Ein Script...

- ist eine Textdatei.
- enthält Befehle zur Automatisierung in maschienenverständlicher Sprache.
- ist ein Ablauf von einem Anfang zu einem Ende.
- ist ein kleines Computerprogramm.
- wird bei Tabellenkalkulationen und Automatisierungen in der Textverarbeitung auch Makro genannt.
- wird in ausführbaren, binären Code übersetzt.
- wird bei jeder Ausführung neu interpretiert und ausgeführt (Programm nur einmal kompiliert -> direkt vom Kompilat ausgeführt.)
- ist ohne Interpreter nicht lauffähig.

13. cat und pipe „<<,<,>,>>"

Mit den Befehlen <<,<,>,>> lassen sich Inhalte umleiten. In einfacher <,> Schreibweise wird das Ziel überschrieben, mit doppelter Ausführung <<,>> einem Dateiinhalt angehängt.

Beispiel:

In die filelist pipen!

```
vmadmin@bmLP1:~/M122_Scripts$ ls –all > filelist
vmadmin@bmLP1:~/M122_Scripts$ mcedit filelist
vmadmin@bmLP1:~/M122_Scripts$ ls >> filelist
vmadmin@bmLP1:~/M122_Scripts$ mcedit filelist
```

In cat filelist pipen!

```
vmadmin@bmLP1:~/M122_Scripts$ cat < filelist
```

Der Interpreter...

- ist auf unterschiedlichen Systemen wie Linux oder Windows ein wiederkehrendes Software Design Konzept.
- übersetzt oder kompiliert eine Scriptdatei(/Textdatei) in eine binäre, ausführbare Datei.
- der Sprachen Python, Perl, PHP, SHELL usw. basieren auf dem Interpreter Design Pattern (bewährtes Entwurfsmuster mit dem sich ein bestimmtes, wiederkehrendes Softwareproblem lösen lässt).

Einsatzgebiet von Scripts

- **Datensicherung:** Bsp. Automatisches Backup alle 24 Stunden.
- **Verwaltung von Benutzern und Gruppen:** Bsp. In einer Domäne sollen 100 neue Benutzer erstellt werden, die in Tabellenform vorliegen.
- **Installation und Aktualisierung von Programmen:** Bsp. Neues Programm auf 40 Arbeitsplätzen installieren.
- **Konfiguration von Programmen:** Bsp. Eine Konfigurationsdatei wird auf mehreren PC's aktualisiert.
- **Netzwerküberwachung:** Bsp. Auslastung des Netzwerks überwachen.
- **Überwachung von Ressourcen:** Bsp. Kontrollieren, ob ein Service noch online ist.
- **Sicherheit:** Bsp. System auf verdächtige Dateien durchsuchen.

14. find

Führen Sie im Verzeichnis M122_Scripts folgende Befehle aus:

```
vmadmin@bmLP1:~/M122_Scripts$ sudo
find –name "*.sh" > ShellScriptList
```

Schauen Sie die Datei an:

```
vmadmin@bmLP1:~/M122_Scripts$ mcedit
ShellScriptList
```

Sc iges:

- D ekannteste Büroanwendung ist Microsoft Office. Als Skriptsprache werden hier Windows Scripting Host mit VBS (Visual Basic Script) sowie JScript und Javascript angewandt.
- OpenOffice.org von SUN nutzt eine eigene, speziell für OpenOffice.org konzipierte Skriptsprache namens OpenOffice.org Basic.

Begriffe

- **Datentyp:** Ein primitiver Datentyp wie z.B. int ist ein eingebauter Datentyp.
- **Objekt:** Objekte sind Datentypen, welche durch Entwickler selbst definiert werden können, indem sie Klassen machen und davon dann diese als Datentyp nutzen
- **Variable:** Ist ein Container zur Aufbewahrung von bestimmten Werten und auf diesen kann man im Verlaufe des Programms zugreifen.
- **Deklaration:** Dem Compiler wird der Typ und der Bezeichner mitgeteilt.

Paradigma

- **Funktional Programmieren:** Bei diesem Paradigma bestehen Programme ausschliesslich aus Funktionen.
- **Imperativ Programmieren:** Hier wird dem Computer mit einer Reihe von Anweisungen vorgegeben, in welcher Reihenfolge was getan werden soll. (Ähnlich wie prozedurale Programmierung)
- **Objektorientiert Programmieren:** Die Idee ist, die Architektur einer Software den Grundstrukturen desjenigen Bereiches der Wirklichkeit auszurichten, der die gegebene Anwendung betrifft. Dazu werden Konzepte wie Klassen, Vererbung, Polym.. etc. benötigt

Inhalt Dokumentation (mindestens)

Spätestens nach der Fertigstellung eines Skripts ist es unerlässlich, dieses State-of-the-art zu dokumentieren.

Programmierer verfolgen dabei zwei Ziele:

1. Die Arbeit bei zukünftigen Änderungen bzw. Wartung zu erleichtern.
2. Dem Benutzer bei der Bedienung zu helfen.

1. Aufgabenbeschreibung
2. Programmbeschreibung
3. rammcode
4. und Struktogramm Darstellung der case Anweisung im Quelltext
5. Testfälle
6. Bedienungsanleitung

grep –i VM /etc/passwd
grep –l (listet betroffene dateiverzeichnise auf)
grep –c (anzahl resultate)
grep –i -l vm /home/vmadmin/M122_Scripts *.sh > VMuser
grep –i Pinguin /home/vmadmin/M122_Scripts/*.sh > PinguSkripte
grep -i -l -E "(wort1|wort2)" /home/vmadmin/M122_Scripts/*.sh
egrep -l -i "(wort1|wort2)" /home/vmadmin/M122_Scripts/*.sh
-l: ignoriert gross klein –E: Enhanced –l: nur dateinamen, die wert beinhalten.

$(expr $x/45) =kommandozeile auch andere befehle $((x/45))
interpreter

Shellprogramme

- Bash (Bourne again shell)
- Csh (c-shell)
- Ksh (Korn-Shell)
- Tcsh

```
a=$1
if [ -e $a ]
        then
          echo "Die Datei \"$a\" existiert."
        else
          echo "Die Datei \"$a\" existiert nicht."
fi
```

```
echo "1. Parameter: $1"
echo "Name dieses Skript: $0"
echo "Sie haben \"$1\" eingeben."
echo "Anzahl Parameter:" $#
echo "Alle Parameter als Einheit:" $*
```

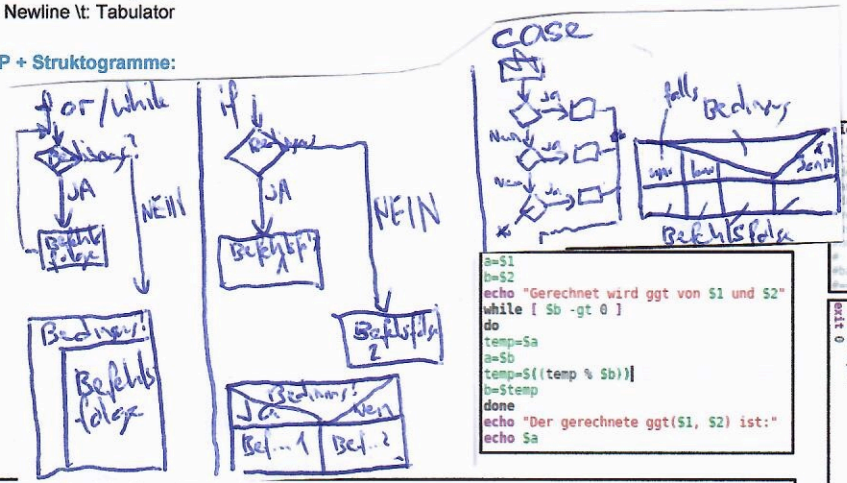
Befehl	Erklärung
Which bash	Zeigt den Pfad zum Befehl (z.B. /bin/bash)
Chmod 700 script.sh (Besitzer-Gruppe-Rest)	Script berechtigen zum Ausführen.
./script.sh oder bash script.sh	Ausführen von Scripts
Man | sort mit –r oder –R | adduser | deluser	Klassische Hilfe | -r = z-a und -R= a-z
Find [-name „	Suchen nach Dateien
Cat | cut mit –d und oder -fZahl	Dateien nacheinander ausgeben | -d = trennzeichen in hochkomma –fX,Y gew feld
<, <<, >>, quelle > ziel	1)überschreiben 2) Anhängen
Ls –al	Alle Sachen anzeigen inklusive Berechtigung
Exit 0	Programm beenden

Sqlite3 Eigenschaften: klein aber fein, serverless, zeroconfiguration, self contained, transactional, plattformunabhängig
.tables / .databases / .help / SELECT * from TABELLE; etc / sqlite3 name.db zum öffnen z.B.
sudo apt-get install sqlite3

Entweder: if test $# -ne 2 oder if [$# -ne 2] | aufbau /etc/passwd = **Benutzername:Kennwort:UID:GID:Kommentar:Heimatverzeichnis:LoginKommando**

Standard berechtigung: -rw-rw-r-- 700: -rwx------ #Lesen mit ls -al
\n: Newline \t: Tabulator

PAP + Struktogramme:




```
if [ -z $(sqlite3 SIX_Bankenstamm.db ".tables" | grep Bankenstamm) ] ; then
  echo "Keine Datenbank vorhanden!!"
else
  sqlite3 SIX_Bankenstamm.db "SELECT * FROM Bankenstamm" | grep $1
fi
```

```
richtig vorallem binbash, title, aufruf, was es macht, autor
#!/bin/bash
# Title           :addPerson.sh
# Description     :This script will add a name to the phonelist.
# Author          :mirio.eggmann@me-solutions.ch
# Date            :20151101
# Version         :1.0
# Usage           :bash addPerson.sh or ./addPerson.sh
# Notes           :You don't have to install any other programs.
#                  Maybe you have to authorize the script first with "chmod 700 addPerson.sh".
# Bash_version    :4.3.11(1)-release
```

```
a=$1
b=$2
echo "Gerechnet wird ggt von $1 und $2"
while [ $b -gt 0 ]
do
temp=$a
a=$b
temp=$((temp % $b))
b=$temp
done
echo "Der gerechnete ggt($1, $2) ist:"
echo $a
```

```
echo "Geben Sie ihren Namen ein (mit quit beenden):"
read Name
while true; do
  if [ "$Name" = "quit" ] ; then
    echo "Exit add person!"
    break
  else
    echo $Name >> Phonelist
    echo "Geben Sie ihren Namen ein:"
    read Name
  fi
done
```

```
# Import the data from the csv into the database
echo -e '.separator ";"\n.import SIX_BankenstammCH.csv Bankenstamm' | sqlite3 SIX_Bankenstamm.db
```

```
counter=$1
factorial=1
while [ $counter -gt 0 ]
do
    factorial=$(( $factorial * $counter ))
    counter=$(( $counter - 1 ))
done
echo "Die Fakultät von $1! ist:"
echo $factorial
```

```
database="$PWD/linuxusers.db"
sqlite3 $database <<EOF
DROP TABLE IF EXISTS LinuxBenutzer;
CREATE TABLE LinuxBenutzer (id INTEGER PRIMARY KEY, uid INTEGER, benutzername TEXT, gid INTEGER, home TEXT);
EOF
grep -i -E "(VM|root|tux)" /etc/passwd > myusers
while read line
do
  name=`echo $line | cut -d ':' -f1`
  gid=`echo $line | cut -d ':' -f4`
  uid=`echo $line | cut -d ':' -f3`
  home=`echo $line | cut -d ':' -f6`
  sqlite3 $database "INSERT INTO LinuxBenutzer(uid, benutzername, gid, home) VALUES ($uid, '$benutzername', $gid, '$home');"
done < myusers
sqlite3 $database "SELECT * FROM LinuxBenutzer;"
rm myusers;
```

```
case "$1" in
   Tux)
    echo "Hallo Pinguin $1";;
   Veronica)
    echo "Hallo Veronica";;
   *)
    echo "Hast du keinen Namen oder was?";;
esac
exit 0
```

```
IFS=' ' read -a userArray <<< $(cut -d':' -f1,3,4,6 /etc/passwd | grep -E 'tux|vm|root')
if [ -z $(sqlite3 LinuxUsers.db ".tables" | grep Users) ] ; then
    sqlite3 LinuxUsers.db "CREATE TABLE Users(id INTEGER PRIMARY KEY, Username TEXT, UID INTEGER, GID INTEGER, Home TEXT)"
fi
for element in ${userArray[@]}
do
    IFS=':' read -a rowArray <<< $element
    sqlite3 LinuxUsers.db "INSERT INTO Users(Username, UID, GID, Home) VALUES ('${rowArray[0]}', '${rowArray[1]}', '${rowArray[2]}', '${rowArray[3]}')"
done
echo -e "Benutzer\tUID\tGID\tHome"
IFS=' ' read -a selectArray <<< $(sqlite3 LinuxUsers.db "SELECT * FROM Users")
for element in ${selectArray[@]}
do
    IFS='|' read -a rowArray <<< $element
    if [ "${#rowArray[1]}" -lt "8" ] ; then
        echo -e "${rowArray[1]}\t\t${rowArray[2]}\t${rowArray[3]}\t${rowArray[4]}"
    else
        echo -e "${rowArray[1]}\t${rowArray[2]}\t${rowArray[3]}\t${rowArray[4]}"
    fi
done
```

```
x=10
# Variant 1: Not for coders! String substitution
# Konservativ: Merken Sie, der Backtick-Operator
# ZEICHEN = +,-,\* (wird nicht als joker angeschaut), /, %
echo `expr $x ZEICHEN 1`

# Variant 2: Liberal und useful
# ZEICHEN = +, -, \* (wird nicht als joker angeschaut), /, %
echo $(expr $x ZEICHEN 45)

# Variant 3: Executed directly by the shell interpreter
# no substitution like in the command expr
# ZEICHEN = +,-,*,/, %
echo $((x ZEICHEN 5))
```

```
for i in $(cat zwischenspeicher); do
...
done
```

```
cut -d':' -f1,2,3,4,5,6,7 /etc/passwd > myusers.csv
```

r	w	x	Kennzahl	Bedeutung
0	0	0	0	keine Rechte
0	0	1	1	Ausführung, aber weder Lesen noch Schreiben
0	1	0	2	Nur Schreibrecht
0	1	1	3	Schreib- und Ausführungsrecht, nicht Lesen
1	0	0	4	Nur Leserecht
1	0	1	5	Lese- und Auführungsrecht
1	1	0	6	Lese- und Schreibrecht, kein Ausführungsrecht
1	1	1	7	Lese-, Schreib- und Ausführungsrecht

Shell-Variable	Bedeutung
$BASH_VERSION	Version des Bash Shell
$HOME	Heimverzeichnis
$HOSTNAME	Computername
$OLDPWD	Letztes Arbeitsverzeichnis
$PS1	Aussehen des Shell-
$PWD	Aktuelles Arbeitsverzeichnis

Kontrollstruktur	
Auswahl	if ... then ... else ... fi
Ein- und zweiseitige Auswahl	
Mehrfachauswahl	case ... in ... esac
For-Schleife	for ... in ... do ... done
While-Schleife	while ... do ... done
Until-Schleife	until ... do ... done

Operator	IT English	Bedeutung
Vergleich von Zeichenketten (strings)		
=	Equal	Gleich
!=	Not equal	Ungleich
-z	Zero	Leere Zeichenkette
-n	Non-zero	Nicht leere zeichenkette
Vergleich von numerischen Werten		
-eq	Equal	Gleich
-ne	Not equal	Ungleich
-lt	Less than	Kleiner als
-le	Less or equal	Kleiner oder gleich
-gt	Greater than	Grösser als
-ge	Greater or equal	Grösser oder gleich
Abfrage von Dateiattributen		
-e	Exist	Existiert die Datei
-r	Read	Leserecht gesetzt
-w	Write	Schreibrecht gesetzt
-x	Execute	Ausführungsrecht gesetzt
-f	File	Normale Datei
-d	Directory	Verzeichnis
!	Not	Nicht

/ Das Wurzelverzeichnis steht ganz oben in der Hierarchie.

/bin Hier befinden sich wichtige Programme für Anwender, die immer verfügbar sein müssen, z. B. die Shells.

/boot Hier befinden sich die zum Hochfahren des Systems unbedingt erforderlichen Dateien. In der Hauptsache ist das der Kernel, im Normalfall eine Datei mit dem Namen vmlinuz. Aber auch andere Namen sind möglich.

/dev Dieses Verzeichnis enthält nur Spezialdateien, sogenannte Gerätedateien. Diese stellen eine einfach zu nutzende Schnittstelle zur Hardware dar.

/etc Hier sind viele der Konfigurationsdateien untergebracht, welche die Einstellungen verschiedener Programme oder auch grundlegende Systeminformationen enthalten.

/home In diesem Verzeichnis liegen traditionell die Heimatverzeichnisse der Benutzer des Systems.

/lib Hier befinden sich die wichtigsten Funktionsbibliotheken des Systems. Eigentlich gibt es nur eine Grundregel im Umgang mit diesem Verzeichnis: Finger weg!

/proc /proc ist eigentlich kein normales Verzeichnis, sondern stellt eine Schnittstelle zum Kernel dar. Jedes laufende Programm wird hier in einem Unterverzeichnis geführt, dessen Dateien viele Informationen z.B. über den aktuellen Programmstatus enthalten.

/root Dies ist das Heimatverzeichnis des Systemverwalters root.Es liegt traditionell im Wurzelverzeichnis, damit root auch dann auf seine Dateien (beispielsweise Diagnoseprogramme) zugreifen kann.

/sbin Ähnlich wie /bin enthält auch /sbin wichtige Programme. Diese sind jedoch hauptsächlich für den Systemverwalter gedacht, da sie Funktionen erfüllen, auf die ein normaler Benutzer keinen Zugriff hat.

/tmp Dieses Verzeichnis kann von jedem Benutzer und jedem Programm als temporäre Ablage für Dateien verwendet werden. Damit sich Benutzer nicht gegenseitig ihre Dateien löschen, ist das sogenannte Sticky-Bit dieses Verzeichnisses gesetzt.

/usr Die umfangreichste Verzeichnisstruktur des Systems. Hier liegt der größte Teil der installierten Software. Auf vielen Systemen befinden sich in und unterhalb von /usr mehr Daten als in allen anderen Dateien zusammen. Die Programmdateien sind meist in /usr/bin, die Spiele in/usr/games.

/var Unter diesem Verzeichnis werden hauptsächlich Dateien gespeichert, die sich ständig verändern. Der Name /var steht für variabel, also veränderlich.

/opt (optionale Software) Kommerzielle Software oder sehr große Programme, die nicht unmittelbar zum System gehören, wie etwa KDE, Netscape, Mozilla usw. finden hier ihren Platz.