MAC0216 – Técnicas de Programação I – 2s2020
Projeto de Software - Bash Script
Data de entrega: 4/11/2020 até 8:00 da manhã

Universidade de São Paulo

1 Problema

A tarefa neste Projeto de Software é implementar, individualmente ou em dupla, um sistema de chat (cliente e servidor) em bash script, em modo texto, para permitir a comunicação local de usuários logados no mesmo computador. Todos os códigos devem ser escritos em bash script para serem executados no GNU/Linux. O vídeo em `https://youtu.be/UHBBcxs2Zd4` mostra como deve ser o comportamento do sistema.

O sistema estará implementado em um único arquivo .sh, por exemplo, `mac0216-chat.sh`. Para executar o servidor é necessário invocá-lo no shell da seguinte forma:

```
./mac0216-chat.sh servidor
```

Basta 1 invocação do servidor.

Para executar o cliente é necessário invocá-lo no shell da seguinte forma:

```
./mac0216-chat.sh cliente
```

Cada usuário que deseje usar o sistema de chat deverá invocar o script em modo cliente da forma acima. Nos exemplos acima considera-se que o script já está com permissão de execução e que ele está localizado no diretório local.

1.1 O servidor de chat

O servidor de chat deve ser iniciado antes de qualquer cliente ser iniciado. Uma vez inicializado, o servidor deve exibir na tela um prompt de comandos aguardando os comandos do usuário:

```
servidor>
```

O servidor deve suportar os seguintes comandos:

- `list`: lista os nomes de todos os usuários logados, um por linha. Se não tiver nenhum usuário logado, não precisa listar nada na tela;
- `time`: informa o intervalo de tempo, em segundos, desde que o servidor foi iniciado;
- `reset`: remove todos os usuários que foram criados nessa instância de execução atual do servidor;

- `quit`: finaliza o servidor.

Não se preocupe em tratar a execução dos comandos `reset` e `quit` se ainda houver algum cliente conectado. Considere que esses comandos só serão executados se não houver clientes conectados. Também não se preocupe em testar o sistema caso o servidor execute e já haja algum cliente ou servidor rodando. Considere que isso nunca vai acontecer.

1.2 O cliente de chat

Os clientes de chat devem ser iniciados após o servidor ser iniciado. Uma vez inicializado, o cliente deve exibir na tela um prompt de comandos aguardando os comandos do usuário:

```
cliente>
```

O cliente deve suportar os seguintes comandos sem necessidade do usuário estar logado:

- `create usuario senha`: cria um novo usuário de nome `usuario` e de senha `senha`. Ambas informações devem ser salvas no servidor e precisam ser mantidas enquanto o servidor estiver em execução. Só podem ser perdidas quando o servidor executar um `reset` ou um `quit`. Se o usuário `usuario` já existir, a string `ERRO` deve ser impressa;

- `passwd usuario antiga nova`: modifica a senha do usuário `usuario` de `antiga` para `nova`. A nova senha deve ser atualizada no servidor. Se o usuário não existir ou se a senha antiga estiver errada, a string `ERRO` deve ser impressa;

- `login usuario senha`: loga como o usuário de nome `usuario` com a senha `senha`. Se o usuário não existir, se a senha estiver errada ou se o usuário já estiver logado no sistema, a string `ERRO` deve ser impressa.

- `quit`: encerra a execução do cliente. Caso o usuário não tenha feito `logout`, faz `logout` antes de encerrar;

O cliente deve suportar os seguintes comandos após o usuário logar:

- `list`: lista os nomes de todos os usuários logados, um por linha, inclusive o próprio usuário;

- `logout`: desloga do sistema mas não encerra a execução do cliente;

- `msg usuario mensagem`: Escreve na tela do usuário `usuario` a mensagem `mensagem`. Se o usuário `usuario` não estiver logado, a string `ERRO` deve ser impressa. Na tela do `usuario` a mensagem deve ser impressa imediatamente com o prefixo `[Mensagem do remetente]:` , onde `remetente` deve ser o nome do remetente da mensagem;

Para todos os três comandos acima, se o usuário não estiver logado, deve ser impressa a string `ERRO`.

Para todos os comandos considere que a `mensagem` é o único parâmetro que poderá ter espaços em branco. Nenhum parâmetro terá tabulações ou quebras de linha.

2 Requisitos

O código deve ser escrito como **um único arquivo** com extensão .sh e as primeiras linhas do arquivo devem ser obrigatoriamente:

```
#!/bin/bash
#  AO PREENCHER(MOS) ESSE CABEÇALHO COM O(S) MEU(NOSSOS) NOME(S) E
#  O(S) MEU(NOSSOS) NÚMERO(S) USP, DECLARO(AMOS) QUE SOU(MOS) O(S)
#  ÚNICO(S) AUTOR(ES) E RESPONSÁVEL(IS) POR ESSE PROGRAMA. TODAS AS
#  PARTES ORIGINAIS DESSE EXERCÍCIO PROGRAMA (EP) FORAM DESENVOLVIDAS
#  E IMPLEMENTADAS POR MIM(NÓS) SEGUINDO AS INSTRUÇÕES DESSE EP E QUE
#  PORTANTO NÃO CONSTITUEM DESONESTIDADE ACADÊMICA OU PLÁGIO. DECLARO
#  TAMBÉM QUE SOU(MOS) RESPONSÁVEL(IS) POR TODAS AS CÓPIAS DESSE
#  PROGRAMA E QUE EU(NÓS) NÃO DISTRIBUÍ(MOS) OU FACILITEI(AMOS) A SUA
#  DISTRIBUIÇÃO. ESTOU(AMOS) CIENTE(S) QUE OS CASOS DE PLÁGIO E
#  DESONESTIDADE ACADÊMICA SERÃO TRATADOS SEGUNDO OS CRITÉRIOS
#  DIVULGADOS NA PÁGINA DA DISCIPLINA. ENTENDO(EMOS) QUE EPS SEM
#  ASSINATURA NÃO SERÃO CORRIGIDOS E, AINDA ASSIM, PODERÃO SER PUNIDOS
#  POR DESONESTIDADE ACADÊMICA.
#
#  Nome(s) :
#  NUSP(s) :
#
#  Referências: Com exceção das rotinas fornecidas no enunciado e em
#  sala de aula, caso você(s) tenha(m) utilizado alguma referência,
#  liste(m) abaixo para que o programa não seja considerado plágio ou
#  irregular.
#
#  Exemplo:
#  - O algoritmo Quicksort foi baseado em
#  http://www.ime.usp.br/~pf/algoritmos/aulas/quick.html
```

Os campos de Nome e NUSP devem ser preenchidos e, se for o caso, as referências devem ser informadas.

Todo o código deve ser escrito em bash script para funcionar no shell sem qualquer interação com o usuário em modo gráfico. **Não é permitido utilizar comandos ou programas que já implementem, mesmo que parcialmente, um sistema de chat no shell. Programas que não respeitem esse requisito terão nota ZERO.**

Caso seja necessário criar arquivos ou diretórios temporários para garantir a execução correta do sistema, esses arquivos devem ser criados no diretório `/tmp/`. Certifique-se de remover tudo que tiver sido criado no sistema de arquivos quando os clientes e o servidor forem encerrados.

3 Sobre a entrega

Deve ser entregue um arquivo .sh em:

`https://edisciplinas.usp.br/mod/vpl/view.php?id=3175316`

até as 8:00 da manhã do dia 4 de Novembro de 2020. No caso de duplas, **apenas 1 entrega deve ser realizada pelo usuário de um dos integrantes da dupla. Caso haja duas entregas de uma mesma dupla, apenas a entrega mais recente será corrigida. Se as duas entregas tiverem sido feitas exatamente no mesmo horário, uma das duas, aleatoriamente, será escolhida para correção**.