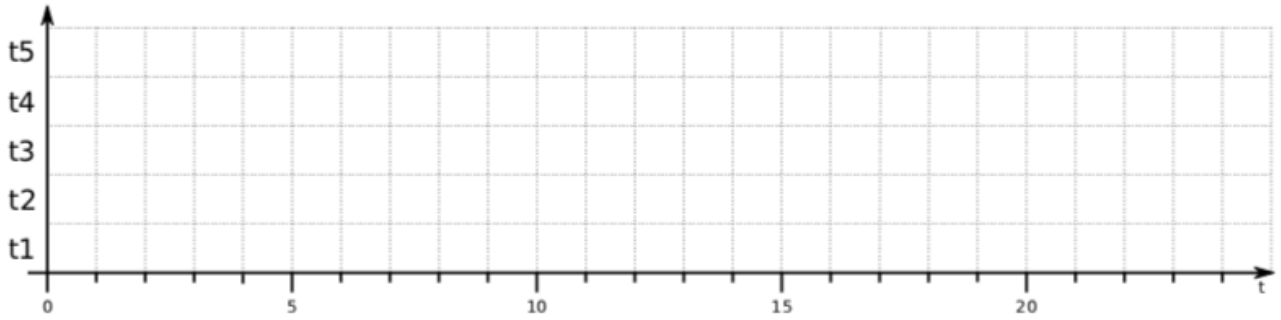
1. Por que a abstração de recursos é importante para os desenvolvedores de aplicações? Ela tem utilidade para os desenvolvedores do próprio sistema operacional?
2. A gerência de atividades permite compartilhar o processador, executando mais de uma aplicação ao mesmo tempo. Identifique as principais vantagens trazidas por essa funcionalidade e os desafios a resolver para implementá-la.
3. O que diferencia o núcleo do restante do sistema operacional?
4. Explique o que é, para que serve e o que contém um PCB - *Process Control Block*.
5. Quais as principais vantagens e desvantagens de *threads* em relação a processos?
6. Indique as transições de estado de tarefas possíveis. (N: Nova, P: pronta, E: executando, S: suspensa, T: terminada).
7. A tabela abaixo representa um conjunto de tarefas prontas para utilizar um processador. Indique a seqüência de execução das tarefas, o tempo médio de vida (*tournaround time*) e o tempo médio de espera (*waiting time*), para as políticas de escalonamento: (a) FCFS cooperativa (b) SJF cooperativa (c) SJF preceptiva (SRTF) (d) RR com `quantum = 2`.

Tarefa	T1	T2	T3	T4	T5
Ingresso	0	0	3	5	7
Duração	5	4	5	6	4
Prioridade	2	3	5	9	6

Considerações: todas as tarefas são orientadas a processamento; as trocas de contexto têm duração nula; em eventuais empates (idade, prioridade, duração, etc), a tarefa ti com menor i prevalece; valores maiores de prioridade indicam maior prioridade.
Para representar a sequência de execução das tarefas use o diagrama a seguir. Use "`x`" para indicar uma tarefa usando o processador, "`–`" para uma tarefa em espera na fila de prontos e '#' para uma tarefa que ainda não iniciou ou já concluiu sua execução.



8. Explique o que são condições de disputa, mostrando um exemplo real. Explique o que é espera ocupada e por que os mecanismos que empregam essa técnica são considerados ineficientes.
9. Usando semáforos, escreva o pseudo-código de um sistema produtor/consumidor com dois buffers limitados organizado na forma `X → B1 → Y → B2 → Z`, onde `X`, `Y` e `Z` são tipos de processos e `B1` e `B2` são buffers independentes com capacidades `N1` e `N2`, respectivamente, inicialmente vazios. Os buffers são acessados unicamente através das operações `insere(Bi, item)` e `retira(Bi, item)` (que não precisam ser detalhadas). O número de processos `X`, `Y` e `Z` é desconhecido. Devem ser definidos os códigos dos processos `X`, `Y` e `Z` e os semáforos necessários, com seus significados e valores iniciais.
10. Suponha três robôs (`Bart`, `Lisa`, `Maggie`), cada um controlado por sua própria `thread`. Você deve escrever o código das `threads` de controle, usando semáforos para garantir que os robôs se movam sempre na sequência `Bart → Lisa → Maggie → Lisa → Bart → Lisa → Maggie → • • •`, um robô de cada vez. Use a chamada `move()` para indicar um movimento do robô. Não esqueça de definir os valores iniciais das variáveis e/ou dos semáforos utilizados. Soluções envolvendo espera ocupada (*busy wait*) não devem ser usadas.