Teste Modelo 2017-2018

① a) i) sem_down -> bloqueante quando 0
sem_up -> não bloqueante

Para já

[0] [1] [1] -> solução mínima

Entra em deadlock se não conseguir fazer o down.

ii) BCABCACBA / BCACBABCA

b) wait -> sempre num while (espera que condição seja true) (bloqueante)

signal -> se bloqueou passa sinal se não perde-se o sinal.

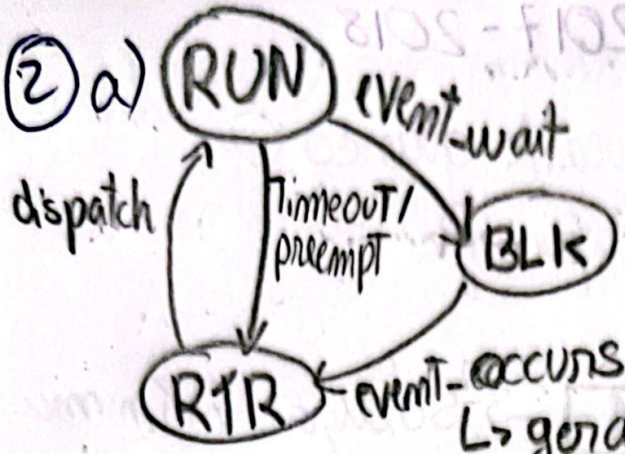
c) down(semx)	up(semx)
⑦ while(condx) cond_wait(vcondx, &mtx) condx=true ⋮ mutex_unlock(&mtx)	⑧ mutex_lock(mtx) condx = false signal(&vcondx) ⋮ mutex_unlock(&mtx)

(cond, vcond) ↳ pthread_cond_T vcondx;

↓

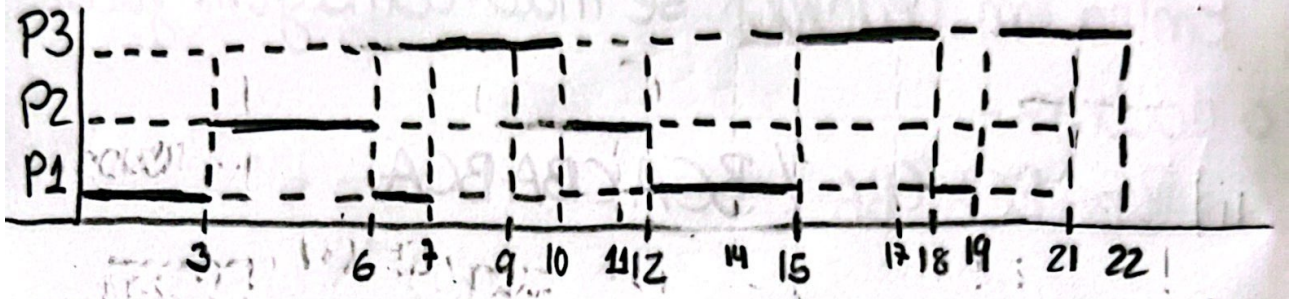
bool condx Preciso também pthread_mutex_T mtx

② a)



↳ gera uma interrupção do CPU

b)



Turnaround: P1 → 19 - 0 = 19

P2 → 12 - 2 = 10

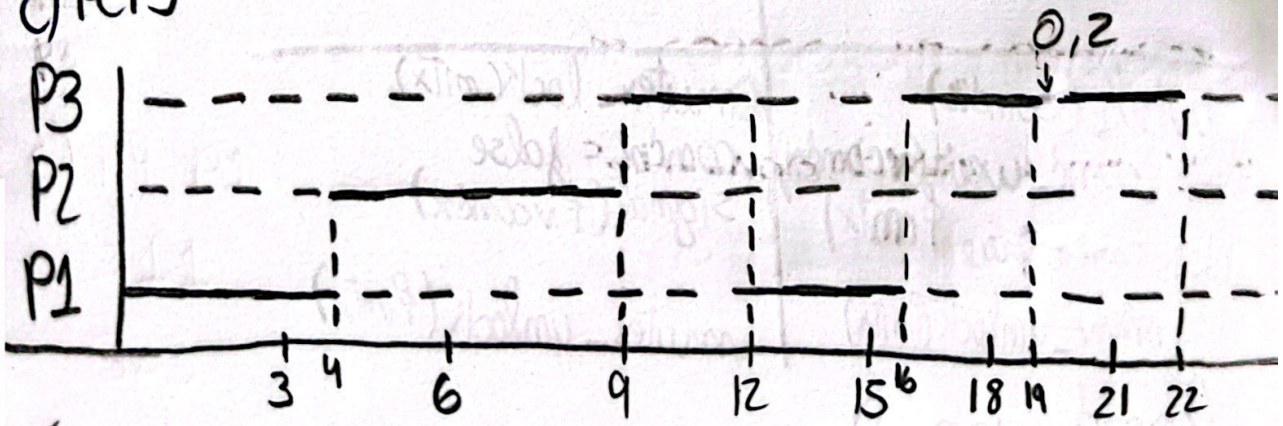
P3 → 22 - 4 = 18

Ready: 9 - 0,4 = 8,6 ← P3

P1 → 11 - 0,5 = 10,5

P2 → 4 - 0 = 4

c) FCFS



Turnaround: P1 → 16 - 0 = 16

P2 → 9 - 2 = 7

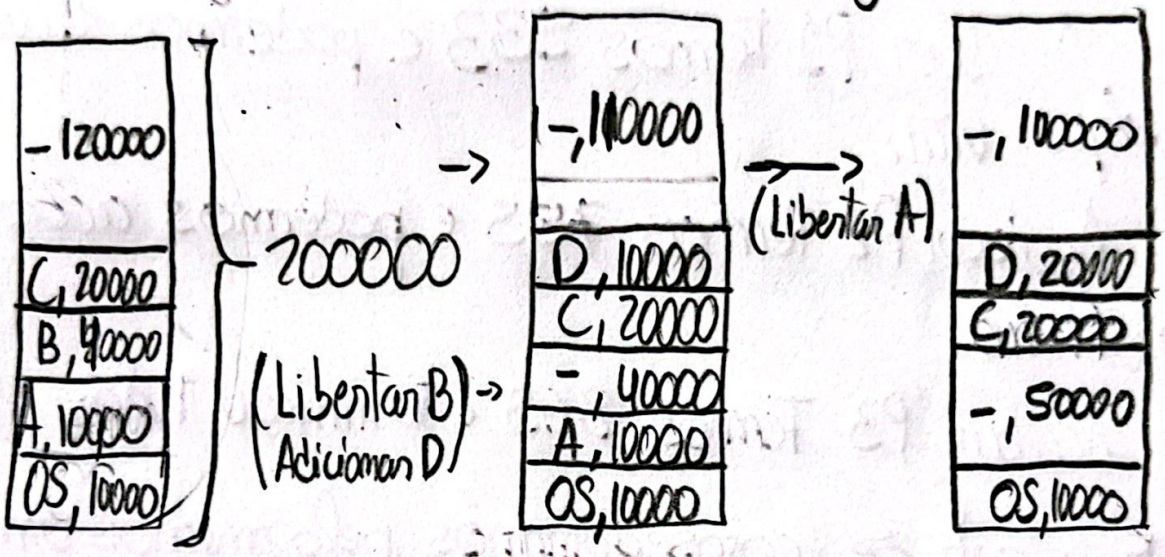
P3 → 22 - 4 = 18

③ b) i) base: tem o endereço inicial
limite: limita cada segmento (medição)

ii) Função dispatch, só lê informação do PCT e carrega os registos com a informação necessária.

iii) Compara o endereço lógico com o limite, se exceder segmentation fault, se não base + lógico.

c)



L1:	-	100000	100000	-
L2:	D	80000	20000	-
L3:	C	60000	20000	L2
L4:	-	10000	50000	L1

free L4
L3

④ a) Deadlock avoidance

b) Tenho condições para Terminar P4.
Atraso pedidos de P1, P2, P3 até P4 Terminar
Acabando P4 temos 221, capaz de Terminar P1.

Acabar P1 Temos 533 e podemos acabar qualquer um.

Acabar P2 Temos 755 e podemos acabar o último.

Acabar P3 Temos 865 e terminou tudo.

É saye se conseguirmos pelo menos uma sequência.

É unsaye se não conseguirmos nenhuma sequência.

É deadlock se nem conseguir começar.

c) P4, P1 (Atribuindo logo 1 recurso do Tipo R2 ao P3)
211
523
R: É possível