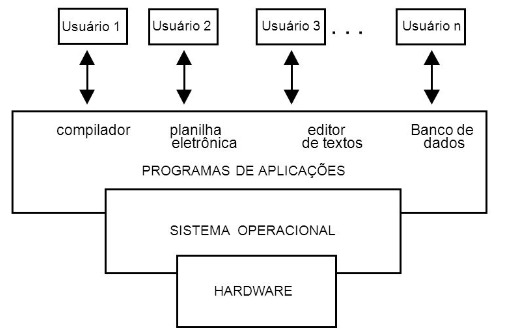
06/02/2024

* O que é um sistema operacional:
  + Um programa que age como intermediário entre o usuário e o hardware, facilitando o uso dos recursos
  + Gerenciam recursos de maneira eficiente:
    - CPU
    - Memoria
    - I/O
  + Programa de controle 🡪 controla execução dos programas e dispositivos de I/O
  + Kernel 🡪 programa executado interruptamente – modo supervisor
* Componentes de um SO
  + Hardware
  + S.O
  + Programas aplicativos
  + Usuários – pessoas , dispositivos, outros...
* Sistemas em lotes
  + Operador: controla a submissão de tarefas
  + Usuário e operador
  + Tarefa em forma de cartões perfuradores
  + Sequenciamento automático
  + Problemas: Baixo desempenho e apenas um programa rodando ao mesmo tempo
  + Solução: Jobs carregados de unidades de fita para memória. A leitura de cartões e tarefas eram feitas offline

Job proof: estrutura de dados que permite selecionar qual job será executado, para aumentar o uso da CPU

* + - Alocação dinâmica na memoria
* Multiprogramação:
  + Características do SO:
    - Escalonamento da CPU - Varios processos rodando ao mesmo tempo – posse da cpu
    - Rotinas de I.O provisionadas pelo sistema
    - Carência de memoria
    - Alocação de dispositivos
* Sistemas de tempos compartilhado:
  + CPU é multiplexada entre diversos Jobs mantidos em memoria primaria e em disco
  + O usuário tem a impressão de que a CPU está dedicado, pois a maquina tem a capacidade de fazer milhões de operações por segundo.
  + Um job é transferido do disco para a memória e da memória para o disco 🡪 swap
* Sistemas de Computadores Pessoais:
  + PC – sistema de computação a um único usuário
  + Dispositivos de I/O
  + Praticidade e tempo de resposta
* Sistemas Paralelos e Distribuídos 🡪 será abordado em outras disciplinas

Estrutura:

* Operação dos Sistemas de Computação:
  + I/O e CPU podem executar de forma concorrente. Ou seja , não preciso parar de imprimir para rodar meu programa
  + Controladora de dispositivos tem um buffer local
  + A CPU transfere dados entre os buffers locais e RAM
  + A controlado interna e CPU termina operações com interrupções
* Interrupções:
  + Como se fosse um try / catch
  + Segmentos de códigos separados determina qual ação deve ser executada a cada interrupção
* Acesso direto à memória:
  + Transferência entre as memorias sem passar por SO
* Estrutura de armazenamento:
  + Memoria principal
  + Memoria secundaria
  + Caching
  + Sistema organizado de armazenamento

08/02/2024 – Estruturas de Sistemas de Computação

* A proteção de memória pode ser feita com auxílio de registradores (apenas dois registradores) que determinam a faixa de memória que pode ser acessada pelo programa usado 🡪 registrar o status = mudança dos registradores para cada Job
* As instruções de load para os registradores de base e limite são instruções privilegiadas
* Proteção de CPU:
  + Timer🡪 interrompe o computador após um período específico

Processos:

* Um SO executa uma variedade de programas:
  + Sistema em batch – Jobs
  + Sistemas de tempo compartilhado – programa de usuário ou tarefas
* Estados 🡪 política de escalonamento:
  + Em espera
  + Pronto
  + Encerrado
* Pseudo-paralelismo 🡪 roda um pedaço de cada programa de cada hora, assim parecendo que eles foram rodados em paralelos.
* Bloco de controle de processos (PCB):
  + Informações associada com cada processo:
    - Estado do processo
    - Controlador do programa
    - Registradores da CPU
    - Informações de escalonamento
    - Informações de gerência de memória
    - Informações de contabilidade
    - Informações de IO
  + Troca de contexto a cada processo

20/02/2024

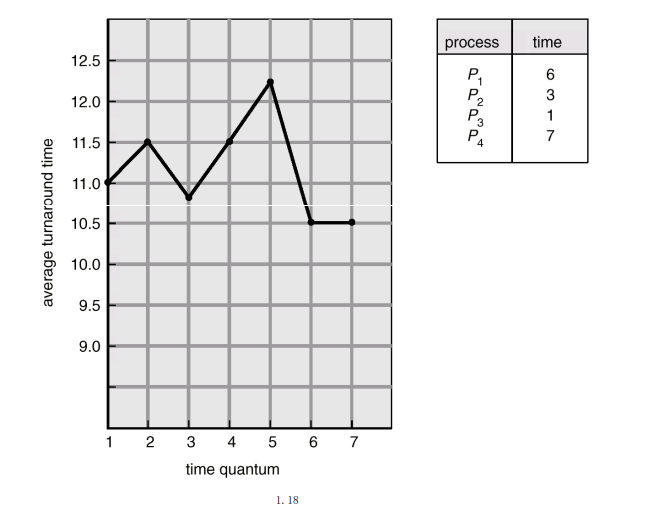
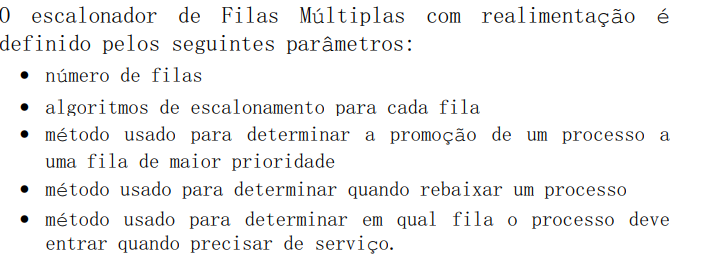
* Diagrama

  Descrição gerada automaticamenteEscalonamento de processos:
  + Toda operação I/o volta para fila de pronto , assim deixando o SO administrar os processos em andamento
* Escalonadores:
  + Longo prazo: selecionam quais processos devem ser levados a memoria , na fila de prontos
    - São invocados menos frequentemente
    - Controla grau de multiprogramação
  + Curto prazo: seleciona qual processo deve ser executado e alocado a CPU
    - Invocados frequentemente e rápido
  + Processos podem ser descritos como: Limitados por I/O – gastam mais tempo fazendo I/O que computação. Limitados por CPU – gastam mais tempo fazendo computação
  + É possível ser introduzido um nível intermediário de escalonador, responsável por reduzir o grau de multiprogramação
* Troca de Contexto:
  + Quando a CPU recebe novo processo, o estado do processo anterior deve ser salvo e o estado do novo processo anterior deve ser salvo e o estado do novo processo carregado
  + O tempo depende de suporte do hardware

Escalonamento de cpu:

* A utilização máxima de CPU é obtida através da multiprogramação
* Existem ciclos de IO e CPU durante a execução do programa – tempo de surto
* Escalonador de CPU:
  + Seleciona um dentre os processos a serem executados aloca CPU pra ele
  + Ocorre quando: execução para espera, execução para pronto, espera para pronto, terminam
  + Quando o escalonador ocorre apenas no caso 1 e 4, são chamados de não preemptivos, ou demais são os preemptivos
* Dispatcher:
  + Do controle da cpu ao processo selecionado pelo escalador de curto prazo.
    - Mudança de contexto
    - Mudança para modo usuário
    - Pular para posição adequada
  + Latência de dispatch:
    - Tempo do dispatcher interromper um processo e iniciar a próxima execução.
    - Se escalonamento é ideal o tempo de latência é 0
* Critérios de Escalonamento:
  + Utilização de CPU
  + Throughput
  + Tempo de retorno
  + Tempo de espera
  + Tempo de resposta – primeira resposta aparecer

Aula 27/02/2024 – Escalonamento de CPU

* A medida que o quantum diminuem aumenta a troca de contexto
* Existe momentos que a mudança do quantum não irá impactar mais
* Filas múltiplas:
  + Cada fila tem seu próprio algoritmo de escalonamento
  + Deve ser feito escalonamento entre as filas:
    - Escalonamento de prioridade fixa:primeiro plano pode ter prioridade sobre o segundo . Possibilidade de starvation
    - Fatia de tempo – cada fila recebe uma certa quantidade de tempo de CPU que pode ser escalonada entre seus processos. Por exemplo, 80% para primeiro plano em RR e 20% para o segundo plano em FCFS
  + Um processo pode passar de uma fila para outro.
* Tempos:
  + RESPOSTA 🡪 Quando inicia
  + Espera 🡪 Quanto tempo total esperado – tempo dele
  + Retorno 🡪 espera + tempo do processo
  + Vazão 🡪 tempo retorno do último / n processos
  + CPU 🡪 soma dos processos/tempo retorno último
  + FOTO NO CELULAR

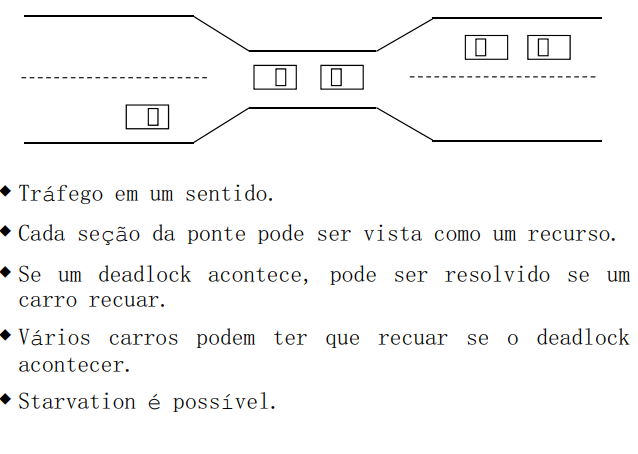
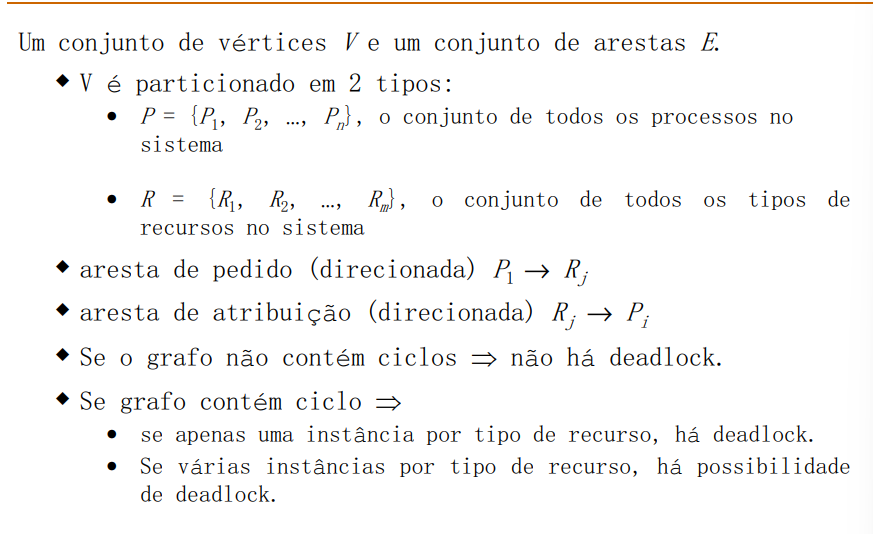
29/02/2024 – Programação concorrente

* Custo do sort = n^2/2
* Paralelamente (2 processos) = n^2/4 + n
* Podemos fazer os sorts em paralelo, entretanto o merge tem que esperar eles acarem , ou seja, **não podemos fazer o merge em paralelo**
* Bloco atômico 🡪 começa a instrução e ninguém pode entrar
* Abstração para problemas de sincronização:
  + A1 e A2 são atividades relativas aos processos P1 e P2
  + A1 e A2 são mutuamente exclusivas se não puderem se sobrepor
  + Um deles será suspenso(sessão critica)
* Exclusão mutua:
  + A abstração é expressa como sequencia: comandos, pre-protocolo, seção critica e pos protocolo
  + Deadlock
  + Lockout / Starvation
* Busy waiting:
  + Um processo aguarda sua vez de entrar em seção critica
  + Testando periodicamente se algum outro processo já executa uma seção critica

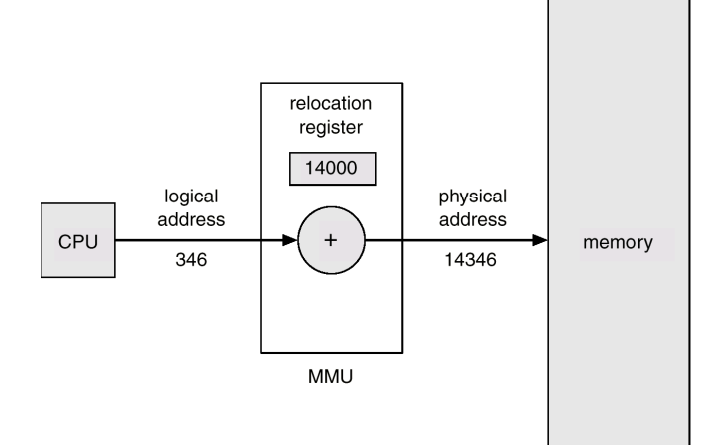
14/03/2024

* Semáforo:
  + Simples de implementar
  + Resolução de Paralelismo
  + É um inteiro que assume valores não negativos
  + Operações: wait and signal
  + wait(s) e signal(s) são operações primitivas como Load e Store
  + São mutualmente exclusivas quando atuam no mesmo semáforo
  + signal(s) não especifica que processo é acordado quando mais de um processo está suspenso no mesmo semáforo
  + Quando dois processos dão wait ao mesmo tempo🡪 Atômico(indivisível ) , porém podemos escalonar
* Produtor x consumidor
  + Problema:
    - Produtor armazena os dados até que consumidor esteja pronto
    - pelo consumidor que não pode processar dados que não foram gerados pelo produtor
  + Solucão: rendez-vouns

21/03/24 - Deadlock

* Um conjunto de processo bloqueados , cada qual mantendo um recurso e esperando receber um recurso mantido por outro processo de conjunto
* Pensar em engarrafamento em ponte.:
* Para ter deadlock:
  + Exclusao multipla
  + Posse e espera
  + Não preempção
  + Espera circular
* Representação usando grafos:
* Como tratar:
  + Garantir que o sistema nunca entrará em estado de deadlock.
  + Permitir que o sistema entre em deadlock e este seja recuperado. 1.11
  + Ignorar o problema e fingir que deadlocks nunca acontecerão; usados pela maioria do S.O.s, incluindo o UNIX!!!

Gerencia de memória

* Programas devem ser trazidos a memória alocados a um processador para serem executados
* Fila de entrada --|> coleção de processos no disco esperando para serem trazidos memoria e executados
* Endereço logico x físico:
  + Endereço lógico – gerado pela CPU; também denominado endereço virtual.
  + Endereço físico – endereço visto pela unidade de memória.
  + Exemplo carteira da sala: vários então na primeira logicamente , porem fisicamente não
* Unidade de Gerenciamento de Memoria:
  + Dispositivo de hardware que mapeia o endereço virtual para o físico.
  + O valor no registro de relocação é adicionado a todo endereço
  + Checa limites e registrador inicial
* Swapping:

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente07/05/2024 – Memória virtual

* Memória virtual🡪 separação da memória logica da física
  + Apenas parte do ograma precisa na memória de execução
  + O espaço de endereçamento logico pode ser muito maior que o espaço de endereçamento físico
  + Precisa permitir que as paginas sejam movidas para memoria
* Pode ser implementada :
  + Página sob demanda
  + Segmentação sob demanda
* Páginas:
  + Primeira referência 🡪 page fault