ANNA JÚLIA FERNANDES BARBOSA

Capitulo 13

1- Nenhuma medida é usada para evitar um impasse. As tarefas executam normalmente suas atividades, alocando e liberando recursos de acordo com a necessidade. Quando ocorrer um impasse o sistema reconhece-o determina suas tarefas e recursos e toma medidas para desfaze-lo.

2- A)T1 t2
R1
R2
R3
R4
T3
T4
b)t1
t2
r1
r2
r3

3- 23

Capitulo 14

1- Codificação: programa escolhe a posição de cada variável e do código do programa (Sistemas embarcados em linguagem de máquina).

Compilação: compilador escolhe a posição das variáveis na memória, código fonte faz e parte do programa deve ser conhecido no momento da compilação para evitar conflito em endereços na memória. Ligação: compilador gera símbolos que representem as variáveis.

Carga: define os objetos de variáveis e funções de carga do código em memória para lançamento de novo processo.

Execução: são analisados e convertidos pelo processador para a memória final (real).

2- Duas seções de tamanho variável (stack e heap) estão dispostas em posições opostas e vizinhas à memória livre. Dessa forma, a memória livre disponível ao processo pode ser aproveitada da melhor forma possível, tanto pelo heap quanto pelo stack, ou por ambos.
STACK
HEAP
BSS
DATA
TEXT

Capitulo 15

1- Endereços de memória gerados pelo processador na medida em que executa algum código, são chamados de endereços lógicos, porque correspondem à lógica do programa, mas não são necessariamente iguais aos endereços reais das instruções e variáveis na memória real do computador, que são chamados de endereços físicos.
2- Segmento 0 1 2 3 4
   Base 44 200 0 2.000 1.200
   Limite 810 200 1.000 1.000 410
3- 2 3
   6 –

   4 5
   9 –

   6 7
   2 –

   8 9
   0 5

   10
   –

   11 12
   –
   –

   13
   7

   14 15
   –
   1

Capitulo 16

1- B) Se usarmos Worst-fit, o tamanho final do buraco B4 será de 15 Mbytes.

Capitulo 17

1- Falta de página é o nome dado à quando uma informação buscada não está na memória, sendo necessário carrega-la do disco para a memória. Neste caso, o processador se comunica com o controlador para gerar uma interrupção. O sistema operacional coloca o processo no estado de bloqueado, faz o pedido de E/S para trazer mais páginas (ou página). Enquanto isso, o sistema operacional escolhe e deixa outro processo e o executa. Quando a página é trazida para a memória, o processo que gerou a falta de páginas entra na fila dos processos em estado pronto e o controlador envia uma interrupção. Essa interrupção significa que o evento foi realizado.

Capitulo 19

1- Para obter rapidez, a maioria dos sistemas operacionais implementa o tratamento de interrupções em dois níveis: um tratador primário (FLIH - First-Level Interrupt Handler) e um tratador secundário (SLIH - Second-Level Interrupt Handler). O tratador primário, também chamado hard/fast interrupt handler ou ainda top-half handler, é ativado a cada interrupção recebida e executa rapidamente, com as demais interrupções desabilitadas. Sua tarefa consiste em reconhecer a ocorrência da interrupção junto ao controlador de interrupções, criar um descritor de evento contendo os dados da interrupção ocorrida, inserir esse descritor em uma fila de eventos pendentes junto ao driver do respectivo dispositivo e notificar o driver. O tratador secundário, também conhecido como soft/slow interrupt handler ou ainda bottom-half handler, tem por objetivo tratar os eventos pendentes registrados pelo tratador primário. Ele geralmente é escalonado como uma thread de núcleo com alta prioridade, executando quando um processador estiver disponível. Embora mais complexa, esta estrutura em dois nível traz vantagens: ao permitir um tratamento mais rápido de cada interrupção, ela minimiza o risco de perder interrupções simultâneas; além disso, a fila de eventos pendentes pode ser analisada para remover eventos redundantes (como atualizações consecutivas de posição do mouse).