Lista de Exercícios 1 - Sistemas Operacionais

812839 - Vinícius Miranda de Araújo

1) As instruções que são permitidas apenas em modo super visor são:

a) Desabilitar todas as interrupções: se um usuário fizer isso, pode travar o sistema.

c) Alterar o relógio da máquina: se um usuário fizer, pode causar problemas de consistência, como em logs e segurança

d) Mudar o mapa de memória: se um usuário fizer, permitiria acesso a áreas restritas de memória ou corrupção de dados de outros jobs.

2) uid = user id, gid = group id, permissões rwx, r-x, ---.
r = read, w = write, x = execute.
(rwx: dono; r-x: grupo; ---: outros)

Como o uid = 6 é um usuário diferente do dono, mas está no mesmo grupo do arquivo, então ele usa as permissões do grupo e não de "outros". Portanto, esse usuário pode ler e executar o arquivo, mas não pode escrever.

3) O SIGALARM é um sinal que notifica um processo quando um temporizador pré-definido por esse processo expirou, ou seja é uma forma de um programa "avisar" a si mesmo que um determinado tempo terminou.
É possível enviar esse sinal com a chamada de uma função como a alarm(time), em C, ou usando o comando kill -SIGALRM <pid>.

Ignorar, geralmente, não faz sentido, pois serve para indicar a expiração de tempo, mas o processo pode tratá-lo em vez de aceitar o comportamento padrão.

4) Para um chaveamento de processos feito por hardware, seriam necessárias as informações dos registradores da CPU, PC, SP, registradores de estado e PCB. O processo por hardware seria: o hardware decide que é hora de trocar, salva o contexto do processo atual, seleciona o próximo processo, por meio de um escalonador embutido, e restaura seu contexto.

5) Em um sistema com threads, cada thread possui sua própria pilha. Isso porque, embora as threads de um mesmo processo compartilhem o mesmo espaço de endereçamento, cada thread tem um fluxo de execução independente.

6) Considerando que $n \leq 10$ e com overhead próximo de zero, cada um dos n processos recebe, em média, $1/n$ da capacidade da CPU. Assim, como T é o tempo que o processo levaria sozinho e $1/n$ é a fração de tempo de CPU que ele recebe, o tempo total de execução será:

$$T'(n) = \frac{T}{1/n} = T \times n.$$

7) Se um processo aparecesse mais de uma vez na lista, ele teria mais de uma fatia de tempo por ciclo, assim, executaria mais vezes podendo desbalancear o escalonamento. Isso faz sentido quando queremos aumentar a prioridade de um processo sem alterar o algoritmo ou acelerar o tempo de processos curtos.

8) eficiência = $\frac{\text{tempo útil}}{\text{tempo útil + overhead}} \rightarrow \eta = \frac{\min(Q,T)}{\min(Q,T)+S}$

a) $Q = \alpha$: $\eta = \frac{\min(\alpha,T)}{\min(\alpha,T)+S}$

b) $Q > T$: $\eta = \frac{T}{T+S}$

c) $S < Q < T$: $\eta = \frac{Q}{Q+S}$

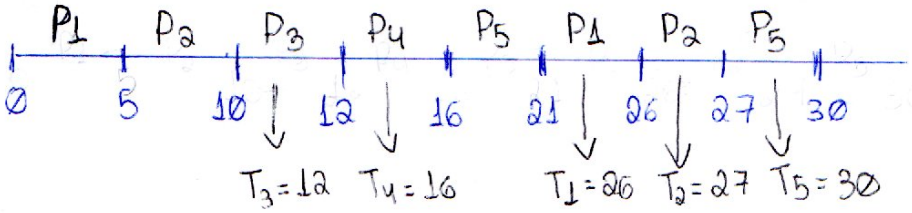
d) $Q = S$: $\eta = \frac{Q}{Q+Q} = \frac{1}{2}$

e) $Q \rightarrow \emptyset$: $\eta = \frac{Q}{Q+S} \rightarrow \emptyset$

9)

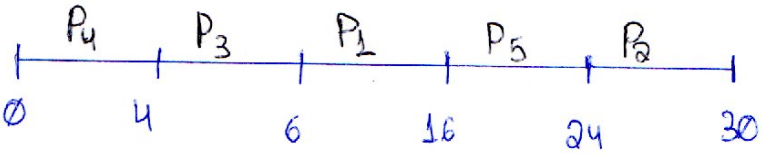
Job	P_1	P_2	P_3	P_4	P_5
Tempo de execução (min)	10	6	2	4	8
Prioridade (1 é o maior)	3	5	2	1	4

a) Round-robin:



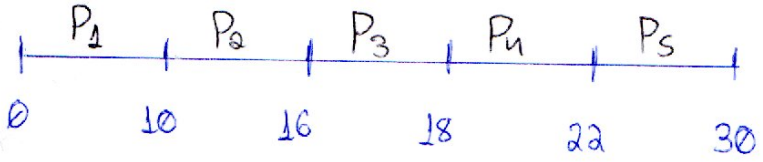
$T' = 12+16+26+27+30 = 111/5 = 22,2$ de médio

b) Escalonamento com prioridade:



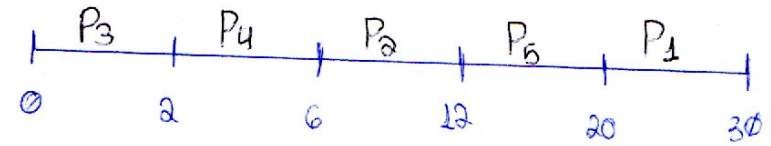
$T' = 4+6+16+24+30 = 80/5 = 16$ de médio

c) FCFS: (first come, first served)



$T' = 10 + 16 + 18 + 22 + 30 = 96/5 = 19,2$ de médio

d) SJF: (shortest job first)



$T' = 2 + 6 + 12 + 20 + 30 = 70/5 = 14$ de médio

10)

Evento	Período (ms)	CPU (ms)
E_1	50	35
E_2	100	20
E_3	200	10
E_4	250	X

Condição
↳ cada tarefa deve terminar antes que seu período se repita.

$$\text{Utilização } (U) = \frac{35}{50} + \frac{20}{100} + \frac{10}{200} + \frac{X}{250}$$

$$U = 0,7 + 0,2 + 0,05 + \frac{X}{250}$$

$$U = 0,95 + \frac{X}{250}$$

$$0,95 + \frac{X}{250} \leq 1$$

$$\frac{X}{250} \leq 1 - 0,95$$

$$\frac{X}{250} \leq 0,05$$

$$X \leq 0,05 * 250$$

$$X \leq 12,5 \text{ ms}$$

O maior valor de x que mantém o sistema escalonável é:

$X = 12,5$ ms