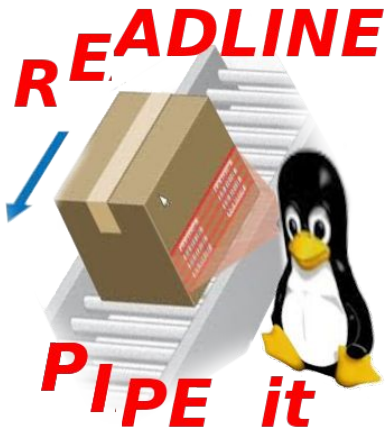
ein-ausblenden

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?

```
#!/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“?***

Von einer Textdatei „.txt“ bzw. CSV-Datei „.csv“ werden nur alle Zeilen eingelesen, wenn jede Zeile – auch die letzte Zeile - mit einem „<RETURN>“ („\n“) abgeschlossen ist.
Ist dies nicht der Fall, geht meist die letzte Zeile, die häufig ohne „\n“ endet, verloren.
Dies kann durch Anwendung des Befehls mapfile, der die einzelnen Zeilen Array-Feldern zuordnet, nicht unterlaufen. Man muss dann aber mit Array-Befehlen weiter arbeiten.

2. Aufgabe

Erkläre die inline-Eingabeumleitung bzw. das so genannte 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

Was beobachten Sie und warum beobachten Sie dieses?

Der „cat“-Befehl dient zum anneinanderbinden mehrerer Einzeldateien. Diese gibt der Befehl über die Standard-Ausgabe „1>“ auf die Konsole aus. Durch ein nachfolgendes Umleitungszeichen „>“ kann im regulären Gebrauch des Befehls eine neue Datei mit allen Inhalten der gelisteten Dateien erzeugt werden.
Jedoch in obigem Beispiel wird an Stelle einer Datei der Text, der dem cat-Befehl nachfolgt, bis zur Textmarke „ENDE_EINLESEN“ von „cat“ eingelesen und ausgegeben. Da im Text selbst Befehle als Substitutionen „`“ (BACK-TICKS) hinterlegt sind, werden diese zuvor noch ausgeführt, so dass die Ausgabe das Ergbnis der Befehle darstellt.

3. Aufgabe

Bauen Sie dem „HERE“-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
$*
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 7. 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

Lösung zu 4.1 und 4.2:

Das bash-Skript enthält von Zeile 3 bis Zeile 5 eine WHILE-Schleife. Der While-Schleife wird über die HERE-Anweisung der in den Zeilen 6 bis zur Marke „TEXT“ folgende Text eingelesen und die Substitutionsergebnisse in der While-Schleife dargestellt.
Würde man nur einen Pfeil nach links „<“ verwenden, so wäre es keine HERE-Anweisung mehr. Es würde dann nach einer Datei mit dem Namen TEXT im Dateisystem gesucht werden. Ist eine solche Datei vorhanden, wird diese in die While-Schleife in Zeile 6 eingelesen. Die restlichen Zeilen würden als separate Befehle interpretiert werden.

5. Aufgabe

Wie setzt man den „IFS“ sinvoll in bash-Skripten ein? - Hierzu zunächst ein einfaches Beispiel:

Beschreiben Sie die Funktion des IFS in nebenstehendem Beispiel!

```
#!/bin/bash
# die voreingestellten Zeichen für IFS werden zunächst gesichert!
BACKIFS="$IFS"
# Minuszeichen als Trenner
IFS=:
if   [  $#  -lt  1  ]
then
   echo "Das Skript: $0 benötigt einen User-login-Namen."
   exit 1
fi
# Ausgabe anhand von Trennzeichen in IFS auftrennen
set `grep ^$1 /etc/passwd`
echo "User            : $1"
echo "User-Nummer     : $3"
echo "Gruppen-Nummer  : $4"
echo "Home-Verzeichnis : $6"
echo "Start-Shell     : $7"
IFS=$BACKIFS
```

Text 6: Skript mit der Anwendung des IFS

Der IFS ***„input field separator“*** wird auch ***„internal field separator“*** genannt. Dieser ist meist mit mehreren Zeichen besetzt: z. B. [[:space:]], tab, \n usw. Diese Zeichen trennen Wörter, Zeichenfolgen Variablen usw. von einander. Wird IFS auf ein neues Zeichen beispielsweise wie im Beispiel von Text 6 auf den Doppelpunkt „:“ gesetzt, kann dies als Trennzeichen verwendet werden um die Zeile mittels „set“-Befehl an den Stellen des Doppelpunktes zu teilen und die Teilstücke als Positions- bzw. set-Parameter ($1 $2 $3 …) zu verwerten. So werden die Eigenschaften des Benutzers, der an das Skript übergeben wird, ausgegeben.

6. Aufgabe

In nachfolgendem Skript (Text 7) werden unterschiedliche Techniken und Befehl in Verbindung mit dem IFS angewandt. Beachten Sie den Unterschied zwischen einem **assoziativ adressierten** und **indizierten Array**. Der Befehl **mapfile** generiert orientiert an den Zeilen der Text-Datei ein Array. Der Index sollte immer von Anführungszeichen umrahmt sein (siehe z.B. “${!adresseni[@]}“), ebenso sollte die Ausgabe des Arrays sollte ebenfalls von Anführungszeichen umgeben sein.
Beschreiben Sie die Funktion des nachfolgenden Skripts!

```
#!/bin/bash
Adressena=()            # assoziatives Array oder gehashtes Array
declare -a adresseni    # indiziertes Array oder Array mit nummerischen Index

if [  $#  -lt  1  ] ||  [  !  -f  $1  ]
then
     Skript=`echo "$0" | sed "s/\///g"`
     echo "###### Achtung #########“
     echo "  Das Skript: $Skript muss "
     echo "  mit einer Adressliste.csv aufgerufen werden!"
     ech######################"
     exit 1
fi                                  #cat "$1" | ( mapfile; echo "${MAPFILE[@]}")
mapfile -t < $1             # printf "%s\n" "${MAPFILE[@]}"
mapfile -t < $1
echo "#########--- print mapfile ---################"
printf "%s" "${MAPFILE[@]}"
    # Adressen=`shuf -e "${MAPFILE[@]}"`
Adressena=( "${MAPFILE[@]}" )
adresseni=( "${MAPFILE[@]}" )
    # wichtig sind die Anführungszeichen!
    # --> Siehe hierzu https://sysware.computer/linux/scripte_variablen_arrays.html
echo -e "\n--------------Adressen[@]----------------------------"
    #printf "%s\n" "${Adressen[@]}"
echo "${Adressena[@]}"
echo "#############-----Adressen[2]------------##########"
echo "${adresseni[2]}"
echo "########################################"
declare -p
echo "-#.-#-##--#-#-#-##-##--#-#-#-#"
echo "Das Array adressini[]:"
for INDEX in "${!adresseni[@]}"
do
   echo "Index=\"${INDEX}\" Wert=\"${adresseni[$INDEX]}\""
done
echo "#+#+#+#+#"
adresseni=(  "${adresseni[@]}" )
for INDEX in "${!adresseni[@]}"
do
   echo "Index=\"${INDEX}\" Wert=\"${adresseni[$INDEX]}\""
done
echo "-jetzt würfeln wir die Zeilen neu aus-"
shuf -e  "${adresseni[@]}"
echo "----------Der Index bleibt aber bestehen!-----------"
z=0
while [ $z -lt ${#adresseni[@]} ]
do
  echo "$z -> ${adresseni[$z]}"
  ((z++))
done
echo "----Formatierte Ausgabe der Adressliste ----------"
for INDEX in "${!adresseni[@]}"
do
    #echo "Index=\"${INDEX}\" Wert=\"${adresseni[$INDEX]}\""
    IFSBACK="$IFS"
    Zeileninhalt="${adresseni[$INDEX]}"
    IFS=","
 set $Zeileninhalt
 IFS="$IFSBACK"
 echo -e "\n--- Id: ${INDEX} ----------------------\nName:    $1"
 echo -e "Vorname: $2 \ngeb. am: $3\nStraße:  $4 $5\nPLZ:     $6"
 echo -e "Ort:     $7\nTelefon: $8"
done
```

Text 7: Konsolen-bash-Adressverwaltung

Fortsetzung der Lösung von Aufgabe 6:

Verwendet man die weiter vorne deklarierte CSV-Adress-Datei als Übergabeparameter, so wird eine Ausgabe wie nebenstehend dargestellt generiert.
Dies wird erreicht durch das nummerisch indizierte Array, das die Adressdaten Zeilenweise adressiert.
Der IFS wir auf das Komma "," gesetzt, so dass die einzelnen Adressteile mittels „set"-Befehl in die Positionsparameter $1 bis $8 übernommen werden.
Diese werden dann formatiert mittels einer FOR-Schleife bezogen auf den Arry-Index wie dargestellt ausgegeben.

```
--- Id: 0 ----------------------
Name:    Meier
Vorname: Peter
geb. am: 03.05.1999
Straße:  Hofgarten 33
PLZ:     91555
Ort:     Ansbach
Telefon: 09123-88842

--- Id: 1 ----------------------
Name:    Huber
Vorname: Rudi
geb. am: 03.06.2000
Straße:  Hufgasse 3
PLZ:     91122
Ort:     Schwabach
Telefon: 09122-82441

--- Id: 2 ----------------------
Name:    Schmid
Vorname: Hans
geb. am: 02.07.1998
Straße:  Zu den Gründen 1
PLZ:     95355
Ort:     Röttenbach
Telefon: 0911-842
```

7. Aufgabe

Testen Sie die beiden Befehle: "echo $TERM"

und "infcomp"!

Welche Ausgaben erhalten Sie?

Der erste Befehl gibt an mit welchem Terminal Sie arbeiten, während der zweite Befehl die Escape-Sequenzen ausgibt, die den z. B. Funktionstasten zugeordnet sind!

8. Aufgabe

Sie wollen ein Passwort unsichtbar eingeben. Hierzu verwenden Sie "stty". Welche Befehlsfolge schaltet die Ansicht der Eingabe ab und wieder an?

Der Befehl zum Abschalten wird der Befehl: "stty -echo" und zum Einschalten "stty echo"
Die Befehlsfolge lautet z. B.: #!/bin/bash; stty -echo; read passwort; stty echo;

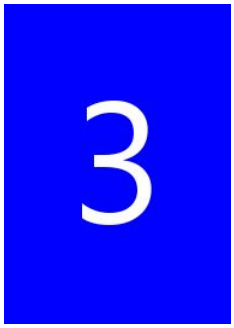
9. Aufgabe

Der zentrale Befehl zur Umleitung der Eingabe lautet **exec**.

Zeigen Sie, wie sich der Befehl exec anwenden lässt hinsichtlich ***stdout, stdin*** und ***stderr***. - Was bedeutet „Kommando" >&fd bzw. „Kommando" >>&fd.

10. Aufgabe

Schreiben Sie ein ARRAY in eine Datei. Beginnen Sie mit ***declare -a Adressen*** oder ***declare -A Adressen***



Und hier ein Rahmen mit Bild

Hier steht noch eine Antwort, im Rahmen mit Umrandung, grau nur die nicht druckbare Textbegrenzung