1- R: PCB é uma estrutura de dados que serve para armazenar as informações relativas ao contexto e os demais dados necessários à gerencia de uma tarefa presente no sistema. Ele serve também para que seja efetuada a Troca de Contexto o que é : interromper a execução de uma tarefa e retornar a ela mais tarde, sem corromper seu estado interno.

2-

R: Time share, em português compartilhamento de tempo, foi um conceito introduzido nos anos 60 para resolver a inviabilização do sistema devido a uma tarefa em execução que nunca termina e nem solicita operações de entrada/saída, monopolizando o processador e impedindo a execução das demais tarefas. Nessa solução, cada atividade que detém o processador recebe um limite de tempo de processamento, denominado quantum. Esgotado seu quantum, a tarefa em execução perde o processador e volta para uma fila de tarefas "prontas", que estão na memória aguardando sua oportunidade de executar.

3- A duração atual do quantum depende muito do tipo de sistema operacional. No Linux ela varia de 10 a 200 milissegundos, dependendo do tipo e prioridade da tarefa.Vários critérios podem ser definidos para a avaliação de escalonadores (quem define a duração dos quantums). Os mais frequentemente utilizados sã o:

?Tempo de execução ou de vida (turnaround time): diz respeito ao tempo total da "vida" de cada tarefa, ou seja, o tempo decorrido entre a criação da tarefa e seu encerramento,

computando todos os tempos de processamento e de espera. É uma medida típica de sistemas em lote, nos quais não há interação direta com os usuários do sistema. Não deve ser confundido com o tempo de processamento, que é o tempo total de uso de processador demandado pela tarefa.

?

Tempo de espera(waiting time): é o tempo total perdido pela tarefa na fila de tarefas prontas, aguardando o processador. Deve-se observar que esse tempo nã o inclui os tempos de espera em operações de entrada/saí da (que sã o inerentes à aplicação).

?Tempo de resposta (response time): é o tempo decorrido entre a chegada de um vento ao sistema e o resultado imediato de seu processamento. Por exemplo, o tempo decorrido entre apertar uma tecla e o caractere correspondente aparecer na tela, em um editor de textos. Essa medida de desempenho é típica de sistemas interativos, como sistemas desktop e de tempo-real; ela depende sobretudo da rapidez no tratamento das interrupções de hardware pelo núcleo e do valor do quantum de tempo, para permitir que as tarefas cheguem mais rápido ao processador quando saem do estado suspenso.

? Eficiência :a eficiência E, conforme definido na Seção 4.2, indica o grau de utilização do processador na execução das tarefas do usuário. Ela depende sobretudo da rapidez da troca de contexto e da quantidade de tarefas orientadas a entrada/saí da no sistema (tarefas desse tipo geralmente abandonam o processador antes do fim do quantum, gerando assim mais trocas de contexto que as tarefas orientadas a processamento).

4-

5-

[N]O código da tarefa está sendo carregado.

[P] A tarefas sã o ordenadas por prioridades.

[E]A tarefa sai deste estado ao solicitar uma operação de entrada/saí da.

[T]Os recursos usados pela tarefa sã o devolvidos ao sistema.

[P]A tarefa vai a este estado ao terminar seu quantum.

[E]A tarefa só precisa do processador para poder executar.

[S]O acesso a um semáforo em uso pode levar a tarefa a este estado.

[E]A tarefa pode criar novas tarefas.

[E]Há uma tarefa neste estado para cada processador do sistema.

[S] A tarefa aguarda a ocorrência de um evento externo

6-

7-

8-

9-

De forma geral, cada fluxo de execução do sistema, seja associado a um processo ou no interior do núcleo,é denominado thread. Threads servem para se executar mais de um processo ao mesmo tempo.

10-

Vantagens: Leve e de fácil implementação. Como o núcleo somente considera uma thread, acarga de gerência imposta ao núcleo é pequena e não depende do número de threads dentro da aplicação.

Desvantagens: Como essas operações sã o intermediadas pelo núcleo, se um thread de usuário solicitar uma operação de E/S (recepção de um pacote de rede, por exemplo) o thread de núcleo correspondente ser suspenso até a conclusão da operação, fazendo com que todos os threads de usuário associados ao processo parem de executar enquanto a operação não for concluí da. Outro problema desse modelo diz respeito à divisão de recursos entre as tarefas. O núcleo do sistema divide o tempo do processador entre os fluxos de execução que ele conhece e gerencia: as threads de núcleo.Assim, uma aplicação com 100 threads de usuário irá receber o mesmo tempo de processador que outra aplicação com apenas um thread (considerando que ambas as aplicações têm a mesma prioridade). Cada thread da primeira aplicação irá portanto receber 1/100 do tempo que recebe o thread único da segunda aplicação, o que não pode ser considerado uma divisão justa desse recurso.

11-

O modelo de threads 1:1(multi-thread) é adequado para a maioria das situações e atende bem às necessidades das aplicações interativas e servidores de rede. No entanto,

é pouco escalável: a criação de um grande número de threads impõe um carga significativa ao núcleo do sistema,inviabilizando aplicações com muitas tarefas (como grandes servidores Web e simulações de grande porte).

12-

[a]Tem a implementação mais simples, leve e eficiente.

[b] Multiplexa os threads de usuário em um pool de threads de núcleo.

[b]Pode impor uma carga muito pesada ao núcleo.

[a] Não permite explorar a presença de várias CPUs pelo mesmo processo.

[c] Permite uma maior concorrência sem impor muita carga ao núcleo.

[b] É o modelo implementado no Windows NT e seus sucessores.

[a] Se um thread bloquear, todos os demais têm de esperar por ele.

[c]Cada thread no nível do usuário tem sua correspondente dentro do núcleo.

[c] É o modelo com implementação mais complexa

13-

14-

Round-robin (RR)é um dos algoritmos mais simples de agendamento de processos em um sistema operacional, que atribui frações de tempo para cada processo em partes iguais e de formacircular, manipulando todos os processos sem prioridades. Escalonamento Round-Robiné simples e fácil de implementar.A adição da preempção por tempo ao escalonamentoFCFS dá origem a outro algoritmo de escalonamento bastante popular, conhecido como escalonamento por revezamento, ou Round-Robin.

FIFO (First in, first out) ouFCFS (First come, first served): Onde como seu próprio nome já diz, o primeiro que chega ser á o primeiro a ser executado;

SJF (Shortest Job First): Onde o menor processo ganhará a CPU e atrás do mesmo formar uma fila de processos por ordem crescente de tempo de execução.

15-

16-

Para evitar a inanição(quando um processo nunca assegura um recurso) e garantir a proporcionalidade expressa através das prioridades estéticas, um fator interno de nominado envelhecimento (task aging) deve ser definido. O envelhecimento indica há quanto tempo uma tarefa está aguardando o processador e aumenta sua prioridade proporcionalmente. Dessa forma,o envelhecimento evita a inanição dos processos de baixa prioridade,permitindo a eles obter o processador periodicamente.

17-

18-

19-A inversão de prioridades consiste em processos de alta prioridade serem impedidos de executar por causa de um processo de baixa prioridade. Um exemplo de como pode ocorrer uma inversão de prioridades:

1. Em um dado momento, o processador está livre é alocado a um processo de baixa prioridade pb;

2. durante seu processamento, pb obt é m o acesso exclusivo a um recurso R e começa a usá-lo;

3. pb perde o processador, pois um processo com prioridade maior que a dele (pm) foi acordado devido a uma interrupção;

4. pb volta ao final da fila de tarefas prontas, aguardando o processador; enquanto ele não voltar a executar, o recurso R permanecer á alocado a ele e ninguém poder á usá-lo;

5.Um processo de alta prioridade pa recebe o processador e solicita acesso ao recurso R; como o recurso está alocado ao processo pb, pa é suspenso até que o processo de baixa prioridade pb libere o recurso. Neste momento, o processo de alta prioridade pa nã o pode continuar sua execução, porque o recurso de que necessita está nas mãos do processo de baixa prioridade pb. Dessa forma, pa deve esperar que pb execute e libere R, o que justifica o nome inversão de prioridades.

20-