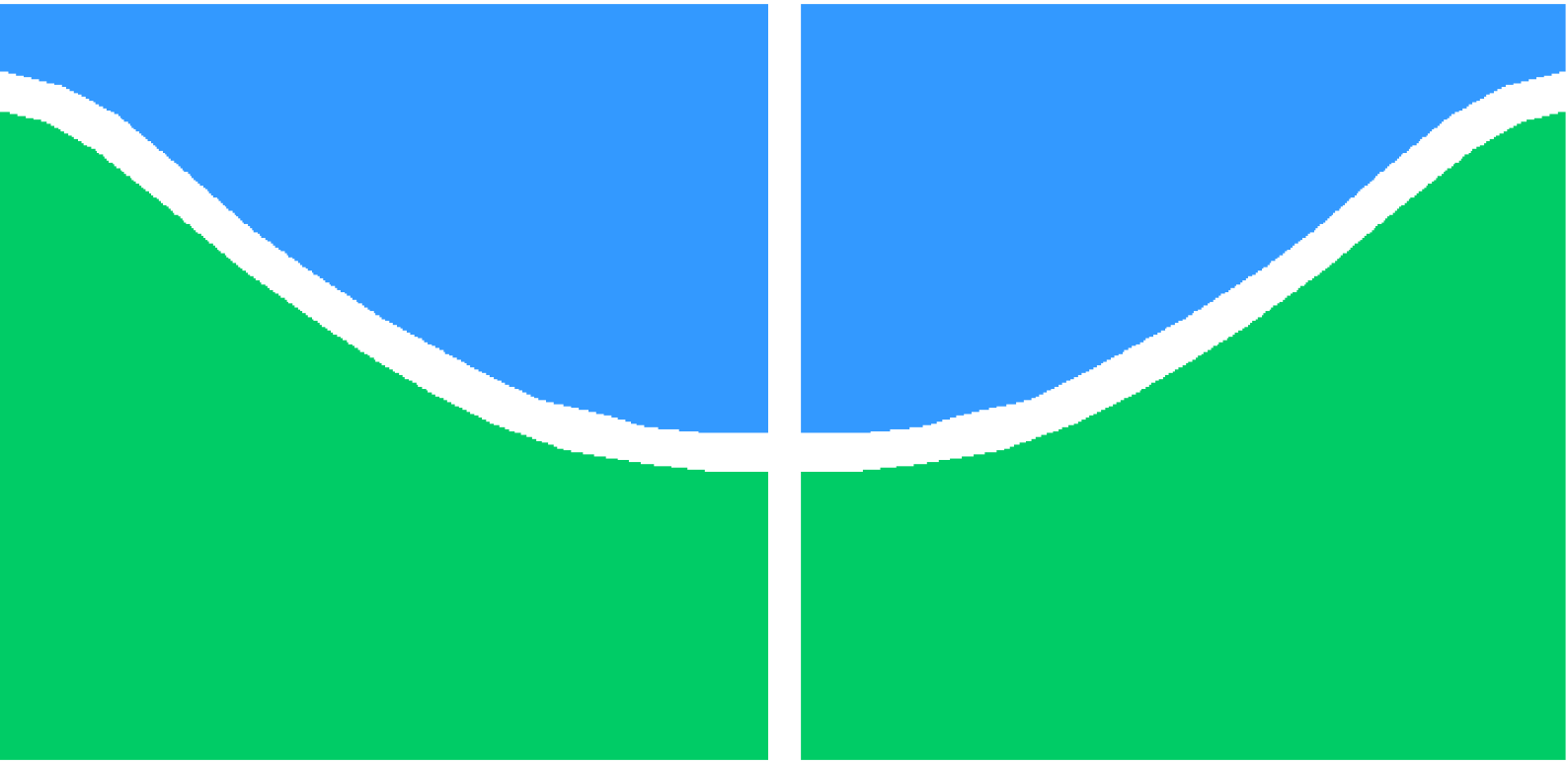

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Relatório de Sistemas Operacionais: Comunicação interprocessos via mecanismos IPC

Autor: Maxwell Almeida Santos

Brasília, DF

2016

Sumário

1 Objetivos

O trabalho tem como objetivo a comunicação entre processos independentes, usando assim mecanismos de filas de mensagens, sockets e memória compartilhada. No trabalho os processos comunicantes simularam um protocolo de rede que será composto por duas camadas, que são eles o host A e host B. A comunicação será realizada entre o pai A com o filho A onde, haverá a criação de uma fila de mensagem e esta será enviada para o filho. Também tem conversação entre o filho A com o filho B usando memória compartilhada e por fim a interlocução entre o filho B com o Pai B, onde o filho vai ler a mensagem que foi compartilhada e enviar para seu pai. Assim, será feito a comunicação entre os processos que foram criados.

1.1 Alternativas Implementadas

Todas as soluções abaixo foram completadas integralmente.

- Comunicação simplex (unidirecional) entre A e B (conforme desenho);
- Comunicação simplex (unidirecional) entre A e B substituindo a memória compartilhada por um socket (tcp ou udp);
- Comunicação duplex (em dois sentidos) entre A e B residentes em hosts distintos.

2 Diario de Atividades

Essa seção descreve em topicos as atividades efetuadas bem como suas respectivas datas.

O trabalho deu ínicio no dia 28 de março, onde os alunos fizeram pesquisas sobre filas de mensagens, no qual eles encontraram os respectivos tutorias que estão listados abaixo e replicaram os exemplos a fim de obter conhecimento sobre o assunto.

- http://www.ime.uerj.br/ alexszt/cursos/so1/troca%20de%20mensagens.pdf
- http://beej.us/guide/bgipc/output/html/singlepage/bgipc.html#mq

No dia 31 de março as comunicação entre o pai A com o filho A e o filho B com o Pai B ja haviam sido implementadas, contudo faltava a comunicação desses dois hosts através do filho a com o filho B pela memória compartilhada. Neste momento, a equipe tinha bastante dificuldade com relação a memória compartilhada, principalmente na recuperação da mensagem pela memória o que ocasionou um certo atraso na comunicação unidirecional. Também fora estudado o seguinte material para melhor esclarecimento sobre a implementação de memoria compartilhada:

- https://www.cs.cf.ac.uk/Dave/C/node27.html

No dia 8 iniciou-se o estudo de implementações voltadas para comunicação via tcp a partir dos seguintes materiais:

- http://www.theinsanetechie.in/2014/01/a-simple-chat-program-in-c-tcp.html
- http://www.programminglogic.com/example-of-client-server-program-in-c-using-sockets-and-tcp/
- http://www.binarytides.com/socket-programming-c-linux-tutorial/

No dia 9 de abril foram iniciado a implementação da comunicação via tcp em um dos hosts e, por fim, no dia 10 de abril fora concluído o trabalho com a comunicação duplex via tcp entre os hosts.

2.1 Problemas Encontrados

Uma das dificuldades da equipe foi em desenvolver uma solução que possibilitasse que o filho do processo 'A' só escrevesse na memória compartilhada e , consequentemente, só pegasse a nova mensagem na fila, quando a mesma já tivesse sido lida pelo filho do Processo B (a).

Na comunicação duplex teve-se dificuldade no envio e recebimento de mensagens na mesma fila de mensagens. O que acarretava em uma leitura equivocada por parte dos processos pai das mensagens que estavam sendo recebidas e enviadas (b).

2.2 Soluçoes Adotadas

Para o problema(a) uma abordagem inicial consistia na remoção da memoria compartilhada, de modo que o filho A utilizasse o código de erro para a criação de uma nova área de memória, e pudesse pegar a mensagem na fila fornecida pelo pai A. A segunda abordagem foi o uso de sinais para a comunicação entre os processo filho A e filho B , de modo que o filho a identificasse quando a leitura feita por B fosse terminada.Entretanto, nenhuma dessas abordagens a equipe obteve êxito, tendo-se muita dificuldade em implementa las. A abordagem que se obteve êxito consiste no uso de um código escrito na área da memória de modo que possibilite que o filho a reconhece quando se deve escrever na memória. Assim, após a leitura da mensagem enviada pelo filho a pela memória, o filho 'B' escreve um código (ex: 8800) na memória, o filho 'A' ao ler esse código identificar que pode ser feita escrita na memória , sendo , consequentemente , lida uma nova mensagem na fila disponibilizada pelo pai 'A'.

Para o problema(b) a solução encontrada para a comunicação duplex, foi a criação de duas filas de mensagens. Uma para envio de mensagens e outra para recebimento de mensagens por parte dos processos pai.

chapterDescrição de Parâmetros Este capitulo descreve os parametros utilizados nas funções relativas à IPC

2.3 Fila de Mensagens

Para criação de fila de mensagens é utilizado o seguinte método:

```
msgget(IPC_PRIVATE, MSG_PRM | IPC_CREAT | IPC_EXCL);
```

O método msgget() é reponsável pela criação de fila de mensagens caso elas não exista , caso existam ele irá conectar a fila de mensagem existente. Os parametros utilizados nesse caso serão descritos a seguir:

- IPC_PRIVATE - É um key type específico para criação de uma nova fila de mensagem;
- MSG_PRM - Definição da permissão da fila de mensagem, no código esse valor ficou definido como 600;
- IPC_CREATE e IPC_EXCL - quando utilizadas em conjunto se a fila de mensagem já existir a função retornará EEXIST.

Para a enviar mensagem na fila é utilizando o seguinte método:

```
msgsnd(qid, msg, sizeof(msg->text),0);
```

O método msgsnd é responsável em colocar uma mensagem na fila, caso esta já esteja criada. São utilizados os seguintes paramentros:

- qid - É o parâmetro que recebe como valor o id da fila de mensagem criada.
- msg - É a mensagem que será enviada na fila pelo pai.
- sizeof(msg->text) - É o tamanho que o texto da mensagem terá.

Para que a mensagem que foi enviada na fila seja lida, usa-se o seguinte método:

```
msgrcv(qid, msg, sizeof(msg->text),0,0);
```

O método msgrcv é responsável por receber e ler a mensagem que foi enviada para ele na fila de mensagem. Alguns parametros são usados e são eles:

- qid - É o parâmetro que recebe como valor o id da fila de mensagem recebida.
- msg - É a mensagem que será enviada na fila pelo pai.
- sizeof(msg->text) - É o tamanho que o texto da mensagem terá.

Para que haja um controle das operações que são realizadas na fila é usado o seguinte método:

```
msgctl(qid, IPC_RMID, NULL);
```

Esse método msgctl executa a operação de controle na fila de mensagem. Os parametros que são usados, são os seguintes:

- qid - É o parâmetro que recebe como valor o id da fila de mensagem enviadas e recebidas.

- IPC_RMID - É o parametro que marca o segmento a ser destruido. Tal segmento só é realmente destruido após o último processo que é desanexado.
- NULL - Retorna nulo para o método.

2.4 Memória Compartilhada

O Método utilizado para criação de memória compartilhada é utilizado msget():

```
shmget(key, SHM_LEN, IPC_CREAT | 0666);
```

A seguir uma descrição dos parâmetros utilizados:

- key - é o valor utilizado para acessar a memória;
- SHM_LEN - é o tamanho em bytes do tamanho da memória a ser criado. No nosso caso o valor de SHM_LEN é 250;
- IPC_CREATE | 0666 - Flag de criação e permissão de acesso;

O shmat() é utlizado para dá o attach no segmento de memória compartilhada. Retorna um ponteiro para o segmento de memória compartilhada.

```
shmat(shmid, NULL, 0);
```

- shmid - É id da memória criada
- NULL - Quando NULL deixa para o sistema a escolha do endereço de memória utilizada.
- No terceiro parametro indica se a mensagem será apenas lida para isso deveria ser passada a flag SHM_RDONLY , não é nosso caso então passa-se a o valor 0.

shmctl() é utlizado para alterar características do segmento de memória compartilhada.

```
shmctl(shmid, IPC_RMID, NULL);
```

- shmid = É o id da memórica criada
- IPC_RMID = É flag utilizada para deletar a memória compartilhada
- O terceiro argumento é um ponteiro para a estrutura shmid_ds , utilizada para obtenção ou alteração de características do segmento da memória. Neste caso o segmento de memória está sendo apagado, portanto não há necessidade de se passar essa estrutura.

2.5 Socket TCP

A seguir sao apresentados os metodos utilizados para comunicaçao tcp;

O método socket() é utilizado para a criação de socket. Neste trabalho este método é utilizado da seguinte maneira:

```
socket(AF_INET, SOCK_STREAM, 0);
```

- AF_INET - Protocolo de internet utilizado , a opção AF_INET indica que o protocolo utilizado será o IP. Uma outra opção seria utilizar o protocolo IPV6 passando o valor AF_INET6
- SOCK_STREAM - Indica o protocolo utilizado na camada de transporte , a opção SOCK_STREAM possbilita que seja utilizado o protocolo tcp. Caso a opção escolhida fosse SOCK_DGRAM o protocolo utilizado seria UDP.
- O terceiro argumento é utlizado para indicar o uso de mais protocolos. O que não é o caso deste trabalho portanto o valor configurado é 0.

O método bind() é utilizado para atribuir a porta e o endereço ao qual socket irá utilizar.

```
bind_port( int socket,struct sockaddr_in server );
```

- int socket - Identificar do socket criado pelo método socket();
- sockaddr_in server - Estrutura de dados utilizada para passar informações como endereço ip e porta a ser utilizada;

O método connect() é utilizado no lado client para conectar ao servidor.

```
connect(socket,(struct sockaddr *)&server, sizeof(server));
```

- socket - Valor do socket criado para estabelecimento da conexão;
- (struct sockaddr *)&server - Estrutura com informações referentes a porta e ip do servidor;
- sizeof(server) - Tamanho em bytes da variável server referente ao servidor.

3 Opnião Sobre o Trabalho

Este trabalho possibilitou o contato com mecanismos que até então eram desconhecidos para mim o que de certo despertou meu interesse. De maneira geral foi proveitoso o desenvolvimento do trabalho, apesar da dificuldade. O trabalho em equipe também foi tranquilo dado que todos se conheciam e ja mantinham contato. Entretanto, por mais que o trabalho tenha sido prazeroso de ser desenvolvido o tempo para conclusão foi demasiado longo o que prejudicou na dedicação de outras disciplinas. O nível do trabalho é ideal para que se possa fixar o conhecimento do funcionamento dos mecanismos apresentados, entretanto a conciliação do mesmo com outros trabalhos acaba por pesar negativamente com relação a linha do nivel ideal para este trabalho. Mais uma vez ressalto, foi proveitoso o desenvolvimento deste trabalho.