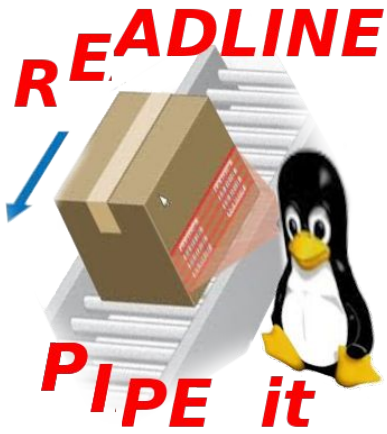
ein-ausblenden

Wir lesen Eingabezeilen, Namen Dateien, Daten usw. mittels der „bash“-Shell



1. Aufgabe

Lesen Sie eine CSV-Datei, die lauter Benutzer enthält ein und generieren Sie Damit Benutzer auf dem linux-System.

Was geschieht hier in Skript areadline2? Eswerden alle zeilen aus dem Csv Augegebn

```
#!/bin/bash

# Die Datei, die als Argument $1 übergeben wurde,
#                                    soll zeilenweise eingelesen werden
cat $1 | while read variable
do
  echo $variable
done
```

Text 1: Skript: areadline2: Kommando-Ausgabe | while read line → do

Die übergebene Datei Namensliste sieht z. B. wie folgt aus:

```
Meier,Peter,03.05.1999,Hofgarten,33,91555,Ansbach,09123-88842
Huber,Rudi,03.06.2000,Hufgasse,3,91122,Schwabach,09122-82441
Schmid,Hans,02.07.1998,Zu den Gründen,1,95355,Röttenbach,0911-842
Kerner,Otto,31.10.1997,Am Bahnhof,1,91421,Hallstadt,09993-83122
```

Text 2: Dateiinhalt der CSV-Datei: „Adressliste.csv“

Führen Sie nachfolgende Test durch!
Die letzte Zeile der der csv-Adressliste ist nicht mit einem Return abgeschlossen. - Was beobachten Sie hinsichtlich des Einlesevorganges bei der Anwendung obigen Skriptes? - Wozu dient der Befehl: ***„mapfile“?***

Er liest das CSV direkt in ein array ein es ist dabei gründlicher als wenn man es mit read macht da es am ende einer zeile kein Return sucht

2. Aufgabe

Erkläre das inline-Eingabeumleitung bzw. das Here-Dokument!

Geben Sie hierzu einfach nachfolgende Zeilen in ein Terminal mit gestarteter bash-Konsole ein! - Was beobachten Sie und wie lässt sich dies erklären?

```
cat <<\ENDE_EINLESEN
Heute ist `date`,
Sie befinden Sie im Verzeichnis `pwd`. Ihr
aktuelles Terminal ist `echo -n $TERM` und
Ihr Heimatverzeichnis finden Sie unter dem Pfad: $HOME.
ENDE_EINLESEN
```

Text 3: Bash-Befehle als inline-Eingabe

Der Text wird so ausgegeben wie er da steht

der / nach den << sorgt dafür das dass dastehende nur als String interpretiert wird das gleiche kann auch durch ' ' der text marke erreicht werden

Was beobachten Sie und warum beobachten Sie dieses?

3. Aufgabe

Bauen Sie dem HERE-Datei-Prinzip folgend einen Taschenrechner für die Konsole, indem Sie nachfolgenden Code verwenden:

```
#!/bin/bash
## Konsolen-Taschenrechner
if [ $# == 0 ]
then
   echo "Sie haben $0 ohne die zusätzlich benötigte Rechenaufgabe gestartet!"
   exit 1
fi

# Option -l für die mathematische Bibliothek
bc -l <<CALC
$*
quit
CALC
```

Text 4: Skript: rechne

4. Aufgabe

4.1 Beschreiben Sie die Funktion von nebenstehendem bash-Skript!

4.2 Welcher Unterschied ergibt sich, wenn die 8. Zeile nicht „done <<TEXT“, sondern „done <TEXT“ lauten würde?

```
#!/bin/bash
i=1

while read line
do
   echo "$i. Zeile: $line"
   i=`expr $i + 1`
done <<TEXT
Eine Zeile
`date`
Homeverzeichnis $HOME
Das Ende
TEXT
```

Code 5: Der read-Befehle und die HERE-Technik