

## Relatório laboratório 4 - IPC

### Código process-A.c e process-B.c

#### EXPLICAÇÃO DO CÓDIGO

Neste código é criada uma fila no sistema com permissões de escrita e leitura.

```
// abre ou cria a fila com permissões 0666
if ((queue = mq_open (QUEUE, O_RDWR|O_CREAT, 0666, &attr)) < 0) {
    perror ("mq_open");
    exit (1);
}
```

Essa fila possui as seguintes características:

- Buffer de até 10 mensagens;
- Cada mensagem possui o tamanho de msg. (msg foi declarado como *inteiro[int]*)

```
// define os atributos da fila
attr.mq_maxmsg = 10;           // capacidade para 10 mensagens
attr.mq_msgsize = sizeof(msg); // tamanho de cada mensagem
attr.mq_flags = 0;
```

Quando o programa **process-A** é executado ele cria ou abre a fila e inicia a enviar mensagens para ela com números aleatórios gerados com a função rand e o numero de PID (process ID) associado ao processo executando o programa.

```
for (;;)
{
    num = random() % 100 ; // valor entre 0 e 99

    // envia o numero como mensagem
    counter = 0;
    do {
        if (counter == 0)
            msg = num;
        else
            msg = getpid();
        counter++;

        if (mq_send (queue, (void*) &msg, sizeof(msg), 0) < 0)
        {
            perror ("mq_send");
            exit (1);
        }
        printf ("Sent message with value %d\n", msg);
    } while (counter != 2);
    sleep (1) ;
}
```

Essas mensagens vão enchendo o buffer até o ponto em que, caso o programa **process-B** não comece a ler, o programa **process-A** não consegue mais enviar mensagens.

Quando o programa **process-B** é executado ele abre a fila de mensagens caso ela exista.

```
// abre a fila, se existir
if ((queue = mq_open (QUEUE, O_RDWR)) < 0)
{
    perror ("mq_open");
    exit (1);
}
```

E inicia a leitura de informações na fila.

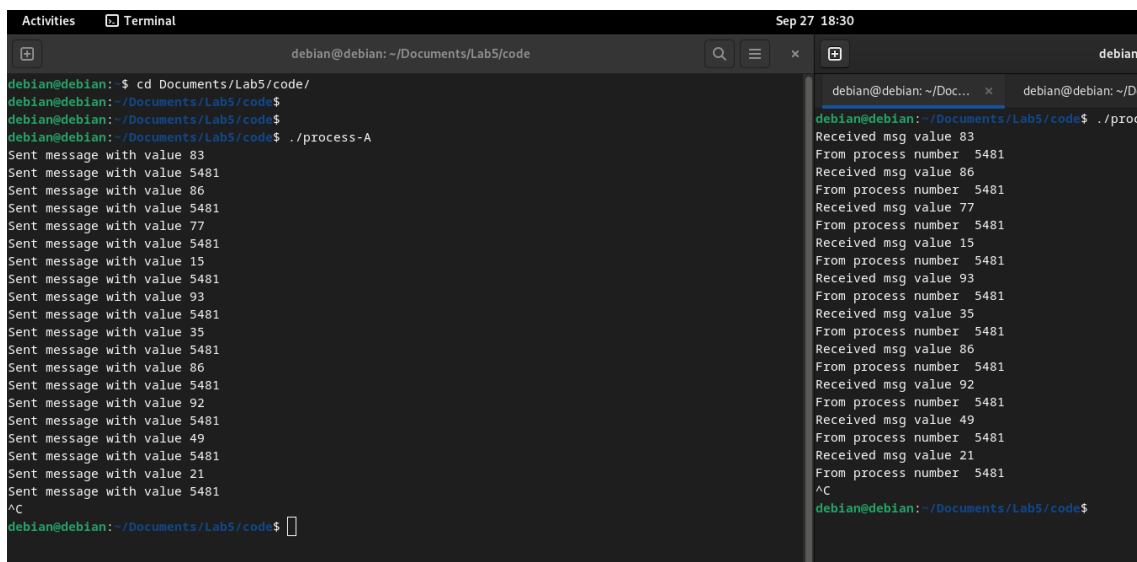
```
// recebe cada mensagem e imprime seu conteúdo
for (;;)
{
    if ((mq_receive (queue, (void*) &msg, sizeof(msg), 0)) < 0) {
        perror("mq_receive:");
        exit (1);
    }

    if (counter == 0) {
        printf ("Received msg value %d\n", msg);
        counter++;
    }
    else {
        printf ("From process number %5d\n", msg);
        counter = 0;
    };
}
}
```

Caso já existam mensagens no buffer da fila ele lê todas de uma vez.

## VALIDAÇÃO DA TROCA DE INFORMAÇÕES

Para dizer que houve troca de informações entre o programa **process-A** e o programa **process-B** foi utilizado a verificação dos valores enviados pelo programa **process-A** e a leitura dos valores do programa **process-B**. Foi percebido que os valores enviados são os mesmos recebidos.



```
Activities  Terminal  Sep 27 18:30

debian@debian: ~/Documents/Lab5/code
debian@debian: ~/Documents/Lab5/code$ cd Documents/Lab5/code/
debian@debian: ~/Documents/Lab5/code$
debian@debian: ~/Documents/Lab5/code$ ./process-A
Sent message with value 83
Sent message with value 5481
Sent message with value 86
Sent message with value 5481
Sent message with value 77
Sent message with value 5481
Sent message with value 15
Sent message with value 5481
Sent message with value 93
Sent message with value 5481
Sent message with value 35
Sent message with value 5481
Sent message with value 86
Sent message with value 5481
Sent message with value 92
Sent message with value 5481
Sent message with value 49
Sent message with value 5481
Sent message with value 21
Sent message with value 5481
^C
debian@debian: ~/Documents/Lab5/code$

debian@debian: ~/Documents/Lab5/code$ ./process-B
Received msg value 83
From process number 5481
Received msg value 86
From process number 5481
Received msg value 77
From process number 5481
Received msg value 15
From process number 5481
Received msg value 93
From process number 5481
Received msg value 35
From process number 5481
Received msg value 86
From process number 5481
Received msg value 92
From process number 5481
Received msg value 49
From process number 5481
Received msg value 21
From process number 5481
^C
debian@debian: ~/Documents/Lab5/code$
```

## Código shared-mem-tx.c e shared-mem-rx.c

### EXPLICAÇÃO DO CÓDIGO

Neste código é reservada, alocada e mapeada uma área na memória em que os dois códigos irão compartilhar.

```
// Passos 1 a 3: abre/cria uma area de memoria compartilhada
fd = shm_open("/sharedmem", O_RDWR|O_CREAT, S_IRUSR|S_IWUSR);
if(fd == -1) {
    perror ("shm_open");
    exit (1) ;
}

// Passos 1 a 3: ajusta o tamanho da area compartilhada
if (ftruncate(fd, sizeof(value)) == -1) {
    perror ("ftruncate");
    exit (1) ;
}

// Passos 2 a 4: mapeia a area no espaco de enderecamento deste processo
ptr = mmap(NULL, sizeof(value), PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);
if(ptr == MAP_FAILED) {
    perror ("mmap");
    exit (1);
}
```

Essa área da memória possui o tamanho da variável **value** que é um **inteiro(int)**. Uma vez que essa área exista o programa **shared-mem-tx** abre esta área e escreve valores aleatórios nela. Após isso escreve no terminal o valor que escreveu.

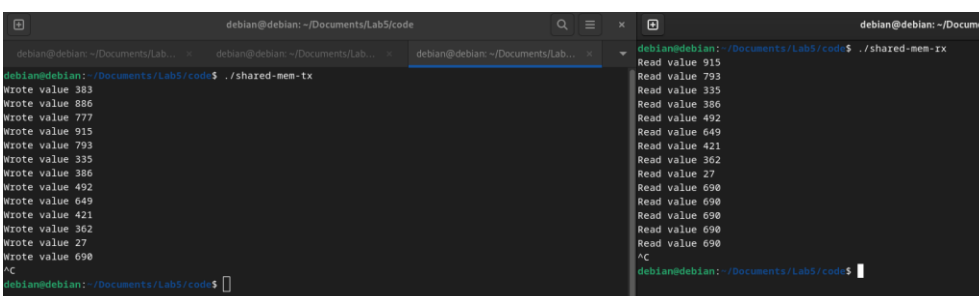
```
for (;;) {
    // Passo 5: escreve um valor aleatorio na area compartilhada
    value = random () % 1000 ;
    (*ptr) = value ;
    printf ("Wrote value %i\n", value) ;
    sleep (1);
}
```

Enquanto que o programa **shared-mem-rx** abre a área e lê os valores e escreve no terminal.

```
for (;;) {
    // Passo 5: lê e imprime o conteúdo da area compartilhada
    value = (*ptr) ;
    printf("Read value %i\n", value);
    sleep (1) ;
}
```

Esta forma de implementação tem um problema, pois como a área de memória compartilhada foi alocada com o valor de um inteiro e a mensagem enviada é um inteiro, é possível guardar apenas um valor por vez na área alocada.

Dessa forma gera o seguinte problema, caso o programa **shared-mem-rx** demore muito a iniciar, a informação será sobrescrita pelo programa **shared-mem-tx**.



```
debian@debian: ~/Documents/Lab5/code$ ./shared-mem-tx
Wrote value 383
Wrote value 886
Wrote value 777
Wrote value 915
Wrote value 793
Wrote value 335
Wrote value 386
Wrote value 492
Wrote value 649
Wrote value 421
Wrote value 362
Wrote value 27
Wrote value 690
^C
debian@debian: ~/Documents/Lab5/code$

debian@debian: ~/Documents/Lab5/code$ ./shared-mem-rx
Read value 915
Read value 793
Read value 335
Read value 386
Read value 492
Read value 649
Read value 421
Read value 362
Read value 27
Read value 690
Read value 690
Read value 690
Read value 690
Read value 690
^C
debian@debian: ~/Documents/Lab5/code$
```

## VALIDAÇÃO DA TROCA DE INFORMAÇÕES

A forma de validação foi a mesma adotada no ultimo código, verificação se a informação enviada pelo **shared-mem-tx** foi recebida pelo **shared-mem-rx**.

### Diferença do código process-A.c/process-B.c para o código shared-mem-tx.c/shared-mem-rx.c

A diferença destes códigos está na forma com que a informação é compartilhada, um deles utiliza a fila para enviar e guardar as informações enquanto o outro utiliza uma área alocada em memória.

A forma de implementação por fila possui um mecanismo de buffer onde, caso o buffer se encha, ele não poderá ser sobrescrito enquanto não for esvaziado. Isso garante que nenhuma informação seja perdida. Enquanto que na implementação de memória compartilhada o cuidado para que a informação não seja perdida tem que ser integralmente do programador. Fazendo com que sejam necessários mecanismos para impedir que a informação se perca.

### Código shared-mem-tx-bonus.c e shared-mem-rx-bonus.c

Para corrigir o problema encontrado no código anterior foi proposto criar um semáforo que permite e impede a escrita na área compartilhada verificando se o lado **rx** do código leu o conteúdo.

Isso foi implementado alocando uma segunda área compartilhada entre os programas.

```
// Passos 1 a 3: abre/cria uma area de memoria compartilhada
fd = shm_open("/sharedmem", O_RDWR|O_CREAT, S_IRUSR|S_IWUSR);
if(fd == -1) {
    perror("shm_open");
    exit(1);
}

s_fd = shm_open("/sharedmem_s", O_RDWR|O_CREAT, S_IRUSR|S_IWUSR);
if(s_fd == -1) {
    perror("shm_s_open");
    exit(1);
}

// Passos 1 a 3: ajusta o tamanho da area compartilhada
if (ftruncate(fd, sizeof(value)) == -1) {
    perror("ftruncate");
    exit(1);
}

if (ftruncate(s_fd, sizeof(s_value)) == -1) {
    perror("ftruncate_s");
    exit(1);
}

// Passos 2 a 4: mapeia a area no espaco de enderecamento deste processo
ptr = mmap(NULL, sizeof(value), PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);
if(ptr == MAP_FAILED) {
    perror("mmap");
    exit(1);
}

s_ptr = mmap(NULL, sizeof(s_value), PROT_READ|PROT_WRITE, MAP_SHARED, s_fd, 0);
if(s_ptr == MAP_FAILED) {
    perror("mmap_s");
    exit(1);
}
```

Essa área funciona da seguinte maneira, quando o código **shared-mem-tx-bonus** quer escrever na área de mensagem ele verifica na área do semáforo se ele pode escrever. Caso possa ele escreve na área de mensagem a mensagem e escreve na área do semáforo informando que escreveu na área de mensagem.

Enquanto ele não for liberado pelo código **shared-mem-rx-bonus** ele não pode escrever um novo valor.

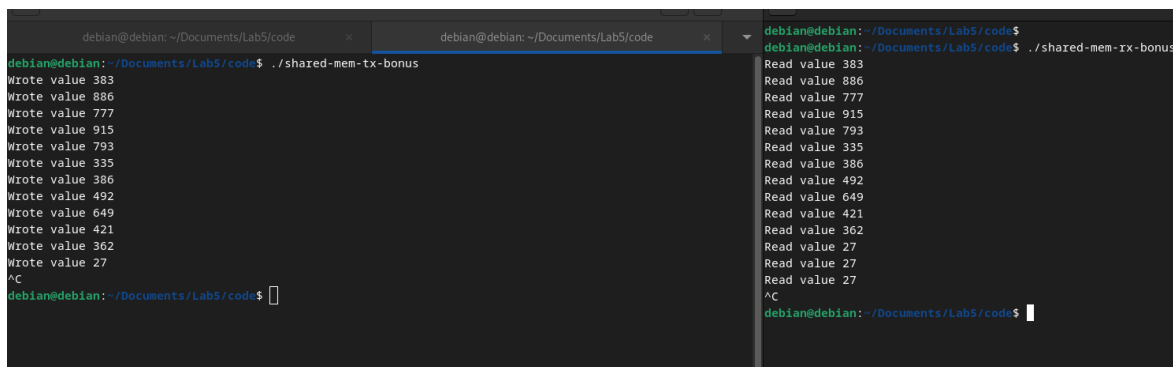
```
for (;;) {
    // Passo 5: escreve um valor aleatorio na area compartilhada se ja tiver sido lida
    if ((*s_ptr) == 1)
        sleep(1);
    else {
        value = random () % 1000 ;
        (*ptr) = value ;
        (*s_ptr) = 1;
        printf ("Wrote value %i\n", value) ;
        sleep (1);
    }
}
```

Caso a área do semáforo informe que está liberada a escrita o código **shared-mem-tx-bonus** escreve um novo valor e o ciclo recomeça.

Do lado do código **shared-mem-rx-bonus** funciona da seguinte maneira, quando o código faz a leitura do valor na área de mensagem ele libera para o **shared-mem-tx-bonus** escrever um novo valor.

```
for (;;) {
    // Passo 5: le e imprime o conteudo da area compartilhada. Libera a nova escrita
    value = (*ptr);
    (*s_ptr) = 0; //libera para escrita;
    printf("Read value %i\n", value);
    sleep(1);
}
```

Dessa maneira é garantido que o código **shared-mem-rx-bonus** não irá perder nenhuma informação enviada pelo código **shared-mem-tx-bonus**.



```
debian@debian: ~/Documents/Lab5/code$ ./shared-mem-tx-bonus
Wrote value 383
Wrote value 886
Wrote value 777
Wrote value 915
Wrote value 793
Wrote value 335
Wrote value 386
Wrote value 492
Wrote value 649
Wrote value 421
Wrote value 362
Wrote value 27
^C
debian@debian: ~/Documents/Lab5/code$

debian@debian: ~/Documents/Lab5/code$ ./shared-mem-rx-bonus
Read value 383
Read value 886
Read value 777
Read value 915
Read value 793
Read value 335
Read value 386
Read value 492
Read value 649
Read value 421
Read value 362
Read value 27
Read value 27
^C
debian@debian: ~/Documents/Lab5/code$
```

## VALIDAÇÃO DA TROCA DE INFORMAÇÕES

A forma de validação foi à mesma adotada no ultimo código, verificação se a informação enviada pelo **shared-mem-tx-bonus** foi recebida pelo **shared-mem-rx-bonus**.