



Praktikum zu LV 7328 - Ruby:

Übung 05

Arbeiten mit
regulären Ausdrücken



Organisatorisches



- Arbeitsverzeichnis:
`~/kurse/ruby/05/`
- Dateinamen:
`05-passwd.rb` # neu erstellen & abgeben
`05-csv.rb` # neu erstellen & abgeben, optional
- Werkzeuge:
`ruby` # Der Interpreter
`emacs` # mit Ruby-Mode. NICHT X-Emacs
`scite` # Ein portabler Editor, auch
mit Ruby-Mode
- Vorlagen:
`marathon-ffm-2003.pdf`
`marathon-ffm-2003.txt`



- 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 / zu folgenden Strings passt:
 - 1) Extraktion des Ortes & der 3 Datumsteile aus:
`"Wiesbaden, den 1.12.03" / "Köln, den 03.11.2003"`
 - 2) Kaufmännische Zahlendarstellungen:
Max. 2 Nachkommastellen, neg. Vorzeichen ggf. folgend:
`"123" / "43,15", "1000-", "15,90-"`
(*) Wandeln Sie derartige Strings in Float um.



- Schreiben Sie Ihre Antworten zu Teilen A & B in einen Kommentarblock zu Beginn Ihrer Datei 05-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, z.B. mit NIS (früher: "yellow pages"). Anstelle des Lesens von `/etc/passwd` verwendet man das Kommando `ypcat 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 nur verschlüsselt. Auf vielen Anlagen 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 `05-passwd.rb`
 - Es soll "ypcat 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 sogenannte Wörterbuch-Attacken:
 - Sind Passwörter leicht zu erraten, lassen sie sich durch Vergleich ihrer Codiersequenz mit den hinterlegten Codes ermitteln.
- Wir wollen dies nun simulieren:
 - `05-passwd.rb` soll so erweitert werden, dass es eine Wörterbuch-Liste von `$stdin` liest und jeden Passwort-Code mit jedem Wort aus der Liste vergleicht. Sollte ein Treffer dabei sein, gibt es den Account des Treffers aus.
 - Sollten Sie zufällig einen fremden Account "knacken": Bleiben Sie anständig und informieren Sie den Kursleiter und/oder einen der Administratoren ;-).



Vorgaben zu (D)



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

```
$ cat wlist | 05-passwd.rb
```
 - 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:
 - Schützen Sie Ihre Wörterliste, bevor Sie Ihr Passwort eintragen!

```
$ chmod 400 wlist
```
 - **Löschen Sie die Liste wieder nach Gebrauch!**



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 vergleichen können.

Deshalb ist es - im eigenen Interesse - **sehr wichtig, kein leicht zu erratendes Passwort zu wählen!**

```
$ yppasswd # Passwort auf der Kommandozeile ändern
```

- Sie verwenden doch ein "sicheres" Passwort, oder? Bedenken Sie, dass Ihre Kommilitonen es gerade versuchen zu erraten...



- 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 `05-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. Dann ist Aufgabe mit ca. 5 Quellcode-Zeilen lösbar!