Laboratório de Sistemas Operacionais

Relatório – Projeto 4

Aluno: Bruno Villas Boas da Costa

RA: 317527

O objetivo desse projeto como dito é aprender como usar os mecanismos básicos de programação concorrente, para isto tive que modificar um programa oferecido pelo professor que quebrava senhas utilizando somente uma thread para o consumidor, fazendo com que este passe a quebrar as senhas utilizando múltiplas threads.

Primeiramente entendi o funcionamento do programa oferecido pelo professor, e depois o compilei para ver o que ele fazia e ver o tempo que este demorou a quebrar a senha passada como parâmetro. O tempo obtido foi:

real 1m46.828s
user 1m46.748s
sys 0m0.059s

Depois modifiquei o programa, e para isto usei a organização produtor/consumidor vista em sala e optei por usar o controle da região critica através de semáforo. Implementei o produtor da seguinte maneira:

- Ele produzirá senhas de 4 caracteres, inicializando com ‘aaaa’ e terminando com ‘zzzz’, após produzir uma senha ele entra na região critica e então colocar na fila de senhas na ultima posição e após sair da região critica ele incrementa a senha (onde ele produzirá a próxima senha).

Já o consumidor foi implementado assim:

- Como o computador onde o programa foi implementado possui duas threads, foram utilizados dois consumidores. Eles resgatarão as senhas armazenadas no buffer. Sendo assim, eles pegarão a primeira senha produzida no buffer e saem da região critica, logo após eles testam a senha para ver se é a senha passada como parâmetro. Se for ele retorna qual a senha e finaliza. Caso contrário, eles continuam pegando outras senhas. O interessante é que os consumidores trabalham concorrentemente testando as senhas. Desta forma o tempo praticamente diminui em 50% para encontrar a senha correta.

Depois que implementei e o programa estava funcionando corretamente, utilizei novamente o comando time do Linux e anotei o tempo que o programa agora com multithreads demorou para encontrar a mesma senha anteriormente testada, que foi:

real 0m54.211s
user 1m47.014s
sys 0m1.264s

Com isso concluo que programas que utilizam múltiplas threads são bem mais rápidos que os que não utilizam. E que o tempo de processamento de dados é inversamente proporcional à quantidade de threads do computador.