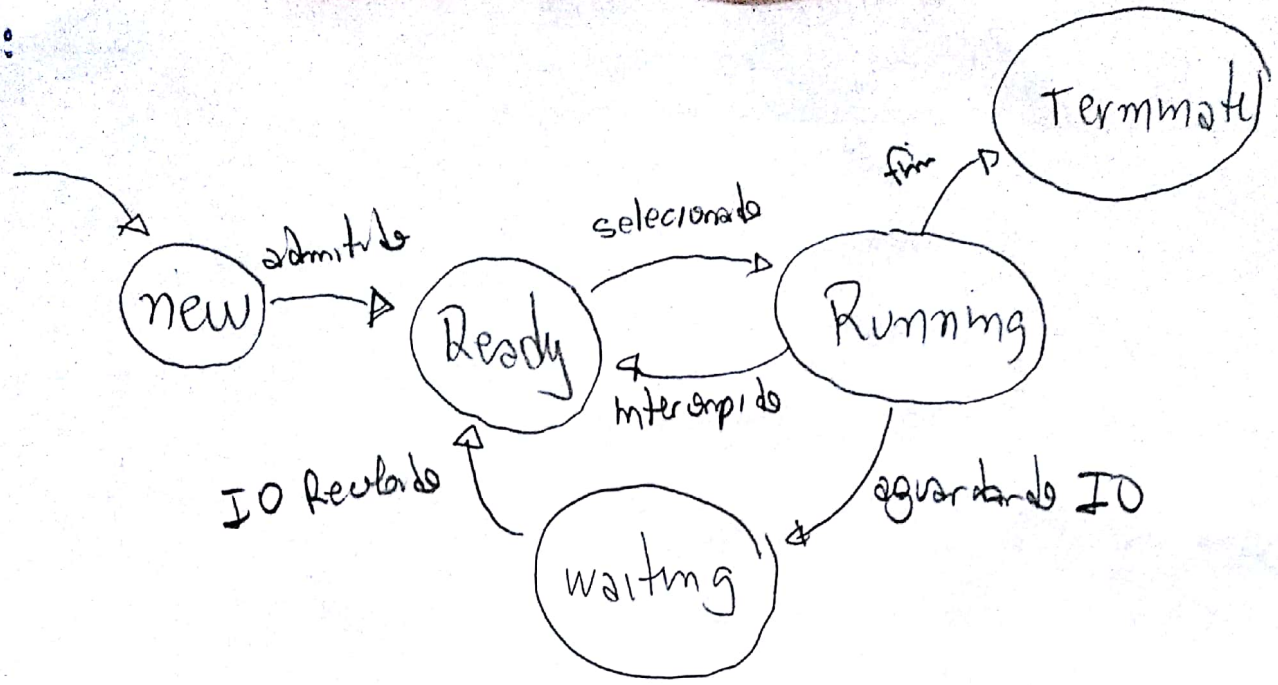
Ex 1:



Ex 2:

Quando um processo está em execução e o SO decide que é hora de dar tempo de CPU para outro processo, é necessário que alguns passos sejam seguidos:

1- O SO executa uma interrupção e o processo atual para de ser executado
2- SO salva o PCB do processo em execução
3- PCB do novo processo é carregado
4- SO inicia a execução do novo processo

Ex 3:

Para minimizar o tempo de execução de vários processos escalonados em uma CPU, o algoritimo ótimo é o "SJF" ou seja, executa os processos de acordo com o seu tempo de execução, do menor para o maior. O problema desta abordagem é que é praticamente impossível saber exatamente qual será o tempo de execução de um processo antes do mesmo ser executado.

4)

	L	A	FCFS TC	FCFS TE	SJF TC	SJF TE	RR TC	RR TE	MLFQ TC	MLFQ TE
1	50	0	50	0	150	100	150	140	150	100
2	40	0	90	50	100	60	140	120	143	103
3	30	0	120	90	30	0	120	90	122	92
4	20	5	135	115	55	35	88	68	94	74
5	10	10	140	130	30	20	47	37	44	34

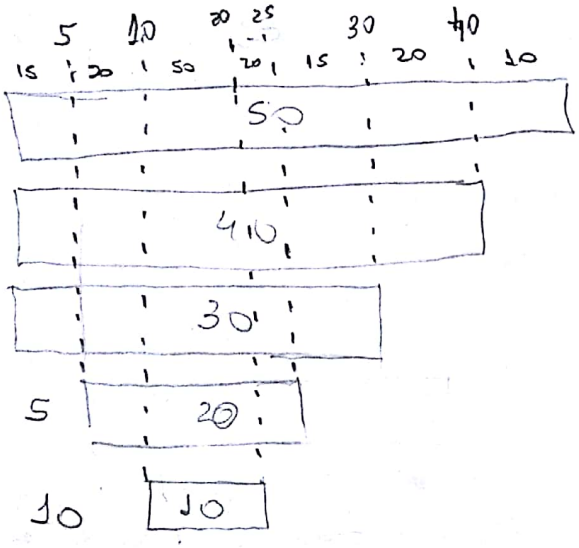
$P_1 = 3.5 + 4.5 + 5.10 + 4.5 + 3.5 + 2.10 + 10$

$P_2 =$

$P_3 =$

$P_4 = 4.5 + 5.10 + 4.5 = 90$

$P_5 = 5.10 = 50$



(5)

Os quatro requisitos para a existência de deadlock são:

a) exclusão mútua: um processo deve reter um recurso no modo não compartilhado;

b) Manter e esperar: o processo que tem a posse do recurso mutuamente exclusivo deve estar esperando outro recurso;

c) Não preempção: Somente o processo de posse do recurso pode liberá-lo mesmo

d) Espera circular: o processo p_0 depende que p_1 libere um recurso e P_1 espera que p_0 libere outro recurso.

(6) $P = \{ P_1, P_2, P_3 \}$

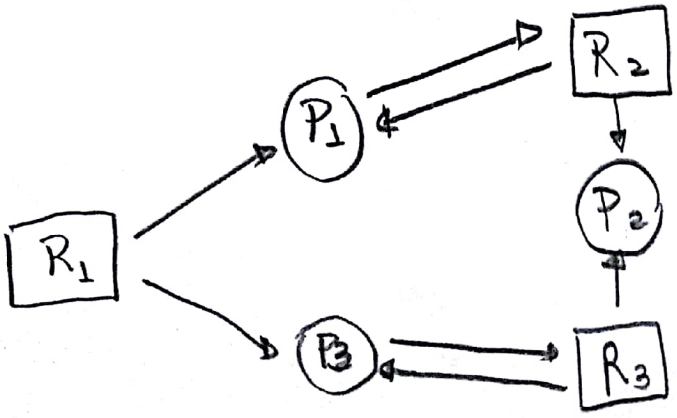
$R = \{ R_1, R_2, R_3 \}$

$A = \{ R_1 \to P_1, R_1 \to P_3, R_2 \to P_1, R_2 \to P_2, R_3 \to P_2, R_3 \to P_3,$
$P_1 \to R_2, P_3 \to R_3 \}$

a) $P_1 \to R_2 \to P_2$

b) $P_3 \to R_3 \to P_2$

O gráfico não apresenta deadlock pois ele não possui espera circular que é uma condição necessária para a existência de deadlock



Quando P_2 liberar R_2 e R_3 os processos P_1 e P_3 poderão continuar sua execução

(2)