Cadastro Seguro de Usuários
SCC0201 - Trabalho 6

Prof. Moacir A. Ponti
Monitores: João Vitor, Giuliano; PAEs: Leonardo, Edesio

19 de novembro de 2018

1 Resumo

Implemente um programa que gerencie um conjunto de usuários e senhas usando tabela hash.

2 Background

Joir e Fernando, preocupados com a atual conjuntura politica e a forma com a qual noticias falsas estavam circulando, decidiram criar sua própria rede para troca de mensagens. Entretanto, eles não sabiam exatamente como desenvolver esse sistema, então resolveram procurar os alunos de ICC2 para ajudá-los.

Em seu projeto inicial, eles queriam que um grupo seleto de pessoas usassem sua rede social, por esse motivo o sistema deveria permitir apenas o acesso de usuário cadastrados. Segundo o documento de especificações apresentado, o sistema deverá ter alguns requisitos peculiares, como que os usuários e as senhas devem ser colocados em uma **tabela Hash** e que ao final, esse projeto fosse adicionado ao sítio run.codes.

3 Especificações

Implemente um programa em linguagem `C` que receba uma série de instruções e as execute.

3.1 Instruções

As instruções são formadas por um código de identificação e uma série de argumentos. As instruções possíveis são `cadastrar`, `autenticar` e `exibir`.

3.1.1 Cadastrar

A instrução `cadastrar` tem o seguinte formato:

```
cad userName password
```

Onde `cad` é o código de identificação da operação `cadastrar`, `userName` é o nome do usuário a ser cadastrado e `password` é a senha a ser cadastrada. Esses dados devem ser colocados em uma tabela Hash. Todos os campos são separados por um espaço e os nomes de usuário devem ser únicos, ou seja, o sistema não deve permitir o cadastro de dois usuários com o mesmo nome.

Para aumentar a segurança do sistema e evitar que algum usuário mal intencionado obtenha a tabela e com isso as senha de todos os usuários, a senha deve ser criptografada e então armazenada. Para isso, a função de dispersão MD5[1] deve ser usada.

Ao final da operação, se tudo der certo a mensagem `Cadastro realizado com sucesso` deve ser impressa. Caso o `userName` já tenha sido cadastrado, a mensagem `Nome de usuario invalido`.

3.1.2 Autenticar

A instrução `autenticar` tem o seguinte formato:

```
aut userName password
```

Onde `aut` é o código de identificação da operção `autenticar`, `userName` é o nome do usuário que deseja acessar o sistema e `password` é a senha desse usuário. Análogo ao processo de cadastro, o nome de usuário e a senha devem ser encontrados na tabela Hash. Todos os campos são separados por um espaço.

Em seguida, a senha deve ser verificada, criptografando a senha digitada e verificando se ela corresponde a senha armazenada. Caso o nome de usuário não seja encontrado ou a senha digitada não corresponda a senha armazenada, o programa deve imprimir a mensagem `Nome de usuario ou senha invalidos`. Se tudo der certo, a mensagem `Autenticacao feita com sucesso` deve ser impresssa.

3.1.3 Deletar

A instrução `deletar` tem o seguinte formato:

```
del userName password
```

Onde `del` é o código de identificação da operação `deletar`, `userName` é o nome do usuário a deletar e `password` é a senha desse usuário. O sistema só deve deletar o usuário se autenticação for feita com sucesso. Caso contrário a instrução não deve ter efeito (nem exibir qualquer mensagem).

3.2 Exibir

A instrução `exibir` tem o seguinte formato:

```
exib
```

Onde `exib` é o código de identificação da operção `exibir`. Ao receber essa instrução, o sistema deve imprimir de forma ordenada, de acordo com o nome de usuário, todos os registros.

A impressão tem o seguinte formato:

```
userName passwordHash\n
```

3.3 Tabela Hash

O tamanho inicial da tabela Hash deve ser 16. Sempre que o fator de carga da tabela exceda $\alpha = 0,85$, então você deverá criar uma nova tabela, com o dobro do tamanho da atual, recomputando o hash code de todos os elementos e armazenando na nova tabela. Nesse processo de reinserção, aproveite para limpar os espaços deletados.

3.4 Exemplo

Abaixo, cadastramos 3 usuários, procuramos por 2, sendo um não existente e imprimos os registros.

```
cad nome1 31415
cad nome3 58979
cad nome4 33315
cad nome2 92653
aut nome1 31415
aut outroNome 12345
del nome4 33315
exib
```

Saída:

```
Cadastro realizado com sucesso.
Cadastro realizado com sucesso.
Cadastro realizado com sucesso.
Autenticacao feita com sucesso.
Nome de usuario ou senha invalidos.
nome1 fd80c4b06025c38f9d6958ebe4f14532
nome2 b864a611f42875942e887b3698c6b121
nome3 37a59e737e77a8796c20e07de64458c2
```

4 Entrega e Avaliação

O trabalho será avaliado levando em consideração:

1. Realização dos objetivos / lógica utilizada
2. Uso de comentários e estrutura no código (e.g. indentação, legibilidade, modularização)
3. Resultado da plataforma run.codes
4. Eficiência da implementação

5 Obersvações e Dicas

- *Table Doubling* é uma alternativa para determinar o tamanho da tabela hash sem antes saber o numero de informações que vão ser armazenadas nela. Veja link para o método abaixo.
- O algoritmo MD5 não precisa ser implementado, ele pode ser obtido a partir de alguma fonte externa[3] (ver links abaixo), mas é importante que você estude como o algoritmo funciona.
- O run codes não possui a biblioteca openssl\md5.
- Para a instrução de exibição ordenada você deve usar um código de ordenação próprio. É proibido usar o Bubble-sort, mas se quiser pode.

Referências

[1] Wikipedia: MD5,
`https://pt.wikipedia.org/wiki/MD5`

[2] Rosetta: MD5,
`https://rosettacode.org/wiki/MD5#C`

[3] Implementação das funções MD5,
`https://openwall.info/wiki/people/solar/software/public-domain-source-code/md5`

[4] OCW: Table Doubling,
`https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/lecture-9-table-doubling-karp-rabin/`