Universidade Federal de Minas Gerais
DCC023: Redes de Computadores
Trabalho Prático

Introdução

Neste trabalho iremos desenvolver um servidor que coleta presença de alunos em sala de aula, bem como o cliente para informar presença ao servidor. No texto abaixo, nomes de funções da biblioteca padrão relativas à tarefa sendo descrita são colocados entre colchetes para facilitar o desenvolvimento do trabalho.

Objetivos

- Obter familiaridade com a interface de programação de soquetes POSIX.
- Entender conceitos de aplicação cliente e aplicação servidor.
- Introduzir os conceitos de codificação e transmissão de dados.

Execução

- O trabalho deve ser desenvolvido em grupo de até dois alunos e vale 5 pontos.
- A data de entrega está disponível no Moodle e no Plano de Curso.

Especificação

Programa Servidor

O programa servidor tem duas funcionalidades:

1. Coletar a lista de alunos presentes em sala;
2. Exportar a lista de alunos presentes em sala.

Seu programa deve inicializar com duas senhas, uma para cada funcionalidade. Uma senha será informada aos alunos em sala de aula para que acessem o servidor e confirmem presença. A outra senha é privada para uso do professor.

Seu servidor deve esperar por conexões na porta 51511/TCP e esperar a conexão de clientes [socket, bind, listen, accept]. Ao receber uma conexão, o servidor deve enviar uma mensagem com cinco bytes, contendo os caracteres “READY” (codificados usando a tabela ASCII) [send]. O servidor deve então esperar por uma mensagem de oito bytes contendo uma das duas senhas acima [recv]. Em caso de erro (senha inválida ou algum erro de comunicação), o servidor deve simplesmente fechar a conexão. Ao receber uma senha válida, o servidor deve continuar operação de acordo com a senha recebida.

Caso a senha recebida seja a senha distribuída aos alunos para confirmação do número de presença, o servidor deve responder com uma mensagem de dois bytes contendo os caracteres “OK” e depois enviar uma mensagem de nove bytes contendo os caracteres “MATRICULA” (sem

acento) [send]. O servidor deve então esperar uma mensagem contendo o número de matrícula do aluno [recv]. O servidor deve receber dados referentes ao número de matrícula do aluno através rede [recv]. O número de matrícula deve ser transmitido como um inteiro de quatro bytes codificado em network byte order [htonl, ntohl]. Após receber o número de matrícula, o servidor deve enviar uma mensagem de dois bytes contendo os caracteres “OK” e fechar a conexão [send, close].

Caso a senha recebida seja a senha do professor, o servidor deve enviar uma lista com os números de matrícula recebidos até o momento (como descrito acima). O servidor deve enviar um número de matrícula por linha; isto é, cada número de matrícula deve ser seguido por uma quebra de linha [\n] [send]. Após transmitir o último número de matrícula (e a quebra de linha correspondente), o servidor deve enviar um byte com o caractere nulo [\0]. O servidor deve então esperar uma mensagem de dois bytes do cliente contendo os caracteres “OK” e em seguida deve fechar a conexão [recv, close].

Programas Clientes

Os programas clientes se conectam ao servidor na porta 51511/TCP [socket, bind, connect]. Após estabelecimento da conexão, os clientes esperam a mensagem de cinco bytes do servidor contendo os caracteres “READY” [recv]. Os programas cliente então enviam a senha em uma mensagem de oito bytes codificada em ASCII [send].

O restante do comportamento dos programas clientes depende da funcionalidade utilizada. O programa cliente para confirmação de presença, utilizado por alunos, deve então esperar duas mensagems do servidor de dois e nove bytes, respectivamente contendo os caracteres “OK” e “MATRICULA” [recv]. O cliente deve então enviar o número de matrícula ao servidor [send] codificado em network byte order [htonl, ntohl]. Por fim, o cliente deve esperar uma mensagem de dois bytes contendo os caracteres “OK” e fechar a conexão [recv, close].

O programa cliente para recebimento da lista de presença, utilizado pelo professor, deve receber números de matrícula da rede e imprimí-los na tela (um número de matrícula por linha) [recv]. O término da lista de presença é marcado por um caractere nulo [\0]. Após receber o caractere nulo, o cliente deve enviar uma mensagem de dois bytes contendo os caracteres “OK” e fechar a conexão [send, close].

Detecção de erros de comunicação

Ambos os programas, cliente e servidor, devem configurar um temporizador (*timeout*) para detectar falhas de comunicação ao chamar a função [recv]. Caso a mensagem esperada não seja recebida dentro do prazo configurado no temporizador, o programa deve imprimir a mensagem de erro “TIMEOUT” na tela e fechar a conexão. O cliente pode terminar prematuramente, mas *o servidor deve continuar funcionando normalmente independente de*

ocorrerem erros de comunicação. O temporizador deve ser configurado com o tempo de 1 segundo.

A configuração de temporizador é feita chamando-se a função [setsockopt]. No Linux, as opções disponíveis para uso na função [setsockopt] são descritas na seção [socket] do manual 7 (acesse usando [man 7 socket]). Procure por [SO_RCVTIMEO].

Detalhes de Implementação

- O servidor deverá escutar no endereço IP 127.0.0.1 (representando a máquina local em que o programa executa) e no porto 51511.
- Todos os caracteres devem ser transmitidos em letra maiúscula, sem acento e seguindo a codificação ASCII.
- As senhas dos alunos e professor devem ser geradas pelo servidor e impressas na tela.
- Os programas não devem imprimir mensagens de depuração na tela. Limite a saída do programa servidor às senhas dos alunos e professor e à mensagem de erro “TIMEOUT” em casos de erro de comunicação.
- O servidor deve executar indefinidamente, até ser terminado pelo usuário. Por exemplo, o servidor pode terminar de executar quando o usuário apertar uma tecla específica, apertar CTRL+C, ou enviar um sinal de terminação para o processo.

Entrega e Avaliação

Você deve entregar o código fonte de seus programas, com os nomes cliente-aluno.c, cliente-prof.c e servidor.c (os nomes precisam ser exatos devido ao sistema de correção automática). Você deve submeter também um arquivo makefile que pode ser utilizado para compilar seu código executando o comando "make all" no diretório. Seu código deve estar bem organizado e comentado. Seu programa será testado semi-automaticamente com a implementação de referência do professor. Para testar o correto funcionamento do seu servidor, teste seu programa com a implementação de colegas. Elas devem ser interoperáveis.

Você deve ainda entregar um PDF de uma página descrevendo:

- As diferenças semânticas entre os programas cliente-aluno.c e cliente-prof.c, em particular discuta a codificação e sinalização do final da mensagem.
- Qual estratégia utilizada para tratar múltiplas conexões pelo programa servidor, e para continuar funcionando em caso de erro de comunicação.