SO exame 2017/2018

1-

a)

i) sem1 – 0; sem2- 1; sem3-1

ii) CBACBACBA; BCACBABCA

b) Ao utilizar up, o semáforo ficará up até ser posto down, o valor não desaparece, enquanto que numa implementação com signal, se nenhuma thread estiver à espera (wait) daquele sinal específico, o sinal não será “aproveitado”. Para além disso, o down não bloqueia necessariamente o código (o semáforo pode já ter um valor > 0), enquanto que o wait bloqueará sempre o código, uma vez que signals que já ocorreram não se aplicam.

c) todos o código dentro dos fors fica entre um lock(mux) e um unlock(mux); cada sem\_down é substituído por while(not cond) waitcond(vcond,mux) cond--; cada sem\_up é substituído por cond++ e signal(vcond)

2-

a)RUN->timeout->READY (o time quantum do processo acaba); RUN->preempt->READY (um processo de maior prioridade surge no estado READY); RUN->wait\_event->BLOCKED (o processo requer que um evento ocorra, então é bloqueado); BLOCKED->event->READY (o evento que tinha sido esperado ocorre); READY->dispatch->RUN (o processo que estava no CPU sai de lá, disponibilizando-o para outro processo)

b) P1(3), P2(3), P1(1), P3(3), P2(3), P1(3), P3(3), P1(1), P3(3)

c) turnaround(P1) = 16, turnaround(P2) = 9 – 2 = 7, turnaround(P3) = 16+3+0.2+3-4 = 18.2

3-

a) Na memória real o processo ou está completamente na memória ou na área de swap e ocupa uma área contígua, ou seja, está de modo contínuo em memória. Do ponto de vista lógico, todos os processos começam no endereço 0, no entanto, há um offset entre o endereço real e o endereço lógico. Numa partição fixa, há um número X de partições que têm um tamanho qualquer à partida, enquanto que num sistema com partições variáveis a memória não terá partições à partida, as partições serão criadas à medida que a memória for alocada, sendo que estas partições podem variar em tamanho.

b)

i) o registo base guarda o endereço do primeiro endereço físico alocado para o processo, enquanto que o registo limite tem o último valor possível para um endereço lógico.

ii) os registos serão alterados quando o processo que está a ser executado mudar

iii) O endereço lógico será comparado com o valor registado no registo limite (a). Se o valor for maior, será gerada uma exceção, uma vez que o registo que o processo está a aceder a não faz parte do seu espaço de memória (n). Se não (s), o endereço lógico será somado ao valor residente no registo base, dando isso origem ao endereço físico do endereço lógico.

c) kernel(0,10000) C(60000, 80000) D(80000,100000)

partições livres -> (10000, 60000), (10000,200000)

4-

a) Deadlock avoidance – a entidade requer que os recursos necessários sejam pedidos antes da execução do processo, o deadlock é prevenido pelo gestor de recursos em vez dos processos

b) P4 -> P1 -> P2 -> P3 (estado safe)

c) Sim, uma vez que essa atribuição de recursos não impossibilita a execução de todos os processos (alocação de R2 para P3 -> P4 -> P1 -> P2 -> P3)

SO exame 2016/2017

1-

a) A ocorrência de deadlock pressupõe a satisfação de exclusão mútua, hold and wait, inexistência de preemption e circular wait.

Exclusão mútua é quando apenas um processo pode usar um certo recurso de cada vez. Hold and wait é quando um processo está a reter um recurso e tem de esperar que outro recurso seja liberto para libertar os que está a usar. Inexistência de preemption significa que, uma vez que um recurso esteja a ser usado por um processo, esse mesmo é o único que o pode libertar. Espera circular ocorre quando processos de espera existem de tal maneira que um grafo de espera tem ciclos.

b) Exclusão mútua é garantida, uma vez que um talher não pode ser usado por dois filósofos ao mesmo tempo. Hold and wait também ocorre – quando um filósofo tem um garfo na mão só o porá para baixo quando acabar a refeição. Também não há elementos exteriores que façam um filósofo largar um garfo, logo,não há preemption. Finalmente, há espera circular graças a todas as regras já mencionadas acima – um exemplo seria todos os filósofos pegarem no garfo esquerdo, uma vez que todos ficariam à espera do garfo direito, mas para esses garfos ficarem disponíveis seria necessário esperar que pelo menos um filósofo comesse, que é impossível com apenas um garfo.

c) Downs só são feitos quando ambos os garfos estão disponíveis --- não haverá hold and wait porque o hold só é feito quando todos os recursos estão disponíveis

2-

a) Processos CPU intensivos são processos que, se estivessem num ambiente monoprocessador FCFS, correriam sem interrupções, enquanto que processos intensivos em IO são processos que requerem bloqueios gerados por interrupções.

b) Em ambas as políticas, o processo a ser atendido primeiro será aquele que está à espera há mais tempo, no entanto, em sistemas round robin os processos podem ser removidos de execução se excedessem o time quantum, enquanto que numa política FCFS o processo só é removido de execução quando ocorre uma interrupção ou termina a execução.

c) Em ambos a ideia de correr vários programas existe, no entanto, multiprocessamento requer mais do que um CPU, enquanto que multiprogramação não. Ainda mais, em multiprogramação a execução de vários programas ao mesmo tempo é meramente aparente, os processadores recebem vários programas que serão executados cada um no seu intervalo de tempo.