Sistemas Operacionais

Módulo 1 – Fundamentos e Processos

ITnerante

Gustavo Vilar

- Mini – CV
  - PPF / DPF – Papiloscopista Policial Federal
  - Pós-Graduado em Docência do Ensino Superior – UFRJ
  - Graduado em Ciência da Computação e Processamento de Dados – ASPER/PB
  - Aprovações: PRF 2002, PF 2004, MPU 2010, ABIN 2010

Gustavo Vilar

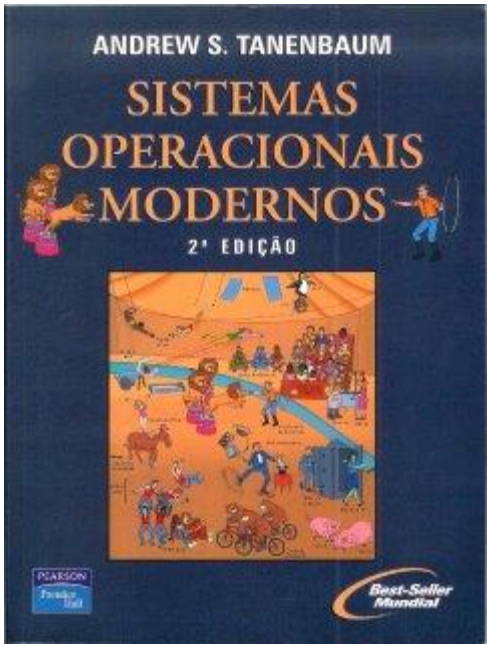
- Contatos:
  - gustavopintovilar@gmail.com
  - p3r1t0f3d3r4l@yahoo.com.br

ITnerante

Escopo

- Abordar os assuntos mais recorrentes e com fortes tendências para concursos atuais
- Familiarizar o concursando com os tipos de questões mais freqüentes.
- Abordar as metodologias de resolução de questões das principais bancas

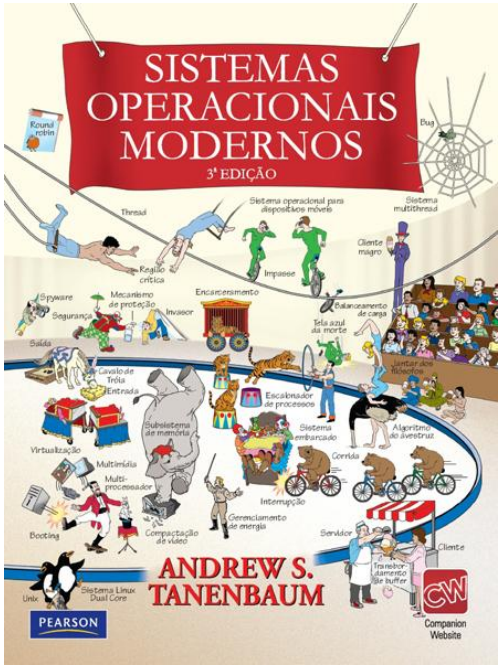
Bibliografia



FUNDAÇÃO CESGRANRIO

cespeUnB

Fundação Carlos Chagas



ITnerante

Sistemas Operacionais 01 de 03 – Carga Horária

- 11 vídeo aulas (04h17m / 00h23m20s)
  - Sistemas Operacionais
    - Conceitos e classificações
    - Ambientes, contextos, visões e kernel
    - Classificações do kernel, modos de operação
    - Controle de Entrada e saída, compartilhamento de recursos
  - Primeira bateria de questões de aprendizagem
  - Processos
    - Conceitos, orientações, vida, morte e estados
    - Modelos, contextos, espaço utilizado
  - Threads
    - Tipos (núcleo, usuário, híbrida)
  - Tópicos avançados
    - Comunicação interprocessos, condições de corrida, concorrência, inconsistências
    - Região Crítica
    - Deadlock
  - Segunda bateria de questões de aprendizagem

ITnerante

Sistemas Operacionais

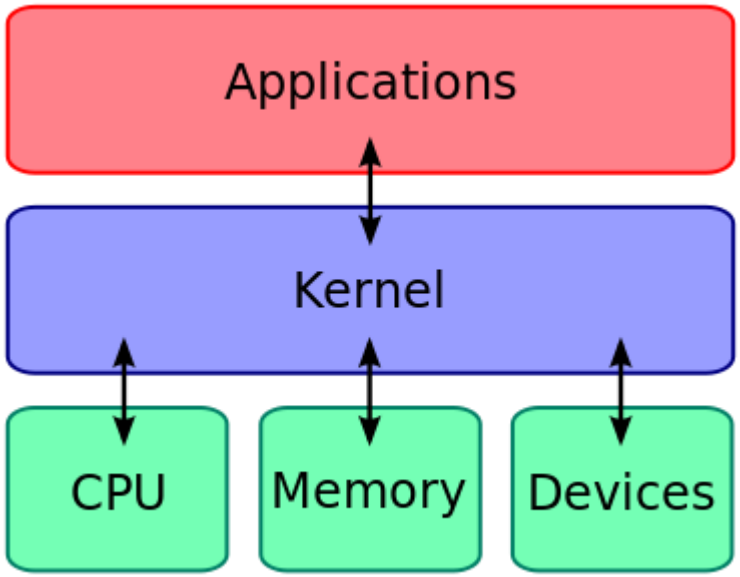
Princípios, fundamentos e classificações

ITnerante

Conceitos Iniciais

- Camada de software localizada entre os aplicativos e o hardware
- Formas de interação (Shell)
  - **CLI**: Comand line interface (Linux)
  - **GUI**: Graphic User Interface (Windows)

Esquema visual



Classificações dos Sistemas Operacionais

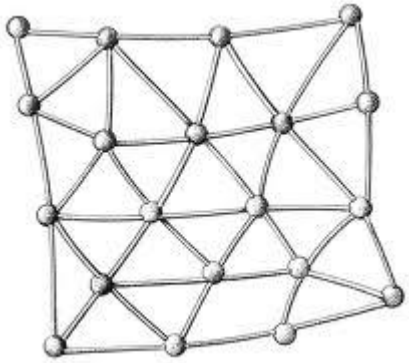
- **Aplicativos**
  - Monotarefa - executa uma tarefa de cada vez
  - Multitarefa - processos podem ser executados simultaneamente/concorrentemente
- **Usuários**
  - Monousuários
  - Multiusuários
- **Qtde Processadores**
  - Monoprocessado
  - Multiprocessado

Classificações dos Sistemas Operacionais

- Política de escalonamento
  - Lote
  - Tempo compartilhado
  - Tempo real

Classificações dos Sistemas Operacionais

- **De rede**
  - Conjunto de módulos que ampliam os sistemas operacionais, complementando-os com um conjunto de funções básicas que tornam transparente o uso de recursos compartilhados da rede
    - Suporte a operação em rede
    - Compartilhamento de recursos em rede
    - Usuário precisa saber onde estão os recursos
    - Recursos pertencem a computadores específicos



Classificações dos Sistemas Operacionais

- **Distribuído**
  - Ambiente computacional virtual
    - visão unificada
    - localização abstrata de armazenamento e processamento
    - Tarefa processada em vários computadores

Classificações dos Sistemas Operacionais

- **De rede**
  - Usuário precisa se preocupar com a localização dos recursos
- **Distribuído**
  - Usuário não tem que se preocupar com a localização dos recursos. O próprio SO “se vira”.

Classificações dos Sistemas Operacionais

- **Embarcado / Incubado / Embutido**
  - Inseridos em hardware com recursos limitados
    - Portáteis
    - Tarefas predefinidas
    - Requisitos específicos
- **PC**
  - Uso doméstico
- **Servidor**
  - Maior robustez

Ambientes dos Sistemas Operacionais

- Sistemas Interativos
- Sistemas Embarcados
  - Funções restritas
- Sistemas de Tempo Real
  - Sofrem restrições de tempo
  - Programa executa o tempo que for necessário
  - Ambientes de missão crítica
  - Não podem parar

Contextos dos Sistemas Operacionais

- Usuário interage com SO pelo Shell
- SOs oferecem APIs
- SO comunica com o hardware via drivers de dispositivos
- Apenas o kernel tem acesso ao hardware
  - Linguagem de baixo nível

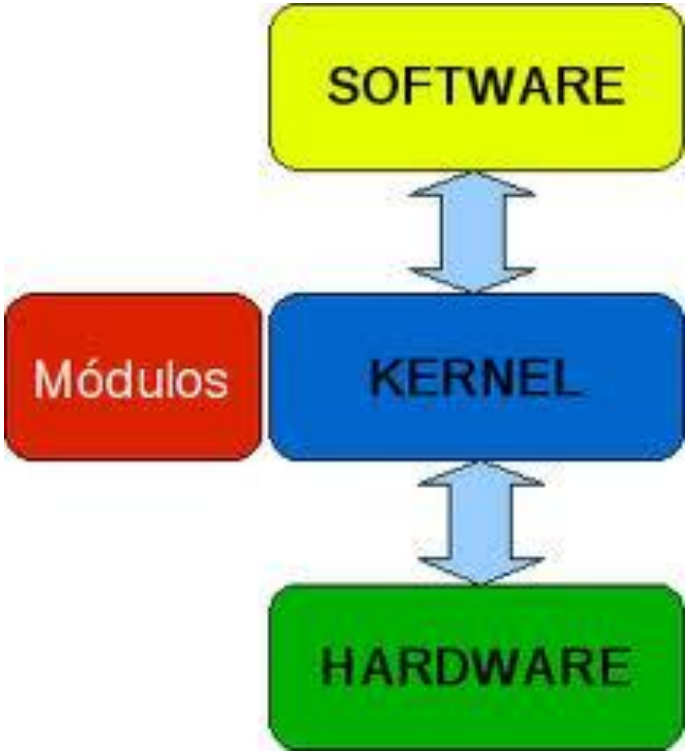
Visões dos Sistemas Operacionais

- **Visão usuário**
  - Interface de comunicação entre usuário e máquina, servindo como plataforma e gerenciador de recursos físicos
  - Pré-requisitos para aplicativos
  - Gerenciamento de memória (fluxo de processamento)
  - Gerência de recursos / arbitragem de tarefas ou processos
    - Memória
    - Processos

Visões dos Sistemas Operacionais

- **Visão programador**
  - Ocultação da complexidade do sistema físico
  - Uso dos comandos disponibilidades pelo SO (API do windows P. ex.)
  - Abstração / máquina estendida / máquina virtual
    - Ocultação da complexidade do sistema físico

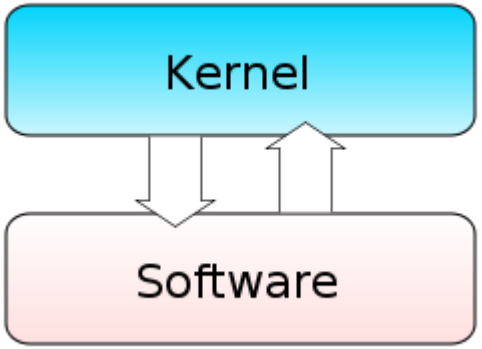
Sistemas Operacionais x Kernel



- Apenas a parte que interage com o hardware é o SO
- O produto de prateleira não é o SO na essência da palavra

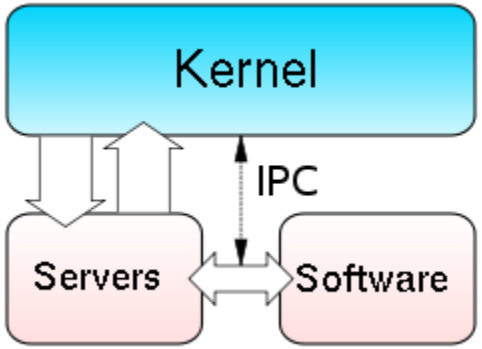
Tipos de Kernel

- **Monolítico**
  - Mais antiga
  - Todas as funções privativas estão no modo kernel
  - Kernel compilado e vai para RAM como bloco único
    - Todo o núcleo é executado em modo privilegiado
  - Detecção de erros difícil
  - Grande desempenho
  - Não existe mudança de modo (usuário / núcleo)
  - SOs vulnerável a programas mal comportados



Tipos de Kernel

- **Micronúcleo (microkernel)**
  - Somente o mínimo roda com privilégios elevados
  - Outras funcionalidades são oferecidas através de programas chamados servidores
  - Funcoes de gerencia de memória, threads e comunicações entre processos IPC
  - A maioria dos serviços roda em modo de usuário
  - Muitas funções do kernel ficam liberadas no modo usuário para acesso aos aplicativos
  - Alto custo nas transições ente modo núcleo e usuário



Tipos de Kernel

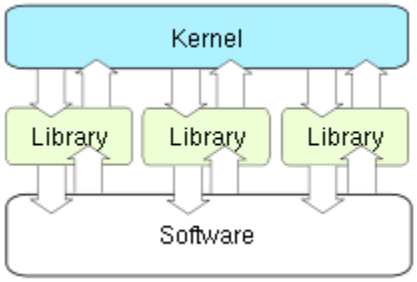
- **Núcleos Híbridos**
  - Núcleos híbridos são um acordo entre o desenvolvimento de micronúcleos e núcleos monolíticos
  - Isto implica executar alguns serviços (como a pilha de rede ou o sistema de arquivos) no espaço do núcleo para reduzir o impacto na performance de um micronúcleo tradicional, mas ainda executar o código no núcleo (como drivers de dispositivos) como servidores no espaço de usuário

Tipos de Kernel

- **Nanonúcleo**
  - Delega virtualmente todos os serviços para drivers de dispositivo para tornar o requerimento de memória do núcleo ainda menor do que o dos tradicionais micronúcleos, incluindo até os serviços mais básicos como controlador de interrupções ou o temporizado

Tipos de Kernel

- **Exonúcleo**
  - Um núcleo faz gerência de recursos do sistema e um conjunto de bibliotecas implementam a abstração de um sistema operacional
  - aplicativo pode utilizar diretamente os recursos dos dispositivos do sistema ou pode utilizar bibliotecas de software e são capazes de compartilhar o mesmo hardware simultaneamente, similarmente a uma máquina virtual
  - em um sistema exonúcleo o hardware é alocado previamente, não podendo ocorrer o acesso indevido de outro sistema
  - Não oferece abstração de hardware
  - VM/370



Modos de Operação

- Modo núcleo / supervisor / sistema / monitor / kernel
  - Manipula diretamente o hardware
    - Registradores, Portas de entrada/saída, Memória
  - Todas instruções do processador podem ser executadas
- Modo usuário
  - Não tem privilégios com o hardware
  - Interage com o Kernel
  - TRAP ou exceção
    - interrupção de software
    - Instrução usada pelo S.O. para alternar entre o modo núcleo <=> usuário

Interrupções

- Eventos podem ocorrer durante o processamento obrigando a intervenção do S.O
- Sistema
  - Externa
  - independe do programa que esta sendo executado
- Trap ou Exceções
  - Interna
  - Depende do programa que esta sendo executado

Controle de ES

- **Controlada por programa**
  - Modo Bloqueado
    - busywait
    - CPU dedicada ao periférico do início ao fim de toda a operação de E/S
  - Pooling
    - certo grau de paralelismo
    - Flip-flop - Flag associado a cada dispositivo de E/S
    - CPU testa os flags
    - Flag = 1 indica necessidade de CPU
  - Interjeição
    - Pooling aprimorado
    - flag adicional para grupo de periféricos
    - Flag adicional = 1 indica que algum dispositivo tem necessidade de CPU

Controle de ES

ITnerante

Controle de ES

- **Controlada por interrupção**
  - Problemas do controle por programa
    - Modo bloqueado a CPU perde muito tempo
    - No Pooling e Interjeição existe a necessidade dos testes
  - O dispositivo envia sinal a CPU
  - CPU interrompe o processamento e desvia a execução para a rotina de atendimento ao dispositivo que enviou o INT

Controle de ES

- **DMA**
  - Permite que dados sejam transferidos entre memória e periféricos sem a intervenção da CPU, exceto no início e no final da transferência.
  - buffer de E/S
  - área de memória utilizada pelo controlador para DMA
  - No momento da transferência DMA, controlador assume controle do barramento e não gera uma interrupção. CPU pode realizar tarefas sem utilização do barramento.

Compartilhamento de recursos

- No tempo
  - Processador e impressora
  - Um utilizador por vez
- No espaço
  - Memória
  - Mais de um utilizador por vez

Bateria de questões de aprendizagem 1

Sistemas Operacionais

ITnerante

MDIC – ESAF 2012 – Analista de Comércio Exterior

1. Nos sistemas multiprogramáveis,

A. um processo é gerido por rotinas de hardware, contexto de arquitetura e espaço de endereçamento.
B. um processo é endereçado pelo contexto de memória, pelo contexto de software e pelo hardware disponível.
C. um processo é programado em hardware, desenvolvido no sistema operacional e endereçado apenas de forma indexada.
D. um programa é formado por contexto de hardware, entrada de software e espaço de memória.
E. um processo é formado por contexto de hardware, contexto de software e espaço de endereçamento.

MDIC – ESAF 2012 – Analista de Comércio Exterior

2. Assinale a opção correta.

A. A principal desvantagem dos sistemas multiprogramáveis é a elevação de custos em função do compartilhamento de recursos.

B. Nos sistemas de tempo real, existe a ideia de fatia de tempo.

C. O processamento em batch exige a interação do usuário com a aplicação.

D. Nos sistemas de tempo real, existe a ideia de fatia de compartilhamento.

E. A principal vantagem dos sistemas multiprogramáveis é a redução de custos em função do compartilhamento de recursos.

MDIC – ESAF 2012 – Analista de Comércio Exterior

3. A técnica de buffering consiste na

A. utilização de uma área na memória principal, denominada cobuff, para a transferência de dados entre a memória principal e a memória secundária.

B. conciliação de buffers de memória com buffers de compilação, para otimização da transferência de dados.

C. otimização da memória principal através de kernel buffers, para a verificação da consistência de dados existentes na memória.

D. utilização de uma área na memória principal, denominada buffer, para a transferência de dados entre os dispositivos de entrada e saída e a memória.

E. utilização de buffers existentes nos dispositivos de entrada e saída para otimização da memória.

PETROBRÁS – CESGRANRIO 2012 – Analista de Sistemas Júnior - Infra

4. A política de escalonamento utilizada pelo sistema operacional para fazer a gerência do processador, que é caracterizada pela possibilidade de o sistema operacional interromper um processo em execução e passá-lo para o estado de pronto, com o objetivo de alocar outro processo no processador, é chamada de escalonamento

A. atemporal
B. temporal
C. seletivo
D. preemptivo
E. não preemptivo

PETROBRÁS – CESGRANRIO 2012 – Analista de Sistemas Júnior - Infra

5. O mecanismo pelo qual programas dos usuários solicitam serviços ao núcleo do sistema operacional é denominado

A. biblioteca do sistema
B. chamada do sistema
C. editor de ligação
D. shell de comandos
E. ligação dinâmica

PETROBRÁS – CESGRANRIO 2012 – Engenheiro de equipamentos Júnior - Eletrônica

6. O processamento de interrupções dentro de um processador oferece suporte ao sistema operacional, pois permite que um programa de aplicação seja suspenso

A. após o final do seu quantum de tempo, para que a interrupção possa ser atendida, sendo a execução do programa retomada mais tarde.

B. após a execução de metade do seu quantum de tempo, para que a interrupção possa ser atendida, sendo a execução do programa retomada mais tarde.

C. imediatamente, para que a interrupção possa ser atendida, sendo a execução do programa retomada mais tarde.

D. assim que o sistema operacional esteja pronto para atender à interrupção, sendo a execução do programa retomada mais tarde.

E. assim que o sistema operacional esteja pronto para atender à interrupção, sendo a execução do programa retomada ou não mais tarde, em função do resultado da interrupção.

BNDES – CESGRANRIO 2012 – Profissional Básico – Análise e Desenvolvimento de Sistemas

7. A gerência do processador é uma das atividades mais importantes em um sistema multiprogramável. Uma política de escalonamento deve ser estabelecida para determinar qual processo será escolhido para fazer uso do processador.

Com relação a essa política, considere as afirmações abaixo.

I - O escalonamento preemptivo é caracterizado pela possibilidade de o sistema operacional interromper um processo em execução e passá-lo para o estado de espera, com o objetivo de colocar outro processo em execução.

II - Com o uso da preempção, é possível ao sistema priorizar a execução de processos, como no caso de aplicações de tempo real onde o fator tempo é crítico.

III - No escalonamento não preemptivo, quando um processo está em execução nenhum evento externo pode ocasionar a perda do uso do processador.

IV - O escalonamento FIFO (First-In First-Out) é um exemplo de escalonamento não preemptivo no qual o processo que chega primeiro ao estado de pronto é colocado em execução e só perde o uso do processador quando termina seu processamento ou quando executa instruções do próprio código que ocasionam uma mudança para o estado de pronto.

É correto **APENAS o que se afirma em**

A. I e II
B. II e III
C. III e IV
D. I, II, III
E. II, III e IV

TCE SE – FCC 2011 – Analista de Controle Externo – Coordenadoria de Informática

8. Sobre uma chamada do sistema operacional, considere:

I. O sistema chama o procedimento de serviço.
II. O programa de usuário gera uma interrupção para o kernel.
III. O controle é retornado para o programa de usuário.
IV. O sistema operacional determina o número do serviço necessário.

A execução natural das operações acontece na sequência

A. II, IV, I e III.
B. II, I, IV e III.
C. IV, I, III e II.
D. IV, III, I e II.
E. III, II, IV e I.

TCE SE – FCC 2011 – Analista de Controle Externo – Coordenadoria de Informática

9. Se um processo de sistema operacional começou a imprimir uma saída, a ação de capturar a impressora e passá-la para outro processo resultará em problemas na saída. As impressoras são recursos

A. sujeitos a deadlock.
B. preemptíveis.
C. não preemptíveis.
D. dependentes de pipe.
E. dependentes de thread.

AL AM – ISAE 2011 – DBA

10. Um computador está dotado de uma UCP, de recursos de memória, de disco rígido e de impressão. Este computador pode executar vários programas em paralelo, de forma concorrente, adotando um sistema operacional que opera em regime de:

A. multiprocessamento.
B. multiprogramação.
C. monotarefa.
D. monoprogramação.
E. monoprocessamento.

Gabarito

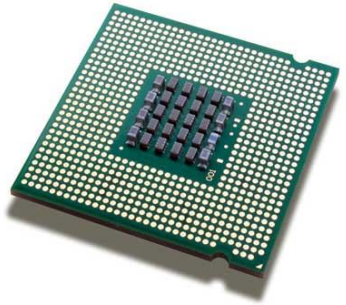
1. E
2. E
3. D
4. B
5. B
6. C
7. B
8. A
9. C
10. B

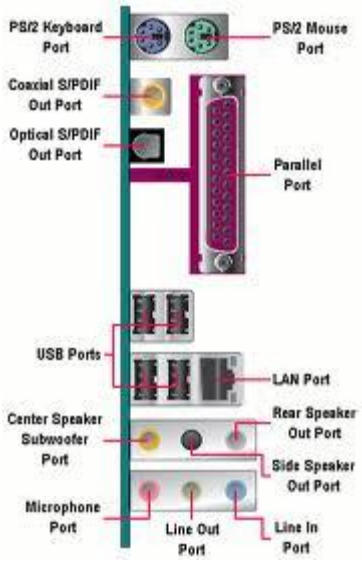
Processos - Conceitos

- Conceito central em qualquer sistema operacional
  - Unidade de gerenciamento de recursos
  - Meio de agrupar recursos relacionados
  - Associado a
    - Espaço de endereçamento (programa + dados + pilha)
    - Conjunto de registradores
  - Há proteção entre os processos: disputa de recursos
  - CPU executa somente um programa por vez
    - Uma CPU = Pseudoparalelismo
    - + de uma CPU/núcleo (multiprocessadores) = Paralelismo

Processos – Orientações

- É um programa em execução
  - Cada processo tem sua CPU virtual
  - Multiprogramação aumenta o uso efetivo da CPU
  - **CPU Bound**
    - Uma CPU proporcionalmente mais lenta ao número de processos
    - Usa muito processador e pouco IO
    - Executa e vai para fila de pronto
  - **I/O bound**
    - Usa muito IO e pouco processador
    - uma CPU um pouco mais lenta para o processo
    - Melhora o grau de multiprogramação
    - Grau de multiprogramação = quantidade de programas que se revezam em execução
      - Definido pelo escalonador de admissão LP / SCHEDULER
- Multiprogramação depende apenas do sistema operacional
- Multitarefa depende do sistema operacional e do hardware do sistema





Processos – A vida

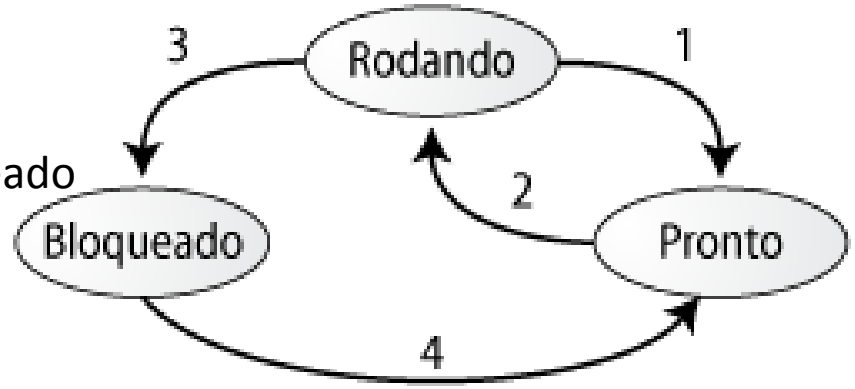
- Criação
  - Início do sistema
    - Foreground
      - Primeiro Plano
      - Interativo
    - Background
      - Daemons
      - Segundo plano
      - Não interativos, ou seja, o usuário não participa
  - Chamada de sistema de criação de processo por um processo em execução
    - Após a inicialização
    - ajuda no trabalho do processo chamador
  - Requisição do usuário
    - Sistemas interativos
    - Linha de comando ou ambiente iconográfico
  - Início de uma tarefa em lote
    - Somente em computadores de grande porte
    - Execução condicionada à existência de recursos
- OBS: Em todos os casos um processo filho e criado por um processo pai
  - UNIX: Fork
    - processo filho executa chamada execve divide a imagem de memória
    - Nenhuma memória para escrita é compartilhada, apenas de leitura
    - Copy-on-write
      - » Imagem de memória iguais até a primeira escrita
      - » Conceito de grupo de processos (Pai, filho, irmãos, netos)
    - Processos pertencem a uma única árvore (init é a raiz)
  - Windows: CreateProccess e outras
    - Espaços de endereçamento diferentes desde o início
    - Não existe conceito de hierarquia entre os processos

Processos – A Morte

- Saída normal (voluntária)
  - Unix - Exit
  - Windows - ExitProccess
  - Terminou a tarefa
- Saída por erro (voluntária)
  - fornecimento de parâmetros errados
  - Geralmente há uma caixa de diálogo para notificação
- Erro fatal (involuntária)
  - Execução de instrução ilegal
  - referência a memória inexistente
  - divisão por zero
  - Erro pode ser tratado pelo SO ou pelo próprio processo (que sinaliza ao SO sua intenção)
- Cancelamento por outro processo (involuntária)
  - Unix - Kill
  - Windows - TerminateProccess
  - Processo que executa a chamada deve ter autorização necessária

Processos - Estados

- Execução
  - No processador
  - Não há fila
  - Trap coloca o processo para bloqueado
- Pronto
  - Fila para o processador
  - Revezando a CPU
  - Escalonamento
- Bloqueado
  - Fila para I/O ou E/S
  - Incapaz de executar enquanto evento externo não ocorrer



- OBS: Existem ainda outros estados: Iniciado...terminado (modelo de 5 estados)
  - Modelo estendido - Diagrama de estados

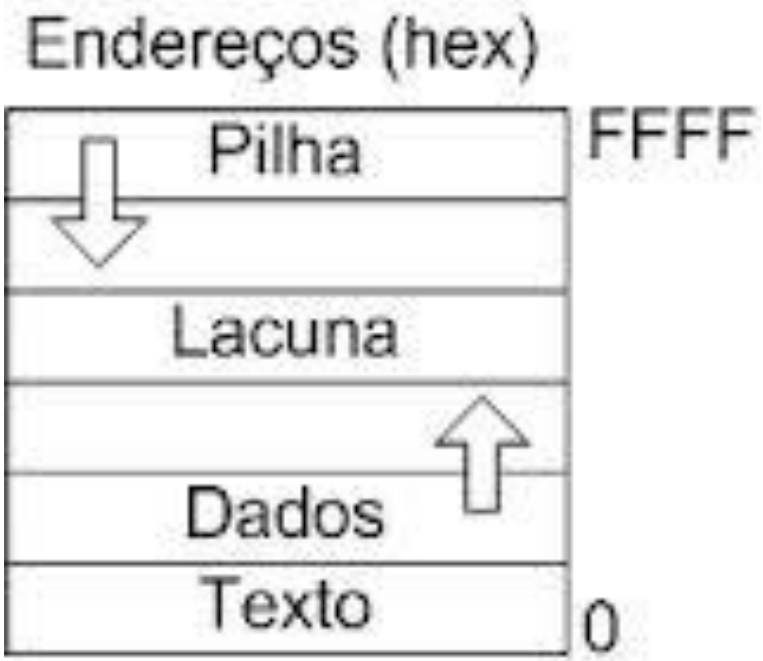
Implementação do modelo de processos

- SO Mantém uma tabela chamada Tabela de Processos
  - Mudança de estados envolvendo o estado de execução
  - Processo é reiniciado como se nunca tivesse bloqueado. "Corta"
- Uma entrada para cada processo
- Informações sobre
  - Estado
  - Contador do Programa
  - Ponteiro da Pilha
  - Alocação de memória
  - Estado dos arquivos abertos
  - Enfim: Contexto

Contextos dos processos

- Hardware
  - registradores gerais
  - registradores PC
    - próxima instrução a ser executada
  - registradores SP
    - Stackpointer
    - Ponteiro para o topo da pilha
  - Registrador psw
    - flags de erros, interrupção, div por 0
- Software
    - Nome do processo
    - PID
    - UID
    - Prioridade de execução
    - Data e hora de criação
    - tempo de processador
    - Quotas
    - Privilégios

Processos – Espaço usado



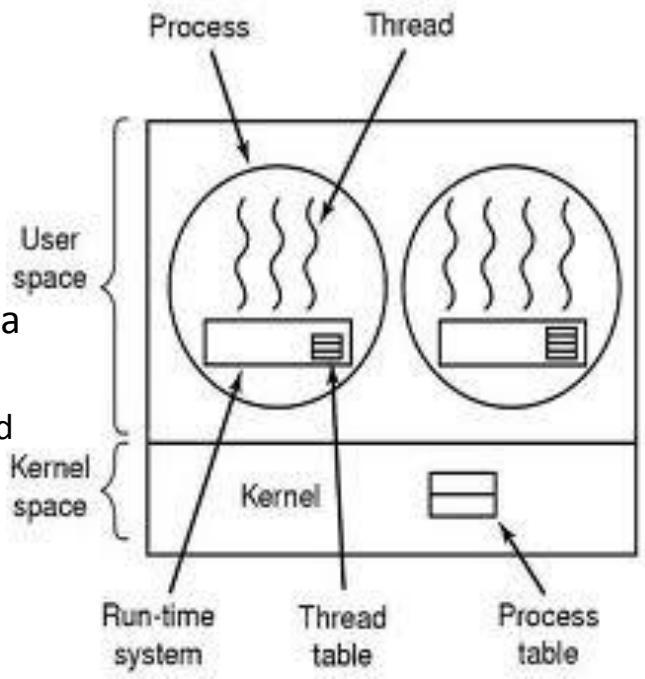
- Espaços de endereçamento
  - Range de endereços
  - Lista de posições da memória
  - Varia entre uma quantidade mínima e máxima em que o processo pode ler e escrever
- Dividido em 3 partes
  - Código (ou texto)
    - código do programa
  - Dados
    - variáveis declaradas no programa
    - variáveis não declaradas no programa
  - Pilha
    - Armazena os apontadores do programa

Threads

- No processo de escalonamento, existem 2 níveis de paralelismo
  - De processos
  - De threads

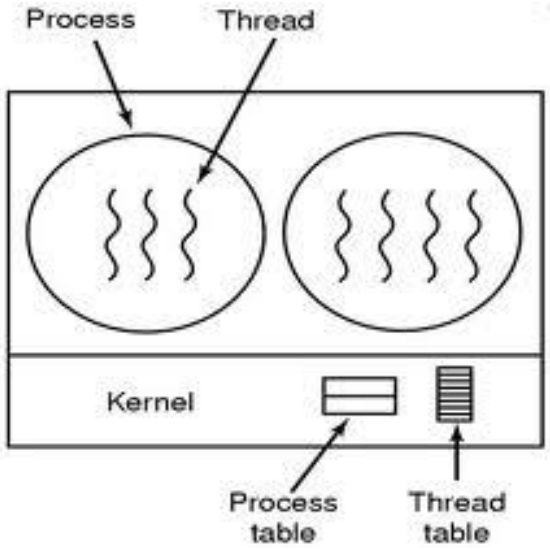
Threads de Usuário

- Núcleo só conhece os processos, desconhece as Threads
  - Quantum é do processo
  - Não existe preempção das threads
- Núcleo trata os processos como se fossem monothread
  - Pode ser implementado em SOs que não suportam Threads
  - Cada processo precisa de sua tabela de Threads
- Não há necessidade de chaveamento de contexto, o que facilita seu escalonamento
- Thread precisa liberar voluntariamente a CPU para que outro o possa fazer
  - Thread do mesmo processo deve ceder voluntariamente seu lugar - yield
  - Escalonador nunca fará isso
  - Thread bloqueante paralisa outros threads do processo
- Executadas instruções que não precisam de privilégios
- Menor sobrecarga
  - Alternância de threads requer poucos recursos de máquina
- SOs não escalona as threads, apenas os processos
  - Necessidade do sistema supervisor
  - Aplicação escalona usando bibliotecas adequadas
  - Cada processo pode possuir seu algoritmo de escalonamento personalizado



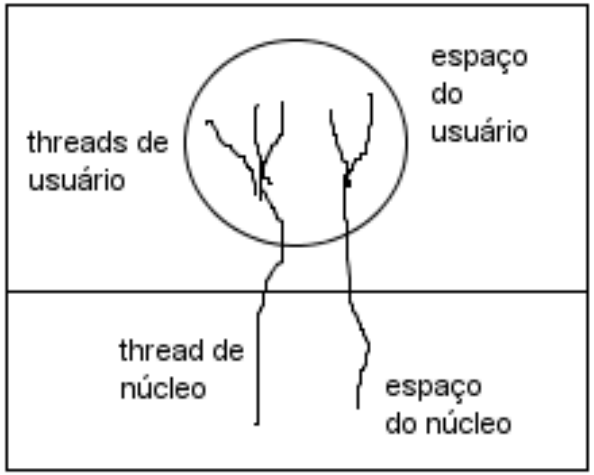
Threads de Núcleo

- Núcleo conhece a existência dos Threads e os gerencia
  - Quantum é da Thread
- Desnecessário o uso das tabelas de threads para cada processo e sistema supervisor
- Núcleo os cria ou destrói
- Núcleo possui tabela de processos e de threads
- Mais lentos que os de usuário
- Necessidades de chamadas de sistemas / chaveamento de modos
  - Alternância requer muitos recursos de máquina
  - Custo de chamadas maior
  - Implementado em nível de hardware
  - Escalonador global
- Suporte vem do kernel, usando instruções privilegiadas
- Portabilidade menor, pois é dependente do SO
- Thread bloqueado para E/S não suspende o processo



Threads – Implementação Híbrida

- Núcleo conhece apenas as Threads de núcleo
- Alguns desses threads multiplexam os threads de usuário
  - Número de Threads de núcleo nunca pode ser maior que o número de threads de usuário
  - Modelo Muitos-Para-Um
    - mapeia muitos threads de nível de usuário para threads do kernel
  - Modelo Um-Para-Um
    - cada thread de usuário para um thread de kernel
    - O modelo de threading suportado por um kernel Linux típico é do tipo um-a-um.
  - Modelo Muitos-Para-Muitos
    - multiplexa muitos threads de nível de usuário para um número menor ou igual de threads de kernel
    - Tenta combinar as vantagens e eliminar as desvantagens



Threads – Pontos relevantes

- São mais fáceis de criar e destruir que os processos
- Inexistência de recursos associados
- 100 x mais rápido do que criar um processo
- Custo de gerenciamento bem menor que processos (comunicação mais simples)
  - Ganho de desempenho só ocorre com a IO/bound
- Uteis em sistemas multiprocessados, onde o paralelismo real é possível
- Processos são usados para agrupar recursos (unidade de gerenciamento de recursos), Threads para escalonamento de CPU
- Escalonamento similar aos dos processos
- Aplicações típicas
  - Servidor web
  - Processador de texto

Comunicação Interprocessos - IPC

- Processos precisam se comunicar entre si
  - Sinais
  - Mensagens
- Questões
  - Como um processo passa informação para outro?
    - Para os Threads do mesmo processo é fácil, pois compartilham o mesmo espaço de endereçamento
  - Como garantir que 2 ou + processos não entrem em conflito?
  - Sequência adequada quando existirem dependências

Processos – Condições de corrida

- Processos que trabalham juntos podem compartilhar algum armazenamento comum (ler / escrever)
  - RAM
  - Arquivo
- 2 ou + processos lendo ou escrevendo algum dado compartilhado
- Problema do produtor consumidor
- Um processo só pode usar algo que já foi produzido pelo produtor

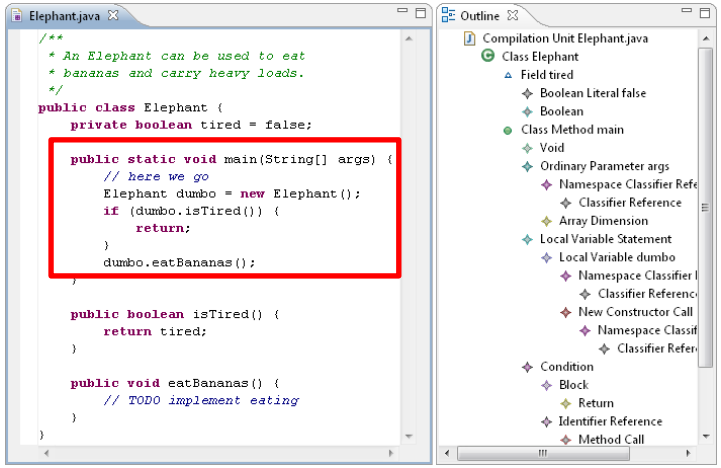
Processos - Concorrência

- Existe quando não há paralelismo
- Pode existir quando há paralelismo
- Problemas
  - Bloqueio de recursos
  - Adiamento indefinido
  - Intermitentes - Quando o paralelismo não previu o bloqueio de recursos
  - Solução - Sincronização entre processos e threads
- Sinais
  - Interrupções de software para notificar ocorrência de eventos
  - Usado para sincronia
  - Não há troca de informações, apenas aviso
  - variável, normalmente binária
    - Mutex - se for binária
    - Semáforo - Variável incrementável
- Mensagens
  - Há troca de informações

Processos – As inconsistências

- Problema da conta bancária
- Evitadas através de exclusão mútua
  - Garantir o uso da região crítica por um processo
    - Garante a multitarefa
    - Acesso ao mesmo recurso
  - Nada mais é do que uma variável indicando se está na região crítica
    - 0 ou 1 significando down e up / P e V
  - Problemas causados pela Exclusão mútua
    - Velocidade de execução de processos
    - Gargalo onde o mais rápido espera pelo mais lento
    - Starvation
    - Sincronização condicional
- Soluções
  - Semáforos
  - Monitores
  - Trocas de mensagens

Processos – Região Crítica



- Região **do código** que deve ser executada de forma atômica
- Apenas um processo por vez na seção crítica
- Parte do programa em que há acesso à memória compartilhada é chamada região crítica ou seção crítica
- Impedir que mais de um processo leia e escreva ao mesmo tempo na memória compartilhada - Exclusão Mútua
- Se 2 processos nunca estivessem em suas regiões críticas ao mesmo tempo, as disputas seriam evitadas
- Serialização

Processos – Região Crítica

- **Solução em hardware**
  - Desabilitar as interrupções
    - Existem operações de interrupções que podem ser mascaráveis
    - Programador deve se preocupar com a habilitação
    - Não é uma boa opção
  - **Uso da instrução do processador (flag) Test-and-Set**
    - Implementada em hardware
    - Espécie de variável global
    - Manuseio via SO

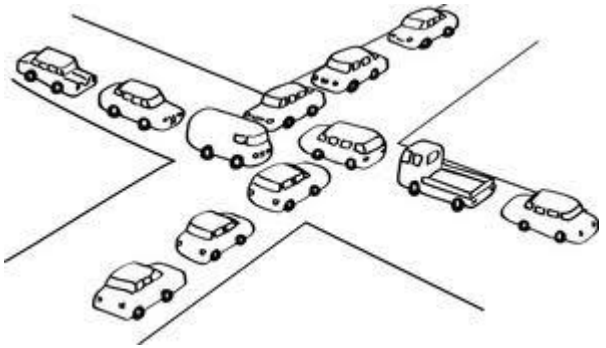
Processos – Região Crítica
Solução em Software

- **Algoritmo de Lamport**
- **Semáforos**
  - Variável inteira não negativa (down, up): Valores 0 ou maiores
  - Operações Down (sleep / P) e Up (wakeup / V)
  - Se binário = mutex
    - Quando não é preciso usar a capacidade do semáforo de contar
    - Não controla quantos consumidores existem
    - Variável que pode estar em um dos dois estados: Impedido ou Desimpedido

Processos – Região Crítica
Solução em Software

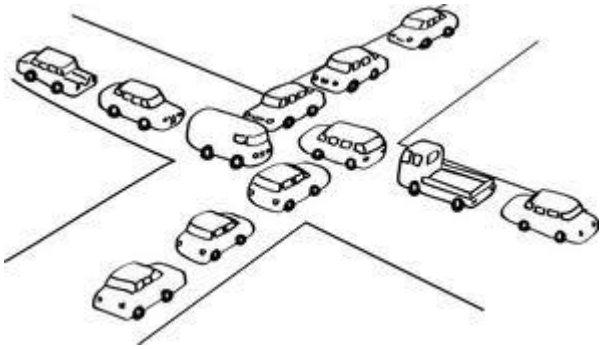
– **Monitores**
  - Mecanismo de sincronização
  - Procedimento que troca parâmetros com o processo
  - Cada processo em estado de pronto chama o monitor
  - Garante a execução da região crítica de cada processo

Deadlock



- Processos bloqueados ad infinitum
- Ocorrem em recursos de hardware ou software
  - Falha de comunicação entre processos
  - A falha é do programa e não do S.O.
- "congestionamento de processos"
- condição específica quando dois ou mais processos estão esperando cada um pelo o outro para liberar um recurso
- mais de dois processos estão à espera de recursos em uma cadeia circular
- problema comum em multiprocessamento onde muitos processos compartilham um tipo específico de recurso mutuamente exclusivo
- Solução: hardware (ou hard lock), que garante o acesso exclusivo aos recursos pelos processos, forçando o acesso serializado

Deadlock



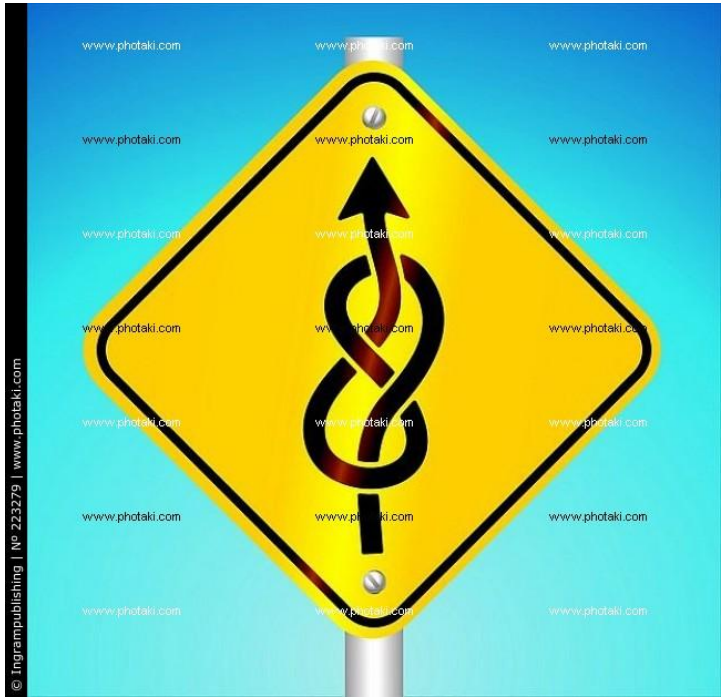
- Não se confunde com adiamento indefinido / starvation / inanição
- Starvation
  - Processo nunca obtém o serviço
  - NÃO está necessariamente em deadlock
  - Preterição indefinida
  - Evitada com FCFS
  - Resolvido através do aging
- A diferença principal é o estado do processo
- No deadlock os processos estão bloqueados, já no starvation estão em execução ou prontos

Deadlock

- Deadlock <> Livelock
  - Livelock - Ação
    - Programa A, chama programa B que por sua vez chama A, ou processo está em algum loop infinito interno
  - Deadlock - Paralisação
    - Impasse

Deadlock – O Problema

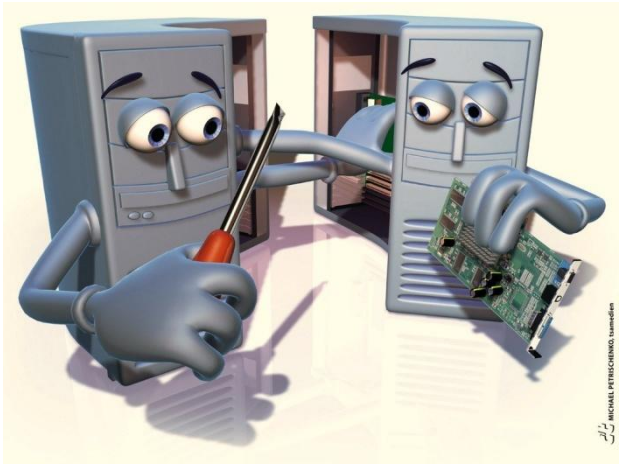
- Condições OBRIGATÓRIAS (necessárias e suficientes) para ocorrência
- 1.Exclusão mútua
  - Ou um recurso está associado a um processo ou está disponível
  - Um recurso que não pode ser usado por mais de um processo de cada vez
- 2.Posse e espera
  - Processos que retêm recursos podem solicitar novos antes da liberação
  - Processos que já possuem recursos podem solicitar novos recursos mantidos por outros processo
- 3.Não preempção
  - Nenhum recurso pode ser retirado à força de um processo, os recursos só podem ser liberados pela ação explícita do processo
  - Recursos concedidos devem ser liberados pelo processo requisitante e não tomados
  - Não haverá interferência do SO
  - Preemptivel: Pode ser retirado do processo, Não há deadlock
  - Não preemptivel: Não pode ser retirado, Há deadlock
- 4.Espera circular
  - dois ou mais processos formar uma cadeia circular onde cada processo espera por um recurso que o próximo processo na cadeia detém



Deadlock – A solução

- Estratégia preventiva
  - Negar uma das 4 condições
    - Exclusão mútua - Contornado com spooling
    - Posse e espera - Requisitar TODOS recursos com antecedência. Ou tudo ou nada
    - Não preempção - Impossível ou a pior de todas
    - Espera circular - Um recurso por vez

Deadlock – A solução



- Estratégia Corretiva
  - Detecção
  - Muitas vezes, a prevenção não pode ser usada
  - Detectando um impasse ocorrido é facilmente possível, pois os recursos que cada processo tem bloqueado e / ou solicitadas são atualmente conhecidos
  - Recuperação: Encerramento de um dos processos e consequente rollback
- OBS: Ignorar o problema até que ele se estabeleça (Unix e windows) - devido ao alto custo envolvido na estratégia preventiva

Bateria de questões de aprendizagem 2

Sistemas Operacionais

ITnerante

MDIC – ESAF 2012 – Analista de Comércio Exterior

1. Em sistemas multiprogramáveis, são estados do processo:

A. Execução (executing). Disponível (avaliable). Espera (log).
B. Execução (running). Pronto (ready). Espera (wait).
C. Direção (driving). Ponto (float). Espera (waste).
D. Execução (running). Inserção (insert). Avaliação (controll).
E. Pronta execução (ready running). Compartilhamento (sharing). Espera (wait).

Pref Ibiporã – AOCP 2011 – Analista de Sistemas

2. Sobre Detecção de Deadlock em sistemas operacionais, analise as assertivas e assinale a alternativa que aponta as corretas.

I. Em sistemas que não possuam mecanismos que previnam a ocorrência de deadlocks, é necessário um esquema de detecção e correção do problema.

II. Não há nenhum algoritmo capaz de detectar deadlock, isso deve-se a complexidade do problema.

III. Detecção de deadlock não é tarefa do Sistema operacional e sim do processador.

IV. A detecção do deadlock é o mecanismo que determina, realmente a existência de um deadlock, permitindo identificar os recursos e processos envolvidos no problema.

A. Apenas I e II.
B. Apenas I e IV.
C. Apenas I, II e IV.
D. Apenas I, III e IV.
E. I, II, III e IV.

TRANSPETRO – CESGRANRIO 2011 – Analista de Sistemas Júnior - Software

3. 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).

FINEP – CESGRANRIO 2011 – Analista - Suporte

4. Em sistemas multiprogramáveis, os processos podem concorrer pelos recursos do sistema. Essa concorrência pode provocar uma situação conhecida como deadlock, que só ocorre quando todas as seguintes condições estão presentes, ao mesmo tempo, em um sistema:

A. posse e espera; não preempção; espera ocupada
B. posse e espera; preempção; espera ocupada
C. exclusão mútua; preempção; espera circular
D. exclusão mútua; posse e espera; não preempção; espera circular
E. exclusão mútua; posse e espera; preempção; espera circular

CORREIOS – CESPE 2011 – Analista de Correios – Suporte de Sistemas

5. Com relação às características e funções básicas de um sistema operacional, julgue os itens seguintes.

[51] Região crítica é o termo utilizado para denominar a parte do código do programa na qual é realizado o acesso a um recurso compartilhado.

[52] Um sistema operacional multiprogramável somente pode executar várias tarefas ao mesmo tempo quando o computador no qual ele está instalado possui mais de um processador de dados.

[53] Um sistema operacional distribuído consiste de múltiplos processadores de dados fortemente acoplados.

CORREIOS – CESPE 2011 – Analista de Correios – Suporte a Sistemas

6. Com relação às características e funções básicas de um sistema operacional, julgue os itens seguintes.

[54] As principais funções do núcleo de um sistema operacional são as seguintes: tratamento de interrupções; criação, eliminação, sincronização e comunicação entre processos; gerência de memória e gerência de arquivos.

[55] Quando o processador trabalha no modo kernel, uma aplicação pode executar somente instruções privilegiadas.

[56] Em um ambiente com múltiplos threads (multithread), não é necessário haver vários processos para se implementarem aplicações concorrentes.

COFEN – CONSULPLAN 2011 – Web Designer

7. Situação em que ocorre um impasse e dois ou mais processos ficam impedidos de continuar suas execuções, ou seja, ficam bloqueados. Trata-se de um problema bastante estudado no contexto dos Sistemas Operacionais, assim como em outras disciplinas, como banco de dados, pois é inerente à própria natureza desses sistemas." Tal processo é denominado:

A. Deadlocks
B. Threads
C. Keyloggers
D. Starvation
E. Fifo

TRE RN – FCC 2011 – Analista Judiciário – Analista de Sistemas

8. Nos sistemas operacionais, múltiplas execuções que ocorrem no mesmo ambiente do processo com um grande grau de independência uma da outra é uma característica do modelo de processo que contempla o conceito de

A. bus.
B. switch.
C. thread.
D. disk array.
E. split-cylinder.

INMETRO – CESPE 2010 – Pesq TMQ – Ciência da Computação

9. Assinale a opção que apresenta um recurso de uso exclusivo de uma thread, em um processo multithreads.

A. variáveis globais
B. arquivos
C. pilhas de execução
D. sinais
E. semáforos

Gabarito

1. B
2. B
3. A
4. D
5. C, E, E
6. C, E, C
7. A
8. C
9. C