1. Um sistema operativo pode ser visto como um gestor de recursos num ambiente de concorrência, que recursos são esses e quais as entidades que se encontram em concorrência? Quais os objectivos principais do sistema Operativo nesta perspectiva?

   Como gestor de recursos creio que o que o SO tenta gerir são o CPU, espaço de memória e threads. As entidades em concorrência penso que sejam os processos. Os objectivos do SO como gestor de recursos são fornecer de forma adequada e controlada a alocação do processador, das memórias e de dispositivos por todos os programas que competem por recursos.

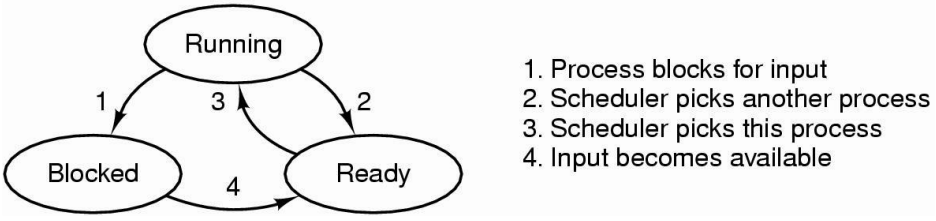
2. Diga o que entende por Multiprogramming, Multiprocessing e Escalonador do SO.

   Multiprogramming: Divide-se a memória em diversas partes com um Job alocado em cada uma delas, enquanto um Job espera a conclusão da sua operação I/O, um outro Job pode estar a utilizar o processador, a ideia é manter na memória simultaneamente uma quantidade de jobs suficiente para ocupar o processador a 100% e aproveitar ao máximo.

   Multiprocessing: Consiste na capacidade que um sistema operacional tem, podendo executar simultaneamente dois ou mais processos. Pressupõe a existência de dois ou mais processadores. Difere da multitarefa, pois esta simula a simultaneidade, utilizando-se de vários recursos, sendo o principal o compartilhamento de tempo de uso do processador entre vários processos.

   Escalonador do SO: é uma actividade organizacional feita pelo escalonador do CPU ou de um sistema distribuído, possibilitando executar os processos mais viáveis e concorrentes, priorizando determinados tipos de processos, como os de I/O Bound e os computacionalmente intensivos.

3. Ilustre, com o auxilio dum diagrama desenhado, quais os estados básicos possíveis para um processo e explica como é feita a transição entre esses estados.



   Ready: Processo disponível para execução, aguardando sua vez.

   - Ao ser selecionado, sofre transição para o estado Running.
   - O sistema passa a interpretar o código.

   Running: Processo actualmente em execução.

- Caso a fatia de tempo máximo de ocupação do processador seja atingida, o processo é interrompido, seus códigos/dados são mantidos na memória ou deslocados para a área de Swap, o processo retorna para a fila de Ready's de acordo com as regras de escalonamento. A transição é feita para o estado Ready.
- Caso o processo realize uma chamada de sistema em que o tempo necessário para cumprí-la seja longo, o processo é interrompido, seus códigos/dados são mantidos na memória ou deslocados para a área de Swap e sofre transição para o estado Blocked. Caso a chamada de sistema seja imediatamente executada, o processo não sofre transição de estado.

Blocked: O processo está aguardando que uma chamada de sistema seja concluída.

- Uma chamada de sistema, ao ser concluída por um processo do sistema, gera uma sinalização (Evento) para o processo ser transferido para o estado de Ready.

4. Considere o seguinte pseudo-codigo:

```
#define N 100;
typedef int semaphoro;
semaphoro mutex = 1;
semaphoro empty = N;
semaphoro full = 0;

void producer(void){
        int item;

        while(TRUE){
                item = producer_item();
                down(&mutex);
                down(&empty);
                insert_item(item);
                up(&full);
                up(&mutex);
        }
}

void consumer(void){
        int item;

        while(TRUE){
                down(&full);
                down(&mutex);
                item = remover_item(item);
```

```
            up(&empty);
            up(&mutex);
            consumer_item(item);
        }
}
```

a. Explique o motivo do semáforo mutex no código ilustrado.
   O semáforo mutex é utilizado para controlar os acessos as regiões criticas.
b. A solução apresentada no código está errada. Descubra porquê e justifique. Altere o código para que a solução seja correcta.
   Na solução o down ao semaforo mutex, está a ser feito antes dos down aos semaforos empy e full, o que não é correcto.
   Resumindo, a solução correcta seria:

```
#define N 100;
typedef int semaphoro;
semaphoro mutex = 1;
semaphoro empty = N;
semaphoro full = 0;

void producer(void){
        int item;

        while(TRUE){
            item = producer_item();
            down(&empty);
            down(&mutex);
            insert_item(item);
            up(&full);
            up(&mutex);
        }
}

void consumer(void){
        int item;

        while(TRUE){
            down(&full);
            down(&mutex);
            item = remover_item(item);
            up(&empty);
            up(&mutex);
            consumer_item(item);
        }
```

5. Considere a seguinte lista de problemas relacionados com a utilização do computador:
    a. Perda de dados do disco;
    b. Privacidade dos dados (segurança);
    c. Robustez do SO;
    d. Rede de dados lenta;
    e. Tempo gasto com instalações e configurações de Software;

    Indique, para um (e um só) destes problemas á sua escolha (indicando qual escolheu), uma solução para cada uma das seguintes estratégias:

        i. Detectar e recuperar o problema;
        ii. Evitar o problema;
        iii. Prevenir o problema;

    Não esqueça de mencionar o problema que escolheu e de justificar a solução/estratégia indicadas adequadamente.

    Problema escolhido:

    c. Robustez do SO;

    Solução/estratégia:

    i- Detecção e recuperação →Deixar deadlocks ocorrer, detectá-los e actuar.
    ii- Evitar o problema →Desviar dinamicamente, através da cuidadosa alocação de recursos.
    iii- Prevenir o problema →Negar 1 das 4 condições necessárias para criar 1 deadlock.