1-) É o conjunto de programas que gerenciam recursos, processadores, armazenamento, dispositivos de entrada e saída e dados da máquina e seus periféricos. O sistema que faz comunicação entre o hardware e os demais softwares. Suas principais funções são: definição da inferface com o usuário, compartilhamento de hardware entre usuários. Compartilhamento de dados entre usuários, gerenciamento dos dispositivos de entrada e saída, tratamento e recuperação de erros e etc.

2-)
a-) Um processo é uma instância de um programa que está sendo executada. Ele contém o código do programa e sua atividade atual.
b-) É um sistema operacional que permite que apenas um programa (tarefa) seja executada de cada vez.
c-) É um sistema operacional que permite que mais de um programa seja executado “ao mesmo tempo”, utilizando ao máximo a CPU, por meio de escalonamento de processos, podendo varias a forma de escalonamento em cada sistema operacional.

3-) Porque em sistemas monoprogramados o processo só sai da CPU quando termina de executar, sendo uma maneira ruim de aproveitar a CPU porque desta maneira o tempo de espera será altíssimo se o processo quiser fazer alguma E/S.

4-) Sistemas de tempo compartilhado permitem que diversos programas sejam executados a partir da divisão do tempo do processador em pequenos intervalos, denominados time-slice. Caso o time-slice não seja suficiente para o processo ser executado, ele é substituído por outro processo enquanto fica aguardando um novo time-slice para executar.

5-) Um programa nada mais é que um arquivo contendo instruções e dados utilizados para inicializar segmentos de instruções e de dados do usuário de um processo e processo é um ambiente de execução que consiste em um segmento de instruções.

6-) Os estados de um processo são, novo (quando o processo é criado), em execução (quando se está associado a um processador que está executando suas instruções), pronto (se o processo aguardar sua vez na fila para executar), bloqueado (se o processo aguarda que ocorra algum evento para continuar a executar, como término de E/S) e término (quando o processo termina sua execução). Um processo passa de “pronto” para “em execução” quando uma CPU fica disponível para este processo, um processo passa de “em execução” para “bloqueado” quando ele necessita de algo para continua executando, um processo passa de “bloqueado” para “pronto” quando a necessidade do processo é satisfeita e um processo passa de “em execução” para “pronto” quando ele ultrapassa seu tempo máximo de execução na CPU.

7-) Os escalonadores agem sobre a fila de processos prontos, de maneira que escolhem qual o próximo a ser executado, depois de selecionado o processo que irá executar, o escalonador aloca a CPU para o processo e depois o retira, se o processo terminou ou se ultrapassado o tempo máximo de uso de CPU por processo. Desta maneira o uso de CPU aumenta consideravelmente nas máquinas, otimizando seu desempenho.

8-) Quando a possa da CPU é atribuída a outro processo, o sistema deve salvar o estado do processo antigo e carregar o estado do processo novo. A troca de contexto é necessária quando o escalonador tira um processo da CPU e coloca outro no lugar, sem a troca de contexto, que salva o estado do processo antes de sair da CPU, o processo não conseguiria continuar executando corretamento de onde parou quando voltasse a CPU.

9-) Aplicações concorrentes são aplicações que brigam pelo uso da mesma CPU, por exemplo quando um processo faz chamadas “fork”, criando assim processos filhos idênticos a ele e que irão concorrer ao uso de CPU com o pai em questão. Existem diversas maneiras de se implementar aplicações concorrentes, como por exemplo as funções “fork()” e “execl()”.

10-) Thread é um pequeno programa que trabalha como um subsistema, sendo uma forma de um processo se autodividir em duas ou mais tarefas. Uma das vantagens da utilização de Threads é que isso facilita o desenvolvimento, visto que torna possível elaborar e criar o programa em módulos, podendo experimentá-los isoladamente no lugar de escrever em um único bloco de código. Uma das desvantagens é que com vários threads o trabalho fica mais complexo, justamente por causa da interação que ocorre entre eles.