Laboratório de Memória Compartilhada

Sistemas de Computação - 3WB

Victor Augusto - 1310784

1. Introdução

Foram propostos 3 exercícios em aula. No primeiro era pedido para executar o somatório de matrizes, onde o resultado da soma delas fica em um espaço de memória compartilhada, e cada linha deve ser somada por um processo diferente. No segundo era necessário trocar mensagens entre dois programas, onde um lia e o outro escrevia. Já no terceiro era necessário pesquisar por um valor em um vetor de inteiros de 100 casas (não ordenado e com valores repetidos) e indicar em quais casas se encontrava o número pedido.

2. Objetivo

Reforçar os conceitos de memória compartilhada aprendidos em aula.

3. Exercícios

3.1 Exercício 1

3.1.1 Enunciado

"Faça um programa para somar matrizes de acordo com o seguinte algoritmo: O primeiro processo irá criar duas matrizes preenchidas e uma terceira vazia na memória compartilhada. Para cada linha da matriz solução, o seu programa deverá gerar um processo para o seu cálculo. Ao final deve ser impressa a matriz resultante."

3.1.2 Código Fonte

```
1.  #include <sys/types.h>
2.  #include <sys/ipc.h>
3.  #include <sys/shm.h>
4.  #include <sys/wait.h>
5.  #include <sys/stat.h>
6.  #include <unistd.h>
7.  #include <stdio.h>
8.  #include <stdlib.h>
9.
10. #define NUM_FILHOS 3
11.
12. int main (void) {
```

```
13.
14.     int matriz1[9] = {1,2,3,1,2,3,1,2,3};
15.     int matriz2[9] = {1,2,3,1,2,3,1,2,3};
16.
17.     pid_t pid;
18.
19.     int shmid;
20.
21.     shmid = shmget(IPC_PRIVATE, 9*sizeof(int), IPC_CREAT | IPC_EXCL | S_IRUSR |
    S_IWUSR);
22.
23.     if (shmid == -1) {
24.         printf("Error in shmget.\n\n");
25.         exit(1);
26.     }
27.
28.     int *matrizResultado = (int*)shmat(shmid, 0, 0);
29.
30.     for(int j = 0; j < NUM_FILHOS; j++) {
31.         pid = fork();
32.         if (pid == 0) {
33.             printf("\nProcesso Filho %d Inicio\n\n", j+1);
34.             for (int i = 0; i < 3; i++) {
35.                 int soma = matriz1[i+3*j] + matriz2[i+3*j];
36.                 matrizResultado[i+3*j] = soma;
37.                 printf("Linha: %d / Coluna: %d / Valor: %d\n", j, i,
    matrizResultado[i+3*j]);
38.             }
39.             printf("\nProcesso Filho %d Fim\n\n", j+1);
40.             exit(0);
41.         }
42.     }
43.
44.     for (int i = 0; i < NUM_FILHOS; i++) {
45.         waitpid(-1, NULL, 0);
46.     }
47.
48.     printf("Matriz Resultante:\n\n");
49.     for (int i = 0; i < 3; i++) {
50.         printf("%d %d %d\n", matrizResultado[i*3], matrizResultado[i*3+1],
    matrizResultado[i*3+2]);
51.     }
52.
53.     shmdt(matrizResultado);
54.     shmctl(shmid, IPC_RMID, 0);
55.
56.     return 0;
57. }
58.
```

3.1.3 Resultado

```
Processo Filho 1 Inicio

Linha: 0 / Coluna: 0 / Valor: 2
Linha: 0 / Coluna: 1 / Valor: 4
Linha: 0 / Coluna: 2 / Valor: 6

Processo Filho 1 Fim


Processo Filho 2 Inicio

Linha: 1 / Coluna: 0 / Valor: 2
Linha: 1 / Coluna: 1 / Valor: 4
Linha: 1 / Coluna: 2 / Valor: 6

Processo Filho 2 Fim


Processo Filho 3 Inicio

Linha: 2 / Coluna: 0 / Valor: 2
Linha: 2 / Coluna: 1 / Valor: 4
Linha: 2 / Coluna: 2 / Valor: 6

Processo Filho 3 Fim

Matriz Resultante:

2 4 6
2 4 6
2 4 6
```

3.2 Exercício 2

3.2.1 Enunciado

“Faça um programa que leia uma mensagem e a armazene na memória compartilhada com id = 8752. Faça um outro programa que utilize o mesmo id (8752) e exiba a mensagem para o usuário.“

3.2.2 Código Fonte

3.2.2.1 Programa que escreve

```
1.  #include <sys/ipc.h>
2.  #include <sys/shm.h>
3.  #include <sys/stat.h>
4.  #include <string.h>
5.  #include <stdio.h>
6.  #include <stdlib.h>
7.
8.  char* getMessage();
9.
10. int main (void) {
11.
12.     int shmid;
13.     key_t key = 8752;
14.
15.     char *message = getMessage();
16.     printf("%s\n", message);
17.
18.     shmid = shmget(key, 1000*sizeof(char), IPC_CREAT | IPC_EXCL | S_IRUSR | S_IWUSR);
19.
20.     if (shmid == -1) {
21.         printf("Error in shmget.\n\n");
22.         exit(1);
23.     }
24.
25.     char *mem = (char*)shmat(shmid, 0, 0);
26.     strcpy(mem, message);
27.     printf("Frase salva com sucesso!\n");
28.     shmdt(mem);
29.
30.     return 0;
31. }
32.
33. char* getMessage() {
```

```
    char buffer[1000];
    scanf("%1000[^\n]", buffer);
    char *message = (char*)malloc(sizeof(char)*(strlen(buffer)));
    return strcpy(message, buffer);
}

```

3.2.2.2 Programa que lê

```
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main (void) {

    int shmid;
    key_t key = 8752;

    shmid = shmget(key, sizeof(char)*1000, IPC_EXCL | S_IRUSR | S_IWUSR);

    if (shmid == -1) {
        printf("Error in shmget.\n\n");
        exit(1);
    }

    char *mem = (char*)shmat(shmid, 0, SHM_RDONLY);
    printf("Frase recuperada:\n");
    printf("%s\n", mem);
    shmdt(mem);
    shmctl(shmid, IPC_RMID, 0);

    return 0;
}

```

3.2.3 Resultado

```
./write
Frase de teste.
Frase de teste.
Frase salva com sucesso!

./read
Frase recuperada:
Frase de teste.
```

3.3 Exercício 3

3.3.1 Enunciado

“Faça um programa paralelo para localizar uma chave em um vetor de 100 posições. Crie o vetor na memória compartilhada com dados numéricos inteiros, desordenados e com repetições. Divida o espaço de busca e gere processos para procurar a chave na sua área de busca, informando as posições onde a chave foi localizada. Ao final o programa deve listar as posições onde a chave foi encontrada.“

3.3.2 Código Fonte

```
1.  #include <sys/types.h>
2.  #include <sys/ipc.h>
3.  #include <sys/shm.h>
4.  #include <sys/wait.h>
5.  #include <sys/stat.h>
6.  #include <unistd.h>
7.  #include <stdio.h>
8.  #include <stdlib.h>
9.
10. #define NUM_FILHOS 10
11. #define NUMBER 1
12.
13. void generateRandomVector(int* randomVector);
14.
15. int main (void) {
16.
17.     int shmid_vector, shmid_count;
18.     pid_t pid;
19.
20.     shmid_vector = shmget(IPC_PRIVATE, 100*sizeof(int), IPC_CREAT | IPC_EXCL | S_IRUSR
    | S_IWUSR);
21.     shmid_count = shmget(IPC_PRIVATE, 100*sizeof(int), IPC_CREAT | IPC_EXCL | S_IRUSR |
    S_IWUSR);
22.
23.     if (shmid_vector == -1) {
24.         printf("Error in shmget.\n\n");
25.         exit(1);
26.     }
27.     if (shmid_count == -1) {
28.         printf("Error in shmget.\n\n");
29.         exit(1);
30.     }
31.
```

```
32.     int* randomVector = (int*)shmat(shmid_vector, 0, 0);
33.     int* results = (int*)shmat(shmid_count, 0, 0);
34.     generateRandomVector(randomVector);
35. 
36.     for(int j = 0; j < NUM_FILHOS; j++) {
37.         pid = fork();
38.         if (pid == 0) {
39.             printf("\nProcesso Filho %d Inicio\n\n", j+1);
40.             for (int i = 0; i < 10; i++) {
41.                 printf("Casa: %d / Valor:%d\n",j*10+i, randomVector[j*10+i]);
42.                 if (randomVector[j*10+i] == NUMBER) {
43.                     printf("Achou %d\n", NUMBER);
44.                     results[j*10+i] = 1;
45.                 }
46.             }
47.             printf("\nProcesso Filho %d Fim\n", j+1);
48.             exit(0);
49.         }
50.     }
51. 
52.     for (int i = 0; i < NUM_FILHOS; i++) {
53.         waitpid(-1, NULL, 0);
54.     }
55. 
56.     printf("\nPosiÃ§Ãµes:\n");
57.     for (int i = 0; i < 100; i++) {
58.         if (results[i] == 1) {
59.             printf("%d\n", i);
60.         }
61.     }
62. 
63.     shmdt(randomVector);
64.     shmdt(results);
65. 
66.     shmctl(shmid_vector, IPC_RMID, 0);
67.     shmctl(shmid_count, IPC_RMID, 0);
68. 
69.     return 0;
70. }
71. 
72. void generateRandomVector(int* randomVector) {
73.     for (int i = 0; i < 100; i++) {
74.         randomVector[i] = i%5;
75.     }
76. }
77. 
```

3.3.3 Resultado

```
Processo Filho 1 Inicio

Casa: 0 / Valor:0
Casa: 1 / Valor:1
Achou 1
Casa: 2 / Valor:2
Casa: 3 / Valor:3
Casa: 4 / Valor:4
Casa: 5 / Valor:0
Casa: 6 / Valor:1
Achou 1
Casa: 7 / Valor:2
Casa: 8 / Valor:3
Casa: 9 / Valor:4

Processo Filho 1 Fim

Processo Filho 2 Inicio

Casa: 10 / Valor:0
Casa: 11 / Valor:1
Achou 1
Casa: 12 / Valor:2
Casa: 13 / Valor:3
Casa: 14 / Valor:4
Casa: 15 / Valor:0
Casa: 16 / Valor:1
Achou 1
Casa: 17 / Valor:2
Casa: 18 / Valor:3
Casa: 19 / Valor:4

Processo Filho 2 Fim

Processo Filho 3 Inicio

Casa: 20 / Valor:0
Casa: 21 / Valor:1
Achou 1
Casa: 22 / Valor:2
Casa: 23 / Valor:3
Casa: 24 / Valor:4
Casa: 25 / Valor:0
Casa: 26 / Valor:1
Achou 1
Casa: 27 / Valor:2
```

```
Casa: 28 / Valor:3
Casa: 29 / Valor:4

Processo Filho 3 Fim

Processo Filho 4 Inicio

Casa: 30 / Valor:0
Casa: 31 / Valor:1
Achou 1
Casa: 32 / Valor:2
Casa: 33 / Valor:3
Casa: 34 / Valor:4
Casa: 35 / Valor:0
Casa: 36 / Valor:1
Achou 1
Casa: 37 / Valor:2
Casa: 38 / Valor:3
Casa: 39 / Valor:4

Processo Filho 4 Fim

Processo Filho 5 Inicio

Casa: 40 / Valor:0
Casa: 41 / Valor:1
Achou 1
Casa: 42 / Valor:2
Casa: 43 / Valor:3
Casa: 44 / Valor:4
Casa: 45 / Valor:0
Casa: 46 / Valor:1
Achou 1
Casa: 47 / Valor:2
Casa: 48 / Valor:3
Casa: 49 / Valor:4

Processo Filho 5 Fim

Processo Filho 6 Inicio

Casa: 50 / Valor:0
Casa: 51 / Valor:1
Achou 1
Casa: 52 / Valor:2
Casa: 53 / Valor:3
Casa: 54 / Valor:4
Casa: 55 / Valor:0
Casa: 56 / Valor:1
Achou 1
```

```
Casa: 57 / Valor:2
Casa: 58 / Valor:3
Casa: 59 / Valor:4

Processo Filho 6 Fim

Processo Filho 7 Inicio

Casa: 60 / Valor:0
Casa: 61 / Valor:1
Achou 1
Casa: 62 / Valor:2
Casa: 63 / Valor:3
Casa: 64 / Valor:4
Casa: 65 / Valor:0
Casa: 66 / Valor:1
Achou 1
Casa: 67 / Valor:2
Casa: 68 / Valor:3
Casa: 69 / Valor:4

Processo Filho 7 Fim

Processo Filho 8 Inicio

Casa: 70 / Valor:0
Casa: 71 / Valor:1
Achou 1
Casa: 72 / Valor:2
Casa: 73 / Valor:3
Casa: 74 / Valor:4
Casa: 75 / Valor:0
Casa: 76 / Valor:1
Achou 1
Casa: 77 / Valor:2
Casa: 78 / Valor:3
Casa: 79 / Valor:4

Processo Filho 8 Fim

Processo Filho 9 Inicio

Casa: 80 / Valor:0
Casa: 81 / Valor:1
Achou 1
Casa: 82 / Valor:2
Casa: 83 / Valor:3
Casa: 84 / Valor:4
Casa: 85 / Valor:0
Casa: 86 / Valor:1
```

```
Achou 1
Casa: 87 / Valor:2
Casa: 88 / Valor:3
Casa: 89 / Valor:4

Processo Filho 9 Fim

Processo Filho 10 Inicio

Casa: 90 / Valor:0
Casa: 91 / Valor:1
Achou 1
Casa: 92 / Valor:2
Casa: 93 / Valor:3
Casa: 94 / Valor:4
Casa: 95 / Valor:0
Casa: 96 / Valor:1
Achou 1
Casa: 97 / Valor:2
Casa: 98 / Valor:3
Casa: 99 / Valor:4

Processo Filho 10 Fim

Posições:
1
6
11
16
21
26
31
36
41
46
51
56
61
66
71
76
81
86
91
96
```

4 Conclusão

4.1 Exercício 1

O exercício proposto foi fundamental para entender o básico de memória compartilhada e como ela deve ser usada. Os comandos `shmget`, `shmat`, `shmdt` e `shmctl` foram fundamentais para conseguir realizar o exercício proposto, e para isso tiveram que ser muito bem entendido (junto com os seus respectivos parâmetros).

Pela sua saída fica claro que todos os processos filhos estão escrevendo na memória compartilhada, e o processo pai consegue recuperar todos esses dados corretamente.

4.2 Exercício 2

O exercício proposto tinha como pré-requisito entender todos os conceitos utilizados pelo exercício 1 e aprofundá-los. Foi entendido que se dois programas acessam a mesma memória compartilhada, a mesma não pode ser fechada ate que não vá mais ser usada. Aprendeu-se também a garantir o uso da mesma área de memória compartilhada entre vários programas através da chave que se passa como parâmetro para o `shmget`.

Pela sua saída fica claro que, se o programa que recupera a mensagem consegue ler o que o programa que escreve a mensagem colocou no espaço de memória dada algumas condições. O programa que escreve que é executado primeiro não pode ter fechado o espaço de memória (`shmctl`) e a chave usada no `shmget` nos dois programas tem que ser a mesma.

4.3 Exercício 3

O exercício proposto foi fundamental para aprender a gerenciar múltiplos processos acessando a mesma memória compartilhada usando os conhecimentos adquiridos no laboratório anterior com o deste relatório.

Pela sua saída fica claro que eles estão executados independentemente e o processo pai espera todos acabarem antes de continuas, recuperando corretamente os dados que estavam no espaço de memória compartilhada sobre a posição do número desejado, os quais foram escritos pelo processo filho.