguir, a que apresenta maior probabilidade de ocasionar o starvation é a

A) Round Robin.

B) de tempo compartilhado.

C) First In First Out.

D) preemptiva.

E) não preemptiva.

Comentários:

Uma política de escalonamento não-preemptiva é aquela que um recurso não pode ser retirado de um processo, a não ser que "ele queira". Imagine um processo que comece a utilizar a CPU e fique utilizando por 1h. Além disso digamos que há uma certa prioridade entre os processos e existe uma demanda muito grande pelo uso da CPU. Pode ocorrer que um processo com prioridade baixa jamais seja executado! Ou seja, starvation! Portanto, a **alternativa E** está correta e é o gabarito da questão.

11. (CESPE/TRE-TO - 2017) Considerando o contexto de gerenciamento de processos dos sistemas operacionais, assinale a opção que apresenta a estrutura de dados responsável por habilitar o sistema operacional a localizar e acessar rapidamente o bloco de controle de processo (PCB) de um processo.

A) árvore de processos.

B) lista de bloqueados.

C) tabela de processo.

D) região de pilha.

E) lista de prontos.

Comentários:

Existe uma estrutura de dados no núcleo do sistema operacional que serve para armazenar a informação necessária para tratar um determinado processo. Trata-se do Bloco de Controle do Processo (PCB - Proccess Control Block). Como o PCB possui informações críticas do processo ele deve ficar armazenado em uma área da memória protegida do acesso de usuários. Geralmente as informações contidas em um PCB incluem:

- Identificador do processo (PID);
- Registradores da CPU;
- O espaço de endereçamento do processo;
- Prioridade do processo;
- Entre outras.

E a estrutura de dados responsável por habilitar o sistema operacional a localizar e acessar rapidamente o bloco de controle de processo (PCB) de um processo é denominada Tabela de Processo. Portanto, a **alternativa C** está correta e é o gabarito da questão.

12. (COMPERVE/UFRN - 2018) Sistemas operacionais modernos têm uma gerência de processos e de threads bem definida. Nesse contexto, é correto afirmar:

A) threads de um mesmo processo compartilham a mesma seção de código na memória.

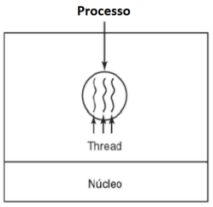
B) threads de um mesmo processo compartilham a mesma seção da pilha na memória.

C) todas as variáveis de uma thread são compartilhadas com as outras threads do mesmo processo.

D) todos os contextos de uma thread são compartilhados com as outras threads do mesmo processo.

Comentários:

A figura abaixo ajuda a entender. Note que existem 3 threads em um único processo, ou seja, 3 fluxos de execução do mesmo código. Por exemplo: um editor de textos com 3 threads é carregado na memória, uma responsável pela correção gramatical, uma pela contagem de palavras e a outra pelo resto (na "vida real" existem bem mais, claro). Então o processo é carregado na memória, no endereçamento X a Y, os fluxos de execução das 3 threads ocorrem “quase em paralelo” e todos eles estarão entre X e Y!



Portanto, a **alternativa A** está correta e é o gabarito da questão.

13. (FUNDEP/CODEMIG - 2018) O escalonamento de processos permite que um computador possa executar diversos programas em pseudoparalelismo, o que viabiliza aspectos como a multiprogramação. Qual entre os algoritmos de escalonamento a seguir seria mais adequado para sistemas de processamento em lote?

A) Primeiro a chegar, primeiro a ser servido.

B) Round Robin.

C) Escalonamento em duas fases.

D) Escalonamento por loteria.

Comentários:

Processamento em lote não possui interação, os processos são colocados lá e podem ser executados na sequência. Quando a natureza das tarefas for adequada a esse tipo de processamento, a grande vantagem é que não há perda de desempenho com chaveamentos de contexto! Desse modo, os algoritmos apresentados o mais adequado seria o “Primeiro a chegar, primeiro a ser servido”, pois segue a ordem de chegada e não tem chaveamento de processos com a designação de um quantum a cada um. Portanto, a **alternativa A** está correta e é o gabarito da questão.

14. (FUNDEP/CODEMIG - 2018) Um dos problemas relacionados ao gerenciamento de um sistema operacional diz respeito ao deadlock, o qual também pode ocorrer em banco de dados. Uma vez que gerenciar o deadlock pode ser uma tarefa que exija muito tempo do processador, a maior parte dos sistemas operacionais não trata desse problema. Em alguns sistemas críticos, entretanto, tratar os deadlocks é uma tarefa importante.

Qual entre as formas de tratamento a seguir se baseia em retirar o recurso do processo?

A) Através de preempção.

B) Revertendo o estado do processo.

C) Matando o processo.

D) Verificando a trajetória do processo.

Comentários:

O exemplo mais comum é retirar um processo do processador após um determinado tempo (quantum) e essa possibilidade de retirar um recurso do processo tem o nome de preempção. Portanto, a **alternativa A** está correta e é o gabarito da questão.

15. (Quadrix/CRA-PR - 2019) A respeito do gerenciamento de processos e do gerenciamento de memória nos sistemas operacionais, julgue o item.

Embora os sistemas operacionais executem diversas operações de processo, como, por exemplo, criar e suspender um processo, eles não são capazes de alterar a prioridade de um processo.

Comentários:

É possível alterar a prioridade de um processo sim! Inclusive um comando bastante utilizado no Linux para isso é o "renice" (alteração da prioridade pelo usuário). Se o usuário pode, é claro que o S.O. também pode! Portanto, a questão está **errada**.

16. (IF-PA/IF-PA - 2019) Em relação à gerência de processo, marque a alternativa CORRETA:

A) o processo é um programa em no estado de pronto.

B) os estados do processo são (execução, pronto, bloqueado ou espera).

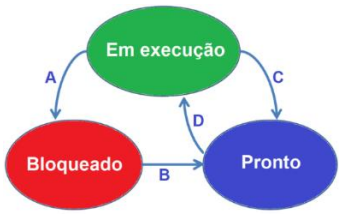
C) a thread permite que apenas uma execução ocorra no mesmo ambiente do processo.

D) os sinais são mecanismos que permitem notificar o sistema operacional de eventos gerados pelo processador.

E) escalonamento é a escolha do processo, em estado de execução.

Comentários:

(A) Processo é um programa em execução; (B) Melhor ver na figura abaixo; (C) Thread permite que fluxos de execução em um mesmo processo ocorram quase que em paralelo; (D) Interrupções são mecanismos que permitem notificar o sistema operacional de eventos gerados pelo processador; (E) escalonamento é a escolha do processo, em estado de pronto.



Obs.: Bloqueado ou em espera é a mesma coisa!

Portanto, a **alternativa B** está correta e é o gabarito da questão.

17. (CESPE/TJ-PA - 2020) No Linux, um processo, por si só, não é elegível para receber tempo de CPU. Essa ação depende, basicamente, do seu estado da execução. O processo está administrativamente proibido de executar no estado

A) pronto.

B) dormente.

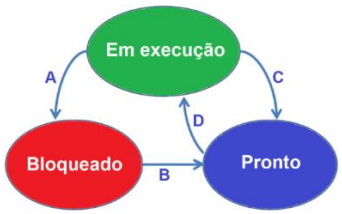
C) executável.

D) parado.

E) zumbi.

Comentários:

Para quem já estudou a aula de Linux fica mais fácil, mas digamos que você não tenha estudado e se depara com uma questão assim. E aí? Vamos ver a figura novamente e tentar por eliminação:



(A) Pronto - podemos ver que pode executar;

(B) Dormente - específico do Linux, não vimos, vamos ver a próxima;

(C) Executável - parece ser o equivalente ao pronto, pois é "executável", pode ser executado;

(D) Parado - parece o equivalente ao "bloqueado", pois está parado esperando algo (ex.: leitura de HD);

(E) Zumbi - específico do Linux, não vimos, vamos ver a próxima.

Com essa análise, mesmo sem ter estudado Linux, daria para marcar a alternativa D, não é? A **alternativa D** está correta e é o gabarito da questão.

GERÊNCIA DE ENTRADA/SAÍDA

Uma das principais funções do S.O. é controlar todos os dispositivos de E/S. Para isso são utilizados comandos que os dispositivos "entendam", interrupções e tratamento de erros. É importante também o fornecimento de uma interface simples e fácil de usar entre o sistema e os dispositivos.

De uma forma geral os dispositivos de E/S são divididos em duas categorias: **dispositivos de bloco** e **dispositivos de caractere**. O primeiro armazena informações em blocos de tamanho fixo (ex.: 512 bytes, 32KB, entre outros). É possível ler ou escrever cada bloco, independentemente dos outros e o principal exemplo são os discos. O sistema de arquivos trata somente com os dispositivos de bloco abstratos e a parte mais "baixo nível" fica sob a responsabilidade de um software denominado **driver de dispositivo**.

O dispositivo de caractere envia ou aceita um fluxo de caracteres, sem nenhuma estrutura de bloco. Ele não tem endereços (como ocorre com os blocos) e não possui operações de busca. Alguns exemplos são as impressoras, interfaces de rede e mouses.

Geralmente as unidades de E/S consistem em um componente mecânico e um eletrônico e é comum que estejam separadas em duas partes, o que permite um projeto mais modular. O componente eletrônico é o que chamamos de **controladora** ou o **adaptador** (pode ser uma placa ou "embutido" na placa-mãe, por exemplo). A controladora normalmente possui algum conector que permite a conexão com o dispositivo em si.

Técnicas de Gerenciamento de E/S

E/S mapeada em memória: cada controladora possui registradores (memória) utilizados para se comunicar com o processador. O sistema operacional escreve nesses registradores, "avisando" para o dispositivo enviar dados, aceitar dados, ligar-se ou desligar-se, etc. Note que a forma de comunicação é através da escrita em registradores. Lendo esses registradores, o S.O. pode saber qual o estado do dispositivo (ex.: pronto para aceitar um novo comando).

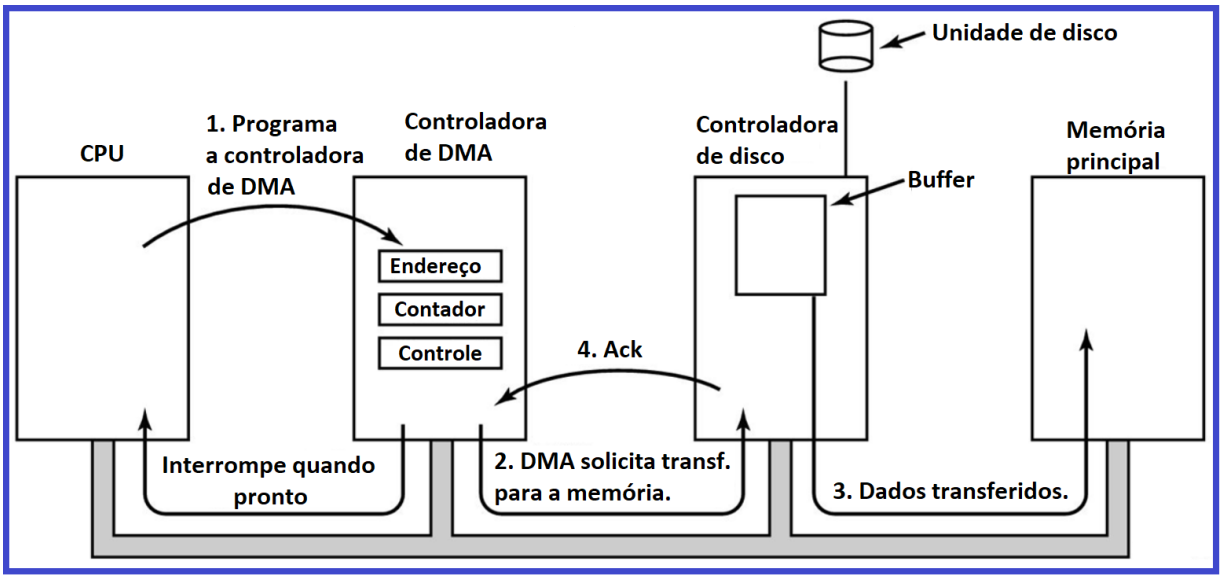
Geralmente os registradores das controladoras possuem bits de *status* que devem ser testados com a intenção de saber se uma operação de saída está concluída ou se novos dados estão disponíveis em um dispositivo de entrada. Para realizar essa consulta a CPU pode executar um laço para consultar os bits de *status*, o que é chamado de **E/S programada**, **consulta sequencial (polling)** ou **espera ativa (busy wait)**.

Interrupções: outra forma de avisar a CPU que os registradores estão prontos para serem lidos ou escritos. A maioria dos dispositivos de interface fornece uma saída logicamente igual ao bit de *status* "operação completa" ou "dados prontos" de um registrador, que é destinada a ser utilizada em uma das linhas de pedido de interrupção (IRQ – *Interrupt ReQuest*) do barramento do sistema. Mas o que é isso? Pense no seguinte: o dispositivo quer interromper o processador para que ele verifique o seu *status* e tome alguma atitude.

Então, quando uma operação termina, ela envia uma interrupção ao processador, que começa a executar a rotina de tratamento da interrupção. Tal código informa ao S.O. que a E/S está concluída, então o S.O. pode verificar os bits de *status* para saber se tudo ocorreu bem (se não ocorreu, pode tentar novamente). Em sistemas antigos a IRQ era configurada por meio de chaves (*jumpers*), mas na atualidade a tecnologia *Plug'n play* já resolve isso de forma automática, evitando conflitos.

Quando o processador estiver no modo usuário, instruções privilegiadas não podem ser executadas. Se houver tentativa de execução dessas instruções, o hardware automaticamente gerará uma interrupção e acionará o sistema operacional para lidar com a situação.

Acesso direto à memória (DMA): para deixar a CPU um pouco mais livre, um esquema denominado *Direct Memory Access* (DMA) é utilizado. Para isso deve haver uma controladora de DMA instalada (o que a maioria dos sistemas possui). Essa controladora pode estar integrada à controladora de disco ou separada. O mais comum é haver uma única controladora de DMA na placa-mãe para controlar as transferências dos diversos dispositivos de E/S. Não importa a localização física da controladora de DMA, ela tem acesso ao barramento do sistema independentemente do processador:



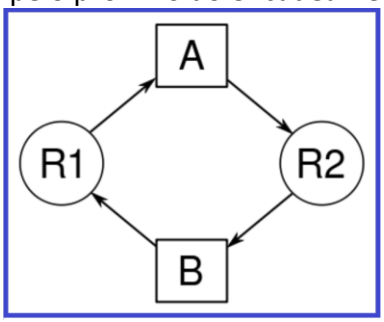
Impasse (Deadlock)

Um **impasse (deadlock)** ocorre quando um conjunto de processos está esperando por um evento que só pode ser causado por outro processo do conjunto. Ou seja, ficam todos "amarrados", sem poder continuar seus processamentos. Alguns exemplos de recursos que só podem ser utilizados por um processo por vez: unidades de fita e impressoras. Ou você acha que é possível dois processos escrevendo em uma fita ou mandando imprimir ao mesmo tempo? Ficaria uma bagunça!

Por isso os sistemas operacionais possuem a capacidade de garantir (por algum tempo) que um processo tenha o acesso exclusivo a determinados recursos, sejam de hardware ou de software. Vamos supor a seguinte situação: os processos A e B desejam digitalizar uma fotografia através de um *scanner* e em seguida gravar em um CD-R. Digamos que A tenha requisitado primeiro o *scanner* e ao mesmo tempo B tenha requisitado o gravador de CD-R. Em seguida A faz requisição do gravador de CD-R, mas ele está ocupado, pois B não o "largou". Assim fica o processo A aguardando o recurso gravador e B aguardando o recurso *scanner*. O que é isso? Um **deadlock**!

As **quatro condições** que devem ser verdadeiras para que ocorra um **deadlock** são:

1. Condição de exclusão mútua: cada recurso ou está correntemente atribuído a exatamente um processo ou está disponível;
2. Condição de posse e espera: os processos que possuem recursos garantidos anteriormente podem solicitar novos recursos (um acumulador de recursos!);
3. Ausência de preempção: os recursos garantidos não podem ser retirados à força de um processo;
4. Condição de espera circular: um encadeamento circular de dois ou mais processos, cada um esperando por um recurso mantido pelo próximo do encadeamento:



Algoritmo do avestruz: estratégia mais simples! O que um avestruz faz? Coloca a cabeça em um buraco... pois é, é isso mesmo! O algoritmo finge que nada aconteceu, simplesmente **ignora**. E, por incrível que pareça, é a estratégia adotada pela maioria dos sistemas operacionais (Windows, Linux, entre outros). Por quê? Porque o preço seria alto em realizar muitas restrições ao usuário. Melhor deixar acontecer...e se for o caso, o usuário reinicia a máquina ou mata um processo, caso ocorra algum travamento.

Detecção e recuperação: o sistema monitora as requisições e liberações de recursos. Sempre que um recurso é solicitado ou liberado, o grafo de recursos é atualizado e uma verificação para saber se há ciclos é realizada. Se houver um ciclo, um dos processos do ciclo é eliminado. Se não resolver o *deadlock*, outro será eliminado e assim por diante.

Prevenção de impasses: estratégia que impõe restrições sobre os processos a fim de tornar os impasses impossíveis. As quatro condições que já vimos dão um indício de possíveis soluções.

Evitação de impasses: não há a imposição de regras, mas sim uma análise criteriosa de cada pedido de recurso. Para isso algumas informações devem estar disponíveis antecipadamente.

QUESTÕES COMENTADAS

1. **(CESPE/TRT5 - 2008) Com relação ao sistema operacional Windows XP Professional, amplamente utilizado em determinados segmentos da área de tecnologia da informação, julgue os itens que se seguem.**

   Os drivers de dispositivos não podem ser alterados pelo administrador da estação porque o kernel assina digitalmente todos os drivers instalados automaticamente.

Comentários:

O administrador de um sistema operacional possui poder irrestrito, pode fazer qualquer coisa, incluindo a instalação ou atualização de um driver de um dispositivo! Portanto, a questão está **errada**.

2. **(CESPE/STF - 2008) Com relação a instalação de impressora local e em rede em sistemas Windows XP, julgue os itens seguintes.**

   No Windows XP, as impressoras laser que possuem drivers previamente instalados conectam-se periodicamente ao sítio do fabricante para modificar os drivers e os parâmetros de impressão.

Comentários:

Se o driver já está instalado, não há motivo para automaticamente se conectar ao sítio do fabricante. Tal busca pode ocorrer se alguém com privilégio de administrador solicitar para atualizar o driver e solicitar a busca na Internet, por exemplo. Portanto, a questão está **errada**.

3. **(FCC/TCM-PA - 2010) A comunicação de uma aplicação com o subsistema de entrada e saída de um sistema operacional é estabelecida por meio de**

A) shell.

B) device drivers.

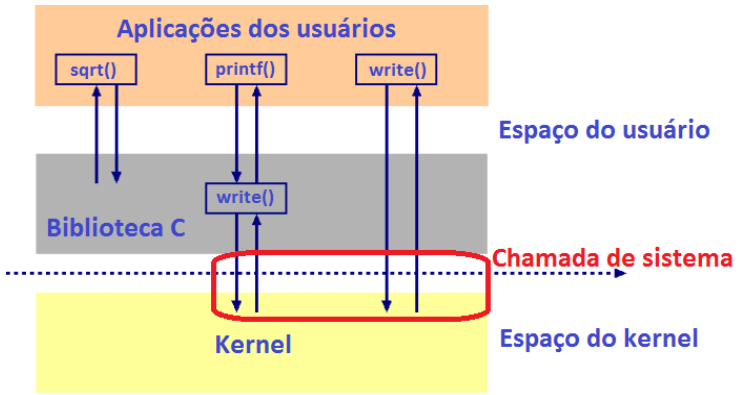
C) system calls.

D) scripting.

E) batch.

Comentários:

Vale a pena lembrar dessa figura:



Podemos ver que para acessar o kernel é necessário fazer uso de uma chamada de sistema (system call), senão um programador qualquer poderia fazer uma "bagunça". Sabemos também que o gerenciamento de E/S é realizado pelo kernel, então a comunicação de uma aplicação qualquer que queira fazer uso de algum dispositivo de E/S deve realizar uma chamada de sistema específica para esse fim.

Portanto, a **alternativa C** está correta e é o gabarito da questão.

4. (CESPE/INMETRO - 2010)

característica	processador pode executar outras instruções enquanto o módulo de E/S executa o seu trabalho	é preciso esperar que a operação de E/S termine
não há envolvimento do processador	técnica I	
processador controla operação de E/S	técnica II	técnica III

A partir das informações da tabela acima, que apresenta características de técnicas de gerenciamento de entrada e saída (E/S) de um sistema de computação, assinale a opção que nomeia corretamente as técnicas I, II e III, respectivamente.

A) E/S programada, E/S controlada por interrupção e acesso direto à memória (DMA).

B) E/S controlada por interrupção, E/S programada e acesso direto à memória (DMA).

C) E/S controlada por interrupção e acesso direto à memória (DMA) e E/S programada.

D) acesso direto à memória (DMA); E/S Programada, E/S Controlada por Interrupção.

E) acesso direto à memória (DMA), E/S controlada por interrupção e E/S programada.

Comentários:

Acesso direto à memória (DMA): para deixar a CPU um pouco mais livre, um esquema denominado Direct Memory Access (DMA) é utilizado. Para isso deve haver uma controladora de DMA instalada (o que a maioria dos sistemas possui). Essa controladora pode estar integrada à controladora de disco ou separada.

Interrupções: outra forma de avisar a CPU que os registradores estão prontos para serem lidos ou escritos. A maioria dos dispositivos de interface fornece uma saída logicamente igual ao bit de status “operação completa” ou “dados prontos” de um registrador, que é destinada a ser utilizada em uma das linhas de pedido de interrupção (IRQ – Interrupt ReQuest) do barramento do sistema. Mas o que é isso? Pense no seguinte: o dispositivo quer interromper o processador para que ele verifique o seu status e tome alguma atitude.

E/S mapeada em memória: cada controladora possui registradores (memória) utilizados para se comunicar com o processador. O sistema operacional escreve nesses registradores, “avisando” para o dispositivo enviar dados, aceitar dados, ligar-se ou desligar-se, etc. Note que a forma de comunicação é através da escrita em registradores. Lendo esses registradores, o S.O. pode saber qual o estado do dispositivo (ex.: pronto para aceitar um novo comando).

Portanto, a **alternativa E** está correta e é o gabarito da questão.

5. (CESGRANRIO/TRANSPETRO - 2011) Os Sistemas Operacionais estão sujeitos a um fenômeno denominado deadlock. Para que uma situação de deadlock seja criada, as seguintes condições devem acontecer simultaneamente

A) exclusão mútua (mutual exclusion), monopolização de recursos (hold and wait), não preempção (no preemption) e espera circular (circular wait).

B) exclusão mútua (mutual exclusion), transferência excessiva de páginas (thrashing), superposição de processos (process overlapping) e espera circular (circular wait).

C) transferência excessiva de páginas (thrashing), superposição de processos (process overlapping), monopolização de recursos (hold and wait) e não preempção (no preemption).

D) exclusão mútua (mutual exclusion), monopolização de recursos (hold and wait), superposição de processos (process overlapping) e falha de escalonamento (scheduling fail).

E) transferência excessiva de páginas (thrashing), não preempção (no preemption), espera circular (circular wait) e falha de escalonamento (scheduling fail).

Comentários:

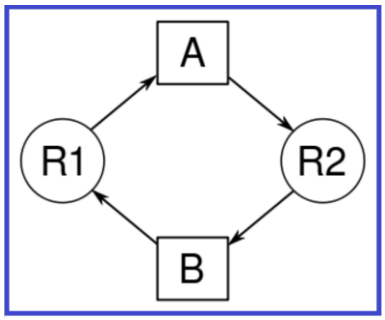
As quatro condições que devem ser verdadeiras para que ocorra um deadlock são:

1. Condição de exclusão mútua: cada recurso ou está correntemente atribuído a exatamente um processo ou está disponível;

2. Condição de posse e espera: os processos que possuem recursos garantidos anteriormente podem solicitar novos recursos (um acumulador de recursos!);

3. Ausência de preempção: os recursos garantidos não podem ser retirados à força de um processo;

4. Condição de espera circular: um encadeamento circular de dois ou mais processos, cada um esperando por um recurso mantido pelo próximo do encadeamento:



Portanto, a **alternativa A** está correta e é o gabarito da questão.

6. (FCC/TRT4 - 2011) Parte da definição da arquitetura de um computador é a especificação do seu sistema de entrada/saída. O esquema de E/S no qual a CPU gasta maior parte do seu tempo em loop, esperando o dispositivo ficar pronto é a E/S

A) por interrupção.

B) DMA.

C) programada.

D) através de canais.

E) por prioridade.

Comentários:

Geralmente os registradores das controladoras possuem bits de status que devem ser testados com a intenção de saber se uma operação de saída está concluída ou se novos dados estão disponíveis em um dispositivo de entrada. Para realizar essa consulta a CPU pode executar um laço para consultar os bits de status, o que é chamado de E/S programada, consulta sequencial (polling) ou espera ativa (busy wait). Portanto, a **alternativa C** está correta e é o gabarito da questão.

7. (CESPE/EBC - 2011) Para que os sistemas operacionais tenham acesso direto à memória, é necessário haver, no computador, recurso de hardware controlador DMA (direct memory access).

Comentários:

Acesso direto à memória (DMA): para deixar a CPU um pouco mais livre, um esquema denominado Direct Memory Access (DMA) é utilizado. Para isso deve haver uma controladora de DMA instalada (o que a maioria dos sistemas possui). Essa controladora pode estar integrada à controladora de disco ou separada. O mais comum é haver uma única controladora de DMA na placa-mãe para controlar as transferências dos diversos dispositivos de E/S. Não importa a localização física da controladora de DMA, ela tem acesso ao barramento do sistema independentemente do processador. Portanto, a questão está **correta**.

8. (CESPE/STF - 2013) No modo de operação do processador denominado modo usuário, instruções privilegiadas não podem ser executadas. Se houver tentativa de execução nesse caso, o hardware automaticamente gerará a interrupção e acionará o sistema operacional.

Comentários:

Relembrando:

Quando o processador estiver no modo usuário, instruções privilegiadas não podem ser executadas. Se houver tentativa de execução dessas instruções, o hardware automaticamente gerará uma interrupção e acionará o sistema operacional para lidar com a situação.

Portanto, a questão está **correta**.

9. (FUNDATEC/BRDE - 2015) Em relação ao acesso direto à memória no gerenciamento de E/S de um sistema operacional, assinale a alternativa correta.

A) A controladora de DMA não permite um acesso independente ao barramento do sistema, dependendo sempre da CPU.

B) A controladora de DMA só depende da CPU quando o processador possui mais de um núcleo.

C) A localização física da controladora de DMA deve ficar próxima à CPU para que o acesso ao barramento do sistema seja independente da CPU.

D) A localização física da controladora de DMA deve ficar distante da CPU para que o acesso ao barramento do sistema seja independente da CPU.

E) Não importa a localização física da controladora de DMA, ela sempre terá acesso ao barramento do sistema de forma independente da CPU.

Comentários:

Ver comentário de duas questões atrás 😊. Portanto, a **alternativa E** está correta e é o gabarito da questão.

10. (FUNRIO/IF-PA - 2016) Sistemas operacionais compartilham recursos, havendo a possibilidade de deadlocks. A literatura especializada indica quatro condições necessárias para que um deadlock ocorra. O algoritmo de Avestruz utiliza uma estratégia para lidar com deadlocks conhecida como

A) detectar.

B) detectar e recuperar.

C) evitar.

D) ignorar.

E) prevenir.

Comentários:

Algoritmo do avestruz: estratégia mais simples! O que um avestruz faz? Coloca a cabeça em um buraco... pois é, é isso mesmo! O algoritmo finge que nada aconteceu, simplesmente ignora. E, por incrível que pareça, é a estratégia adotada pela maioria dos sistemas operacionais (Windows, Linux, entre outros). Por quê? Porque o preço seria alto em realizar muitas restrições ao usuário. Melhor deixar acontecer...e se for o caso, o usuário reinicia a máquina ou mata um processo, caso ocorra algum travamento. Portanto, a **alternativa D** está correta e é o gabarito da questão.

11. (CESPE/MPE-PI - 2018) Julgue o item a seguir, acerca de sistemas operacionais.

Uma das causas de deadlocks em sistemas operacionais é a disputa por recursos do sistema que podem ser usados apenas por um processo de cada vez.

Comentários:

Essa é a primeira das quatro condições, a condição de exclusão mútua, a qual define que cada recurso ou está correntemente atribuído a exatamente um processo ou está disponível. Portanto, a questão está **correta**.

12. (COPESE/Câmara de Palmas-TO - 2018) Os sistemas operacionais modernos possuem diversos mecanismos para detecção e tratamento de situações de deadlock. Assinale a alternativa que NÃO apresenta um destes mecanismos.

A) O sistema irá escolher criteriosamente um processo e o terminará. Se a situação de deadlock não for resolvida, outros processos serão eliminados até que tudo esteja resolvido.

B) Os recursos são retirados dos processos e entregue aos outros até que o deadlock seja eliminado.

C) Os processos podem ser capazes de detectar um deadlock e voltar ao estado de execução anterior antes de pedir um recurso.

D) Um processo que detém um recurso fica esperando pela liberação de outro recurso, eliminando assim o deadlock.

Comentários:

A única alternativa que não fala em preempção (retirar algum recurso) ou voltar a um estado anterior (que não havia deadlock) é a letra D. Pelo contrário, ainda afirma que detém um recurso e FICA ESPERANDO pela liberação de outro recurso, o que não ajuda em nada a liminar a situação de impasse. Portanto, a **alternativa D** está correta e é o gabarito da questão.

13. (FCC/DPE-AM - 2018) Em um sistema operacional de computador, três processos estão na seguinte situação:

– o processo P1 tem a posse do recurso R1.

– o processo P2 tem a posse do recurso R2.

– o processo P3 tem a posse do recurso R3.

O processo P1 solicita o recurso R2, o processo P2 solicita o recurso R3, e o processo P3 solicita o recurso R1. Sobre essa situação, é correto afirmar que

A) não haverá deadlock, pois o processo P1 não solicitou o recurso R3.

B) tem-se uma condição de deadlock.

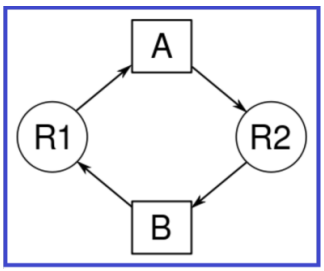
C) não haverá deadlock, pois o processo P3 não solicitou o recurso R2

D) só ocorrerá deadlock caso P1 solicite o recurso R3, P2 solicite o recurso R1 e P3 solicite o recurso R2.

E) não haverá deadlock, pois o processo P2 não solicitou o recurso R1.

Comentários:

Podemos ver que nenhum processo "larga" o recurso que possui. Além disso, cada processo solicita um recurso que está alocado por outro processo, de forma circular. Isso caracteriza um deadlock! Na figura abaixo vemos com 2 processos:



Portanto, a **alternativa B** está correta e é o gabarito da questão.

LISTA DE QUESTÕES

1. (FCC/DPE-SP - 2010) NÃO é uma função do sistema operacional:

A) Permitir aos programas armazenar e obter informações.

B) Controlar o fluxo de dados entre os componentes do computador.

C) Responder a erros e a pedidos do usuário.

D) Impor escalonamento entre programas que solicitam recursos.

E) Gerenciar apenas a base de dados.

1. (FCC/TRT16 - 2014) Um Sistema Operacional (SO) realiza o gerenciamento

..I.. , que inclui o fornecimento do sistema de arquivos para a representação de arquivos e diretórios e o gerenciamento do espaço em dispositivos com grande capacidade de armazenamento de dados.

..II.. , que são a unidade básica de trabalho do SO. Isso inclui a sua criação, sua exclusão e o fornecimento de mecanismos para a sua comunicação e sincronização.

..III.. , controlando que partes estão sendo usadas e por quem. Além disso, é responsável pela alocação e liberação dinâmica de seu espaço.

As lacunas I, II e III são, correta e respectivamente, preenchidas por:

A) de armazenamento - de processos - de memória

B) em memória secundária - de serviços - em memória principal

C) de arquivos - de barramentos - de discos

D) de discos - de threads - de cache

E) de I/O - de tempos de CPU - de RAM

2. (FCC/TRF3 - 2016) Um Técnico Judiciário de TI do TRF3, ao estudar os princípios dos sistemas operacionais, teve sua atenção voltada ao processo que perfaz a interface do usuário com o sistema operacional. Observou que este processo lê o teclado a espera de comandos, interpreta-os e passa seus parâmetros ao sistema operacional. Entendeu, com isto, que serviços como login/logout, manipulação de arquivos e execução de programas são, portanto, solicitados por meio do interpretador de comandos ou

A) Kernel.

B) System Calls.

C) Shell.

D) Cache.

E) Host.

3. (Quadrix/COFECI - 2017) O gerenciador de memória é a parte do sistema operacional que gerencia, parcialmente, a hierarquia de memórias.

4. (UPENET-IAUPE/UPE - 2017) O software responsável pelo gerenciamento dos recursos do hardware para o usuário, a fim de que os softwares aplicativos não tenham que interagir diretamente com os dispositivos periféricos, é definido como

A) compilador.

B) driver.

C) sistema operacional.

D) drive.

E) controlador.

5. (IESES/IGP-SC - 2017) Considere as afirmativas abaixo referentes as funções que são de responsabilidade de um Sistema Operacional Moderno:

I. Controlar os dispositivos de entrada/saída.

II. Efetuar o gerenciamento de programas em execução.

III. Oferecer mecanismos de proteção aos recursos básicos do computador.

Estão corretas as afirmativas:

A) I e III

B) II e III

C) I, II e III

D) I e II

6. (COSEAC/UFF - 2019) Os sistemas operacionais normalmente possuem uma casca, que é a parte visível com a qual o usuário entra em contato, e outra parte interna. Essas duas partes são conhecidas, respectivamente, por:

A) API e shell.

B) GUI e cluster.

C) shell e kernel.

D) kernel e CPU.

E) buffers e spooling.

7. (ESAF/CGU - 2008) Analise as seguintes afirmações, levando em conta as chamadas de sistemas usadas com semáforos, e assinale a opção verdadeira.

I. A chamada de sistema UP adiciona uma unidade ao valor corrente de um semáforo.

II. Se o valor do semáforo é zero, uma chamada de sistema DOWN não será completada e o processo será suspenso.

III. Quando um processo inicia a execução de uma chamada de sistema UP ou DOWN, nenhum outro processo terá acesso ao semáforo até que o processo complete a execução ou seja suspenso.

A) Apenas I e II são verdadeiras.

B) Apenas I e III são verdadeiras.

C) Apenas II e III são verdadeiras.

D) I, II e III são verdadeiras.

E) I, II e III são falsas.

8. (CESPE/ABIN - 2010) No contexto de sistemas operacionais, semáforos são tipos de variáveis que podem ser verificadas e alteradas em instruções atômicas, ou seja, sem possibilidades de interrupções. Esse tipo de variável é empregado em tarefas como o compartilhamento de recursos entre processos.

9. (IADES/PG-DF - 2011) O escalonamento de tarefas é uma atividade de processamento realizada pela CPU de um computador. Esta atividade permite executar de forma mais eficiente os processos considerados prioritários para o sistema operacional. Assinale a alternativa que apresenta o escalonamento de tarefas em um computador, utilizado como servidor de arquivos de uma rede.

A) O escalonamento garantido busca atender a demanda da rede, priorizando ações de leitura

B) O algoritmo de escalonamento FIFO (First In, First Out) atua na gravação de arquivos em disco, implementando o conceito de pilha de escalonamento.

C) Os algoritmos de escalonamento preemptivos devem permitir que um processo seja interrompido durante sua execução.

D) O algoritmo de escalonamento de múltiplas filas permite o acesso simultâneo a arquivos e banco de dados disponibilizados na rede.

E) O escalonador de longo prazo seleciona os processos na interface de rede, dando prioridade às ações de I/O (Input/Output).

10. (CESPE/STF - 2013) Em um algoritmo de escalonamento FIFO, os processos são executados na mesma ordem que chegam à fila. Quando um processo do tipo cpu-bound está na frente da fila, todos os processos devem esperá-lo terminar seu ciclo de processador.

11. (FCC/Câmara Municipal-SP - 2014) No escalonamento usando o algoritmo Round-Robin,

A) o escalonador seleciona o processo à espera com o menor tempo de execução estimado até a conclusão, reduzindo o tempo médio de espera, mas aumentando a variância dos tempos de resposta.

B) processos são despachados na ordem FIFO (First-in-First-Out), mas recebem uma quantidade limitada de tempo de processador denominada quantum.

C) a prioridade de cada processo é uma função não apenas do seu tempo de serviço, mas também do tempo que passou esperando pelo serviço.

D) o escalonador ajusta dinamicamente o comportamento do processo, de tal forma que o próximo processo a obter o processador seja aquele que chegar à frente da fila de nível mais alto, que não estiver vazia, na rede de filas.

E) o processo que tem o prazo de execução mais curto é favorecido, medindo a diferença entre o tempo que um processo requer para finalizar e o tempo restante até atingir o seu prazo final.

12. (CESPE/TRE-PI - 2016) A respeito das características do algoritmo de escalonamento SPF (shortest process first), assinale a opção correta.

A) Os processos são executados na ordem em que chegam à fila de espera e executados até o final, sem nenhum evento preemptivo.

B) No SPF, um processo recém-chegado e em espera, cujo tempo estimado de execução completa seja menor, provoca a preempção de um processo em execução que apresente tempo estimado de execução completa maior.

C) O SPF favorece processos longos em detrimento dos mais curtos. Estes, ao chegarem à fila de espera, são obrigados a aguardar a conclusão dos processos longos que já estiverem em andamento, para, então, entrar em execução.

D) Os processos são despachados na ordem em que são colocados em espera e recebem uma quantidade limitada de tempo do processador para execução; além disso, são interrompidos caso sua execução não se conclua dentro do intervalo de tempo delimitado.

E) O escalonador seleciona o processo que estiver à espera e possuir o menor tempo de execução estimado e o coloca em execução até a sua conclusão.

13. (IESES/IGP-SC - 2017) Acerca da gerência de processos dos sistemas operacionais, assinale a alternativa correta:

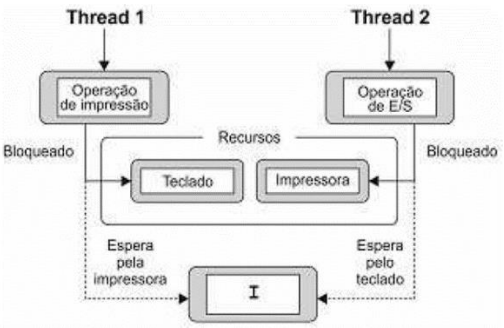
A) Um conjunto de processos está em estado de deadlock quando todos os processos no conjunto estão esperando por um evento que só pode ser causado por outro processo do conjunto.

B) Em um escalonamento preemptivo, um processo só perde o processador se terminar ou entrar em estado de espera.

C) No algoritmo de escalonamento de processos Round Robin, o escalonador sempre escolhe para execução o processo com menor expectativa de tempo de processamento. Esse algoritmo baseia-se no fato de que privilegiando processos pequenos o tempo médio de espera decresce.

D) Starvation é uma situação que não pode ocorrer quando um sistema operacional provê prioridades a processos.

14. (FCC/DPE-RS - 2017) Considere a figura abaixo.



Do ponto de vista do sistema operacional, a situação indica que a caixa I deve ser preenchida com?

A) starvation.

B) multithreading.

C) superthreading.

D) deadlock.

E) hyperthreading.

15. (CONSULPLAN/TRE-RJ - 2017) Quando um processo aguarda por um recurso que nunca estará disponível ou mesmo um evento que não ocorrerá, acontece uma situação denominada deadlock (ou como alguns autores denominam: impasse ou adiamento indefinido). Para que um deadlock ocorra, quatro condições são necessárias. Uma delas tem a seguinte definição: "cada recurso só pode estar

alocado a um único processo em um determinado instante". Assinale a alternativa que apresenta tal condição.

A) Espera circular.

B) Exclusão mútua.

C) Não-preempção.

D) Espera por recurso.

16. (FCC/DPE-RS - 2017) Dentre as políticas de escalonamento de processos a seguir, a que apresenta maior probabilidade de ocasionar o starvation é a

A) Round Robin.

B) de tempo compartilhado.

C) First In First Out.

D) preemptiva.

E) não preemptiva.

17. (CESPE/TRE-TO - 2017) Considerando o contexto de gerenciamento de processos dos sistemas operacionais, assinale a opção que apresenta a estrutura de dados responsável por habilitar o sistema operacional a localizar e acessar rapidamente o bloco de controle de processo (PCB) de um processo.

A) árvore de processos.

B) lista de bloqueados.

C) tabela de processo.

D) região de pilha.

E) lista de prontos.

18. (COMPERVE/UFRN - 2018) Sistemas operacionais modernos têm uma gerência de processos e de threads bem definida. Nesse contexto, é correto afirmar:

A) threads de um mesmo processo compartilham a mesma seção de código na memória.

B) threads de um mesmo processo compartilham a mesma seção da pilha na memória.

C) todas as variáveis de uma thread são compartilhadas com as outras threads do mesmo processo.

D) todos os contextos de uma thread são compartilhados com as outras threads do mesmo processo.

19. (FUNDEP/CODEMIG - 2018) O escalonamento de processos permite que um computador possa executar diversos programas em pseudoparalelismo, o que viabiliza aspectos como a multiprogramação. Qual entre os algoritmos de escalonamento a seguir seria mais adequado para sistemas de processamento em lote?

A) Primeiro a chegar, primeiro a ser servido.

B) Round Robin.

C) Escalonamento em duas fases.

D) Escalonamento por loteria.

20. (FUNDEP/CODEMIG - 2018) Um dos problemas relacionados ao gerenciamento de um sistema operacional diz respeito ao deadlock, o qual também pode ocorrer em banco de dados. Uma vez que gerenciar o deadlock pode ser uma tarefa que exija muito tempo do processador, a maior parte dos sistemas operacionais não trata desse problema. Em alguns sistemas críticos, entretanto, tratar os deadlocks é uma tarefa importante.

Qual entre as formas de tratamento a seguir se baseia em retirar o recurso do processo?

A) Através de preempção.

B) Revertendo o estado do processo.

C) Matando o processo.

D) Verificando a trajetória do processo.

21. (Quadrix/CRA-PR - 2019) A respeito do gerenciamento de processos e do gerenciamento de memória nos sistemas operacionais, julgue o item.

Embora os sistemas operacionais executem diversas operações de processo, como, por exemplo, criar e suspender um processo, eles não são capazes de alterar a prioridade de um processo.

22. (IF-PA/IF-PA - 2019) Em relação à gerência de processo, marque a alternativa CORRETA:

A) o processo é um programa em no estado de pronto.

B) os estados do processo são (execução, pronto, bloqueado ou espera).

C) a thread permite que apenas uma execução ocorra no mesmo ambiente do processo.

D) os sinais são mecanismos que permitem notificar o sistema operacional de eventos gerados pelo processador.

E) escalonamento é a escolha do processo, em estado de execução.

23. (CESPE/TJ-PA - 2020) No Linux, um processo, por si só, não é elegível para receber tempo de CPU. Essa ação depende, basicamente, do seu estado da execução. O processo está administrativamente proibido de executar no estado

A) pronto.

B) dormente.

C) executável.

D) parado.

E) zumbi.

24. (CESPE/TRT5 - 2008) Com relação ao sistema operacional Windows XP Professional, amplamente utilizado em determinados segmentos da área de tecnologia da informação, julgue os itens que se seguem.

Os drivers de dispositivos não podem ser alterados pelo administrador da estação porque o kernel assina digitalmente todos os drivers instalados automaticamente.

25. (CESPE/STF - 2008) Com relação a instalação de impressora local e em rede em sistemas Windows XP, julgue os itens seguintes.

No Windows XP, as impressoras laser que possuem drivers previamente instalados conectam-se periodicamente ao sítio do fabricante para modificar os drivers e os parâmetros de impressão.

26. (FCC/TCM-PA - 2010) A comunicação de uma aplicação com o subsistema de entrada e saída de um sistema operacional é estabelecida por meio de

A) shell.

B) device drivers.

C) system calls.

D) scripting.

E) batch.

27. (CESPE/INMETRO - 2010)

característica	processador pode executar outras instruções enquanto o módulo de E/S executa o seu trabalho	é preciso esperar que a operação de E/S termine
não há envolvimento do processador	técnica I	
processador controla operação de E/S	técnica II	técnica III

A partir das informações da tabela acima, que apresenta características de técnicas de gerenciamento de entrada e saída (E/S) de um sistema de computação, assinale a opção que nomeia corretamente as técnicas I, II e III, respectivamente.

A) E/S programada, E/S controlada por interrupção e acesso direto à memória (DMA).

B) E/S controlada por interrupção, E/S programada e acesso direto à memória (DMA).

C) E/S controlada por interrupção e acesso direto à memória (DMA) e E/S programada.

D) acesso direto à memória (DMA); E/S Programada, E/S Controlada por Interrupção.

E) acesso direto à memória (DMA), E/S controlada por interrupção e E/S programada.

28. (CESGRANRIO/TRANSPETRO - 2011) Os Sistemas Operacionais estão sujeitos a um fenômeno denominado deadlock. Para que uma situação de deadlock seja criada, as seguintes condições devem acontecer simultaneamente

A) exclusão mútua (mutual exclusion), monopolização de recursos (hold and wait), não preempção (no preemption) e espera circular (circular wait).

B) exclusão mútua (mutual exclusion), transferência excessiva de páginas (thrashing), superposição de processos (process overlapping) e espera circular (circular wait).

C) transferência excessiva de páginas (thrashing), superposição de processos (process overlapping), monopolização de recursos (hold and wait) e não preempção (no preemption).

D) exclusão mútua (mutual exclusion), monopolização de recursos (hold and wait), superposição de processos (process overlapping) e falha de escalonamento (scheduling fail).

E) transferência excessiva de páginas (thrashing), não preempção (no preemption), espera circular (circular wait) e falha de escalonamento (scheduling fail).

29. (FCC/TRT4 - 2011) Parte da definição da arquitetura de um computador é a especificação do seu sistema de entrada/saída. O esquema de E/S no qual a CPU gasta maior parte do seu tempo em loop, esperando o dispositivo ficar pronto é a E/S

A) por interrupção.

B) DMA.

C) programada.

D) através de canais.

E) por prioridade.

30. (CESPE/EBC - 2011) Para que os sistemas operacionais tenham acesso direto à memória, é necessário haver, no computador, recurso de hardware controlador DMA (direct memory access).

31. (CESPE/STF - 2013) No modo de operação do processador denominado modo usuário, instruções privilegiadas não podem ser executadas. Se houver tentativa de execução nesse caso, o hardware automaticamente gerará a interrupção e acionará o sistema operacional.

32. (FUNDATEC/BRDE - 2015) Em relação ao acesso direto à memória no gerenciamento de E/S de um sistema operacional, assinale a alternativa correta.

A) A controladora de DMA não permite um acesso independente ao barramento do sistema, dependendo sempre da CPU.

B) A controladora de DMA só depende da CPU quando o processador possui mais de um núcleo.

C) A localização física da controladora de DMA deve ficar próxima à CPU para que o acesso ao barramento do sistema seja independente da CPU.

D) A localização física da controladora de DMA deve ficar distante da CPU para que o acesso ao barramento do sistema seja independente da CPU.

E) Não importa a localização física da controladora de DMA, ela sempre terá acesso ao barramento do sistema de forma independente da CPU.

33. (FUNRIO/IF-PA - 2016) Sistemas operacionais compartilham recursos, havendo a possibilidade de deadlocks. A literatura especializada indica quatro condições necessárias para que um deadlock ocorra. O algoritmo de Avestruz utiliza uma estratégia para lidar com deadlocks conhecida como

A) detectar.

B) detectar e recuperar.

C) evitar.

D) ignorar.

E) prevenir.

34. (CESPE/MPE-PI - 2018) Julgue o item a seguir, acerca de sistemas operacionais.

Uma das causas de deadlocks em sistemas operacionais é a disputa por recursos do sistema que podem ser usados apenas por um processo de cada vez.

35. (COPESE/Câmara de Palmas-TO - 2018) Os sistemas operacionais modernos possuem diversos mecanismos para detecção e tratamento de situações de deadlock. Assinale a alternativa que NÃO apresenta um destes mecanismos.

A) O sistema irá escolher criteriosamente um processo e o terminará. Se a situação de deadlock não for resolvida, outros processos serão eliminados até que tudo esteja resolvido.

B) Os recursos são retirados dos processos e entregue aos outros até que o deadlock seja eliminado.

C) Os processos podem ser capazes de detectar um deadlock e voltar ao estado de execução anterior antes de pedir um recurso.

D) Um processo que detém um recurso fica esperando pela liberação de outro recurso, eliminando assim o deadlock.

36. (FCC/DPE-AM - 2018) Em um sistema operacional de computador, três processos estão na seguinte situação:

– o processo P1 tem a posse do recurso R1.

– o processo P2 tem a posse do recurso R2.

– o processo P3 tem a posse do recurso R3.

O processo P1 solicita o recurso R2, o processo P2 solicita o recurso R3, e o processo P3 solicita o recurso R1. Sobre essa situação, é correto afirmar que

A) não haverá deadlock, pois o processo P1 não solicitou o recurso R3.

B) tem-se uma condição de deadlock.

C) não haverá deadlock, pois o processo P3 não solicitou o recurso R2

D) só ocorrerá deadlock caso P1 solicite o recurso R3, P2 solicite o recurso R1 e P3 solicite o recurso R2.

E) não haverá deadlock, pois o processo P2 não solicitou o recurso R1.

GABARITO

1. E
2. A
3. C
4. Certo
5. C
6. ANULADA
7. C
8. D
9. Certo
10. C
11. Certo
12. B
13. E
14. A
15. D
16. B
17. E
18. C
19. A
20. A
21. A
22. Errado
23. B
24. D
25. Errado
26. Errado
27. C
28. E
29. A
30. C
31. Certo
32. Certo
33. E
34. D
35. Certo
36. D
37. B

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.