PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS – PUC- CAMPINAS

Experimento 3

Sistemas Operacionais A

ALUNO	RA
Beatriz Morelatto Lorente	18071597
Cesar Marrote Manzano	18051755
Fabricio Silva Cardoso	18023481
Pedro Ignácio Trevisan	18016568

Sumário

Introdução

O experimento realizado permitiu o entendimento do uso de semáforos do System V e do uso de memória compartilhada. O experimento foi dividido em duas tarefas, com o objetivo de imprimir um vetor (que continha todas as letras maiúsculas e minúsculas e todos os números), com e sem o uso do mecanismo de exclusão mútua.

Na primeira tarefa foi executado um programa exemplo, no qual três processos filhos chamavam a mesma função para imprimir o vetor. No programa há um recurso compartilhado, um inteiro usado como índice para acessar o vetor de caracteres, e os processos filhos, por sua vez, tentam exibir os caracteres ao mesmo tempo, por isso o uso do semáforo é necessário.

Na segunda tarefa, o programa foi modificado de forma que houvesse 8 filhos e metade seriam produtores de caracteres e os outros consumidores de caracteres. Os produtores acessavam o vetor e colocavam os caracteres em um buffer compartilhado. Um consumidor substitui um caractere produzido pelo caractere '#'. Quando o buffer estivesse cheio, era necessário que um produtor e um consumidor exibissem o conteúdo de todo o buffer. Cada filho produtor também mostrava os caracteres produzidos.

Apresentação dos erros do programa exemplo e suas soluções

Ao compilar o programa pela primeira vez, foi mostrado no prompt alguns erros de sintaxe e lógica de programação. Os problemas estão listados abaixo, seguidos de suas soluções (as correções estão destacadas em negrito e itálico).

Problema 1

/* #define PROTECT */

Problema corrigido:

A definição PROTECT estava comentada, não possibilitando que a exclusão mútua ocorresse. Para corrigirmos, apenas retiramos o comentário.

#define PROTECT

Problema 2

#include <errno.h>

#inlcude <sys/time.h>

#include <unistd.h>

#include <sys/types.h>

#inlcude <sys/wait.h>

#include <signal.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/sem.h>

Problema corrigido:

Faltava a biblioteca <stdlib.h> para o uso da função exit().

#include <stdlib.h>

#include <errno.h>

#inlcude <sys/time.h>

#include <unistd.h>

#include <sys/types.h>

#inlcude <sys/wait.h>

#include <signal.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/sem.h>

Problema 3

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

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

Problema corrigido:

A mesma estrutura estava sendo usada para travar e destravar os semáforos.

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

g_sem_op2[0].sem_num = 0;
g_sem_op2[0].sem_op = 1;
g_sem_op2[0].sem_flg = 0;
```

Problema 4

```
if( semop( g_sem_id, g_sem_op1, 1 ) == -1 ) {
        fprintf(stderr,"chamada semop() falhou, impossivel inicializar o semaforo!");
        exit(1);
}
```

Problema corrigido:

O semáforo necessita começar a sua execução destravado, diferente do que ocorre.

```
if( semop( g_sem_id, g_sem_op2, 1 ) == -1 ) {
        fprintf(stderr,"chamada semop() falhou, impossivel inicializar o semaforo!");
        exit(1);
}
```

Problema 5

```
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);
```

```
}
```

Problema corrigido:

A permissão para criar a memória compartilhada está errada, usamos a permissão 0666.

```
if( (g_shm_id = shmget( SHM_KEY, sizeof(int), IPC_CREAT | 0666)) == -1 ) {
        fprintf(stderr,"Impossivel criar o segmento de memoria compartilhada!\n");
        exit(1);
}
```

Problema 6

```
rtn = 1;
for( count = 0; count < NO_OF_CHILDREN; count++ ) {
        if( rtn != 0 ) {
                pid[count] =rtn = fork();
        } else {
                exit
        }
}
```

Problema corrigido:

Na diretiva else, há a palavra ‘exit’, que deveria ser a função exit(). Para corrigirmos o problema, usamos o break.

```
rtn = 1;
for( count = 0; count < NO_OF_CHILDREN; count++ ) {
        if( rtn != 0 ) {
                pid[count] =rtn = fork();
        } else {
                break;
        }
}
```

Problema 7

```
kill(pid[0], SIGKILL);
kill(pid[1], SIGKILL);
kill(pid[2], SIGKILL);
kill(pid[3], SIGKILL);
kill(pid[4], SIGKILL);
```

Problema corrigido:

O pai matava os filhos, porém no programa só havia 3 filhos e não 5. Para deixar a tarefa automatizada, matamos os filhos com um for.

int child;

for (child = 0; child < NO_OF_CHILDREN; child++) {

kill(pid[child], SIGKILL);

}

Problema 8

```
for ( i = 0; i < number; i++ ) {
        if( ! (tmp_index + i > sizeof(g_letters_and_numbers)) ) {
                fprintf(stderr,"%f7", g_letters_and_numbers[tmp_index + i]);
                fputc(g_letters_and_numbers[tmp_index + i], arq);
                usleep(1);
        }
}
```

Problema corrigido:

Ao imprimir o vetor, estava sendo usado o formato para float. Trocamos para o formato char.

```
for( i = 0; i < number; i++ ) {
        if( ! (tmp_index + i > sizeof(g_letters_and_numbers)) ) {
                fprintf(stderr,"%c", g_letters_and_numbers[tmp_index + i]);
                fputc(g_letters_and_numbers[tmp_index + i], arq);
                usleep(1);
        }
}
```

Problema 9

```
#ifdef PROTECT
                if( semop( g_sem_id, g_sem_op1, 1 ) == -1 ) {
               fprintf(stderr,"chamada semop() falhou, impossivel liberar o recurso!");
                exit(1);
                }
#endif
```

Problema corrigido:

Ao liberarmos o recurso usando o semáforo, este estava sendo trancado e não liberado, para corrigir usamos o g_sem_op2.

```
#ifdef PROTECT
                    if( semop( g_sem_id, g_sem_op2, 1 ) == -1 ) {
                   fprintf(stderr,"chamada semop() falhou, impossivel liberar o recurso!");
                    exit(1);
                    }
#endif
```

Resultados da execução do programa exemplo

Abaixo são mostrados os resultados da execução da tarefa 1 (programa exemplo). Foram feitas 10 rodadas de teste, sendo 5 rodadas com o mecanismo de exclusão mútua, e outras 5 rodadas sem o mecanismo.

```
COM PROTECT DEFINIDO

1ªexecução
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 12345678

2ªexecução
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmno

3ªexecução
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 1234567890
AB

4ªexecução
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 1234567

5ªexecução
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcde
```

Figura 1 - Resultados do programa exemplo com protect definido

```
SEM PROTECT DEFINIDO

1ªexecução
AABABBCDCEFGHIDCJDEFGGEHHFIIJJGKKLHHLMIJIIKJJLKKMLNOLMPNOQRPMSNQTRUSVTWUXVYWZX YabZcc
dddefeefggfhghiihjjkijlkkmllnmomnnpoqopprqqsrrtssttuuvuwvvwwxxyyy

2ªexecução
ABAB ABCDCDEEFGFGCDEHIFGHJKLMNIJKFGHOPLMIJKLJKLMNJKMNLMOPQNORSPOPTUVWQWRXYZSTUSTU
VWVWXabcXYZddefgefgefghijhihijkkjkllmnmnlmopopqnopqrrrststustuvvvwwxwxyzxyzyz
1 123423564564567897897890090  ABCD ABABCDCDCDEEFFEFGGHIGHIHIJJKLJMNMNKLOPMNOOPQRRQRSSTUSTUTUVWVWVWXYXYZXYZZ
abcd abcdecfghcdedeffijkglmghihijnopqrjklpqmnrrstsuvtuvtwxywxyz z uvwxy123zxy 4512z 134523674678985670899 0A0B
A ABCCDDECDEEFFGHGHHHII

3ªexecução
ABABCDCDEEFEGHIJKLGHIIJKJMNOKLPQKMLNMNOLMOPQPQRRRSTSTUVSTWXUVWXUVWYZ
XYZXYZabcd efeefghghghijkijklmmnonpqrsrstslmnvuo

4ªexecução
ABCABCDEFDEFGGHHIJIJKKLMLMKLMNNOOPQOPQRSPQRTUSTURSTVVWXUVWXYYZWXY Z aZ
aabcbcddedefefghihijkfghjklmlmnijklnoopqprsmnoqrstuvtupqrvstwxywxyzz 1z 23
12123345645456677898890 AB AB A ABABCDECDEFGHIJKLMNOIJFGHPQRKLIJKMNOMNOPPLMNQRSQRSOPQTRSTUVUTUVWXWXYVWXYZ aZ
abb acdecdebcffgghighihijkjkjkllmlmnmnonopnoppqrqrssstutuvwvwwxyzvwxz yz yz 121234534556676788908990 00
ABBCCD

5ªexecução
ABCCDEDFDGGHEHIJIKJLMMNNNOPPQOQPPQRRSSTQURSVTTWUVXYWXZYZ Ua abbcdcdeeffgghhijikljmknlVVmWWnXXYoYpZ qrasZbbccdddeeef
```

Figura 2 -Resultados do programa exemplo sem protect definido

Resultados da execução do programa modificado

Abaixo são mostrados alguns dos resultados da execução da tarefa 2 (programa modificado). Foram feitas 10 rodadas de teste, sendo 5 rodadas com o mecanismo de exclusão mútua, e outras 5 rodadas sem o mecanismo.

Nessa tarefa será colocado apenas o resultado da primeira execução de cada uma das rodadas.

```
COM PROTECT DEFINIDO

1ªexecução
Caracteres produzidos pelo filho 2:  A
Caracteres produzidos pelo filho 4: BC
Caracteres produzidos pelo filho 6: DE
Caracteres produzidos pelo filho 2: FGHIJ
Caracteres produzidos pelo filho 8: KL
Caracteres produzidos pelo filho 4: MNO
Caracteres produzidos pelo filho 6: PQ
Caracteres produzidos pelo filho 2: RST
Caracteres produzidos pelo filho 8: U
Caracteres produzidos pelo filho 4: VWXY
Caracteres produzidos pelo filho 6: Z
Caracteres produzidos pelo filho 2: abcd
Caracteres produzidos pelo filho 8: efg
Caracteres produzidos pelo filho 4: hijk
Caracteres produzidos pelo filho 6: lmnop
Caracteres produzidos pelo filho 2: qrstu
Caracteres produzidos pelo filho 8: vwxyz
Caracteres produzidos pelo filho 4:
Caracteres produzidos pelo filho 6: 123
Caracteres produzidos pelo filho 2: 4567
Caracteres produzidos pelo filho 8: 89
Caracteres produzidos pelo filho 4: 0
```

Figura 3 - Resultado da primeira execução com protect definido (parte 1)

```
Consumidor - Buffer Cheio
##BCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqr##################0#

Caracteres produzidos pelo filho 6:  ABC
Caracteres produzidos pelo filho 2: DEFG
Caracteres produzidos pelo filho 8: HIJKL
Caracteres produzidos pelo filho 4: MN
Caracteres produzidos pelo filho 6: OPQRS
Caracteres produzidos pelo filho 2: TUVWX
Caracteres produzidos pelo filho 8: YZ
Caracteres produzidos pelo filho 4: abcde
Caracteres produzidos pelo filho 6: fghij
Caracteres produzidos pelo filho 2: kl
Caracteres produzidos pelo filho 8: mn
Caracteres produzidos pelo filho 4: opq
Caracteres produzidos pelo filho 6: rstu
Caracteres produzidos pelo filho 2: vwx
Caracteres produzidos pelo filho 8: y
Caracteres produzidos pelo filho 4: z 12
Caracteres produzidos pelo filho 6: 3456
```

Figura 4 - Resultado da primeira execução com protect definido (parte 2)

```
Produtor - Buffer Cheio
########################################nopqrstuvwxyz 1234567890
Caracteres produzidos pelo filho 2: 7890
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 4: AB
Caracteres produzidos pelo filho 6: CD
Caracteres produzidos pelo filho 2: E
Caracteres produzidos pelo filho 8: FGHIJ
Caracteres produzidos pelo filho 4: KLMN
Caracteres produzidos pelo filho 6: OPQRS
Caracteres produzidos pelo filho 2: TUV
Caracteres produzidos pelo filho 8: WXYZ
Caracteres produzidos pelo filho 4: abc
Caracteres produzidos pelo filho 6: def
Caracteres produzidos pelo filho 2: ghijk
Caracteres produzidos pelo filho 8: l
Caracteres produzidos pelo filho 4: mn
Caracteres produzidos pelo filho 6: opqrs
Caracteres produzidos pelo filho 2: tu
```

Figura 5 - Resultado da primeira execução com protect definido (parte 3)

```
SEM PROTECT DEFINIDO

1ªexecução
Caracteres produzidos pelo filho 4:  AB
Caracteres produzidos pelo filho 2:  ABCD
Caracteres produzidos pelo filho 4: EFG
Caracteres produzidos pelo filho 2: EFG
Caracteres produzidos pelo filho 4: H
Caracteres produzidos pelo filho 2: HI
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 6:
Caracteres produzidos pelo filho 4: HJKLMNIH
Caracteres produzidos pelo filho 4: JOKILJ
Caracteres produzidos pelo filho 4: PQ
Caracteres produzidos pelo filho 8: P
Caracteres produzidos pelo filho 4: RST
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 6: RSTUVPQQRS
Caracteres produzidos pelo filho 4: TUVR
Caracteres produzidos pelo filho 2: TU
```

Figura 6 - Resultado da primeira execução sem protect definido (parte 1)

```
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 2: VWV
Caracteres produzidos pelo filho 6:
Caracteres produzidos pelo filho 4:
Caracteres produzidos pelo filho 2: WWXYVWZXY
Caracteres produzidos pelo filho 8: WX
Caracteres produzidos pelo filho 4: X
Caracteres produzidos pelo filho 2: YXZYZ
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 8:  a
Caracteres produzidos pelo filho 2: bc
Caracteres produzidos pelo filho 8: defg
Caracteres produzidos pelo filho 2: defgh
Caracteres produzidos pelo filho 4:  abc
Caracteres produzidos pelo filho 6: YZ a
Caracteres produzidos pelo filho 4: d
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 4: efddef
Caracteres produzidos pelo filho 6: eeffgghhig
Caracteres produzidos pelo filho 4: hijkl
Caracteres produzidos pelo filho 2: jkl
```

Figura 7 - Resultado da primeira execução sem protect definido (parte 2)

```
Consumidor - Buffer Cheio
###

Consumidor - Buffer Cheio
#################PQRS
Caracteres produzidos pelo filho 4:
Caracteres produzidos pelo filho 8: TmnU

Consumidor - Buffer Cheio
VW XAYBZC #a#b#c#d#efghij

Consumidor - Buffer Cheio
k l
Caracteres produzidos pelo filho 4: monpoqpq##
Caracteres produzidos pelo filho 2: #################P#Q#RPSQTRl
##
####
Caracteres produzidos pelo filho 6: #####m#n#o####p#q###
mAn
Caracteres produzidos pelo filho 4: #rsAB
Caracteres produzidos pelo filho 6: tCDu#
Caracteres produzidos pelo filho 8: #v##
Caracteres produzidos pelo filho 4: v
Caracteres produzidos pelo filho 6: wwxx######
Caracteres produzidos pelo filho 2: w#xyz
Caracteres produzidos pelo filho 6:  1#
```

Figura 8 - Resultado da primeira execução sem protect definido (parte 3)

```
Caracteres produzidos pelo filho 8: yz
Caracteres produzidos pelo filho 4: y#zP
Caracteres produzidos pelo filho 6: 1 12QRSTUVWXYZ abcd
Caracteres produzidos pelo filho 2: e2f3g4hij
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 4: 23k425
Caracteres produzidos pelo filho 6: 2
Caracteres produzidos pelo filho 2: 3465
Caracteres produzidos pelo filho 4: 678lmnop

Produtor - Buffer Cheio

Caracteres produzidos pelo filho 6: ##
Caracteres produzidos pelo filho 4: #9#0###q#7######r

Produtor - Buffer Cheio


Produtor - Buffer Cheio
################################c#d#e#f#g#h#i#j#k#l#m#n#o#p#q#rcsdteufvgwhxiyjzk l1m2n3o4p5q6r7s8t9u0vw
Caracteres produzidos pelo filho 2: x9y0z z12348567890s9
Caracteres produzidos pelo filho 4: #AB######
Caracteres produzidos pelo filho 2: t uABCvwx
Caracteres produzidos pelo filho 4: y zA B#1234#5678#90
##########
Caracteres produzidos pelo filho 2: CD#E#F#
Caracteres produzidos pelo filho 4: #C#D#####EFG####
Caracteres produzidos pelo filho 6: # #A#BC
```

Figura 9 - Resultado da primeira execução sem protect definido (parte 4)

```
Caracteres produzidos pelo filho 2: HIJ
Caracteres produzidos pelo filho 6: H
Caracteres produzidos pelo filho 4: HIJK###opqrstuv
Caracteres produzidos pelo filho 4: LM
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 4: OPw
Caracteres produzidos pelo filho 6: KLxyMNzL MN1234567890
Caracteres produzidos pelo filho 4: OPQ
Caracteres produzidos pelo filho 8: 7890
Caracteres produzidos pelo filho 6: OP
Caracteres produzidos pelo filho 2: OPQRS
Caracteres produzidos pelo filho 4: R
Caracteres produzidos pelo filho 4:
Caracteres produzidos pelo filho 8: STUS
Caracteres produzidos pelo filho 6: ST
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 4: U
Caracteres produzidos pelo filho 2: SXTUVWVWV
Caracteres produzidos pelo filho 6: YZ
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 8: YZYZ

Consumidor - Buffer Cheio
 A
Caracteres produzidos pelo filho 6:
Caracteres produzidos pelo filho 8:   BCDEFGH
Caracteres produzidos pelo filho 2: I JaKbLcMNOaabcPQbcRSdTUVWXYZ abcdef##
Caracteres produzidos pelo filho 2: d##ef
Caracteres produzidos pelo filho 6: d############################
```

Figura 10 - Resultado da primeira execução sem protect definido (parte 5)

```
Caracteres produzidos pelo filho 2: ef
Caracteres produzidos pelo filho 6: ef
Caracteres produzidos pelo filho 8: defgh
Caracteres produzidos pelo filho 2: gh
Caracteres produzidos pelo filho 6: ij
Caracteres produzidos pelo filho 8: ijklk
Caracteres produzidos pelo filho 2: ij
Caracteres produzidos pelo filho 6:
Caracteres produzidos pelo filho 8: kkll
Caracteres produzidos pelo filho 2: klm
Caracteres produzidos pelo filho 8: mn
Caracteres produzidos pelo filho 6: nopq
Caracteres produzidos pelo filho 2: nopqr
Caracteres produzidos pelo filho 8: op
Caracteres produzidos pelo filho 6: rs
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 2: qq
Caracteres produzidos pelo filho 6: rst
Caracteres produzidos pelo filho 6: u
Caracteres produzidos pelo filho 8: rstuv
Caracteres produzidos pelo filho 6: vwxy

Consumidor - Buffer Cheio
#######
Caracteres produzidos pelo filho 4: #z# #########

Consumidor - Buffer Cheio
##S#T#U#V#W#X#Y#Z# #a#b#c#d#e#f#g#h#i#j
Caracteres produzidos pelo filho 4: k lmn
Caracteres produzidos pelo filho 8: owpxqyrzst
```

Figura 11 - Resultado da primeira execução sem protect definido (parte 6)

```
Caracteres produzidos pelo filho 6: uvwxSzTUyz ###### ##V###
WXYZ abcdefghijklmno
Caracteres produzidos pelo filho 8:  123pqrs
Caracteres produzidos pelo filho 4: t u1v2w3xyz 12
Caracteres produzidos pelo filho 6: 3 41#2######

Caracteres produzidos pelo filho 4: 45
Caracteres produzidos pelo filho 4: 6
Caracteres produzidos pelo filho 6: 3456
Caracteres produzidos pelo filho 6: 7
Caracteres produzidos pelo filho 8: 67890
Caracteres produzidos pelo filho 4: 7890

Produtor - Buffer Cheio
 #############
Caracteres produzidos pelo filho 4:  ##
Caracteres produzidos pelo filho 8: ###ST UVWXYZ abcdefghijklmno
Caracteres produzidos pelo filho 2: rstu
Caracteres produzidos pelo filho 4: ABCD
Caracteres produzidos pelo filho 8: AB
Caracteres produzidos pelo filho 2: Epqrstuvwxyz 1234567890
Caracteres produzidos pelo filho 6: 456
Caracteres produzidos pelo filho 2: CDEF
Caracteres produzidos pelo filho 4: CDEF
Caracteres produzidos pelo filho 8: CDEF
Caracteres produzidos pelo filho 2: G
Caracteres produzidos pelo filho 6: CDHEF
Caracteres produzidos pelo filho 8: IJKLM
Caracteres produzidos pelo filho 4: IJKLM
Caracteres produzidos pelo filho 8: NO
```

Figura 12 - Resultado da primeira execução sem protect definido (parte 7)

```
Caracteres produzidos pelo filho 6: GHI
Caracteres produzidos pelo filho 2: JGHIJ
Caracteres produzidos pelo filho 8: K
Caracteres produzidos pelo filho 6: K
Caracteres produzidos pelo filho 4: PQR
Caracteres produzidos pelo filho 2: SKL
Caracteres produzidos pelo filho 8: M
Caracteres produzidos pelo filho 2: M
Caracteres produzidos pelo filho 6: M
Caracteres produzidos pelo filho 4: MNOPQ
Caracteres produzidos pelo filho 8: NOPQ
Caracteres produzidos pelo filho 4: RS
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 6: NNOOPPQQ
Caracteres produzidos pelo filho 4: R
Caracteres produzidos pelo filho 2: R
Caracteres produzidos pelo filho 6: RSS
Caracteres produzidos pelo filho 8: RSTUV
Caracteres produzidos pelo filho 6: T
Caracteres produzidos pelo filho 4:
Caracteres produzidos pelo filho 2: TUVWTU
Caracteres produzidos pelo filho 4: V
Caracteres produzidos pelo filho 6: UVWX
Caracteres produzidos pelo filho 8: UVWX
Caracteres produzidos pelo filho 2:
```

Figura 13 - Resultado da primeira execução sem protect definido (parte 8)

```
Consumidor - Buffer Cheio
V WABCDEFGHIJKLMNOPQRSTUVWX
Caracteres produzidos pelo filho 8: Y Z ##
Caracteres produzidos pelo filho 2: ### ##########################

Consumidor - Buffer Cheio
# #A#B#C##

Consumidor - Buffer Cheio


Consumidor - Buffer Cheio

 DAB AECFDGEHFBIGCJHDKIELJFMKGNLHOMIPNJQOKRPLSQMTRNUSOVTPWUQXVRYWZSX TYaUZbV #Wa#Xb#Y##Z## ##a##b###
###
#####

Caracteres produzidos pelo filho 4: YZ ab
Caracteres produzidos pelo filho 6: YZ ab
Caracteres produzidos pelo filho 2: ABCD
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 4: EE
Caracteres produzidos pelo filho 8: FAGBCDEH
Caracteres produzidos pelo filho 2: FGH
Caracteres produzidos pelo filho 6: EFGHI
Caracteres produzidos pelo filho 8: FGHI
Caracteres produzidos pelo filho 4: FGHI
Caracteres produzidos pelo filho 2: IJ
Caracteres produzidos pelo filho 4: K
```

Figura 14 - Resultado da primeira execução sem protect definido (parte 9)

```
Caracteres produzidos pelo filho 2: K
Caracteres produzidos pelo filho 6: JKLM
Caracteres produzidos pelo filho 8: JKLMN
Caracteres produzidos pelo filho 2: NOP
Caracteres produzidos pelo filho 4: NOPQ
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 6: RNSOTPQR
Caracteres produzidos pelo filho 2: ST
Caracteres produzidos pelo filho 4: RSTUV
Caracteres produzidos pelo filho 8: RST
Caracteres produzidos pelo filho 2: WU
Caracteres produzidos pelo filho 6: STU
Caracteres produzidos pelo filho 4: XY
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 2: XYZ
Caracteres produzidos pelo filho 4: Z
Caracteres produzidos pelo filho 6: ZV WaXbYc
Caracteres produzidos pelo filho 8: d
Caracteres produzidos pelo filho 6: de
Caracteres produzidos pelo filho 4: defg
Caracteres produzidos pelo filho 2: hdefgh
Caracteres produzidos pelo filho 6: f
Caracteres produzidos pelo filho 8: efgh
Caracteres produzidos pelo filho 4: g
Caracteres produzidos pelo filho 6: ijkh
Caracteres produzidos pelo filho 6: lm
Caracteres produzidos pelo filho 2: ijkl
Caracteres produzidos pelo filho 8: ijklm
Caracteres produzidos pelo filho 2: n
Caracteres produzidos pelo filho 4: nop
```

Figura 15 - Resultado da primeira execução sem protect definido (parte 10)

```
Caracteres produzidos pelo filho 6: mnop
Caracteres produzidos pelo filho 4: q
Caracteres produzidos pelo filho 2: q
Caracteres produzidos pelo filho 8: rstqrst
Caracteres produzidos pelo filho 2: u
Caracteres produzidos pelo filho 6: qrstu
Caracteres produzidos pelo filho 8: vw
Caracteres produzidos pelo filho 4: rst
Caracteres produzidos pelo filho 4: u
Caracteres produzidos pelo filho 2: vwx
Caracteres produzidos pelo filho 6: x
Caracteres produzidos pelo filho 4: yzyz
Caracteres produzidos pelo filho 8: uvwxy
Caracteres produzidos pelo filho 2: yz 1
Caracteres produzidos pelo filho 6: z
Caracteres produzidos pelo filho 4:  1234
Caracteres produzidos pelo filho 8: 1234
Caracteres produzidos pelo filho 2: 1234
Caracteres produzidos pelo filho 6: 1234
Caracteres produzidos pelo filho 6: 5
Caracteres produzidos pelo filho 4: 567
Caracteres produzidos pelo filho 2: 5
Caracteres produzidos pelo filho 8: 5678

Produtor - Buffer Cheio


Produtor - Buffer Cheio
############
```

Figura 16 - Resultado da primeira execução sem protect definido (parte 11)

```
Produtor - Buffer Cheio
########################################
Caracteres produzidos pelo filho 8:  A#BCD#######

Consumidor - Buffer Cheio
# AB

Consumidor - Buffer Cheio
#C ADEFG###

Consumidor - Buffer Cheio
 A#
Caracteres produzidos pelo filho 8: EF#G###
Caracteres produzidos pelo filho 2: 904hACBDEFCGHBCD##################B##C#D#E#F#G##HI

Consumidor - Buffer Cheio
 #DEEFAGBHCJFGHIJ#####################################
Caracteres produzidos pelo filho 2: #KL######D
Caracteres produzidos pelo filho 8: #HIJKEFGHIJKLH###IJKL#L#########################################################

Caracteres produzidos pelo filho 8: IL###J#########
Caracteres produzidos pelo filho 2: ####N#############################################################
#
KM####################
Caracteres produzidos pelo filho 4: 9034L#M###########################################

Caracteres produzidos pelo filho 2: NOPQROSPQRSTUVW#################
Caracteres produzidos pelo filho 4: TUVW###
Caracteres produzidos pelo filho 8: #O#P#Q#R#S##################
Caracteres produzidos pelo filho 6: 904
```

Figura 17 - Resultado da primeira execução sem protect definido (parte 12)

```
Caracteres produzidos pelo filho 4: T
Caracteres produzidos pelo filho 2: TUV
Caracteres produzidos pelo filho 8: WX
Caracteres produzidos pelo filho 6: WXY
Caracteres produzidos pelo filho 4: WXY
Caracteres produzidos pelo filho 8: Z
Caracteres produzidos pelo filho 6: Z
Caracteres produzidos pelo filho 4: Z a
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 6: bb
Caracteres produzidos pelo filho 4: bc
Caracteres produzidos pelo filho 4: defg
Caracteres produzidos pelo filho 8: defg
Caracteres produzidos pelo filho 6: def
Caracteres produzidos pelo filho 4: hijg
Caracteres produzidos pelo filho 6: k
Caracteres produzidos pelo filho 4: klmno
Caracteres produzidos pelo filho 8: klmno
Caracteres produzidos pelo filho 4: pqrs
Caracteres produzidos pelo filho 6: pqrs
Caracteres produzidos pelo filho 6: tu
Caracteres produzidos pelo filho 8: pqrs
Caracteres produzidos pelo filho 4: ttuvwvw
Caracteres produzidos pelo filho 2: WXY
Caracteres produzidos pelo filho 8: xyz
Caracteres produzidos pelo filho 2: Z
Caracteres produzidos pelo filho 8:  a
Caracteres produzidos pelo filho 4: xyz
Caracteres produzidos pelo filho 6: xyz
Caracteres produzidos pelo filho 2:
```

Figura 18 - Resultado da primeira execução sem protect definido (parte 13)

```
Caracteres produzidos pelo filho 8:  1a2b3
Caracteres produzidos pelo filho 6: 41
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 4: 21324345
Caracteres produzidos pelo filho 6: 234
Caracteres produzidos pelo filho 8: 234
Caracteres produzidos pelo filho 2: 5678
Caracteres produzidos pelo filho 2: 90
Caracteres produzidos pelo filho 4: 5678

Produtor - Buffer Cheio
#########################YZ ab##############pqrstuvwxyz 1234567890
Caracteres produzidos pelo filho 2: 078
Caracteres produzidos pelo filho 4:
Caracteres produzidos pelo filho 4: ABC
Caracteres produzidos pelo filho 2: ABC
Caracteres produzidos pelo filho 4: DE
Caracteres produzidos pelo filho 2: DEF
Caracteres produzidos pelo filho 4: F
Caracteres produzidos pelo filho 4: GHIJ
Caracteres produzidos pelo filho 2: GHIJ
Caracteres produzidos pelo filho 2: KLMN
Caracteres produzidos pelo filho 4: KLMN
Caracteres produzidos pelo filho 4: O
Caracteres produzidos pelo filho 2: OPQR
Caracteres produzidos pelo filho 4: PQR
Caracteres produzidos pelo filho 2: STUVW
Caracteres produzidos pelo filho 4: STU
```

Figura 19 - Resultado da primeira execução sem protect definido (parte 14)

```
Consumidor - Buffer Cheio
 ABCDEFGHIJKLMNOP

Consumidor - Buffer Cheio
 ABCDEFGHIJKLMNOPQRSTUVWXYZ ab########
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 4: V#W#X##########################


Consumidor - Buffer Cheio
#ABCDEFGHIJKLMNOPQRSTUVWXYZ ab###################################
VWXQRSTUVWXYZ ab#######
Caracteres produzidos pelo filho 2: ####YZ## a######################

Caracteres produzidos pelo filho 4:  A
Caracteres produzidos pelo filho 2: BbCc
Caracteres produzidos pelo filho 2: d
Caracteres produzidos pelo filho 4: d
Caracteres produzidos pelo filho 2: efgh
Caracteres produzidos pelo filho 4: efgh
Caracteres produzidos pelo filho 2: ijkl
Caracteres produzidos pelo filho 6: 5678
Caracteres produzidos pelo filho 8: 5678
Caracteres produzidos pelo filho 4: ijkl
```

Figura 20 - Resultado da primeira execução sem protect definido (parte 15)

```
Produtor - Buffer Cheio
 ABC##################V
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 6: mWnXopYZ abmcdneofpghijk
Caracteres produzidos pelo filho 4: lmnompnqopq##################90
Caracteres produzidos pelo filho 2: 901
Caracteres produzidos pelo filho 6: rs
Caracteres produzidos pelo filho 8: rs
Caracteres produzidos pelo filho 4: rst
Caracteres produzidos pelo filho 6: t
Caracteres produzidos pelo filho 8: uv
Caracteres produzidos pelo filho 2: tuvw
Caracteres produzidos pelo filho 6: uvw
Caracteres produzidos pelo filho 2: x
Caracteres produzidos pelo filho 4: u
Caracteres produzidos pelo filho 8: vxw
Caracteres produzidos pelo filho 6: xy
Caracteres produzidos pelo filho 4: yz
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 2: yyz
Caracteres produzidos pelo filho 6: zz
Caracteres produzidos pelo filho 4: 12
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 8: 1 2123
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 6: 4 5123
Caracteres produzidos pelo filho 4: 34567
Caracteres produzidos pelo filho 8: 45
Caracteres produzidos pelo filho 4:
```

Figura 21 - Resultado da primeira execução sem protect definido (parte 16)

```
Produtor - Buffer Cheio
 ABC#6########

Produtor - Buffer Cheio
# #A#B########
Caracteres produzidos pelo filho 8: C6##############################################st#u#vwx#y#z 1#2#3456#7#8#9#0#
Caracteres produzidos pelo filho 6: 89#0############tuvwxyz 1234567890
Caracteres produzidos pelo filho 2: 890W
Caracteres produzidos pelo filho 4:  ABC
Caracteres produzidos pelo filho 8:  AB
Caracteres produzidos pelo filho 4: C
Caracteres produzidos pelo filho 8: C
Caracteres produzidos pelo filho 6:  ABC
Caracteres produzidos pelo filho 2: DEFGH
Caracteres produzidos pelo filho 8: DEFG
Caracteres produzidos pelo filho 4: DEFG
Caracteres produzidos pelo filho 6: DEFG
Caracteres produzidos pelo filho 2: IJKH
Caracteres produzidos pelo filho 6: LM
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 8: HILJMK
Caracteres produzidos pelo filho 4: IJKLM
Caracteres produzidos pelo filho 8: L
Caracteres produzidos pelo filho 2: N
Caracteres produzidos pelo filho 8: OP
Caracteres produzidos pelo filho 4: OP
Caracteres produzidos pelo filho 6: LMNOP
Caracteres produzidos pelo filho 2: OPQ
Caracteres produzidos pelo filho 6:
```

Figura 22 - Resultado da primeira execução sem protect definido (parte 17)

```
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 8: RR
Caracteres produzidos pelo filho 4: RR

Consumidor - Buffer Cheio
 ABCDEFGHIJKLM
Caracteres produzidos pelo filho 8: NS

Consumidor - Buffer Cheio
T AU

Consumidor - Buffer Cheio
BOPC QDAREBSFCTGDUH#EIFJGKHLIMJK#L##M#NOP#QRSTUV##############
Caracteres produzidos pelo filho 6: #S#T#U#V##################
Caracteres produzidos pelo filho 8: # ##A#######
N
Caracteres produzidos pelo filho 2: OSPTQURST#UV#W#X#######################################################
Caracteres produzidos pelo filho 8: ##B################
######

Caracteres produzidos pelo filho 2: B
Caracteres produzidos pelo filho 4: CSTUV
Caracteres produzidos pelo filho 6: WXYZ
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 6:  a
Caracteres produzidos pelo filho 8: CDEF
Caracteres produzidos pelo filho 4:  a
Caracteres produzidos pelo filho 2: GHIJ
Caracteres produzidos pelo filho 6: bcde
Caracteres produzidos pelo filho 2: fg
```

Figura 23 - Resultado da primeira execução sem protect definido (parte 18)

```
Caracteres produzidos pelo filho 8: bcdef
Caracteres produzidos pelo filho 4: bcd
Caracteres produzidos pelo filho 6: fgehf
Caracteres produzidos pelo filho 8: i
Caracteres produzidos pelo filho 6: i
Caracteres produzidos pelo filho 4: ij
Caracteres produzidos pelo filho 2: ghijk
Caracteres produzidos pelo filho 6: jkl
Caracteres produzidos pelo filho 4: l
Caracteres produzidos pelo filho 2: m
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 6: mjnklm
Caracteres produzidos pelo filho 2: nop
Caracteres produzidos pelo filho 4: nop
Caracteres produzidos pelo filho 6: opqrs
Caracteres produzidos pelo filho 2: qr
Caracteres produzidos pelo filho 8: opqrs
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 4: stuv
Caracteres produzidos pelo filho 6: stuvw
Caracteres produzidos pelo filho 6: x
Caracteres produzidos pelo filho 2: tuv
Caracteres produzidos pelo filho 8: tuv
Caracteres produzidos pelo filho 4: tuvw
Caracteres produzidos pelo filho 6: wxy
Caracteres produzidos pelo filho 2: wxyz
Caracteres produzidos pelo filho 4: xy
Caracteres produzidos pelo filho 8: xy
Caracteres produzidos pelo filho 8: z
Caracteres produzidos pelo filho 4: z
```

Figura 24 - Resultado da primeira execução sem protect definido (parte 19)

```
Caracteres produzidos pelo filho 6: z 12
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 4: z3 41523
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 4: 4345
Caracteres produzidos pelo filho 6: 3456
Caracteres produzidos pelo filho 2: 56
Caracteres produzidos pelo filho 4: 789
Caracteres produzidos pelo filho 8: 78

Produtor - Buffer Cheio
9##B####################################
Caracteres produzidos pelo filho 6: #7#8#
Caracteres produzidos pelo filho 4:  A#####tuvwxyz 129345607890
Caracteres produzidos pelo filho 2: 0123
Caracteres produzidos pelo filho 4:
Caracteres produzidos pelo filho 6: BB
Caracteres produzidos pelo filho 8:  AB
Caracteres produzidos pelo filho 4:
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 2: CCDCDE
Caracteres produzidos pelo filho 6: CDDEE
Caracteres produzidos pelo filho 8:
Caracteres produzidos pelo filho 2: FF
Caracteres produzidos pelo filho 4: G
Caracteres produzidos pelo filho 6: GH
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 8: GGHH
Caracteres produzidos pelo filho 4: IJKL
Caracteres produzidos pelo filho 2: IJKL
```

Figura 25 - Resultado da primeira execução sem protect definido (parte 20)

```
Caracteres produzidos pelo filho 6: IJKL
Caracteres produzidos pelo filho 8: IJKLM
Caracteres produzidos pelo filho 4: MNO
Caracteres produzidos pelo filho 6:
Caracteres produzidos pelo filho 2: MMNNOOPP
Caracteres produzidos pelo filho 4: QRS
Caracteres produzidos pelo filho 8: QRS
Caracteres produzidos pelo filho 2: QRST
Caracteres produzidos pelo filho 6: QRST
Caracteres produzidos pelo filho 8: TU
Caracteres produzidos pelo filho 4: TU

Consumidor - Buffer Cheio
 ABCDEFGHIJKLMNOPQRSTUVWX

Consumidor - Buffer Cheio
# #A#B#C#D#E######

Consumidor - Buffer Cheio
# #A#B#C#D#E#F

Consumidor - Buffer Cheio
 #AB#CG#D#E#F#GH#H#I#J#K#L#M#N#IO#P#Q#R#SJT#U#V#W#X
##K####L####F##G#H#I##J##
Caracteres produzidos pelo filho 2:
Caracteres produzidos pelo filho 4: MN#OPQRSTUVWXY######
Caracteres produzidos pelo filho 8: #V#W#X#Y################################
KL
Caracteres produzidos pelo filho 6: MNXUY#VW####################
VOWXYPQRSTUVWXYZ
```

Figura 26 - Resultado da primeira execução sem protect definido (parte 21)

```
Caracteres produzidos pelo filho 4: aZb##
Caracteres produzidos pelo filho 8: Z# #a#b################################

Caracteres produzidos pelo filho 2: Z abc
Caracteres produzidos pelo filho 6:  ab
Caracteres produzidos pelo filho 8: cd
Caracteres produzidos pelo filho 4: efcdef
Caracteres produzidos pelo filho 2: cde
Caracteres produzidos pelo filho 8: g
Caracteres produzidos pelo filho 4: hgihjij
Caracteres produzidos pelo filho 6: ghi
Caracteres produzidos pelo filho 2: fg
Caracteres produzidos pelo filho 8: jkl
Caracteres produzidos pelo filho 4: jkl
Caracteres produzidos pelo filho 6: hijk
Caracteres produzidos pelo filho 4: m
Caracteres produzidos pelo filho 2: nhij
Caracteres produzidos pelo filho 8: mnkl
```

Figura 27 - Resultado da primeira execução sem protect definido (parte 22)

Respostas das perguntas

Perguntas do relatório

Pergunta 1: Uma região por ser crítica tem garantida a exclusão mútua? Justifique.

Resposta: Não. Para que a região tenha a exclusão mútua, é necessário o programador implementar algum método, como semáforo (usado no experimento).

Pergunta 2: É obrigatório que todos os processos que acessam o recurso crítico tenham uma região crítica igual?

Resposta: Não, basta que estejam compartilhando o mesmo recurso, a mesma variável para que tenham uma região crítica.

Pergunta 3: Porque as operações sobre semáforos precisam ser atômicas

Resposta: Porque no momento de escalonamento um processo B pode acessar uma variável que o processo A ainda não terminou de usá-la. Sendo assim o dado desta variável foi manipulado incorretamente, gerando uma inconsistência e para isto não ocorrer é necessário que o momento de manipulação de uma variável compartilhada seja atômico, uma vez iniciado não será interrompido até sua finalização, só pode acorrer por inteiro.

Pergunta 4: O que é uma diretiva ao compilador?

Resposta: As diretivas de compilação são comandos que não são compilados, sendo dirigidos ao pré-processador, executado pelo compilador antes da execução do processo de compilação propriamente dito.

Pergunta 5: Porque o número é pseudo aleatório e não totalmente aleatório?

Resposta: Porque não existem funções que gerem números genuinamente aleatórios, eles são formados por operações matemáticas, e uma vez que a semente da operação se repetir, toda a sequência irá se repetir também.

Perguntas do programa

Pergunta 1: Se usada a estrutura g_sem_op1 terá qual efeito em um conjunto de semáforos?

Resposta: A estrutura g_sem_op1 é usada para travar o semáforo, na hora de entrar em uma região crítica.

Pergunta 2: Para que serve esta operação semop(), se não está na saída de uma região crítica?

Resposta: A operação serve para que o semáforo comece a execução do programa destravado.

Pergunta 3: Para que serve essa inicialização da memória compartilhada com zero?

Resposta: A variável *g_shm_addr é inicializada com zero, porque ela é um ponteiro inteiro que aponta para o segmento de memória compartilhada, portanto é necessário inicializar no início da memória.

Pergunta 4: Se os filhos ainda não terminaram, semctl e shmctl, com o parâmetro IPC-RMID, não permitem mais o acesso ao semáforo / memória compartilhada?

Resposta: No caso do programa exemplo, os filhos são mortos antes que os semáforos e a memória compartilhada sejam excluídos, logo não ocorrerá o acesso. Porém, se forem removidos antes, os filhos não terão acesso.

Pergunta 5: Quais os valores possíveis de serem atribuídos a number?

Resposta: 1, 2 ou 3, na primeira tarefa. 1, 2, 3, 4 ou 5, na segunda tarefa.

Análise dos Resultados

Tarefa 1 (programa exemplo)

A tarefa 1 possibilitou o melhor entendimento sobre o uso de semáforos e sua importância. Antes de discutirmos o efeito do semáforo no programa, é necessário a discussão do conceito de race condition, exclusão mútua e região crítica.

Race condition (condição de corrida), ocorre quando dois ou mais processos querem acessar o mesmo recurso e o resultado depende de quem foi escalonado primeiro. Portanto, dependendo da ordem de escalonamento, o resultado obtido não será o esperado. Para evitar que processos concorrentes acessem o mesmo recurso, é necessário algum mecanismo de sincronização. A exclusão mútua garante que, se um processo esteja acessando uma variável ou um arquivo compartilhado, os outros processos estarão bloqueados, garantindo acesso exclusivo de um único processo. A região (ou seção) crítica, é onde se localiza os recursos compartilhados.

Como já citado, o recurso compartilhado é um inteiro, usado para acessar uma string de caracteres. O uso do semáforo é necessário para que a impressão ocorra de forma correta, como apresentada na seção de resultados.

Percebemos que quando o protect está definido, a impressão ocorre de forma correta, as letras e números são imprimidos exatamente na ordem em que estão na string. Portanto, podemos concluir que o uso do protect é essencial para que o semáforo faça sua função corretamente.

Já quando o protect não está definido, a impressão não ocorre da maneira esperada. Isso se deve ao fato de que vários processos tentaram acessar a variável compartilhada, mudando o valor da mesma, de maneira incorreta, a cada execução, fazendo com que a impressão ocorresse de maneira errada.

Tarefa 2 (programa modificado)

Os mesmos conceitos de race condition, exclusão mutua e região crítica valem para esta tarefa. Também foi observado como o uso do protect é necessário para evitar que dois ou mais processos acessem a região crítica ao mesmo tempo.

Como foi observado nos resultados desta tarefa, quando usamos o protect, a impressão ocorreu de forma correta, mostrando o que cada filho produziu e o buffer quando estivesse cheio. Mais uma vez, verificamos que o uso de semáforos nessa tarefa foi essencial.

A não definição do protect, assim como na tarefa 1, levou a uma impressão completamente desordenada. Também observamos a extensão dos resultados (podemos observar que os resultados colocados aqui no relatório, foram divididos em vinte e duas partes). Isso se deve ao fato de que os processos filhos (8 nessa tarefa), não são bloqueados pelos semáforos, e como cada processo é independente, estes executam até o pai executar o comando kill.

Conclusão

Através deste experimento, foi observado o funcionamento de semáforos e da memória compartilhada. Conceitos como exclusão mútua, região crítica e condições de corrida, também foram discutidos e o entendimento de cada um ficou mais fácil.

Também foi possível verificar o funcionamento de algumas funções par manipulação de semáforos e memoria compartilhada, como: semget(), semop(), semctl(), shmget(), shmat() e shmctl(). As três primeiras, são relacionadas à semáforos e as três últimas à memória compartilhada (o comando man do Linux auxilia no entendimento de cada uma).