Resumo SO

Processo é uma entidade ativa no sistema e representa um programa em execução

Um processo é representado no SO por uma estrutura de dados chamada PCB

O PCB possui informações sobre o contexto de execução do processo

Algumas informações contidas no PCB

Id do processo

Estado de execução

Dados do contador de programas

Dados dos registradores da CPU

Info para o escalonamento de CPU, como prioridade por exemplo

Informações sobre memória

Informações sobre memória

Informações de contabilidade

Informações sobre as operações de entrada e saída

Cada SO possui chamadas de sistema que são responsáveis pela criação e destruição dos processos

Fork () – cria processo novo

Exit () – força destruição do processo

Getpid () retorna id do processo

Getppid() retorna o id do pai do processo

Wait () aguarda execução do processo filho

Sleep () aguarda um tempo sem segundos

Thread : Thread é o fluxo de execução de um processo

Processos podem ter mais de um thread (multithread)

Cada thread pode executar uma tarefa específica dentro do processo

Threads são usadas para decompor um processo em partes menores com custo menor de troca de contexto e explorar múltiplos processadores

MODELO N1

Os primeiros SO’s não reconheciam as threads, assim era necessário o uso de uma biblioteca de threads em nível de usuário

Neste modelo os threads implementados em nível de usuario são representadas por uma única thread no nível de kernel

A vantagem deste modelo é que torna o SO mais eficiente no gerenciamento de threads

A desvantagem é que se uma thread bloquear para realizar operações de entrada/saída, por exemplo, as demais também param

MODELO 1:1

Neste modelo cada thread presente no nível de usuario tem uma correspondente no nível de kernel

A vantagem deste modelo é que se uma thread bloquear, as demais continuam sendo escalonadas para serem executadas

Desvantagem: torna o so mais complexo em virtude do gerenciamento individual de cada thread

MODELO N:M

Este modelo é uma junção dos dois anteriores. Neste caso, as N threads presentes no nível de usuario possuem M threads no nível de kernel M<N

Vantagem: Thread continua sendo vista individualmente pelo SO

Desvantagem: Mais difícil implementar que 1:1 e N:1

Vantagens multithreading: Em um programa interativo é possível manter parte do programa executando enquanto outra parte dele estiver esperando bloqueada esperando uma operação E/S (capacidade de resposta)

Um sistema multithreading possui várias threads compartilhando o mesmo espaço de endereçamento (compartilhamento de recursos)

A criação e o gerenciamento de threads pelo SO é mais econômico do que a criação e o gerenciamento de processos (economia)

Se o computador possuir vários processadores (ou núcleos de processamento) é possível manter cada thread executando um processador distinto (escalabilidade)

A troca de uma tarefa (processo ou thread) em execução, ou seja, troca de posse do processador que estiver com ela é definida como troca de contexto

Os dados de contexto de uma tarefa são armazenados em sua PCB

A troca de contexto é realizada pelo despachante e a escolha de qual tarefa deverá assumir o processador é realizada pelo algoritmo de escalonamento de tarefas

A tarefa (processo ou thread) pode estar em um dos seguintes estados

**Novo** acabou de ser criada

**Pronto** pronta para ser executada

**Executando** está em execução

**Aguardando** aguardando um evento

**Concluído** execução concluída

Troca de contexto ocorre quando o tempo de execução do processo atual se esgotou, ocorrer uma interrupção de hardware, interrupção de software (execução de uma chamada de sistema chamada trap), ocorrer um erro de execução

Em um sistema multiprogramado existe a necessidade de selecionar qual e por quanto tempo um determinado programa ocupará a CPU. Processos em execução alternam entre picos de CPU e picos de entrada/saída

A CPU é escalonada quando um processo passa de execução pra espera ou pronto, espera para pronto ou quando termina

Critérios para a escolha de um algoritmo escalonamento de recursos: utilização da cpu, quantidade de processos que são concluídos por unidade de tempo, intervalo de tempo a partir da criação do processo até a sua conclusão, tempo de espera e tempo de resposta

O escalonamento de threads pode ser realizado de duas maneiras:

No processo: a disputa pelo processador acontece no nível e processo, isto é, a biblioteca de gerenciamento de threads é responsável por determinar qual thrad entrará em execução

No sistema: A disputa pelo processador acontece no nível de sistema. O SO determinará qual thread entrará em execução no modelo 1:1 de gerenciamento de threads

Escalonamento em sistemas multiprocessadores: Abordagem assimétrica (mestre escravo): existe um processador mestre q manipula as estruturas de dados do sistema, os demais processadores (escravos) são destinados a execução de processos que o processador mestre determinar

Abordagem simétrica: cada processador é responsável pelo seu próprio escalonamento

Nesta abordagem pode existir uma única fila de prontos para todos os processadores ou cada processador pode ter sua própria fila de prontos. Deve haver mecanismos que impeçam que dois processadores não selecionem o mesmo processo para executar

**Escalonamento em sistemas de tempo real**: Um sistema de tempo real deve produzir resultados dentro de um limite de tempo (deadline)

Resultados após o deadline podem não ter valor real

**Sistema de tempo real critico:** possuem restrições de tempo mais perigosas, atrasos na execução de alguma tarefa não são permitidos. Uma tarefa que não cumpra o deadline pode ser considerada inútil, atrasos podem levar o sistema ao colapso

**Sistema de tempo real não critico:** Atrasos levam o sistema a degradação, mas não ao colapso, restrições de tempo mais brandas

Sistemas de tempo real, críticos ou não possuem: uso especifico, tamanho reduzido, produzidos em massa e requisitos de tempo especifico.

**Tipos de tarefas em um sistema de tempo real**

**Periódicas**: exigem a cpu em intervalos de tempos constantes. Possuem um tempo de processamento fixo t após adquirir a cpu, um deadline d durante o qual deverá ser atendido pela CPU, e um período p. 0<t<d<p. São usadas em sistemas de tempo real critico

**Aperiódicas:** Sua ativação reponde a eventos internos ou externos, definindo uma característica aleatória. São usadas em tempo real brando

**Esporádicas: .** Possuem como característica a restrição de um intervalo mínimo conhecido entre duas ativações consecutivas, por isso podem ter atributos de tarefas críticas. Usadas em sistemas de tempo real crítico.

**Algoritmo de taxa monotônica:** Define uma política de escalonamento baseada na alocação de prioridade estática com preempção. Cada tarefa recebe uma prioridade inversamente proporcional ao seu período, quanto menor o período, mais alta a prioridade. Considera o tempo de processamento de uma tarefa periódica é o mesmo em cada pico de CPU