Relatório Sobre Threads

William Cardoso Barbosa

[1] Universidade Federal de Rondônia (UNIR)

[2]Departamento de Ciência da Computação

1. Informações Gerais

A compreensão sobre threads é crucial para o desenvolvimento do raciocínio sobre a sincronização e comunicação entre os processos de um programa. A thread é caracterizada como a menor unidade de processamento de um programa, semelhante ao bit, que é a menor unidade lógica para as máquinas.

2. Entendimento de Threads

Essa ferramenta pode ser entendida como um pequeno programa que trabalha como um subsistema, seria uma forma de um processo computacional ser subdividido em duas - tendo em vista que um processo possui uma thread principal - ou mais tarefas. Em outras palavras, podemos denotar thread como sendo um encadeamento de execução.

3. Explicação

Em adição, as threads que existem em um programa podem trocar dados e informações entre elas e compartilhar os recursos postos a elas durante o funcionamento do programa, isso também inclui a mesma célula de memória. Desse modo, um usuário pode usar outras ferramentas do sistema enquanto as threads trabalham de forma oculta no sistema operacional. Podemos ilustrar esse cenário da seguinte maneira: Gabriel é usuário do Linux e ele permitiu que seu computador pudesse ser acessado de forma remota via ssh. Gabriel precisa criar 20 pastas em seu computador para organizar as disciplinas de sua faculdade como, por exemplo, pastas “Geometria Análitica”, “Sistemas Operacionais”, etc. William, um certo dia, decidiu se oferecer para facilitar essa atividade para seu amigo. De forma genial, Gabriel deu a ideia de passar seu usuário ssh para William e assim o convidado possa criar as pastas. A forma mais interessante de fazer essa atividade seria William e Gabriel trabalharem juntos na criação das pastas, com a condição de ambos não criarem uma pasta com o mesmo nome ao mesmo tempo.

Seguindo o problema anterior, o William e Gabriel seriam duas threads em funcionamento, criando diversas pastas - tarefa a ser processada - e o ssh seria a porta que permitiria essa ocorrência. O espaço de memória compartilhado seria a pasta raiz, na qual as outras subpastas seriam criadas. Logo, conseguimos ver de forma macro como as threads funcionam em prol de uma atividade.

Embora isso pareça encantador, as threads possuem desvantagens também, uma delas é que o trabalho fica mais complexo com a quantidade de threads e isso surge justamente por causa da interação que ocorre entre elas. Vimos isso quando impomos uma condicional para todo orquestramento da criação das pastas, mesmo que 2 pessoas estejam criando pastas de locais diferentes sem se comunicar, ambas podem colidirem, fazendo uma mesma sub-atividade, que seria criar pastas com os mesmo nomes.

Também, é importante distinguirmos a diferenciação de processos e threads, ambos parecem iguais, entretanto um é a unidade do outro. O que melhor distingue uma thread de um processo é o espaço de endereçamento. Todas as threads de um processo trabalham no mesmo espaço de endereçamento, que é a memória lógica do "processo hospedeiro". Isto é, quando se tem um conjunto de threads dentro de um processo, todas as threads executam o código do processo e compartilham as suas variáveis. Por outro lado, quando se tem um conjunto de processos, cada processo trabalha num espaço de endereçamento próprio, com um conjunto separado de variáveis.

4. Concorrência e Threads

O sistema de concorrência é o princípio básico para o projeto e a implementação dos sistemas multiprogramados. A possibilidade de o processador executar várias tarefas ao mesmo tempo permite que vários programas sejam executados concorrentemente pelo sistema operacional. Destacando, essa possibilidade só existe pelo fato das threads serem os operários que irão distribuir as pequenas atividades de um processo, facilitando a concorrência.

A utilização concorrente da UCP deve ser implementada de maneira que, quando um programa perde o uso do processador e depois retorna para continuar o processamento seu estado deve ser idêntico ao do momento em que foi interrompido. O programa deverá continuar sua execução exatamente na instrução seguinte àquela em que havia sido interrompido. Para o usuário este processo é transparente, ficando ao usuário a percepção de que o computador está totalmente dedicado ao usuário e a mais nenhum outro processo. Com isso, é exatamente este mecanismo de concorrência que permite o compartilhamento de recursos pelos vários programas sendo executados pelo processador.