– Praktikumsaufgabe 4 –

Thema: *Reguläre Ausdrücke*

Zielstellung: Verstehen des Konzepts des *regulären Ausdrucks*, periodische Shellskripts

1. Machen Sie sich mit *regulären Ausdrücken* vertraut. Nutzen Sie `grep` (und u. U. weitere Kommandos), um innerhalb des Bibeltextes aus dem zweiten Praktikum die folgenden Suchoperationen durchzuführen.

   a) Welche dreibuchstabigen Wörter gibt es, welche vierbuchstabigen?

   b) Wieviel verschiedene Versanfänge gibt es? (Als Versanfang wollen wir die beiden ersten Worte jedes Satzes interpretieren.)

   c) Wieviel verschiedene Substantive gibt es in der Bibel? Es reicht, großgeschriebene Worte zu identifizieren.

   d) Finden Sie alle Verse, in denen „Löwe“ zweimal vorkommt! Gibt es auch welche mit mehr als zwei Löwen?

   e)* Was ist das längste Wort der Bibel? Und das längste Wort, das keine Zahl ist? Hinweis: Die Länge einer Zeichenkette kann man mit einem kleinen `awk`-Programm bestimmen:

   ```
   awk '{ print length " " $1 }'
   ```

   Das Programm gibt die Länge der von `stdio` gelesenen Zeichenkette aus, dann ein Leerzeichen und dann die Zeichenkette selbst.

   Ein ähnlicher Mechanismus ist in der Bash bereits eingebaut:

   ```
   ${#STRING}
   ```

   substituiert die Länge der in der Variablen `STRING` gespeicherten Zeichenkette.

   f) Finden Sie alle Worte, die mit 'g' beginnen und mit 'n' enden! Finden Sie auch alle Worte, die mit 'a' beginnen und auf 'ing' oder 'ung' enden.

2. Schreiben Sie ein Shellskript, das (im Terminal) sekundengenau (d.h., jede Sekunde) die Zeit ausgibt. Löschen Sie vor jeder Zeitausgabe den Bildschirm. (Hinweis: Schauen Sie sich die Kommandos `date`, `sleep` und `clear` an).

3. Erstellen Sie ein Histogramm (eine Tabelle der absoluten Häufigkeiten) der von Ihnen genutzten Befehle!

   Hinweise:

   - In der Datei `~/.bash_history` stehen alle Befehle, die Sie eingegeben haben.

- Wir wollen nur die Kommandos, nicht ihre Optionen (sonst wird es kein schönes Histogramm), daher empfiehlt es sich, etwas wegzuschneiden.
- Damit Sie `uniq` im nächsten Schritt einsetzen können, müssen Sie zunächst `sort`ieren, um alle Dubletten untereinander zu bekommen.
- Schauen Sie sich die `man`-Page des Kommandos `uniq` noch einmal genau an. Gibt es eine Möglichkeit, die Zeilenanzahlen auszugeben?

4.** Erzeugen Sie einen Kindprozess und senden Sie anschließend mittels eines *Shared-Memory-Segmentes* (unsynchronisiert) unidirektional Daten von einem Prozess zum anderen. Was stellen Sie fest?

Hinweise:

- Außer `fork()` benötigen Sie `shmget()`, `shmctl()` und `shmat()`.
- Nur einer der beiden Prozesse darf das Segment anlegen! Wie können Sie diese Präzedenz realisieren?
- Wie wird das Segment identifiziert? Wie übertragen Sie den Identifikator vom anlegenden Prozess zum passiven Partner?

Was sind reguläre Ausdrücke?

Reguläre Ausdrücke (*regular expressions, RegExp*) sind Muster-Zeichenketten, die zur flexiblen Suche und Generierung von Zeichenketten eingesetzt werden. Sie werden von vielen Unix-Kommandos und Applikationen verstanden, z. B. von `grep`, `sed` und `emacs`.

Die Syntax regulärer Ausdrücke differiert u. U. (z. B. gibt es *Basic* und *Extended* Regular Expressions), das Funktionsprinzip ist jedoch identisch.

Nutzung z. B. mittels `grep`:

```
grep -o "RegExp" <suchdatei>
```

Die wesentlichen Regeln für *Basic Regular Expressions* sind:

- Jedes Zeichen, das kein Metazeichen (s. u.) ist, passt (nur) auf sich selbst.
- `'.'` passt auf ein einzelnes (beliebiges) Zeichen (auch Leerzeichen!).
- `'*'` passt auf *beliebig viele* (auch 0) Vorkommen des vorangegangenen Zeichens.
- `'?'` passt auf 0 oder 1 Vorkommen des vorangegangenen Zeichens.
- `'+'` steht für 1 oder mehr Vorkommen des vorangegangenen Zeichens.

- Zeichen in eckigen Klammern (z. B. `[aou]`) stehen für *genau eines* aus diesen Zeichen.
- Ist das allererste Zeichen in eckigen Klammern ein `^`, so ist eine *Invertierung* gemeint, d.h., der Ausdruck passt auf alle Zeichen *außer* denjenigen in der Klammer.
- Bereiche von Zeichen geben Sie mittels – an, beispielsweise `[0-9]` oder `[a-z]`[1]
- Die meisten Metazeichen, wie `'+'` und `'?'` sowie auch das Leerzeichen müssen mit dem Backslash maskiert („escapt“) werden (eckige Klammern nicht und `'*'` auch nicht). Die man-Page von `grep` führt die korrekte Syntax auf.
- Der Zeilenanfang wird durch `^` symbolisiert, das Zeilenende durch `$`.
- Der Anfang eines Wortes wird durch `\<` gekennzeichnet, das Wortende durch `\>`
- `[[:space:]]` steht für ein Whitespace (Leerzeichen, Tabulator).
- `[[:alpha:]]` steht für einen (beliebigen) Buchstaben.
- `[[:alnum:]]` oder `\w` stehen für einen Buchstaben oder eine Ziffer
- Beliebige Anzahlen des vorhergehenden Zeichens werden in {}-Klammern gesetzt (diese müssen maskiert werden). `{m,n}` bedeutet wenigstens `m` und höchstens `n` Vorkommen des vorstehenden Zeichens. `{m,}` bedeutet: mindestens `m` und unbeschränkt viele Vorkommen des Zeichens.
- Anstatt eines einzelnen Zeichens kann man sogenannte *Atome* vereinbaren, die in `()` gesetzt werden (escapen!). Beispielsweise sucht `"\(Väter\)\{1,2\}"` nach `Väter` und `VäterVäter`, jedoch nicht nach `VäterVäterVäter` (genauer brauchen wir's nicht).
- Der `|`-Operator repräsentiert die Alternative für Atome.

  ```
  grep -o "\(Vater\)\|\(Mutter\)" bibel.txt
  ```

  sucht nach allen Auftreten von Vater *oder* Mutter.

Beispiele:

```
grep -o "\<[[:alpha:]]*a\{2\}[[:alpha:]]*\>" bibel.txt
```

sucht alle Worte mit 'aa'.

```
grep -o "\<[[:alpha:]]*au[[:alpha:]]*au[[:alpha:]]*\>" bibel.txt
```

selektiert alle Worte, in denen „au“ zweimal vorkommt.

[1] Achtung: Mit Bereichsangaben können Sie keine Annahmen über Groß- oder Kleinschreibung treffen (`[A-Z]` selektiert *nicht* nur die Großbuchstaben); diese ist vielmehr *locale*abhängig. Sichere und portable Ausdrücke zur Selektion von Groß- bzw. Kleinbuchstaben sind `[[:upper:]]` bzw. `[[:lower:]]`.

```
grep -o "\(\<[[:alpha:]]*au[[:alpha:]]*eu[[:alpha:]]*\>\)\| \(\<[[: ↩
    alpha:]]*eu[[:alpha:]]*au[[:alpha:]]*\>\)" bibel.txt
```

selektiert alle Worte, die die Silben „au“ und „eu“ in beliebiger Reihenfolge enthalten, z.B. das hübsche Wort *Heuchelmaul*.

Beschreiben Sie, welche Zeichenketten der folgende Ausdruck selektiert.

```
grep -o "\<\(25[0-5]\|2[0-4][0-9]\|[01]\?[0-9][0-9]\?\.\)\{3\}  ↩
    \(25[0-5]\|2[0-4][0-9]\|[01]\?[0-9][0-9]\?\)\>
```

Literatur:

- `man 7 regex`
- *http://de.wikipedia.org/wiki/Regul%C3%A4rer_Ausdruck*
- Jan Goyvaerts, Steven Levithan: *Regular Expressions Cookbook*. O'Reilly, 2009