Universidade de Brasília — Faculdade UnB Gama

Fundamentos de Sistemas Operacionais — Período: 2o./2017

Prof.: Fernando W Cruz — *Data entrega: 17/09/2017 (postado no Moodle)*

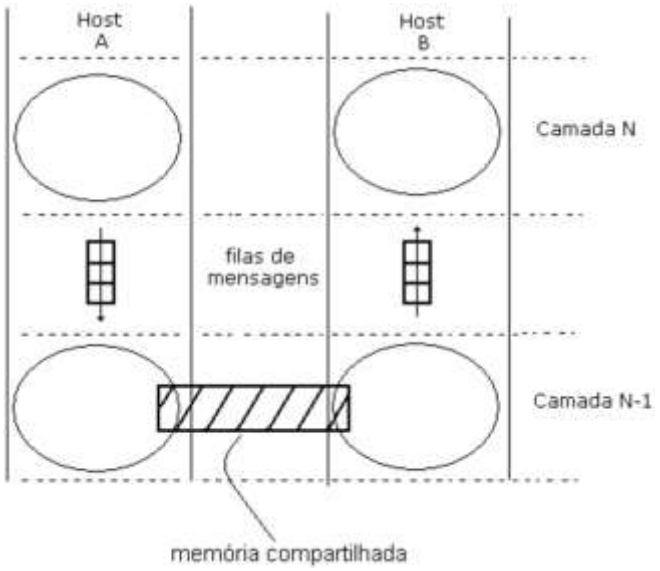
1º. Projeto de pesquisa

1. Título:

Comunicação interprocessos via mecanismos IPC (*Interprocess Communication*) de sistemas Unix/Linux.

2. Objetivos do projeto e descrição Geral

O objetivo desse trabalho é a implementação da comunicação entre processos independentes usando os mecanismos de filas de mensagens e memória compartilhada em sistemas Unix/Linux. Nesse caso, os processos comunicantes estão simulando um protocolo de rede composto por duas camadas, conforme descrito na figura abaixo.



Conforme pode ser visto na figura, existem 4 processos, sendo dois relativos à Camada N e dois relativos à Camada N-1. A comunicação entre esses processos ocorre da seguinte forma: *(i)* A Camada N-1 se comunica com a Camada N por filas de mensagens, e *(ii)* as entidades da camada N-1 comunicam-se entre si usando memória compartilhada. O objetivo do projeto é fazer com que o processo superior (camada N) do Host A leia dados do usuário (proveniente de teclado ou arquivo) e seja capaz de enviá-los para a entidade remota (processo superior no lado do Host B), utilizando os mecanismos de comunicação descritos na figura acima. Por sua vez, o destinatário (processo superior do Host B) deve imprimir as informações recebidas na tela.

Este projeto deve ser escrito em ambiente Linux, linguagem C e os processos devem ser disparados em janelas do Sistema Linux para simular a figura apresentada e facilitar o teste da aplicação. Sugere-se ainda colocar mensagens impressas em tela nos processos da camada N-1 para facilitar a depuração de repasse das mensagens de um lado para o outro.

3. Dicas/Requisitos

- Os processos de nível inferior devem ser criados pelos processos de nível superior através da *system call fork()*. Em outras palavras, o processo inferior do Host A é filho do processo superior do Host A. O mesmo ocorre no lado do Host B.

- Fica a critério dos alunos, a escolha de como será feita a criação das filas de mensagens e da área de memória compartilhada, assim como os direitos de leitura/escrita desses mecanismos. Uma sugestão é fazer com que esses mecanismos sejam criados pelos processos da Camada N.
- Os alunos devem definir um esquema para garantir que as informações lidas do teclado no lado A sejam colocadas em mais de uma posição na fila de mensagens.
- Uma nova mensagem só pode ser lida da fila depois que a anterior for enviada completamente pela memória compartilhada. Esse processo se repete até que a fila de envio se esvazie.
- Os alunos devem definir algum esquema para não haver sobreposição de dados na memória compartilhada. Ou seja, o remetente só pode escrever na área compartilhada depois que o destinatário já leu a informação anterior.
- Nos testes da aplicação, a camada superior (lado Host A) deve ler do teclado e escrever na fila até que o usuário não deseje mais (via uma *flag* de controle, por exemplo). Esse *flag* pode ser repassado para os demais processos para que eles também finalizem suas execuções.

4. Pontuação sobre a implementação do projeto

Neste projeto, o grupo de alunos receberá notas relativas à implementação de código, além da nota de relatório e apresentação oral. Para a parte de implementação, a pontuação está dividida conforme as regras abaixo:

Atividades do projeto	Pontos
1. Comunicação simplex (unidirecional) entre A e B (conforme desenho)	6,0
2. Comunicação simplex (unidirecional) entre A e B substituindo a memória compartilhada por um *socket* (tcp ou udp)	1,5
3. Comunicação duplex (em dois sentidos) entre A e B residentes em hosts distintos	2,5

Observações:

- O item 1 é pre-requisito para o item 2 que, por sua vez, é pré-requisito para o item 3.
- No item 3, sugere-se (não obrigatório) utilizar a *system call select()* ou similar para sincronizar os momentos de envio e recepção de dados

5. Requisitos e regras para o trabalho

Este trabalho deve ser feito obedecendo as seguintes regras:

- O trabalho pode ser feito por grupos de até 3 alunos.
- O grupo deve apresentar a aplicação funcionando no laboratório (e entregá-la em mídia digital)
- Cada aluno do grupo deve entregar um relatório INDIVIDUAL, contendo:
  - i) Os objetivos do trabalho (incluindo uma declaração sobre quais as alternativas de implementação citadas no tópico 4 que foram resolvidas e se foram resolvidas integralmente).
  - ii) Um diário de atividades, relatando o que foi resolvido ou estudado em que dia até chegar a versão final do trabalho. Nesse diário, relatar uma descrição resumida sobre os problemas encontrados e as soluções adotadas ao longo dos estudos.
  - iii) Uma descrição das funções utilizadas para garantir o sincronismo no diálogo entre as partes comunicantes.

iv) Uma descrição dos parâmetros usados nas funções relativas à *system call fork()*, e aos mecanismos de IPC (*Interprocess Communication Call*) usados no projeto: filas de mensagens, memória compartilhada e *sockets*. Ou seja, apresentar conhecimento das variações possíveis de uso desses mecanismos.

v) Uma análise do grau de falhas e limitação da solução apresentada e as melhorias que poderiam ser providas, se for o caso.

vi) Sua opinião pessoal sobre o projeto.

- A nota é individual e o valor máximo será dado ao aluno que demonstrar conhecimento e aprendizado satisfatório com o projeto. Outros requisitos como cumprimento das datas, trabalho coordenado em grupo e inserção de novas funcionalidades no projeto também serão consideradas para emissão da nota final.
- Datas de entrega: *(i)* relatório escrito e códigos – devem ser postados no Moodle até às 23h do dia 17/09/2017, *(ii)* apresentação oral – a ser feita no dia 19/09/2017. No dia, haverá sorteio para as apresentações. Alunos que não estiverem prontos, perderão a nota de apresentação. Tempo de apresentação: 20 minutos p/ grupo.

6. Referências

[Tanenbaum, 2003] Tanenbaum, A. Sistemas Operacionais Modernos. 2a. Ed (livro texto do curso) *(essa referência é importante para compreensão dos conceitos relativos à comunicação interprocessos.*

[Tanenbaum, 2004] Tanenbaum A. Computer Networks. 3a. e 4a. Ed. *(referência é útil caso o aluno queira conhecer melhor o conceito de protocolo de comunicação. Nesse caso, ver capítulo 1 desse livro).*

Procurar materiais na Internet e livros apropriados sobre a system call fork e mecanismos de comunicação interprocessos (ipcs, message queue e shared memory em ambiente Linux).