Universidade Federal
de Ouro Preto

Trabalho Prático 3: FORK/PIPE/PID/IPC/THREADS

Aluno em Graduação da Universidade Federal de Ouro Preto do curso Ciência da Computação:

Halliday Gauss Costa dos Santos.

Matrícula: 18.1.4093.

Área: Sistemas Operacionais.

Introdução:

Sabe-se que threads é a divisão de processos dentro de um processo pesado e que pode ser necessário criar vários processos pesados para aumentar a eficiência de um programa ou até mesmo para dividir tarefas. Este documento apresenta a análise da implementação de processos pesados em **C.**

Desenvolvimento:

No trabalho prático, a função fork foi utilizada e é uma função que duplica o processo atual dentro do sistema operacional. O processo que inicialmente chamou a função fork é chamado de processo pai. O novo processo criado pela função fork é chamado de processo filho. Todas as áreas do processo são duplicadas dentro do sistema operacional (código, dados, pilha, memória dinâmica).

A função fork é chamada uma única vez (no pai) e retorna duas vezes (uma no processo pai e outra no processo filho). O processo filho herda informações do processo pai. Porém, cada processo terá um único PID (Process Identification) dentro do sistema. O PID é um número inteiro que serve para identificar o processo dentro do sistema operacional.

Diferente do uso de threads, o uso de fork para criação de processos pesados gera um processo com uma área de memória separada dos outros, e que não é compartilhada entre si. No entanto, pode haver a necessidade de comunicação entre processos, logo, é necessário o uso de PIPE para estabelecer uma comunicação entre dois processos. Isso também é chamado de IPC (Inter-process Communication).

O PIPE em C é um vetor com duas posições, a posição 0 para leitura e posição 1 para escrita. Um processo escreve o número de bytes que será enviado para outro processo, e o processo receptor é capaz de ler os dados enviados, e assim vice-versa.

O TP calcula a série de Fibonacci de 0 até n, onde n é o parâmetro passado ao executar o programa. Inicialmente, um canal de comunicação entre processos

(PIPE) é criado assim como o fork do processo principal, gerando um processo pai e um processo filho.

O processo pai escreve no PIPE o valor de n, e o processo filho é responsável de ler esse valor e calcular o Fibonacci de 0 até n, gerando uma sequência. Essa sequência é escrita no PIPE pelo processo filho enquanto o processo pai imprime na tela o PID dos processos e espera a execução do processo filho. Finalmente, após o cálculo do Fibonacci, o processo pai lê a sequência do PIPE e a imprime na tela.

Conclusão:

A realização desse trabalho foi de suma importância para o melhor entendimento do funcionamento dos processos pesados e sua implementação em uma linguagem de programação, para a compreensão dos conceitos de PID, IPC e PIPE, e para entender a diferença entre processos pesados e threads.