Trabalho de Implementação – Sistemas Operacionais. 2023/1

Elaborar um programa onde o processo pai cria uma área de memória compartilhada (shmem) e 3 processos filhos (através da system call *fork*). A área de memória compartilhada indicada acima deve ser suficiente para conter duas variáveis do tipo inteiro.

A primeira das variáveis será iniciada com o valor 120 (cento e vinte) e outra com 0 (zero). Dois processo filho vão manipular diretamente a área de memória compartilhada, e o outro processo filho cria duas threads A e B para essa manipulação. A manipulação da área de memória compartilhada compreende os passos indicados abaixo, e será realizada em um loop de 30 iterações (30 para o processo filho1, 30 para o processo filho 2, 30 para a thread A e 30 para thread B) com a seguintes rotinas:

1. Copia o valor da 1a variável da área de memória compartilhada para uma variável local;

2. Decrementa de forma local o valor da variável copiada;

3. "Dorme" (sleep) um tempo aleatório (variando entre 200ms e 2s);

4. Armazena o valor decrementado na 1a variável da memória compartilhada;

5. Incrementa o valor da 2a variável (observe que esta 2a variável não é copiada para a memória local do processo filho/thread, e sim alterada diretamente na memória compartilhada);

6. Exibe na tela uma mensagem contendo o seu PID (no caso dos processos filhos que manipulam a memória compartilhada) ou TID (no caso das Threads), um número de série sequencial iniciado em 1 (cada processo/thread terá uma contagem própria), os valores atualizados das duas variáveis da memória compartilhada e outras informações que o programador achar relevante;

7. "Dorme" (sleep) um tempo aleatório (variando entre 200ms e 2s).

Quando o loop terminar, cada filho deve avisar ao processo "pai" que já terminou. O pai então informa que está ciente de que o filho 1, 2 ou 3 terminou (imprimir na tela uma mensagem informativa quando cada um desses 3 eventos ocorrer). No caso dos processos criados através de fork () vamos utilizar a troca de mensagens como mecanismo de comunicação entre processos filhos com o pai. Ou seja, os processos filho vão enviar uma mensagem para o processo pai (msgsnd), onde o processo pai já deve estar aguardando as mensagens dos processos filho (msgrcv). No caso das threads, elas irão avisar o seu término de forma simples, imprimindo elas mesmo na tela uma mensagem informativa com o seu TID e indicando que foram finalizadas.

Observe que o processo que cria as threads A e B tem que ficar aguardando elas terminarem e o processo "pai de todos" tem que ficar bloqueado aguardando o recebimento de uma mensagem de cada filho (ou seja, msgrcv deve ser usada de forma bloqueante). Quando todos os processos filhos terminarem o processo "pai de todos" imprime duas últimas mensagens, informando que o programa será finalizado e exibindo os valores atuais das duas variáveis da memória compartilhada. Após essas 2 últimas mensagens, o processo pai é finalizado.

Observe que a memória compartilhada deve ser manipulada de forma consistente, ou seja, é preciso garantir exclusão mutual no acesso à mesma por meio de semáforos POSIX. Desta forma, a exclusão mútua deve ser iniciada imediatamente antes do passo 1 do loop, e ser finalizada imediatamente após passo 6). Observar que o passo 7 (uma nova dormida entre 200ms e 2s) é feito fora da região crítica, portanto sem proteção de semáforos para exclusão mútua. Para que haja um escalonamento aleatório do processo e threads que manipulam a memória compartilhada, o tempo de "dormir" desses elementos (passos 3 e 7 do loop) deve ser efetivamente aleatório (variando entre 200ms e 2s). Caso contrário eles podem executar em sequência.

Os manuais sobre semáforos POSIX (sem_open, sem_init, sem_wait, sem_post, etc), memória compartilhada (shmget) e troca de mensagens (msgsnd e msgrvc) são uma boa fonte de consulta para se iniciar a programação. De qq forma, vcs podem consultar livremente a internet para pesquisar exemplos de uso e esclarecer dúvidas sobre a implementação dessas rotinas. Atentar para a necessidade de desconexão e exclusão da área de memória compartilhada, semáforos e fila de troca de mensagens ao final da execução dos processos e threads programa, para que não haja "lixo" de memória no sistema após a finalização do programa.

Prazo para entrega: Até as 23:59 do dia 23/05. Pontuação do trabalho: 0-10 pontos.

Nosso retorno a aulas presenciais será apenas no dia 24/05 (dia seguinte ao prazo de entrega), para que a carga horária da disciplina nessas 3 semanas possa ser dedicada à realização deste trabalho. Qq dúvida é só chamar no Teams para esclarecimentos virtuais ou agendamento de esclarecimento presencial no CEFET.

Artefatos de entrega na tarefa do Teams:

- Arquivos de código fonte (.c ou .h) em formato .zip;

- Link do vídeo de gravação de tela com a explicação do código fonte e a execução do programa.

Bons estudos!