Lösung von Übungsblatt 11

Aufgabe 1 (Virtualisierung und Emulation)

1. Was ist der Unterschied zwischen Emulation und Virtualisierung?

   Emulation bildet die komplette Hardware eines Rechnersystems nach, um ein unverändertes Betriebssystem, das für eine andere Hardwarearchitektur (CPU) ausgelegt ist, zu betreiben.

2. Nennen Sie einen Nachteil der Emulation gegenüber Virtualisierung.

   Die Entwicklung ist sehr aufwendig und die Ausführungsgeschwindigkeit ist geringer als bei Virtualisierung.

3. Wie funktioniert Partitionierung?

   Bei Partitionierung können auf den Gesamtressourcen eines Computersystems Teilsysteme definiert werden. Jedes Teilsystem kann eine lauffähige Betriebssysteminstanz enthalten. Jedes Teilsystem ist wie ein eigenständiges Computersystem verwendbar.

4. Welche Komponente eines Rechners verteilt beim Virtualisierungskonzept Partitionierung die physischen Ressourcen an die virtuellen Maschinen?

   Die Ressourcen (Prozessor, Hauptspeicher, Datenspeicher...) werden über die Firmware des Rechners verwaltet und den VMs zugeteilt.

5. Welche Art von Computer-Systemen verwendet üblicherweise Partitionierung?

   ☐ Mobiltelefone ☐ Desktop PCs ☐ Mainframes ☒ Workstations

6. Wie funktioniert Anwendungsvirtualisierung?

   Anwendungen werden unter Verwendung lokaler Ressourcen in einer virtuellen Umgebung ausgeführt, die alle Komponenten bereitstellt, die die Anwendung benötigt.

7. Nennen Sie ein Beispiel für Anwendungsvirtualisierung.

   Java Virtual Machine oder VMware ThinApp.

8. Wie funktioniert vollständige Virtualisierung?

   Vollständige Virtualisierungslösungen bieten einer VM eine vollständige, virtuelle PC-Umgebung inklusive eigenem BIOS. Jedes Gastbetriebssystem erhält eine eigene VM mit virtuellen Ressourcen (u.a. CPU, Hauptspeicher, Laufwerken, Netzwerkkarten).

9. Was ist die Aufgabe des Virtuellen Maschinen-Monitors (VMM)?

   Der VMM verteilt Hardwareressourcen an VMs.

10. Wo läuft der Virtuelle Maschinen-Monitor (VMM)?

    ☒ Der VMM läuft *hosted* als Anwendung im Host-Betriebssystem.
    ☐ Der VMM läuft *bare metal* und ersetzt das Host-Betriebssystem.

11. Können bei vollständiger Virtualisierung alle physischen Hardwareressourcen virtualisiert werden? Wenn das nicht möglich ist, nennen Sie ein Beispiel, wo es nicht geht und begründen Sie Ihre Antwort.

    Es ist nicht möglich. Ein Beispiel sind Netzwerkkarten. Netzwerkkarten sind nicht dafür ausgelegt, von mehreren Betriebssystemen gleichzeitig verwendet zu werden.

12. Wie viele Privilegienstufen enthalten x86-kompatible CPUs?

    Es gibt 4 Privilegienstufen.
    In Privilegienstufe 0 (= Kernelmodus) läuft der Betriebssystemkern.
    In Privilegienstufe 3 (= Benutzermodus) laufen die Anwendungen.

13. In welcher Privilegienstufe läuft der VMM?

    In Privilegienstufe 3.

14. In welcher Privilegienstufe laufen die VMs?

    In Privilegienstufe 1.

15. Wie greifen VMs bei vollständiger Virtualisierung auf Hardwareressourcen zu?

    Nur über den VMM.

16. Nennen Sie ein Beispiel für vollständige Virtualisierung.

    - *VMware Server, VMware Workstation und VMware Fusion.*
    - *Microsoft Virtual PC (in der Version für x86).*
    - *Parallels Desktop und Parallels Workstation.*
    - *VirtualBox.*
    - *Kernel-based Virtual Machine (KVM).*
    - *Mac-on-Linux (MoL).*

17. Wie funktioniert Paravirtualisierung?

    Es wird keine Hardware virtualisiert oder emuliert. Gast-Betriebssystemen steht keine emulierte Hardwareebene, sondern eine API zur Verfügung. Die Gast-Betriebssysteme verwenden eine abstrakte Verwaltungsschicht (⟹ Hypervisor), um auf physische Ressourcen zuzugreifen. Der Hypervisor ist ein auf ein Minimum reduziertes Metabetriebssystem. Der Hypervisor verteilt Hard-

wareressourcen unter den Gastsystemen, so wie ein Betriebssystem dieses unter den laufenden Prozessen tut.

18. Wo läuft der Hypervisor bei Paravirtualisierung?

    ☐ Der Hypervisor läuft *hosted* als Anwendung im Host-Betriebssystem.
    ☒ Der Hypervisor läuft *bare metal* und ersetzt das Host-Betriebssystem.

19. In welcher Privilegienstufe läuft der Hypervisor bei Paravirtualisierung?

    In Privilegienstufe 0 (= Kernelmodus).

20. Warum ist bei Paravirtualisierung ein Host-Betriebssystem nötig?

    Ein Host-Betriebssystem ist wegen der Gerätetreiber nötig.

21. Was ist eine unprivilegierte Domain (DomU) bei Xen?

    VMs heißen unprivilegierte Domain (DomU).

22. Was ist die Domain 0 (Dom0) bei Xen?

    Der Hypervisor ersetzt das Host-Betriebssystem. Die Entwickler können aber nicht alle Treiber selbst schreiben und pflegen. Darum startet der Hypervisor eine (Linux-)Instanz mit ihren Treibern und leiht sich diese Treiber. Diese spezielle Instanz heißt Domain0 (Dom0).

23. Nennen Sie einen Nachteil der Paravirtualisierung.

    Kernel der Gast-Betriebssysteme müssen für den Betrieb im paravirtualisierten Kontext angepasst sein.

24. Wie wurden die Privilegienstufen x86-kompatibler CPUs verändert, um Hardware-Virtualisierung zu realisieren?

    Eine neue Privilegienstufe ($\Longrightarrow$ Privilegienstufe -1) für den Hypervisor ist hinzugefügt.

25. Nennen Sie einen Vorteil von Hardware-Virtualisierung.

    Unveränderte Betriebssysteme können als Gast-Systeme ausgeführt werden.

26. Wie funktioniert Betriebssystem-Virtualisierung (Container/Jails)?

    Unter ein und demselben Kernel laufen mehrere voneinander abgeschottete identische Systemumgebungen.

27. Nennen Sie einen Nachteil der Betriebssystem-Virtualisierung (Container/Jails).

Alle virtuellen Umgebungen nutzen den gleichen Kernel. Es werden nur unabhängige Instanzen eines Betriebssystems gestartet. Verschiedene Betriebssysteme können nicht gleichzeitig verwendet werden.

28. Nennen Sie ein Beispiel für Betriebssystem-Virtualisierung (Container/Jails).

    - SUN/Oracle Solaris
    - OpenVZ für Linux
    - Linux-VServer
    - FreeBSD Jails
    - Virtuozzo (kommerzielle Variante von OpenVZ)
    - FreeVPS

29. Wie funktioniert Speichervirtualisierung?

    *Speicher wird in Form virtueller Laufwerke (*Volumes*) den Benutzern zur Verfügung gestellt. Logischer Speicher wird vom physischen Speicher getrennt.*

30. Wie funktioniert Netzwerkvirtualisierungvia Virtual Local Area Networks (VLAN)?

    Verteilt aufgestellte Geräte können via VLAN in einem einzigen virtuellen (logischen) Netzwerk zusammengefasst werden. VLANs trennen physische Netze in logische Teilnetze (Overlay-Netze). VLAN-fähige Switches leiten Pakete eines VLAN nicht in andere VLANs weiter. Ein VLAN ist ein nach außen isoliertes Netz über bestehende Netze.

Aufgabe 2 (Shell-Skripte, Schleifen)

1. Schreiben Sie ein Shell-Skript, das mit Schleifen folgende Ausgabe erzeugt:

```
1
22
333
4444
55555
```

```
#!/bin/bash
#
# Skript: schleifen_beispiel1.bat
#
# Aufruf zum Beispiel: .schleifen_beispiel1.bat 5
# $1 enthält das erste Kommandozeilenargument

i=0
while [ $i -lt $1 ]    # Schleife für die Zeilen
do
  i=`expr $i + 1`

```

```
13   j=0
14   while [ $j -lt $i ]  # Schleife für die Spalten
15   do
16     j=`expr $j + 1`
17     echo -n "$i"
18   done
19
20   echo ""              # Newline am Zeilenende
21 done
```

2. Schreiben Sie ein Shell-Skript, das mit Schleifen folgende Ausgabe erzeugt:

```
1
12
123
1234
12345
```

```
1  #!/bin/bash
2  #
3  # Skript: schleifen_beispiel2.bat
4  #
5  # Aufruf zum Beispiel: .schleifen_beispiel2.bat 5
6  # $1 enthält das erste Kommandozeilenargument
7
8  i=0
9  while [ $i -lt $1 ]    # Schleife für die Zeilen
10 do
11   i=`expr $i + 1`
12
13   j=0
14   while [ $j -lt $i ]  # Schleife für die Spalten
15   do
16     j=`expr $j + 1`
17     echo -n "$j"
18   done
19
20   echo ""              # Newline am Zeilenende
21 done
```

3. Schreiben Sie ein Shell-Skript, das mit Schleifen folgende Ausgabe erzeugt:

```
|_
| |_
| | |_
| | | |_
| | | | |_
```

```
1  #!/bin/bash
2  #
3  # Skript: schleifen_beispiel3.bat
4  #
5  # Aufruf zum Beispiel: .schleifen_beispiel3.bat 5
```

```
6  # $1 enthält das erste Kommandozeilenargument
7
8  i=0
9  while [ $i -lt $1 ]     # Schleife für die Zeilen
10 do
11   i=`expr $i + 1`
12
13   j=0
14   while [ $j -lt $i ]   # Schleife für die Spalten
15   do
16     j=`expr $j + 1`
17     echo -n " |"        # Senkrechten Strich ausgeben
18   done
19
20   echo "_"              # Horizontalen Strich ausgeben
21 done
```

4. Schreiben Sie ein Shell-Skript, das mit Schleifen folgende Ausgabe erzeugt:

```
*
**
***
****
*****
```

```
1  #!/bin/bash
2  #
3  # Skript: schleifen_beispiel4.bat
4  #
5  # Aufruf zum Beispiel: .schleifen_beispiel4.bat 5
6  # $1 enthält das erste Kommandozeilenargument
7
8  i=0
9  while [ $i -lt $1 ]     # Schleife für die Zeilen
10 do
11   i=`expr $i + 1`
12
13   j=0
14   while [ $j -lt $i ]   # Schleife für die Spalten
15   do
16     j=`expr $j + 1`
17     echo -n "*"
18   done
19
20   echo ""               # Newline am Zeilenende
21 done
```

5. Schreiben Sie ein Shell-Skript, das mit Schleifen folgende Ausgabe erzeugt:

```
*
**
***
****
```

```
*****
*****
****
***
**
*
```

```
#!/bin/bash
#
# Skript: schleifen_beispiel5.bat
#
# Aufruf zum Beispiel: .schleifen_beispiel5.bat 5
# $1 enthält das erste Kommandozeilenargument

i=0
while [ $i -lt $1 ]     # Schleife für die Zeilen
do
  i=`expr $i + 1`

  j=0
  while [ $j -lt $i ]  # Schleife für die Spalten
  do
    j=`expr $j + 1`
    echo -n "*"
  done

  echo ""               # Newline am Zeilenende
done

i=$1
while [ $i -gt 0 ]      # Schleife für die Zeilen
do
  j=0
  while [ $j -lt $i ]  # Schleife für die Spalten
  do
    j=`expr $j + 1`
    echo -n "*"
  done

  i=`expr $i - 1`
  echo ""               # Newline am Zeilenende
done
```

6. Schreiben Sie ein Shell-Skript, das mit Schleifen folgende Ausgabe erzeugt:

```
    *
   ***
  *****
 *******
*********
```

```
#!/bin/bash
#
# Skript: schleifen_beispiel6.bat
```

```
4  #
5  # Aufruf zum Beispiel: .schleifen_beispiel6.bat 5
6  # $1 enthält das erste Kommandozeilenargument
7
8  i=0                     # Zählvariable für die Zeilen
9  while [ $i -lt $1 ]     # Schleife für die Zeilen
10 do
11   i=`expr $i + 1`       # j=j+1 => Nächste Zeile
12   grenze_li=`expr $1 - $i + 1` # Position erster *
13   grenze_re=`expr $1 + $i - 1` # Position letzter *
14
15   # Variable $s leer initialisieren.
16   # Das legt den Abstand zum Rand fest
17   s=''
18
19   j=0                   # Zählvariable für die Spalten
20
21   # Breite der Zeilen ist 2 * $1
22   # $1 = Anzahl der Zeilen
23   breite=`expr 2 \* $1`
24
25   while [ $j -lt $breite ]  # Schleife für die Spalten
26   do
27     # Überprüfen, ob $j im Bereich der *-Positionen liegt
28     if [ $j -ge $grenze_li ] && [ $j -le $grenze_re ]
29     then
30       s=$s'*'           # An $s einen * anhängen
31     else
32       s=$s' '           # An $s ein Leerzeichen anhängen
33     fi
34
35     j=`expr $j + 1`     # j=j+1 => Nächste Spalte
36   done
37
38   echo -e "$s"          # Komplette Zeile ausgeben
39 done
```