WHATSPROG – INTERCAMBIADOR DE MENSAGENS
PROFESSOR: ADELARDO ADELINO DANTAS DE MEDEIROS

DESCRIÇÃO

O objetivo é desenvolver em C++ um aplicativo cliente-servidor, denominado WhatsProg, capaz de trocar mensagens de texto entre usuários em máquinas diferentes. O servidor se conecta com todos os clientes dos usuários e encaminha as mensagens para os destinatários corretos. A aplicação utilizará threads e sockets TCP. A comunicação entre cliente e servidor deverá utilizar a porta 23456.

Todo usuário, antes de utilizar o WhatsProg, deverá se cadastrar no servidor com um login e senha. O login deve ser único e ter entre 6 e 12 caracteres. A senha também deve ter entre 6 e 12 caracteres. Tanto no login quanto na senha os caracteres maiúsculos e minúsculos são diferentes.

O destinatário deve ser um usuário previamente cadastrado: caso não exista, a mensagem é rejeitada. As mensagens destinadas a um usuário que esteja conectado devem ser transmitidas imediatamente. Caso o usuário esteja desconectado, a mensagem deve ser armazenada em um buffer de mensagens. Quando o usuário se conectar, todas as mensagens armazenadas devem ser transmitidas e eliminadas do buffer.

Quando uma mensagem é lida (ou seja, visualizada em tela pelo usuário), uma notificação é enviada do cliente do destinatário para o cliente do remente, via servidor. Também são enviadas, pelo servidor, notificações quando a mensagem é recebida e quando é transmitida para o destinatário. O programa cliente exibe para o usuário a situação das mensagens que foram enviadas por ele:

- `MSG_ENVIADA` (?): a mensagem foi enviada para o servidor.
- `MSG_RECEBIDA` (✓): o servidor confirmou a recepção.
- `MSG_ENTREGUE` (✓✓): a mensagem foi entregue (transmitida) ao destinatário pelo servidor.
- `MSG_LIDA` (✓✓): a mensagem foi lida (visualizada) pelo usuário no cliente destinatário.

As comunicações entre o cliente e o servidor, ou vice-versa, seguem um padrão:

1. Os primeiros 4 bytes sempre contêm um inteiro (`int32_t`) que indica o comando (ver lista de comandos a seguir)
2. Em seguida, seguem os parâmetros do comando, caso existam (cada comando tem um número diferente de parâmetros).
3. Os comandos (`int32_t`) e os parâmetros (`int32_t` ou `string`) são enviados utilizando o padrão da biblioteca MySocket.

COMANDOS

NOME	VALOR	PAR 1	PAR 2	PAR 3	SIGNIFICADO (*)
`CMD_NEW_USER`	1001	login: string	senha: string	-	Criação de um novo usuário (C→S)
`CMD_LOGIN_USER`	1002	login: string	senha: string	-	Conexão com um usuário já existente (C→S)
`CMD_LOGIN_OK`	1003	-	-	-	Conexão OK (S→C)
`CMD_LOGIN_INVALIDO`	1004	-	-	-	Conexão inválida (S→C)
`CMD_NOVA_MSG`	1005	Id: int32_t	usuário: string	texto: string	Nova mensagem (Cr→S e S→Cd)
`CMD_MSG_RECEBIDA`	1006	Id: int32_t	-	-	Mensagem recebida pelo servidor (S→Cr)
`CMD_MSG_ENTREGUE`	1007	Id:	-	-	Mensagem entregue ao des-

NOME	VALOR	PAR 1	PAR 2	PAR 3	SIGNIFICADO (*)
		int32_t			tinatário (S→Cr)
CMD_MSG_LIDA1	1008	Id: int32_t	usuário: string	-	Mensagem visualizada pelo destinatário (Cd→S)
CMD_MSG_LIDA2	1009	Id: int32_t	-	-	Mensagem visualizada pelo destinatário (S→Cr)
CMD_ID_INVALIDA	1010	Id: int32_t	-	-	ID da mensagem inválida (S→Cr)
CMD_USER_INVALIDO	1011	Id: int32_t	-	-	Usuário (remetente) da mensagem inválido (S→Cr)
CMD_MSG_INVALIDA	1012	Id: int32_t	-	-	Texto da mensagem inválido (S→Cr)
CMD_LOGOUT_USER	1013	-	-	-	Desconexão do usuário (C→S)

(*) Sentido de envio: S=Servidor, C=Cliente, Cr=Cliente remetente, Cd=Cliente destinatário

ESTADOS (STATUS) DAS MENSAGENS NOS BUFFERS

A – No cliente

ESTADO ANTERIOR	EVENTO	AÇÃO A SER REALIZADA	NOVO ESTADO
-	Mensagem digitada	Criar conversa, se necessário Inserir no buffer da conversa Enviar CMD_NOVA_MSG	MSG_ENVIADA
-	Recebe CMD_NOVA_MSG	Criar conversa, se necessário Inserir no buffer da conversa Exibir aviso de chegada (alteração no número de mensagens da conversa) Exibir mensagem, se conversa estiver visualizada	MSG_ENTREGUE
MSG_ENVIADA (remetente)	Recebe CMD_MSG_RECEBIDA	Exibir status (✓), se conversa estiver visualizada	MSG_RECEBIDA
	Recebe CMD_MSG_ENTREGUE	Exibir status (✓✓), se conversa estiver visualizada	MSG_ENTREGUE
	Recebe CMD_MSG_LIDA2	Exibir status (✓✓), se conversa estiver visualizada	MSG_LIDA
	Recebe CMD_ID_INVALIDA Recebe CMD_USER_INVALIDO Recebe CMD_MSG_INVALIDA	Remover do buffer Exibir mensagem de erro	–
MSG_RECEBIDA (remetente)	Recebe CMD_MSG_ENTREGUE	Exibir status (✓✓), se conversa estiver visualizada	MSG_ENTREGUE
	Recebe CMD_MSG_LIDA2	Exibir status (✓✓), se conversa estiver visualizada	MSG_LIDA
MSG_ENTREGUE (remetente)	Recebe CMD_MSG_LIDA2	Exibir status (✓✓), se conversa estiver visualizada	MSG_LIDA
MSG_ENTREGUE (destinatário)	Conversa visualizada	Enviar CMD_MSG_LIDA1	MSG_LIDA

B – No servidor

ESTADO ANTERIOR	EVENTO	AÇÃO A SER REALIZADA	NOVO ESTADO
-	Recebe `CMD_NOVA_MSG` do remetente	Inserir no buffer de mensagens pendentes Enviar `CMD_MSG_RECEBIDA` p/ remetente Se possível, enviar `CMD_NOVA_MSG` p/ destinatário	`MSG_RECEBIDA`
`MSG_RECEBIDA`	Envia `CMD_NOVA_MSG` p/ destinatário	Se possível, enviar `CMD_MSG_ENTREGUE` p/ remetente	`MSG_ENTREGUE`
`MSG_ENTREGUE`	Recebe `CMD_MSG_LIDA1`	Se possível, enviar `CMD_MSG_LIDA2` p/ remetente	`MSG_LIDA`
`MSG_LIDA`	Envia `CMD_MSG_LIDA2` p/ remetente	Remover do buffer	-

CASOS DE USO - CLIENTE[1]

C.1 – Cadastro de novo usuário ou conexão de usuário já existente

1) O usuário digita login e senha.
2) O cliente se conecta ao servidor e envia `CMD_NEW_USER` (novo usuário) ou `CMD_LOGIN_USER` (usuário existente), com parâmetros:
   - login: string
   - senha: string
3) Cliente lê (aguarda) comando do servidor, com tempo de espera máximo:
   a) Se for `CMD_LOGIN_OK`:
      i) Lê o arquivo com os dados de conexão anterior, caso exista.
      ii) Lança a thread de leitura do socket.
      iii) Reexibe toda a interface.
      iv) O cliente pode começar a utilizar a conexão para troca de mensagens.
   b) Se não for `CMD_LOGIN_OK` ou ocorrer timeout:
      i) O cliente deverá fazer nova conexão.

C.2 – Envio de mensagem (do cliente remetente para o servidor)

1) O usuário escolhe uma conversa (ou cria uma nova conversa, caso ainda não exista uma conversa associada ao destinatário) e digita o texto de uma mensagem.
2) O cliente remetente armazena a mensagem no buffer da conversa associada ao destinatário, com dados:
   - id única da mensagem: int32_t
   - remetente (o próprio usuário) e destinatário (o usuário da conversa)
   - texto da mensagem: string
   - status `MSG_ENVIADA`.
3) O cliente remetente envia `CMD_NOVA_MSG` para o servidor com parâmetros:
   - id única da mensagem: int32_t
   - nome do usuário (destinatário): string

[1] Observação: nos casos de uso estão omitidos os testes e as operações relacionados a erros na comunicação (erro de leitura/escrita no socket, etc.), que normalmente geram o fechamento da conexão.

- texto da mensagem: string

4) Testa se envio deu certo:
   a) Se deu certo, faz a conversa passar a ser a primeira no buffer.
   b) Se deu errado, remove mensagem do buffer.

C.3 – Recepção de comando pela thread de leitura do cliente

1) Se for `CMD_NEW_USER`:
   Se for `CMD_LOGIN_USER`:
   Se for `CMD_LOGIN_OK`:
   Se for `CMD_LOGIN_INVALIDO`:
   Se for `CMD_MSG_LIDA1`:
   Se for `CMD_LOGOUT_USER`:
   Se for um comando desconhecido:
   a) Ignorar (erro do servidor).
2) Se for `CMD_NOVA_MSG`:
   a) Lê id, remetente e texto do socket.
   b) Se não houver uma conversa associada com o remetente recebido como parâmetro:
      i) Cria nova conversa.
      ii) Exibe a lista de conversas atualizadas, se for o caso.
   c) Insere nova mensagem no buffer da conversa associada ao remetente, com dados:
      - id
      - remetente (o usuário da conversa) e destinatário (o próprio usuário)
      - texto
      - status `MSG_ENTREGUE`.
   d) Faz a conversa passar a ser a primeira no buffer
   e) Incrementa e exibe o número de mensagens da conversa (aviso ao usuário que chegou nova mensagem).
   f) Caso a conversa esteja sendo visualizada, executa o procedimento previsto para visualização de uma conversa pelo usuário (caso de uso C.4).
3) Se for `CMD_MSG_RECEBIDA`:
   a) Cliente lê id do socket.
   b) Testa se existe no buffer de alguma das conversas uma mensagem que:
      - tenha Id igual à que foi recebida como parâmetro; e
      - esteja armazenada no buffer tendo o usuário como remetente:
      i) Se existir:
         (1) Caso tenha status `MSG_ENVIADA`:
            (a) Muda o status da mensagem para `MSG_RECEBIDA`.
            (b) Exibe para o usuário o novo status da mensagem, caso a conversa esteja sendo visualizada.
         (2) Caso tenha outro status:
            (a) Ignorar (erro do servidor ou do cliente).
      ii) Caso não exista:
         (1) Ignorar (erro do servidor ou do cliente)
4) Se for `CMD_MSG_ENTREGUE`:
   a) Cliente lê id do socket.
   b) Testa se existe no buffer de alguma das conversas uma mensagem que:
      - tenha Id igual à que foi recebida como parâmetro; e
      - esteja armazenada no buffer tendo o usuário como remetente:
      i) Se existir:
         (1) Caso tenha status `MSG_ENVIADA` ou `MSG_RECEBIDA`:

(a) Muda o status da mensagem para `MSG_ENTREGUE`.
(b) Exibe para o usuário o novo status da mensagem, caso a conversa esteja sendo visualizada.
(2) Caso tenha outro status:
(a) Ignorar (erro do servidor ou do cliente).
ii) Caso não exista:
(1) Ignorar (erro do servidor ou do cliente)

5) Se for `CMD_MSG_LIDA2`:
a) Cliente lê id do socket.
b) Testa se existe no buffer de alguma das conversas uma mensagem que:
- tenha Id igual à que foi recebida como parâmetro; e
- esteja armazenada no buffer tendo o usuário como remetente:

i) Se existir:
(1) Caso tenha status `MSG_ENVIADA` ou `MSG_RECEBIDA` ou `MSG_ENTREGUE`:
(a) Muda o status da mensagem para `MSG_LIDA`.
(b) Exibe para o usuário o novo status da mensagem, caso a conversa esteja sendo visualizada.
(2) Caso tenha outro status:
(a) Ignorar (erro do servidor ou do cliente).
ii) Caso não exista:
(1) Ignorar (erro do servidor ou do cliente)

6) Se for `CMD_ID_INVALIDA`, `CMD_USER_INVALIDO` ou `CMD_MSG_INVALIDA`:
a) Cliente lê id do socket.
b) Testa se existe no buffer de alguma das conversas uma mensagem que:
- tenha Id igual à que foi recebida como parâmetro;
- esteja armazenada no buffer tendo eu como remetente; e
- tenha status `MSG_ENVIADA`.

i) Se existir:
(1) Remove a mensagem do buffer da conversa.
(2) Atualiza a exibição do número de mensagens da conversa e, caso a conversa esteja sendo visualizada, das mensagens da conversa.
(3) Exibe para o usuário a mensagem de erro apropriada.
ii) Caso não exista:
(1) Ignorar (erro do servidor ou do cliente)

7) Se for timeout:
a) Aproveita o período de inatividade para salvar os dados do cliente (caso de uso C.5)

C.4 – Visualização de conversa pelo usuário

1) Quando uma conversa for visualizada (exibida em tela), o cliente procura todas as mensagens daquela conversa com as seguintes características:
- O remetente é o usuário da conversa, e não o próprio usuário.
- O status é status `MSG_ENTREGUE`.

2) Para cada mensagem encontrada:
a) Altera o status para `MSG_LIDA`.
b) Envia `CMD_MSG_LIDA1` para o servidor, com parâmetros:
- Id da mensagem: int32_t
- Nome do usuário (remetente): string

3) Atualiza a exibição do número de mensagens da conversa e, caso a conversa esteja sendo visualizada, das mensagens da conversa.

C.5 – Salvamento dos dados [OPCIONAL]

1) O programa cliente, ao se encerrar ou em caso de inatividade, salva as informações em arquivo para que, ao ser lançado novamente, esteja no mesmo estado anterior. Devem ser salvos:
   - Servidor (ip)
   - Usuário (login)
   - Última id enviada
   - Número de conversas
   - Para todas as conversas:
     - Correspondente (login)
     - Número de mensagens
     - Para todas as mensagens:
       - id
       - status
       - remetente
       - destinatário
       - texto

CASOS DE USO - SERVIDOR[2]

S.1 – Espera por atividade no servidor

1) Forma uma fila de sockets com:
   - Socket de conexões
   - Todos os sockets de clientes conectados
2) Espera por dados disponíveis em algum socket da fila.
3) Se chegou dado em algum socket de cliente:
   a) Lê o comando recebido (caso de uso S.2)
4) Se chegou nova conexão:
   a) Lê o novo cliente (caso de uso S.3)
5) Se for timeout:
   a) Aproveita o período de inatividade para salvar os dados do cliente (caso de uso S.4)

S.2 – Recepção de comando pelo servidor

1) Se for `CMD_NEW_USER`:
   Se for `CMD_LOGIN_USER`:
   Se for `CMD_LOGIN_OK`:
   Se for `CMD_LOGIN_INVALIDO`:
   Se for `CMD_MSG_RECEBIDA`:
   Se for `CMD_MSG_ENTREGUE`:
   Se for `CMD_MSG_LIDA2`:
   Se for `CMD_ID_INVALIDA`:
   Se for `CMD_USER_INVALIDO`:
   Se for `CMD_MSG_INVALIDA`:
   Se for um comando desconhecido:
   a) Fecha a conexão (erro do cliente).
2) Se for `CMD_NOVA_MSG`:

[2] Observação: nos casos de uso estão omitidos os testes e as operações relacionados a erros na comunicação (erro de leitura/escrita no socket, etc.), que normalmente geram o fechamento da conexão.

a) Lê id, destinatário e texto do socket do cliente correspondente.
b) Cria mensagem com dados:
   - id
   - remetente (login do cliente correspondente ao socket) e destinatário (recebido)
   - texto
   - status `MSG_RECEBIDA`.
c) Testa se:
   - a id é válida (a última id anterior desse cliente é menor que id atual);
   - o destinatário é válido (tamanho correto) e está cadastrado; e
   - o texto é válido (tamanho correto)
   i) Se válido:
      (1) Armazena a mensagem no buffer.
      (2) Atualiza a última id recebida desse cliente.
      (3) Envia `CMD_MSG_RECEBIDA` para o remetente com parâmetro id;
      (4) Se o destinatário estiver conectado:
         (a) Envia `CMD_NOVA_MSG` para o destinatário com parâmetros:
            - Id da mensagem que foi atribuída pelo remetente: int32_t
            - Nome do usuário (remetente): string
            - Texto da mensagem: string
         (b) Muda o status da mensagem para `MSG_ENTREGUE`.
         (c) Envia `CMD_MSG_ENTREGUE` para o remetente com parâmetro id.
   ii) Se não for válido:
      (1) Envia `CMD_ID_INVALIDA`, `CMD_USER_INVALIDO` ou `CMD_MSG_INVALIDA` para o remetente, conforme o tipo de erro.

3) Se for `CMD_MSG_LIDA1`:
   a) Lê id e remetente do socket do cliente correspondente.
   b) Testa se:
      - existe no buffer mensagem com aquela Id; e
      - o remetente dessa mensagem é o usuário enviado como parâmetro.
      i) Caso exista:
         (1) Altera o status para `MSG_LIDA`.
         (2) Se o remetente estiver conectado:
            (a) Envia a confirmação de leitura `CMD_MSG_LIDA2` para o remetente com parâmetro id.
            (b) Remove a mensagem do buffer de mensagens.
      ii) Caso não exista:
         (1) Fecha a conexão (erro do cliente).
4) Se for `CMD_LOGOUT_USER`:
   a) Fecha a conexão.

S.3 – Pedido de nova conexão no servidor

1) Aceita a conexão com um socket temporário.
2) Lê o comando.
3) Se o comando não for `CMD_NEW_USER` nem `CMD_LOGIN_USER`:
   a) Fecha o socket temporário e encerra esse pedido de conexão.
4) Lê login e senha.
5) Se o login e/ou a senha não têm o tamanho apropriado:
   a) Envia `CMD_LOGIN_INVALIDO`.
   b) Fecha o socket temporário e encerra esse pedido de conexão.
6) Se for `CMD_NEW_USER`:

a) Se já existir usuário cadastrado com esse login na lista de usuários:
   i) Envia `CMD_LOGIN_INVALIDO`.
   ii) Fecha o socket temporário e encerra esse pedido de conexão.
b) Cria novo usuário, com o login e a senha recebidos e associado ao socket t.
c) Cadastra o usuário na lista de usuários.
d) Envia `CMD_LOGIN_OK`.

7) Se for `CMD_LOGIN_USER`:
   a) Se não existir usuário cadastrado com esse login na lista de usuários; ou
   se a senha não conferir; ou
   se o usuário já estiver conectado (associado a outro socket):
      i) Envia `CMD_LOGIN_INVALIDO`.
      ii) Fecha o socket temporário e encerra esse pedido de conexão.
   b) Associa o usuário com o socket t.
   c) Envia `CMD_LOGIN_OK`.
   d) Testa se existem mensagens armazenadas que:
      - sejam destinadas ao usuário; e
      - tenham status `MSG_RECEBIDA`.

      Caso existam, para cada mensagem (similar ao caso S2.2.c.i.4):
      i) Envia `CMD_NOVA_MSG` para o destinatário com parâmetros:
         - Id da mensagem que foi atribuída pelo remetente: int32_t
         - Nome do usuário (remetente): string
         - Texto da mensagem: string
      ii) Muda status da mensagem para `MSG_ENTREGUE`.
      iii) Caso o remetente esteja conectado, envia `CMD_MSG_ENTREGUE` para o remetente com parâmetro id.
   e) Testa se existem mensagens armazenadas que:
      - foram remetidas pelo usuário; e
      - tenham status `MSG_LIDA`.

      Caso existam, para cada mensagem (similar ao caso S2.3.b.i.2)::
      i) Envia a confirmação de leitura `CMD_MSG_LIDA2` para o cliente recém-conectado com parâmetro id.
      ii) Remove a mensagem do buffer de mensagens.

S.4 – Salvamento dos dados [OPCIONAL]

1) O programa servidor, ao se encerrar ou em caso de inatividade, salva as informações em arquivo para que, ao ser lançado novamente, esteja no mesmo estado anterior. Devem ser salvos:
   - Número de usuários cadastrados
   - Para todos os usuários:
     - login
     - senha
     - última id recebida
   - Número de mensagens cujo processamento ainda não foi concluído (ou seja, estão no buffer)
   - Para todas as mensagens:
     - id
     - status
     - remetente
     - destinatário
     - texto