

Sistema Sistemas Operacionais
Márcia Silvio Alves de Barros

202100360105

1 - As principais funções de um SO, de acordo com o livro de Tanenbaum, são:

- * Fornecer uma interface de alto nível para os usuários, agindo como uma máquina estendida e fazendo forte uso da abstração para administrar e tornar amigável a complexidade real do hardware.
- * Ser um gerenciador de recursos, administrando processos (escalonamento de processos), controlando os dispositivos de entrada e saída, gerência da memória e dos processadores. E tudo isso é feito de maneira que visa garantir eficiência e segurança no utilização dos recursos.

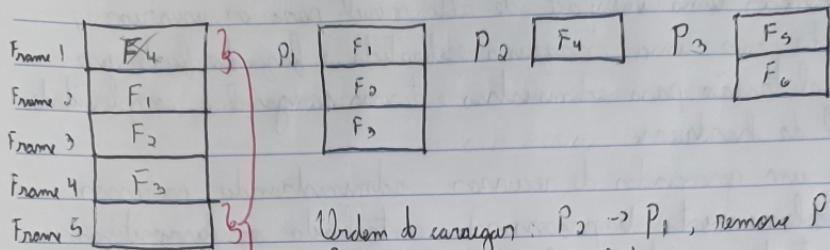
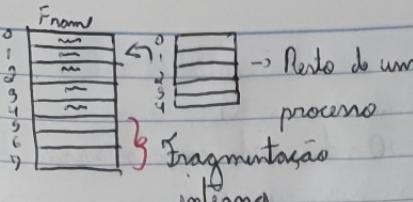
2. Chamadas de sistema são uma interface entre o sistema operacional e os programas do usuário. Fornecem uma maneira para o usuário acessar serviços do SO. Chamadas de sistemas precisam de instruções do tipo TRAP, que basicamente troca o sistema para Modo kernel.

Existem chamadas para diversas funcionalidades, a seguir estão as três categorias principais e exemplos:

- 1º gerência de processos: fork() e exit(status)
- 2º gerência de arquivos: open(f, t, ...), lseek(fd, offset, whence)
- 3º gerência de diretórios/arquivos do sistema: mkdir(), rmdir()

3 - Fragmentação interna ocorre quando um processo não utiliza o bloco de memória todo que lhe foi alocado. É como se, dentro do frame, sobrassem endereços sem uso.

• Fragmentação externa é quando existe memória de forma não contínua.



Ordem de carregar: P₂ → P₁, remove P₂ → P₃
 Após remover P₂, existe espaço para P₃,
 mas de forma descontínua e então é uma
 demonstração de fragmentação externa

Fragmentação
externa

4- Endereços virtuais são gerados pela CPU para que no ambiente de memória virtual os processos não precisem dos endereços físicos. Esse conceito é importantíssimo pois além de permitir uma abstração a ponto de cada processo no seu endereçamento virtual específico (garantindo segurança nos acessos e uso eficiente da memória) faz com que o espaço virtual seja maior que o físico, isto é, processos podem ser maiores que a memória física (os algoritmos de falta de página cuidam dessas situações de troca) já que existe essa abstração lógica de endereços.

Endereços físicos são um ponto específico real na memória RAM do computador. Sabendo que para que os endereços virtuais sejam utilizáveis, devem ser convertidos para os endereços físicos adequados.

5. Em sistemas paginados cada processo tem seu endereço virtual específico e quando esse endereço é convertido para o endereço físico correspondente, não deve ocorrer acessos nas áreas de outros processos por questões de segurança e integridade de dados.

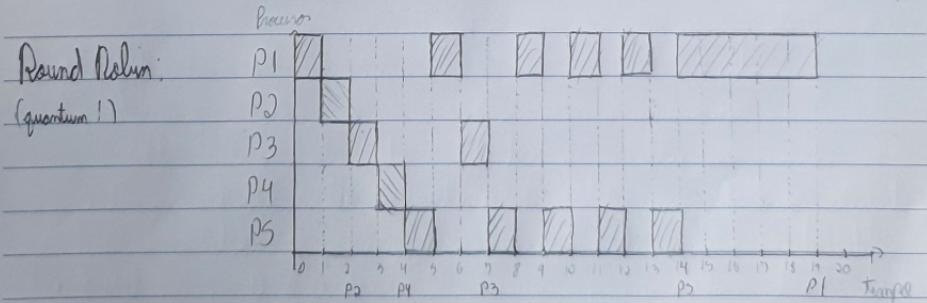
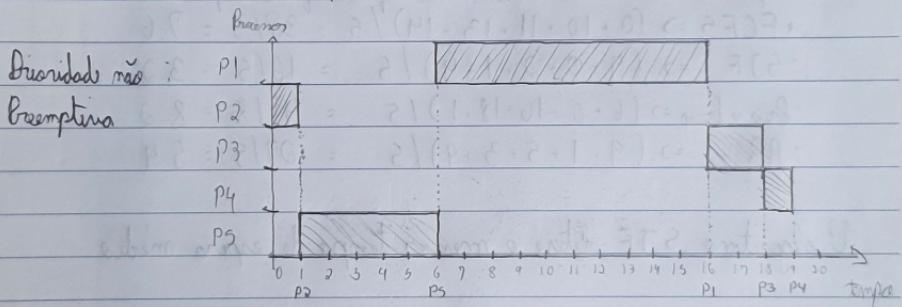
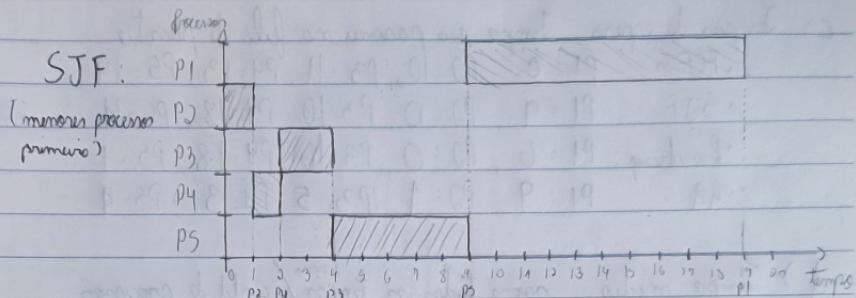
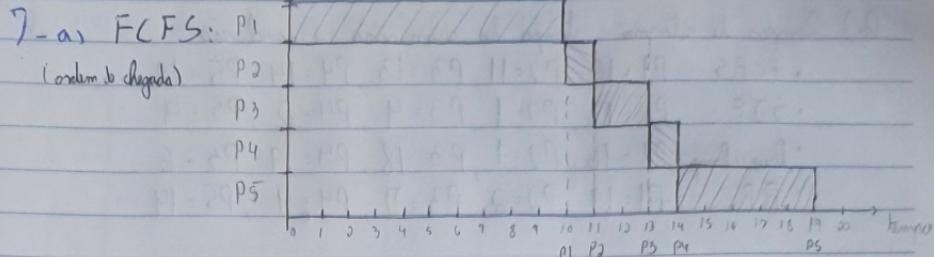
O SO pode criar e gerenciar áreas compartilhadas de memória para informações que não gerariam conflitos. Por exemplo, informações apenas para leitura, executáveis e outras coisas que vários processos podem precisar mas não mutáveis poderiam ser compartilhadas na memória, enquanto áreas de escrita que não mutáveis devem ser exclusivas para cada processo.

Isto evitaria blocos de processos duplicados desmembrando, portanto um uso mais eficiente da memória, afinal se 10 processos usam o mesmo executável não preciso escrever em executável 9 regras extras sendo que 1 ocorrência e acesso compartilhado bastaria.

6. • CPV 13%, DISCO 97%: Como o disco está sendo muito acedido, deve estar ocorrendo muitos page faults e a CPV está sendo mal utilizada. A paginação não está sendo útil, aumentar o grau de multiprogramação só ia piorar a situação (deixaria a mais page faults e a CPV ficaria mais subutilizada)

• CPV 87%, DISCO 3%: Nesse caso a CPV está sendo bem utilizada, com poucos acessos ao disco, o que indica poucas page faults. Provavelmente aumentar a multiprogramação aqui aumentaria uso da CPV e manteria o DISCO no nível semelhante, já que a política de paginação nesse caso está sendo muito útil.

• CPV 13%, DISCO 3%: Nesse caso em que ambos estão sendo pouco usados, aumentar o grau de multiprogramação seria bom pois deixaria a CPV menos ociosa. A técnica de paginação pode estar sendo útil ou não, depende de como seriam os acessos ao disco com uma CPV menos ociosa.



b) Tempos de retorno : tempo da submissão à conclusão

- FCFS : $P_1 = 10, P_2 = 11, P_3 = 13, P_4 = 14, P_5 = 19$
- SJF : $P_1 = 19, P_2 = 1, P_3 = 4, P_4 = 2, P_5 = 9$
- Fio e Preemp : $P_1 = 16, P_2 = 1, P_3 = 18, P_4 = 19, P_5 = 6$
- RR : $P_1 = 19, P_2 = 2, P_3 = 7, P_4 = 4, P_5 = 14$

c) Tempos de espera : tempo que passou na fila de pronto

- FCFS : $P_1 = 0, P_2 = 10, P_3 = 11, P_4 = 13, P_5 = 14$
- SJF : $P_1 = 9, P_2 = 0, P_3 = 2, P_4 = 1, P_5 = 4$
- Fio e Preemp : $P_1 = 6, P_2 = 0, P_3 = 16, P_4 = 18, P_5 = 1$
- RR : $P_1 = 9, P_2 = 1, P_3 = 5, P_4 = 3, P_5 = 9$

d) Tempo médio : soma todos os $\frac{\text{tempo de espera}}{\text{qntd de processos}}$

- FCFS $\Rightarrow (0 + 10 + 11 + 13 + 14) / 5 = 38 / 5 = 7.6$
- SJF $\Rightarrow (9 + 0 + 2 + 1 + 4) / 5 = 16 / 5 = 3.2$
- Fio e Preemp $\Rightarrow (6 + 0 + 16 + 18 + 1) / 5 = 41 / 5 = 8.2$
- RR $\Rightarrow (9 + 1 + 5 + 3 + 9) / 5 = 27 / 5 = 5.4$

O algoritmo SJF obtém o menor tempo de espera médio.

8. De acordo com a tabela da página 104 (135 no PDF), vemos que espaço de endereço, variáveis globais, arquivos abertos, processos filhos, alarmes pendentes são alguns itens compartilhados por todas as threads do processo. Enquanto contadores de programa, registradores, pilha e estados são alguns itens privados e específicos para cada thread.

Portanto a resposta é : b) Memória do heap

c) Variáveis globais

9- LRU: ~~1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6~~

1	1	1	1
2	2	2	2
3	6	6	6
4	4	7	7
5	5	5	3

7 page faults

FIFO: ~~1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6~~

1	6	6	6	6	6	6
2	1	1	1	1	1	1
3	3	3	2	2	2	2
4	4	4	4	3	3	3
5	5	5	5	5	7	

9 page faults

10. Paginação em vários níveis usa uma hierarquia de tabelas de páginas. Pode ser interessante quando o espaço de endereçamento virtual é muito grande e assim numa organização hierárquica apenas as tabelas de páginas atualmente em uso precisam ser carregadas, economizando espaço na memória.

O funcionamento:

1. Divide-se o endereço virtual em várias partes, em que cada uma corresponde a um nível na hierarquia da tabela de páginas.
2. O primeiro nível da tabela de páginas é indexado pela primeira parte do endereço virtual. Aponta para a tabela do segundo nível
3. O segundo nível é indexado pela segunda parte do endereço virtual. Isso aponta para o terceiro nível e assim por diante.
4. O último nível da tabela de páginas é indexada pela última parte do endereço virtual e aponta para a página de memória física
5. A parte final do endereço virtual é um offset dentro da página