Experimento #3
Sistemas Operacionais A

Introdução:

O experimento realizado envolve a implementação da exclusão mútua na execução da função PrintChars().

Na primeira parte, o programa gera 3 filhos que irão concorrer entre si pela utilização da função de exibição do conteúdo da string, o código deverá ser reparado para que possa realizar as operações devidas e, além disso, os resultados serão das execuções com e sem a diretiva Protect serão analisados e comparados entre si.

Na segunda parte, o programa exemplo será modificado para conter 10 filhos, 5 produtores e 5 consumidores. Enquanto os produtores irão inserir até 4 letras num buffer de memória compartilhado, os consumidores irão retirá-los, de 0 a 4 caracteres, do buffer, sendo que cada vez que o buffer estiver cheio ou os consumidores tenham chegado ao seu fim, se conteúdo é exibido e a interação se mantem até o fim do tempo proporcionado pelo processo pai.

Apresentação Dos Erros de Sintaxe e/ou Lógica do programa Exemplo:

- **Erros de Sintaxe**
  - Declarações:
  - Linha 189: chama incompleta da função exit()

```
exit
```

  - Linha 309: tentativa de exibir as letras com o tipo indevido de variável. Deveria ser do tipo char(%c);

```
fprintf(stderr,"%f7", g_letters_and_numbers[tmp_index + i]);
```

- **Erros de Lógica:**
  - Linha 189: chamada da função exit sem nexo, pois finalizaria os programas filhos. Deveria ser break;

```
exit
```

  - Linhas 143 a 145: inicialização subscrita de g_sem_op1

```
g_sem_op1[0].sem_num =  0;
g_sem_op1[0].sem_op  =  1;
g_sem_op1[0].sem_flg =  0;
```

  - Linha 156: inicialização do recurso como fechado

```
if( semop( g_sem_id, g_sem_op1, 1 ) == -1 )
```

  - Linha 167: criação de uma memória compartilhada sem permissões de leitura e escrita

```
if( (g_shm_id = shmget( SHM_KEY, sizeof(int), IPC_CREAT | 0000)) == -1 ) {
    fprintf(stderr,"Impossivel criar o segmento de memoria compartilhada!\n");
    exit(1);
}
```

- Linha 213 a 215: tentativa de matar filhos inexistentes.

```
kill(pid[2], SIGKILL);
kill(pid[3], SIGKILL);
kill(pid[4], SIGKILL);
```

- Linha 337: fechamento dos semáforos após a saída da região crítica de PrintChars().

```
if( semop( g_sem_id, g_sem_op1, 1 ) == -1 ) {
```

Respostas às perguntas:

- **Contidas no texto do Experimento:**
  - **1 - Uma região por ser crítica tem garantida a exclusão mútua? Justifique.**
    Depende se há implementação de exclusão mútua dos recursos feita pelo programador.
  - **2 - O mecanismo de exclusão mútua protege a região crítica e, dessa maneira, indiretamente, protege o recurso compartilhado. Explique se há algo de errado nessa afirmação.**
    Estão corretas ambas afirmações, pois ao chegar na região crítica a exclusão mútua permite a utilização de recursos compartilhados do sistema á apenas um único processo, o garantindo exclusividade ao mesmo e com isso, a proteção do recurso que está sendo usufruído, por não permitir mudanças redundantes no arquivo ou ainda o retorno de informações desatualizadas para um processo por exemplo.
  - **3 - Descreva a relação entre exclusão mútua e região crítica?**
    A região crítica é a zona de um processo que faz referências aos recursos compartilhados para sua execução e com a utilização da exclusão mútua há garantia do uso exclusivo de tal determinado recurso necessário para a execução do programa.
  - **4 - É obrigatório que todos os processos que acessam o recurso crítico tenham uma região crítica igual?**
    Não, por mais que não seja obrigatório, seria o ideal, os processos podem requisitar recursos iguais, porém diferentes em sequências de acordo com sua implementação.

- **5 - Por que as operações sobre semáforos precisam ser atômicas?**
  Pois caso haja qualquer tipo de interrupção na execução da chamada do semáforo, precisa garantir dentre todos os fatores a atualização devida do status da exclusão mútua, para não ocorrer erros como DeadLock da CPU.

- **6 - O que é uma diretiva ao compilador?**
  Uma diretiva ao compilador é uma alteração no modo que um programa será compilado, no experimento disponibilizado é citado uma diretiva que permite a compilação dos comandos que protegem os recursos compartilhados, se a mesma não for definida o código executável criado não terá tais comandos.

- **7 - Porque o número é pseudo-randômico e não totalmente randômico?**
  O número é considerado pseudo-randômico pois a sua criação não é completamente aleatória, por depender do se valor inicial que já está pré-estabelecido em uma seed na memória.

- **Contidas no código fonte:**
  - **1: Se usada a estrutura g_sem_op1 terá qual efeito em um conjunto de semáforos?**
    g_sem_op1 realiza o fechamento do recurso que está sendo utilizado, para assim garantir a exclusão mútua deste.
  - **2: Para que serve esta operação semop(), se não está na saída de uma região crítica?**
    Ela irá realizar o fechamento dos semáforos para garantir a exclusão mútua enquanto o um dos processos filhos executa a função.

  - **3: Para que serve essa inicialização da memória compartilhada com zero?**
    Para garantir que ao exibir as letras, o primeiro a ser executado filho acesse o início da cadeia de caracteres.

  - **4: Se os filhos ainda não terminaram, semctl e shmctl, com o parametro IPC-RMID, não permitem mais o acesso ao semáforo / memória compartilhada?**
    Sim, pois removem ambos do sistema operacional.

  - **5: Quais os valores possíveis de serem atribuídos a number?**
    Number pode variar de 0 à 3, dependendo do valor do tempo atual que foi retornado pelo gettimeofday().

Resultados da Execução do Programa Exemplo (Parte 1):

- **Dados recolhidos com Protect definido**

  1ª Execução:
  ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 1234567890

  ABCDEFGHIJKLMNOPQRSTUVWXYZ a

- **Dados recolhidos sem Protect Definido**

  1ª Execução:
  A
  ABABCBDCCEDDFEEGFFHGGIHHJIIKJJLKKLMLNMMONNOPOQPPRQQSRRTS
  SUTTVUUWVVWXWYXXYZYZZ a
  baabcbccdeddefeffghgghihiijjkjlkklmlnmmonnopoppqqqrrrsstsuttuvuvvwwwxxxyyyzzz
  1 1123223345445656768778989O9
  00

- **Comparação e Análise dos Resultados**

  Ao observar o retorno das execuções com a diretiva protect definida, nota-se que em todas as execuções não ocorrem repetição de letras ou erros na sequência da produção entre os filhos, comprovando a proteção da região crítica, pois dois ou mais filhos não estão acessando a memória compartilhada com índices iguais.

  Porém ao olhar para as execuções sem a diretiva definida, há repetição e desordem das letras por conta desse acesso incontrolado da memória compartilhada, pois dois ou mais filhos acessam o recurso com índices iguais.

Resultados da Execução do Arquivo Modificado (Parte 2):

- **Dados Recolhidos Com Protect definido**

  1ª Execução:

  AB CDEF GHIJ KLMN OPQR STUV W X Y Z a bc d efgh ijkl m n op q rstu v wx y z 1 2 34 567 890
  Fim do buffer(Produtor):##34567890
  AB CDEF G HIJ KLMN O PQR S TUVW XY Z a bcd ef gh ij klmn opq r st uvw x yz 123 4 567 8 9 0
  Fim do buffer(Produtor): ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 12#####890
  AB CDE FG HIJ KL MNOP QRST UVWX YZ abcd efg hij kl mn o pqr stu v w xy z 1 2345 6 7 890
  Fim do buffer(Produtor): ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 12345678###
  AB C D EFG H I JKL M N O PQRS TUVW XY Z a bcd e fg h ij kl

- **Dados Recolhidos Sem Protect definido**

  1ª Execução:

  A AAAAB B BBBCCCBBBB DDDCCC CE EE DDCC F F FE EDD DDG GGFFE EH H H GGE II I F FEJJJHH G F KKK I IGF LLL JH GM MM KJI HG NNNLKJH OOOMI HPPP N L K IQQQO JIRRRMLJ SS PKJ STNMKT QL KTU O NLVUUR ML V WPVS ON M QX WTWPON MRY XUX QPZ YSON V YR Q T ZW POUaZ SR XVb Q P WcYaTS aRQ ZbXbd UT SR Ycc e V UT SZdad fWV UTeb eg X WV U afc f YhXWV gdbg ZhiY XWce h ij f ZY X dj ig ak ek ZYjhflb lia kgm cm Zj hlnba nd imok c bnjp oe akqlodc pfrb lpmed gsqcqmn fet ho nrr dugi o pss feh jpq ttvg f ur q wiku hvg sl vxjr hi wtmwy ksjixxuzl tk n vyj myu lozk nw1zvm pl wo x2 n pmx q1y3o nyqr1z4 p2r zq53 os 2r 64 1 spt352 71tuq4s u3682vr 5v79 43st0ww45t68ux56 7u x 9 v 7 vy 608w yz789w
  Fim do buffer(Produtor):################################d################uvwxyz 1234567890
  xz 8 90xy A 19
  Fim do buffer(Produtor): A################################d################uvwxyz 1234567890
  0 y zB1 0 2 A z C 2 3
  Fim do buffer(Produtor): ABC##############################d################vwxyz 1234567890
  B 1 D34

Fim do buffer(Produtor):
ABCD############################d##################vwxyz 1234567890
A C54
Fim do buffer(Produtor):
ABCD############################d##################vwxyz 1234567890
2E1 D A 365F2BAB 473GC6BEDCH7 F 4 85C

- **Comparação e Análise dos Resultados**

  Ao observar o programa com a diretiva Protect definida, nota-se, assim como na parte 1, que os filhos mantem a ordem ao executar o programa, se ajudando para preencher o buffer por completo. E além disso o buffer sempre apresenta todos os caracteres preenchidos ao fim, por mais que uma parte esteja consumida (#).

  Com a diretiva não definida o comportamento é completamente diferente, há repetição e divergência da sequência das letras tanto na exibição da produção de cada filho quanto no valor final do buffer que são exibidos.

Conclusão:

Ao utilizar semáforos para a garantia de exclusão mútua nota-se que sua implementação, ainda que seja mais delicada com recursos mais demandados, apresenta um ótimo resultado que não permite, como neste experimento, mudanças redundantes ou atualizações errôneas nas informações de um arquivo/buffer por vários processos concorrentes por este.