INF101 – Introdução à Programação II
Roteiro de Prática: 15 de abril de 2021

Introdução

Nesta aula prática, vamos exercitar o uso de arquivos com ênfase em arquivos de saída. Observe que os arquivos que estamos usando em INF101 são arquivos de texto. Por ora, não usaremos diretamente nenhum arquivo binário. Caso seja necessário, usaremos alguma biblioteca que encapsule as operações necessárias para arquivos binários. Por exemplo, em INF100 e no início de INF101, usamos a biblioteca `PIL` para esconder a manipulação de arquivos binários contendo imagens. Se precisássemos usar arquivos de áudio, poderíamos importar o módulo `pydub` ou o `playsound`, por exemplo. E para manipular arquivos de vídeo, poderíamos usar a biblioteca `cv2` do projeto de código aberto OpenCV.

Vamos resolver hoje o problema: administrar um arquivo de senhas *(passwords)* associadas a nomes de usuários *(user login names)*. Todo sistema operacional manipula arquivos semelhantes aos com que vamos trabalhar nesta aula. É óbvio que um arquivo desta modalidade em um sistema operacional real contém muito mais informações além do simples par *login name, password*. Ademais, o componente *password* teria de ser criptografado. Hoje não vamos mexer com criptografia. Os dois componentes do par serão armazenados por linha e serão separados por meio do caractere tab (`'\t'` em Python). A senha *(password)* ficará armazenada em texto claro *(clear text)*. O programa pedirá ao usuário entrar com um par *login name, password*. Ato contínuo, o programa pesquisará pelo *login name* no arquivo de texto contendo esses dados. Se for encontrado, o programa verificará se a senha digitada pelo usuário bate com a respectiva senha armazenada no arquivo. Se as senhas se casarem, então o programa emite uma mensagem de autenticação bem sucedida. Se não se casarem, o programa emite uma mensagem de advertência de falha na autenticação. Caso o *login name* não se encontra no arquivo, ele inserirá o par *login name, password* no final do arquivo.

Instruções

1. Faça o *download* do esqueleto do programa que se encontra no sistema de entrega do LBI. Abra o IDLE e mude o nome do arquivo-fonte para `p10.py`. Não se esqueça de salvá-lo de tempos em tempos, porque pode ocorrer falha de energia elétrica durante a aula prática.
2. Complete os comentários obrigatórios (nome, matrícula, data e uma breve descrição sobre o que o programa faz).
3. O programa está estruturado em quatro funções: `main()`, `autenticado(s, es)`, `pesquise(nome)` e `insira(nome, senha)`.
4. A função `main()` faz o seguinte: lê o *login name* de um usuário e, em seguida, a sua respectiva senha. Depois ela pesquisa (sequencialmente) no arquivo de senhas se o usuário se encontra lá. Caso a pesquisa tenha sucesso, ela testa se a senha retornada pela pesquisa bate com a senha que foi dada pela entrada do usuário. Se bater, emite uma mensagem de sucesso na autenticação do usuário. Se não bater, emite uma mensagem de advertência: o usuário não foi autenticado. Caso a pesquisa no arquivo de senhas pelo *login name* obtiver fracasso, a função `main` insere o novo *login name* e a respectiva senha no arquivo de senhas. A função `main` já está pronta no esqueleto. Favor não a modificar.
5. A função `autenticado(s, es)` é trivialmente simples aqui: ela apenas retorna o booleano resultante da igualdade entre `s` e `es`. Ou seja, se a senha lida pela entrada do usuário for igual à senha armazenada no arquivo de senhas referente ao *login name*, ela retorna `True`. Caso contrário, retorna `False`. Esta função já está pronta no esqueleto. Não mexa nela.
6. Implemente a função `pesquise(nome)`. Ela deve manipular o arquivo de senhas no modo *read*, caso o arquivo já exista no diretório corrente. Caso o arquivo ainda não exista no diretório corrente, a função cria o arquivo do zero, simplesmente abrindo-o, por exemplo, no

modo *write*. Coloque cada uma destas duas opções dentro de blocos `try-except` respectivamente. No bloco `try`, a função processará a pesquisa sequencial pelo arquivo em busca do *login name* passado como parâmetro. No bloco `except`, use a exceção específica `OSError` que sinaliza problema na abertura de arquivo. Dentro deste bloco, imprima uma mensagem acerca da não existência do arquivo no diretório corrente e que será providenciada a criação do arquivo. Veja no Exemplo 1 de Teste do Programa, qual deve ser exatamente essa mensagem. Não se esqueça de fechar o arquivo de senhas em qualquer das situações: o arquivo existe e o arquivo não existe. Inspire-se na implementação da função `insira(nome, senha)` descrita no item 7 abaixo. Lembre-se também do retorno da função de pesquisa: caso a pesquisa tenha sucesso, ela retorna a respectiva senha armazenada no arquivo referente à chave `nome` passada como parâmetro; caso a pesquisa fracasse, ela retorna o caractere nulo, isto é, `'\0'`.

7. A função de inserção é muito fácil, porque o Python permite abrir um arquivo em modo *append* que auxilia enormemente no trabalho de inserção do novo par *login name, password* no final do arquivo. Esta função já está pronta. Não a modifique por favor!
8. Teste seu programa com os dados mostrados nos dois Exemplos de Teste do Programa abaixo. Os dados digitados pelo usuário estão enfatizados em fundo amarelo. Observe que o programa termina a execução ao entrar com um *login name* vazio. Então o programa emite uma mensagem de término.
9. Se seu programa entrar em *laço infinito* ou travar por alguma razão, digite `CTRL-C` na janela do *Python Shell* para interromper a execução do programa.
10. Inspecione visualmente o arquivo `passwords.txt` produzido pelo seu programa, usando para tanto um editor de texto puro, tipo `bloco de notas` no Windows ou o `gedit` no Linux, ou ainda o próprio IDLE em qualquer dos dois sistemas operacionais. Veja se seu arquivo bate com o exemplo de teste abaixo.

☞ Não se esqueça de preencher o cabeçalho do código fonte com seus dados, a data de hoje e uma breve descrição do programa.

Após certificar-se de que seu programa esteja correto, envie o arquivo com o código fonte (`p10.py`) através do sistema de entrega do LBI.

Exemplo 1 de Teste do Programa

```
Início do processamento do arquivo de senhas 'passwords.txt'.

Login name: lcaa
Password: 123456
O arquivo de senhas 'passwords.txt' não existe. Está sendo criado...
O usuário 'lcaa' não existe. Está sendo criado...

Login name: 90876
Password: carvalho
O usuário '90876' não existe. Está sendo criado...

Login name: flordelis
Password: camelia
O usuário 'flordelis' não existe. Está sendo criado...

Login name: 90876
Password: carvalho
```

```
O usuário '90876' foi autenticado pelo sistema.

Login name: flordelis
Password: camélia
O usuário 'flordelis' NÃO foi autenticado pelo sistema.

Login name:

Fim do processamento do arquivo de senhas.
```

Exemplo 2 de Teste do Programa

```
Início do processamento do arquivo de senhas 'passwords.txt'.

Login name: lcaa
Password: 123456
O usuário 'lcaa' foi autenticado pelo sistema.

Login name: 90876
Password: pau-brasil
O usuário '90876' NÃO foi autenticado pelo sistema.

Login name: admin
Password:
O usuário 'admin' não existe. Está sendo criado...

Login name: guest
Password: guest
O usuário 'guest' não existe. Está sendo criado...

Login name:

Fim do processamento do arquivo de senhas.
```

Arquivo `passwords.txt` produzido pelos Exemplos de Teste

```
lcaa  123456
90876 carvalho
flordelis   camelia
admin
guest guest
```