MAC0242 - Laboratório de Programação II

prof. Dr. Alfredo Goldman

Segundo mini exercício programa

De forma bastante simplificada, administradores de sistemas são os responsáveis por manter a infraestrutura de organizações funcionando. Isso envolve desde administrar bancos de dados e gerenciar usuários, até fornecer informações úteis que são de interesse de todos. Bons administradores de sistemas automatizam ao máximo suas tarefas. Neste mini-EP, vocês terão a oportunidade de fazer esse tipo de automatização em Python. Ele vai consistir de duas etapas, podendo ser realizado **em dupla** ou **individualmente**.

Primeira Etapa

Escolha uma das três opções:

1. **Usuários duplicados**

Um problema corriqueiro ao se criar contas aqui no IME é que nem sempre os administradores sabem se os requisitantes já possuem ou não uma conta. A forma adotada de verificar se dois usuários têm chance de ser o mesmo é comparar o primeiro nome e o último, por meio de um *script*.

Outro problema é verificar se dois usuários têm o mesmo UID (Unique ID). Isso poderia acontecer, por exemplo, depois de uma migração de sistemas de autenticação ou mesmo por causa de um *bug* em um *script* malfeito.

Seu objetivo como novo sysadmin é justamente descobrir se há essas inconsistências.

Seu *script* deverá trabalhar com dois parâmetros:

- ➔ `-a: indica que queremos procurar a partir dos nomes;`
- ➔ `-u: indica que queremos procurar a partir dos UIDs.`

A **entrada** consiste num arquivo de texto passado via **entrada padrão** com o seguinte formato:

```
<usuário>:x:<UID>:<GID>:<Nome do usuário>:<home>:<shell>
```

Saída:

- ➔ No primeiro caso, o que deve ser devolvido para cada homônimo é:
  - ◆ o nome e o último nome;
  - ◆ uma lista separada por vírgulas contendo todos os usuários homônimos;
  - ◆ o nome e a lista devem estar separados por ':'.
- ➔ No segundo caso, para cada UID com conflito devolveremos:
  - ◆ UID;
  - ◆ uma lista separada por vírgulas contendo todos os usuários com o mesmo UID;
  - ◆ o UID e a lista devem estar separados por ':'.

Exemplo de execução:

- ➔ Primeiro caso:

```
[root@zillertal ~]# script -a < /etc/passwd
Fulano Tal:fulano,fulano2
```

- ➔ Segundo caso:

```
admin@ldap:~# getent passwd | script -u
512:corinthians,campeao
397:gold,goldat,agoldman
```

2. Bandex!

Depois da volta do Bandejão, você, o sysadmin que virou a noite tentando consertar o problema envolvendo a migração recente do módulo de autenticação e a

atualização das máquinas, resolveu verificar o que tinha no cardápio, mas... o driver de vídeo do seu computador parou de funcionar. Agora só resta o modo texto!

Seres humanos comuns não conseguem processar com tanta facilidade hipertextos, então é sua missão (e seu interesse) escrever um *script* que processe as páginas do cardápio de todos os restaurantes Bandejão que podem ser obtidas em

http://www.usp.br/coseas/COSEASHP/COSEAS2010_cardapio.html

e devolve justamente o que teremos de comida.

O seu script deverá trabalhar com os seguintes parâmetros:

- `-b <nome do bandejão: fisica, quimica, pco ou central>`
- `-d <dia: segunda, terca, quarta, quinta, sexta, sabado e domingo>`
- `-j ou -a`

Ele também deverá trabalhar com um arquivo `~/.bandexrc` cujo conteúdo é justamente qual o seu restaurante favorito dentre: `fisica`, `quimica`, `pco` ou `central`. Isso determinará a execução do *script* sem o parâmetro `-b`.

Caso o parâmetro `-d` não apareça, seu *script* deve assumir o dia atual. Os parâmetros `-j` e `-a` indicam se é almoço ou janta, caso nenhum deles esteja presente, deveremos exibir almoço e janta.

Saída:

Comidas disponíveis no bandejão no dia.

Exemplo de execução:

```
user@debian:~$ bandex -b fisica -d terca -a
     ---Almoço---
     Arroz/feijão/arroz integral
     Carne picada à paulista
     Abobrinha ao forno com queijo
     Salada de alface
     Opcional: PVT à paulista
     Gelatina/refresco
```

3. Rastreamento de muambas

Compras pela Internet tornaram-se algo corriqueiro no dia-a-dia das pessoas. Com a facilidade de alguns cliques e um cartão de crédito generoso, podemos comprar até onde nosso limite permitir. Contudo, junto com as compras tem a demorada espera pela entrega.

Os mecanismos de entrega trabalham com um código de rastreamento que diz qual a situação da entrega. Por exemplo, o site dos Correios possui um sistema de rastreamento online no *link*

http://www2.correios.com.br/sistemas/rastreamento/

Para rastrear um objeto, é preciso de um código de 13 dígitos que segue a expressão: `[A-Z]{2}[0-9]{9}[A-Z]{2}`. Detalhes acerca do significado do código estão disponíveis em

http://www.correios.com.br/para-voce/precisa-de-ajuda/como-rastrear-um-objeto

A página de rastreamento trabalha com um formulário cujo conteúdo é "postado" na página de resultado. Basta procurar por **<form>** no código para descobrir onde os

dados são "postados" e o método que é utilizado. Isso é crucial para se fazer o preenchimento automático.

O objetivo deste exercício é construir um *script* que preencha automaticamente o formulário do site dos Correios e consiga acompanhar o transporte de um dado item. Ele deve verificar se houve alguma alteração no resultado da página desde a última vez que foi consultada e informar ao usuário que houve, de fato, a alteração. *Seja criativo no método de informar o usuário!*

Seu script deve contar com um parâmetro: `<cod. rastreamento>`. Inclusive, ele deve verificar se o código é válido.

Saída:

Se **houver alteração**, seu script deverá devolver exatamente qual foi a alteração. Ela pode ser encontrada na `<div class="status">`. Caso contrário, ele não deve devolver nada.

Exemplo de execução:

```
n00b@ubuntu:~$ script RC130449759HK #haverá alteração
Objeto entregue ao destinatário
n00b@ubuntu:~$ script RC130449759HK #não houve alteração
n00b@ubuntu:~$
```

Observações: esse tipo de script faz mais sentido quando executado automaticamente por algum computador. Haverá um bônus de até 0.5 para quem explicar uma forma de fazê-lo. Também podemos discutir um bônus pela criatividade ao avisar das atualizações. :-)

Testes

Vocês também deverão entregar um script shell (ou python) que valide automaticamente a saída do mini-EP, para algumas entradas. Nesse script, vocês devem fornecer uma entrada, rodar o programa e avaliar a saída. Se a saída for a esperada, o

script deve imprimir na saída padrão “O teste passou com sucesso!”. Caso contrário, ele deverá imprimir “O teste falhou! =(“.

Procure sempre testar casos extremos como a entrada vazia ou uma entrada muito grande. Geralmente programas quebram com elas. Bem como, teste o caso médio para garantir.

Não se esqueça de junto com o script enviar todas as entradas necessárias para os testes.

Relatório

Junto com o programa escolhido, é esperado um relatório no arquivo README contendo os nomes e número USP da dupla (ou pessoa), especificando qual foi a opção e demais informações que julgarem importantes para a correção. Pode ser um simples arquivo texto.

Lembrem-se de adicionar nesse arquivo tudo que for necessário para executar seu EP, incluindo uma maneira de instalar os módulos externos que você usar (caso seja necessário). Recomenda-se que seja usado a ferramenta *virtualenv* para criar um ambiente de desenvolvimento isolado do sistema, além de usar o comando *pip freeze > requirements.txt* para gerar uma lista de todas as dependências.

Um tutorial de como usar o virtualenv com o pip pode ser encontrado aqui (recomendo que ignorem a parte que ele fala comoele fala como inst instalar o pip, já que o mesmo geralmente pode ser instalado usando o gerenciador de pacotes da sua distribuição Linux).

Importante

- Lembrem-se que os programas devem ser feitos no Python 3. Na maioria das distros Linux, significa chamar o comando *python3* ao invés do *python*.

- O grupo deverá, inclusive, pesquisar módulos/bibliotecas que possam ajudá-los a resolver os exercícios como, por exemplo, uma biblioteca do Python que processe requisições HTTP, outra que processe argumentos por linha de comando ou mesmo uma que trabalhe com expressões regulares.

- O trabalho é estritamente em dupla ou individual. Veja a política do Departamento de Ciência da Computação para casos de plágio ou cola.

- Escreva de forma clara e estruturada todos os seus códigos e relatórios. Organize e nomeie todos os arquivos entregues de forma que eles possam ser facilmente identificados. A avaliação levará em conta todas essas questões! Uma apresentação ruim, ou a falta de clareza, poderá prejudicar sua nota.

- O programa deve ser entregue por meio do sistema Paca.

- Todos os arquivos devem ser entregues em um arquivo zip no formato **mini-ep2-<seu-número-USP>-<número-USP-dupla>.zip**. Exemplo: Se seu número USP for 12345678, você deverá entregar um arquivo com o nome mini-ep2-12345678.zip. **Apenas uma pessoa deve fazer essa entrega.**

- Enquanto o prazo de entrega não expirar, você poderá entregar várias versões do mesmo mini exercício-programa. Apenas a última versão entregue será guardada pelo sistema. Encerrado o prazo, você perderá 1 ponto para cada hora de atraso. Não deixe para entregar seu exercício na última hora!

- Guarde uma cópia do seu mini exercício-programa pelo menos até o final do semestre.