Fábio Henrique Alves Fernandes

19.1.4128

Link do video: https://youtu.be/xG8py-ggMhU

Diferenças entre Fork e Thread

Quando usamos fork, criamos processos distintos, onde o código passa a ser executado na linha seguinte à execução do comando que criou o fork. Normalmente, forks são utilizados quando a comunicação entre as execuções não é tão frequente, já que a comunicação entre tais processos só pode ocorrer mediante a modos de comunicação entre processos.

Já no thread, a execução do programa até o momento de criação das threads são compartilhados. Isso facilita a comunicação entre as threads, porém torna necessário o uso de mecanismos de exclusão mútua em alguns casos.

Primeira Questão

A função fork 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).

Caso a função fork retorne 0 (zero), está se executando o processo filho. Caso a função retorne um valor diferente de 0 (zero), mas positivo, o processo pai está sendo executado. O valor retornado representa o PID do processo filho criado. A função retorna -1 em caso de erro (provavelmente devido a se ter atingido o limite máximo de processos por usuário configurado no sistema).

Uso pipes para que os processos não tenham interferência durante o uso dos processos pai e filhos.

A função ‘kill’ **[int kill(pid_t pid, int sig);]** é usada para uma chamada no sistema que fecha um processo ou uma árvore de processos de acordo com o PID.

A função getpid **[pid_t getppid(void);]** é usada para receber o PID do processo atual.

Segunda Questão

As threads como pequenos processos. O grande diferencial das threads é que compartilham os mesmos recursos e endereçamento de memória. Ou seja, é como se tivéssemos um processo dividido em processos menores, onde há um chaveamento entre eles, cada um executando um pouco, porém compartilhando os mesmos dados e recursos, e com o mesmo objetivo.

Temos algumas funções importantes a serem usadas quando falamos em threads:

pthread_create - Essa função inicia uma determinada thread no processo atual.

pthread_join - Essa função suspende o funcionamento de uma determinada thread, até que outra termine sua execução.

pthread_exit - Essa função finaliza uma determinada thread no processo atual, e retorna um determinado valor.

pthread_kill - envia o sinal “sig” para a thread, uma thread no mesmo processo da que o chamou. O sinal é direcionado de forma assíncrona para o encadeamento. Essa função acabou sendo descontinuada, não se encontrando mais nas bibliotecas mais recentes dos compiladores de C.