CENTRO DE EDUCAÇÃO SUPERIOR DE BRASÍLIA

CENTRO UNIVERSITÁRIO INSTITUTO DE EDUCAÇÃO SUPERIOR DE BRASÍLIA

Curso de Ciência da Computação - Projeto Integrador IV - Banco de Dados I

Mário Sérgio Oliveira de Queiroz

Valdelar Muniz Martins Sobrinho

Comunicação entre Processos no MINIX

BRASÍLIA, DF

15 de Maio de 2013

Mário Sérgio Oliveira de Queiroz

Valdelar Muniz Martins Sobrinho

Comunicação entre Processos no MINIX

Projeto da disciplina Projeto Integrador V – Sistemas Operacionais, do Centro Universitário Instituto de Educação Superior de Brasília, DF.

Orientador: Prof. Flávia Maria Alves Lopes

BRASÍLIA, DF

15 de Maio de 2013

Mário Sérgio Oliveira de Queiroz

Valdelar Muniz Martins Sobrinho

Comunicação entre Processos no MINIX

BANCA EXAMINADORA - APROVADO POR:

_____

Prof. xxxx(título e nome completo)xxxxxx
Centro Universitário Instituto de Educação Superior de Brasília, DF

_____

Prof. xxxx(título e nome completo)xxxxxx
Centro Universitário Instituto de Educação Superior de Brasília, DF

Brasília, DF, _____ de_____ de 2013.

Resumo

O trabalho proposto será voltado para a abordagem e implementação das trocas de mensagens feitas no contexto de comunicação entre processos no MINIX-3, direcionado para o desenvolvimento dos principais paradigmas deste módulo do sistema.

Para isso, utiliza-se a documentação oficial do MINIX baseada nos estudos explicitados no Projeto e Implementação, dos contribuidores Tanenebaum e Woodhull, como também o código fonte disponibilizado pela organização do MINIX-3.

Devido os argumentos apresentados, entende-se que será apresentado uma abordagem sobre como é feito o processo de comunicação entre processos no MINIX-3 de forma que apresentar-se-á o contexto teórico e implementação.

Palavras-chaves: MINIX-3, troca de mensagens, implementação, processos.

Lista de Figuras

Lista de Abreviaturas e Sigla

I/O – Input/Output (Entrada e Saida).

SENDREC – Send/Receive (envio e recebimento).

Sumário

1 Introdução

Este trabalho acadêmico se refere à pesquisa e implementação das principais formas de trocas de mensagens utilizadas pelo kernel do MINIX na comunicação entre os processos executados pelo sistema operacional.

1.1 Motivação

A principal motivação de se abordar a característica de comunicação entre processos no MINIX é buscar compreender a complexidade do núcleo de um sistema operacional, logo dessa forma pode-se integrar os conhecimentos do hardware e software.

1.2 Objetivos

Conforme os estímulos de motivação citados o trabalho deseja atingir os objetivos gerais e específicos, apresentados a seguir.

1.2.1 Geral

O projeto está direcionado a realizar a implementação das principais primitivas de troca de mensagens do MINIX, e apresentar todo o conteúdo teórico que sustenta o modelo de comunicação do sistema.

1.2.2 Específicos

Dentro do que foi estabelecido no objetivo geral deriva-se os seguintes objetivos específicos:

- Documentar todos os métodos de comunicação entre processos utilizados pelo MINIX;
- Implementar as principais funções de troca de mensagem, como *send*, *receive*, *sendrec* e *modify*;
- Contextualizar como é feita as funções que utilizam troca de mensagem no sistema operacional, como interrupção de hardware e chamadas de aplicações.

1.3 Organização do Trabalho

Este trabalho está organizado em 3 capítulos, da seguinte forma:

- Capítulo 1 - Definição dos objetivos e introdução do problema a ser resolvido – Evidenciando as motivações que levam a desenvolver o projeto.
- Capítulo 2 - Apresentação breve da estrutura do MINIX e apresentação das estruturas de comunicação interprocessuais.
- Capítulo 3 - Implementação das principais funções de troca de mensagens entre processos no MINIX-3.

2 Primitivas de Comunicação

A seguir, serão apresentadas as definições das regras da arquitetura para a implementação das funções de troca de mensagem no MINIX3.

2.1 Estrutura do MINIX

Diferente do UNIX (Linux), em que o kernel é monolítico, o MINIX 3 é baseado em uma estrutura do tipo micro-kernel. Esse design permite uma estrutura mais flexível e modular.

O MINIX 3 é estruturado em quatro camadas, sendo que, cada camada tem uma função bem definida. As camadas podem ser vistas na *figura 1*:

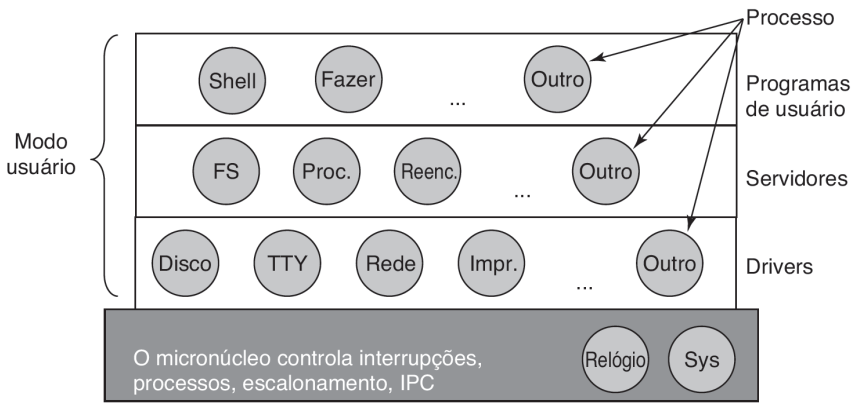

figura 1 – Modelo de camadas MINIX 3

2.2 Troca de Mensagens

Dentre as técnicas de comunicação entre processos fornecidas por modelos de sistemas operacionais, convém citar os seguintes métodos:

- Por memória compartilhada: através de diretivas shm pode-se disponibilizar uma área de

memória comum à vários processos. Simples: quando um processo quer enviar dados a outro processo, insere a notificação na memória.

- Pipes: um método muito interessante onde se cria um canal bidirecional na tabela de descritores, como se fosse arquivo, mas não é (e não tem I/O).

- Sinais: quando um processo quer sinalizar algo a outro processo. Ao contrário das demais nesta não se envia dados mas apenas um valor numérico.

Existem outras formas de comunicação como semáforos, fila de mensagens e até mesmo morte de filho, que é considerada uma comunicação.

A comunicação entre os processos no Minix 3 é feito por meio de mensagens. Sendo que, são feitas na mesma camada ou da camada superior com a inferior, que faz o escalonamento e fornece modelo de processos sequenciais independentes que se comunicam por mensagens, conforme ilustra a *figura 2*.

figura 2 – Exemplo de comunicação interprocessual

Quando um processo envia uma mensagem para um processo que não está esperando uma mensagem, o remetente é bloqueado até que o destino execute uma operação receive. Logo, o Minix 3 utiliza um método de somente enviar e receber mensagens, para que sejam evitados os problemas de armazenamento em *buffer*.

Neste contexto de troca de mensagens interprocessoais utilizadas no MINIX, existem algumas primitivas importantes, sendo elas apresentadas a seguir:

- O procedimento Send é utilizado para enviar mensagem a um processo de destino. *send (dest, &message)*
- O processo Receive é utilizado para receber uma mensagem de um processo fonte. *receive (source, &message)*
- O processo Sendrec é utilizado para enviar uma mensagem e esperar uma resposta do mesmo processo. *sendrec (src_dest, &message)*
- A primitiva Notify é usada quando um processo precisa informar outro processo que algo importante aconteceu.

3 Implementação

O modelo de implementação da comunicação entre processos no MINIX será feita na linguagem C, sem a utilização de qualquer função já existente que faça a abstração de procedimentos executados por um processo, como inicialização de um processo ou bloqueios.

Um processo será representado por uma estrutura que contém os campos id, status e entrada de mensagens, que representam respectivamente o identificador do processo, o status do processo para que seja verificado se o mesmo se encontra 'pronto', 'em execução' ou 'bloqueado', e a entrada de mensagens é necessária para que o processo possa verificar se ele tem alguma mensagem para receber de outro processo.

```
typedef struct Processo
{
        int id;
        int status;
        int entrada_ms;
}Processo;
```

Quando um processo executa uma função send, a camada inferior do núcleo verifica se o destino está esperando uma mensagem do remetente. Se estiver, a mensagem será coopiada do buffer do remetente para o buffer do destinatário.

```
#A função send recebe o id do remetente,  o destino e o status do destino, desta forma se cria duas estruturas de processos internamente para verificar se o processo destinatário está bloqueado, caso não esteja a função retorna 1.
int send(int origem,int destino, int status_d){
Processo processo;
Processo processo2;
processo.id = origem;
processo2.id = destino;
processo2.status = status_d;
        if (processo2.status == 0)
                return 1;
        else
                return 0;
}
```

Quando um processo executa uma função receive, núcleo verifica se algum processo está enfileirado, tentando enviar para ele. Logo, se isto estiver ocorrendo a mensagem será

recebida pelo destinatário. Caso nenhum outro processo tenha encaminha uma mensagem o destinatário será bloqueado.

```
#A função receive recebe o id do remetente,  o destino e a fila do destino, desta forma se cria duas estruturas de processos internamente para verificar se o processo destinatário com alguma mensagem na "fila", caso haja, a mensagem será recebida.
int receive(int origem, int destino, int fila){
   Processo processo;
   processo.id = destino;
   processo.fila = fila;
   if (processo.fila==origem)
      return 1;
   else
         return 0;
}
```

Na primitiva sendrec um processo envia uma mensagem e espera a resposta do destinatário. O mecanismo de passagem de mensagens no núcleo copia a mensagem do remetente no destinatário, sendo que a resposta sobrepõe à mensagem original.

```
int sendrec(int origem,int destino, int status_d){
Processo processo;
Processo processo2;
processo.id = origem;
processo2.id = destino;
processo2.status = status_d;
   printf("Enviando mensagem ao processo de destino\n");
   if(processo2.status == 0)
         return 1;
   else
         return 0;
}
```

A primitiva notify envia uma mensagem simples, sendo que nenhum processo é bloqueado neste procedimento de notificação.

4 Conclusão

Com base nas propostas e argumentos apresentados, nota-se que o MINIX-3 adotou um padrão de modelagem do sistema, que permite que a troca de mensagens entre processos fosse implementada respeitando a integridade de todo o sistema, além de organizar as operações de notificação nas camadas do sistema, desde o núcleo até o modo usuário.

Sendo assim, é relevante evidenciar a importância de se trabalhar em um projeto de sistemas operacionais, que aplique conhecimentos válidos para entender o funcionamento de todo o sistema, pois assim as chances de se obter um acréscimo de conhecimento são muito altas, tendo em vista a maturidade que é adquirida no estudo de sistemas operacionais.

5 Bibliografia

(TANENBAUM, WOODHULL 2008) TANENBAUM, Andrew S. Tanenbaum; WOODHULL Albert S. Woodhull. Sistemas Operacionais – Projeto e Implementação, 3ª Edição. Porto Alegre: Bookman, 2008.

(MINIX3 2013) MINIX3. Disponível em: <http://www.minix3.org/>. Acesso em 13 de Março de 2013.

(VIVA O LINUX 2013) Comunicação entre processos. Disponível em: <http://www.vivaolinux.com.br/artigo/Sinais-em-Linux/>. Acesso em 10 de Junho de 2013.

6 Apêndice

7 Anexos