Praktikum zu LV 7328 - Ruby:
Übung 06

Arbeiten mit regulären Ausdrücken

Organisatorisches

- Arbeitsverzeichnis:
  `~/lv/ruby/06/`
- Dateinamen:
```
06-passwd.rb    # neu erstellen & abgeben
06-csv.rb       # neu erstellen & abgeben, optional
```
- Werkzeuge:
```
ruby            # Der Interpreter
emacs           # mit Ruby-Mode. Auch X-Emacs ok
scite, irb, ri  # Optionale Tools, wie üblich
                # mit Ruby-Mode
```
- Vorlagen:
```
marathon-ffm-2003.pdf
marathon-ffm-2003.txt
```

Vorübung

- A: Reguläre Ausdrücke verstehen
  - Geben Sie je 2-3 Beispiele für Strings, die zu folgenden regulären Ausdrücken passen:
    1) **/a.+b/**
    2) **/\d{3,}[.,]?\d*/**
    3) **/\BA/**
    4) **/(\||\+)(.*?)\1/**
- B: Reguläre Ausdrücke entwerfen
  - Geben Sie jeweils einen regulären Ausdruck an, der folgende Aufgaben erfüllt bzw. zu folgenden Strings passt:

    1) Extraktion des Ortes & der 3 Datumsteile aus:
    "Wiesbaden, den 1.12.05" , "Köln, den 03.11.2005"
    2) Kaufmännische Zahlendarstellungen:
    Max. 2 Nachkommastellen, neg. Vorzeichen ggf. folgend:
    "123" , "43,15", "1000-", "15,90-"
    (*) Wandeln Sie derartige Strings in Float um.

Vorgaben zu A & B

- Schreiben Sie Ihre Antworten zu Teilen A & B in einen Kommentarblock zu Beginn Ihrer Datei 06-passwd.rb
  - Zuerst das Übliche:

```
#!/usr/bin/env ruby
# Name, Vorname, Matrnr, Datum
```

  - Nun der Antwort-Block

```
=begin
Zu A1:
  ...
Zu A2:
  ...
(usw.)
Zu B2:
  ...
=end
```

  - Jetzt normal weiter mit Ihrem Programmcode...

Die Aufgabe

- Hintergrund-Information
  - Auf Unix-Systemen erfolgt die Benutzerverwaltung traditionell (u.a.) mit der Datei `/etc/passwd`.
  - Gemeinsam administrierte Rechnernetze werden zentral verwaltet, am Fachbereich mit einer LDAP-Datenbank. Anstelle des Lesens von `/etc/passwd` verwendet man das Kommando **`getent passwd`**.
  - Sie erhalten einzelne Zeilen. Diese sind entweder Kommentarzeilen (beginnend mit '#'), Leerzeilen oder Nutzdaten-Zeilen, die aus 7 (mit ':' getrennten) Feldern bestehen.
  - Einzelheiten zum Aufbau und der Bedeutung der Inhalte erhalten Sie mittels **`man 5 passwd`**.
  - Passwörter finden sich nie im Klartext in dieser Datei, sondern ggf. verschlüsselt. Bei uns werden sie aus Sicherheitsgründen in eine unzugängliche Stelle ausgelagert, z.B. in `/etc/shadow` .
  - Vereinfachungen: Ignorieren Sie Teilfelder (mit ',' getrennt). Gehen Sie davon aus, dass kein Feld das Trennzeichen ':' enthält.

Die Aufgabe

- C: Schreiben Sie ein Programm 06-passwd.rb
  - Es soll „getent passwd" ausführen und dessen Output zeilenweise auswerten.
  - Kommentarzeilen und Leerzeilen soll es ignorieren.
  - Nutzdaten-Zeilen soll es in seine Bestandteile zerlegen und damit folgende Teilaufgaben lösen:

1. Zahl der User ausgeben, die *nicht* die "bash" als Login-Shell verwenden.
2. Liste der Accountnamen ausgeben, die der Gruppe "200" angehören.
3. Report (als Tabelle) ausgeben:
   Gruppennummer vs. Anzahl Mitglieder,
   sortiert nach aufsteigender Gruppen-Nummer.

  - Hinweis: Beachten Sie die Vorgaben auf der folgenden Seite!

Vorgaben zu (C)

1. Verwenden Sie für Teil (1) einen regulären Ausdruck, der "passt", wenn die Zeile das Suchkriterium erfüllt. Zählen Sie die passenden Zeilen und geben Sie die Summe aus.

2. Verwenden Sie für Teil (2) einen regulären Ausdruck, der
   - "passt", wenn die Zeile das Suchkriterium erfüllt
   - in $1 die gewünschte Information bereitstellt

3. Hinweis zu Teil (3):
   - Erwägen Sie den Einsatz eines Hashes zum Zählen

- Allgemeine Hinweise:
  – Die Reihenfolge der drei Ausgaben spielt keine Rolle.
  – Fehlerbehandlung steht hier nicht im Vordergrund...

Die Aufgabe

- D: **(*)** Passwort-Codierung
  – Das Speichern verschlüsselter Passwörter ist zwar viel besser als das Speichern der Passwörter im Klartext, aber es ermöglicht immer noch so genannte Wörterbuch-Attacken:
  – Sind Passwörter leicht zu erraten, lassen sie sich durch Vergleich ihrer Codiersequenz mit den hinterlegten Codes ermitteln.

  Wir wollen dies nun simulieren:
  – Datei `passwd_sim` enthält *fiktive* Accounts mit verschlüsselten, ehemals real gewählten Passwörtern.
  – Erweitern Sie `06-passwd.rb` wie folgt

```
Lesen der Zeilen von "passwd_sim"
Lesen einer Wörterbuch-Liste von $stdin
Vergleich jedes Passwort-Codes mit jedem Wort aus der
Liste, Ausgabe des Accounts bei Treffer.
```

  – Melden Sie dem Kursleiter Ihre Trefferzahl. Wer findet die meisten?
  – BEACHTEN: „Experimente“ mit **realen** Accounts sind illegal!

Vorgaben zu (D)

- Vorgehen:
  - Lesen Sie die Wörterliste in ein Array ein mittels `$stdin.gets`, so dass folgender Aufruf funktioniert:

    ```
    $ 06-passwd.rb < wlist
    ```
  - Entfernen Sie ggf. *whitespace* von den Wörtern!
  - Beenden Sie das Programm, wenn die Liste leer ist.
  - Extrahieren Sie die ersten zwei Zeichen jedes verschlüsselten Passworts. Sie benötigen es als "*seed value*" später.
  - Das Testwort verschlüsseln Sie mittels `String#crypt`.
  - Skizze für einen Test:

    ```
    puts "#{acct} cracked" if pw_crypt==testword.crypt(seed)
    ```
- Hinweise:
  - Testen Sie Ihr Program mit Passwörtern Ihrer Wahl, die Sie im Klartext in die Wörterliste und verschlüsselt in passwd_sim eintragen (vorhandene PW dabei einfach ersetzen).
  - **Zumindest Ihre Passwörter sollten gefunden werden.**

Bemerkungen

- Reguläre Ausdrücke sind an den verschiedensten Stellen nützlich!
- Es gibt fertige, hocheffizient arbeitende Programme, die eine große Zahl an Wörtern und deren Varianten in kurzer Zeit gegen Listen verschlüsselter Passwörter – und systematischer Varianten - vergleichen können.

  Deshalb ist es - im eigenen Interesse - sehr wichtig, kein leicht zu erratendes Passwort zu wählen!
- Sie verwenden doch ein "sicheres" Passwort, oder? Bedenken Sie, wie einfach man es erraten könnte …

Extra-Aufgabe

- E: **(*)** PDF-Daten weiterverwenden
  - Datei `marathon-ffm-2003.pdf` enthält die Ergebnisliste des Frankfurt-Marathons am 27.10.03. Zum Nachlesen gut, aber schlecht zum Weiterverwenden!
  - Per copy/paste ist es möglich, die reinen ASCII-Daten zu extrahieren. Versuchen Sie es mal. Falls Probleme: Datei `marathon-ffm-2003.txt` enthält die Daten.
  - Leider geht die Formatierung dabei verloren! Die Daten erscheinen gar als *eine* lange Zeile...
- Aufgabe:
  - Rekonstruieren Sie die Tabellenstruktur mit `06-csv.rb`: Erzeugen Sie eine CSV-Datei (mit ';' als Trennzeichen) zur Weiterverarbeitung z.B in einer Datenbank.
  - Vorgehen: Zerlegung mit einem (komplizierten !) regex. möglich, `String#scan` und `Array#join` sind geeignete Methoden!
  - Ignorieren Sie die störenden eingestreuten Kopf/Fuß-Texte. Ohne diese ist die Aufgabe mit ca. 5 Quellcode-Zeilen lösbar!