

## Übungsblatt 4: Shellskripte 2

13.11.2015

### Allgemeine Hinweise

- Abgabetermin für die Lösungen ist
  - **Freitag, 20.11.2015, 11:00**
- Schickt die Lösungen bitte per Email an Euren Tutor.

### Aufgabe 4.1: Backup-Skript (5 Punkte)

Erstelle ein Skript, das alle *wichtigen* Dateien in deinem Home-Verzeichnis sichert. Benutze den Befehl `tar` um ein gzip komprimiertes Archiv zu erstellen.

- Die Backup-Dateien werden nach dem Zeitpunkt des Backups in der Form „backup-JJJJ-MM-TT-HH-MM.tar.gz“ benannt und in ein eigenes Unterverzeichnis `~/backup` gespeichert. Ein am 2. 11. um 13:00 Uhr erzeugtes Backup soll also „backup-2015-11-02-13-00.tar.gz“ heißen. (2 Punkte)
- Existiert das Verzeichnis `~/backup` noch nicht, soll es erzeugt werden. Existiert es, ist aber eine Datei, soll es eine Fehlermeldung geben. (1 Punkt)
- Das Skript soll stets das gesamte Home sichern, egal, von wo aus es aufgerufen wird. Ausnahmen sind die Verzeichnisse `~/backup` (klar, dort liegen ja die Backups) `~/local` und `~/cache` (diese können sehr groß werden). Diese Verzeichnisse bitte auslassen. (1 Punkt)
- Das Skript soll den Benutzer genau über alles informieren, was es tut. (1 Punkt)

### Hinweise:

- Das Datum bekommt man mit Hilfe von `date`. Der Backtick (```), den Du brauchst, um die Ausgabe des Befehls per `'date'` in eine Zeichenkette zu verwandeln, befindet sich auf der englischen Tastatur bei der Tilde (`~`) links oben.
- Der Übungsleiter kann Fehler nur korrigieren, wenn er verstehen kann, was Deine Intention war. Daher bitte reichlich kommentieren. Ein Kommentar beschreibt *nicht*, was die Befehle tun, sondern, was der Gedanke dahinter ist!
- Beim Testen bitte auf den verfügbaren Speicherplatz (200MB) achten. `du -sh ~` sagt Dir jederzeit, wieviel Platz Du belegst. Eventuell musst Du dann Backups löschen.
- Wenn Du in Schritten vorgehst, ist es einfacher. Probiere zunächst auf dem Terminal, wie Du den Dateinamen des Backups erzeugen kannst, und benutze erst dann `tar`.

## Aufgabe 4.2: Zahlensysteme (5 Punkte)

- **4.2.1** Berechne die folgenden Zahlen  $a$  bis  $k$ , indem Du zwischen verschiedenen Zahlensystemen umrechnest. Dabei steht  $1234_7$  für die Zahl 1234 im Zahlensystem zur Basis 7.  $a_{10} = 1234_7$  bedeutet also, daß die Zahl 1234 im Zahlensystem zur Basis 7 ins Zahlensystem zur Basis 10 (Dezimalsystem) umgerechnet werden soll. (3 Punkte)
- Ihr könnt euch entweder ein bash-Skript schreiben, dass die Umrechnung für euch vornimmt, oder aber die Umrechnung per Hand durchführen (Lasst Umrechnungen von und ins Hexadezimalsystem hierbei aus).

**Hinweis:** In einigen Fällen dürfte die nebenstehende Tabelle nützlich sein.

- $a_{10} = 1234_7$
- $b_{16} = 1234_7$
- $c_{16} = 1234_{10}$
- $d_8 = 1234_{10}$
- $e_7 = 1234_{10}$
- $f_2 = CD_{16}$
- $g_2 = 27_8$
- $h_8 = 10000001_2$
- $i_8 = 10100101_2$
- $j_{16} = 10000001_2$
- $k_{16} = 10100101_2$

	2	7	8	10	16
0	0	0	0	0	0
1	1	1	1	1	1
10	2	2	2	2	2
11	3	3	3	3	3
100	4	4	4	4	4
101	5	5	5	5	5
110	6	6	6	6	6
111	10	7	7	7	7
1000	11	10	8	8	8
1001	12	11	9	9	9
1010	13	12	10	A	A
1011	14	13	11	B	B
1100	15	14	12	C	C
1101	16	15	13	D	D
1110	20	16	14	E	E
1111	21	17	15	F	F
10000	22	20	16	10	10

- **4.2.2** Im Computerumfeld wird häufig das Hexadezimalsystem ( $B = 16$ ) verwendet.
  - Welchen Vorteil bietet das System gegenüber dem Dezimalsystem ( $B = 10$ ) im Computerumfeld? (1 Punkt)
  - Welchen Vorteil bietet es gegenüber dem Oktalsystem ( $B = 8$ ) im Computerumfeld? (1 Punkt)

### Hinweise:

- In bash kann ein String mittels `echo $string | grep -o .` in ein Array von Zeichen umgewandelt werden.
- Mit `rev` kann die Reihenfolge umgekehrt werden.
- bash unterstützt nativ nur Integerarithmetik, besonders hilfreich sind dabei die *modulo* und die Divisions Operation `%` und `/`.
- Der Kommandozeilen Taschenrechner `bc` kann auch verwendet werden.  
`result=$(echo '1+2'| bc -l)`
- Geht schrittweise vor, wandelt die Zahl zu Beginn um, in zum Beispiel das Dezimalsystem, und danach in eine andere Basis. Ignoriert hierbei das Hexadezimalsystem.