UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA APLICADA

INF01151 – SISTEMAS OPERACIONAIS II N
SEMESTRE 2015/1
TRABALHO 3: COMUNICAÇÃO INTER-PROCESSOS USANDO SOCKETS UNIX

---

1. ESPECIFICAÇÃO DO TRABALHO:

O trabalho consiste em implementar em C (não em C++) uma aplicação de chat usando a API de sockets Unix. O programa deverá executar obrigatoriamente em ambientes Unix (Linux) mesmo que o trabalho tenha sido desenvolvido sobre outras plataformas.

Uma descrição sobre comunicação entre processos utilizando sockets é apresentada em vários livros de sistemas operacionais e de programação concorrente. Em caso de dúvidas, é possível consultar os livros do Silberchatz (Operating System Concepts, 8th edition, seção 3.6) e Coulouris (Distributed Systems Concepts and Design, 5th edition, seção 4.2). Ainda, na Internet se encontram diversos tutoriais (alguns deles mencionados nas notas de aula) que descrevem em detalhes a API de sockets Unix.

2. SERVIDOR DE CHAT

Um servidor de chat deve permitir que até um certo número de aplicações cliente se conectarem ao servidor, e facilitar a troca de mensagens entre o grupo de clientes que estejam utilizando uma mesma sala do chat simultaneamente. O programa deve **obrigatoriamente** ser implementado utilizando a **API de sockets Unix**. O tipo de socket a ser utilizado deve **obrigatoriamente** ser **orientado a conexão (TCP)**, e o servidor deve **obrigatoriamente** ser um **servidor concorrente** (capaz de tratar simultaneamente requisições de vários clientes). O servidor deverá implementar o comportamento típico de um servidor de chat, ou seja, quando um cliente enviar uma mensagem de chat para uma sala específica do servidor, o servidor deverá reenviar essa mensagem para todos os outros clientes conectados àquela sala do chat.

Para implementar o servidor concorrente você deverá criar um novo processo ou thread para tratar cada nova conexão. Sugere-se portanto a **implementação com threads** para facilitar o compartilhamento de memória entre as diferentes unidades de execução (lembre-se que threads compartilham o espaço de endereçamento do processo que as criou). Os detalhes de como o servidor será implementado ficam a critério do aluno, inclusive a **especificação de qualquer estrutura de dados** para manter os dados dos clientes conectados ao chat. Lembre-se do que estudamos desde o início do semestre a respeito de sincronização: o acesso concorrente a dados compartilhados pode causar inconsistências, e você deverá **garantir sincronização entre as unidades de execução** através de primitivas como **mutex, semáforos, e variáveis de condição**, quando for necessário.

Além da funcionalidade básica descrita no primeiro parágrafo dessa seção, o programa **deverá incluir pelo menos as funcionalidades abaixo** (implementadas através de comandos específicos enviados do cliente para o servidor):

- Alteração de nickname do cliente;
- Criação de novas salas de chat;
- Permitir que um cliente entre ou saia de uma sala específica (\join e \leave);
- Fechar o chat (garantindo que as estruturas no servidor serão deixadas em estado consistente).

Note que você deverá definir alguma forma de delimitar esses comandos. Além disso, note que mensagens de chat são compostas de duas informações: “o que foi dito” e “quem disse”. Você deverá definir um protocolo simples de mensagens de tamanho fixo, ou alguma convenção através do uso de caracteres delimitadores especiais de modo que o servidor saiba quando mensagens enviadas por um cliente devem ser interpretadas como comandos ou chat, e também para que o servidor/cliente saiba quando as mensagens enviadas por cada cliente foram completamente transmitidas. DICA 1: ler discussão na seção 7.5 em Beej's Guide to Network Programming, Using Internet Sockets (leitura adicional passada nas notas de aula). DICA 2: lembre-se que em um socket orientado à conexão os limites das mensagens podem não ser preservados. Portanto, se um cliente escrever 20 bytes em um socket, não é garantido que o servidor irá ler 20 bytes em uma operação de leitura. O servidor poderá ler 20 bytes; mas o servidor também poderá ler menos, se houver alguma fragmentação do pacote.

Funcionalidades adicionais àquelas listadas acima irão fazer parte da nota final, e serão contabilizadas no quesito “Interface e Usabilidade”.

3. DESCRIÇÃO DO RELATÓRIO A SER ENTREGUE

Deverá ser produzido um relatório fornecendo os seguintes dados:

- Descrição do ambiente de teste: versão do sistema operacional e distribuição, configuração da máquina (processador(es) e memória) e compiladores utilizados (versões).
- Explique suas decisões e justificativas a respeito de:
  - (A) Como foi implementada concorrência no servidor;
  - (B) Quais estruturas de dados são mantidas pelo servidor com informações dos clientes;
  - (C) Em quais áreas do código foi necessário garantir sincronização no acesso a dados;
  - (D) Funcionalidades adicionais que você implementou.
- Também inclua no relatório problemas que você encontrou durante a implementação e como estes foram resolvidos (ou não).

Entregar dois programas C diferentes (cliente e servidor). A **nota será atribuída baseando-se nos seguintes critérios**: (1) qualidade do relatório produzido conforme os itens acima, (2) correta implementação das funcionalidades requisitadas, e (3) qualidade do programa de chat em si (incluindo uma interface limpa e amigável, documentação do código, funcionalidades adicionais implementadas, etc).

4. DATAS E MÉTODO DE AVALIAÇÃO

O trabalho pode ser feito em grupos de **2 OU 3 INTEGRANTES**. Não esquecer de identificar claramente os componentes do grupo no relatório.

Faz parte do "pacote" de entrega os fontes e o relatório em um arquivo ZIP. O trabalho deverá ser entregue **até às 08:25 horas do dia 26/05/2015 (turma A),** e **até às 08:25 horas do dia 25/05/2015 (turma B)**. A entrega deverá ser via moodle (link para submissão na Aula 12). As demonstrações ocorrerão no mesmo dia, no horário da aula.

Após a data de entrega o trabalho deverá ser entregue via e-mail para alberto@inf.ufrgs.br (subject do e-mail deve ser “INF01151: Trabalho 3”). Neste caso, será descontado 02 (dois) pontos por semana de atraso. O atraso máximo permitido é de duas semanas, isto é, nenhum trabalho será aceito após 09/06/2014 (Turma A), e 08/06/2015 (Turma B).