Solution of Exercise Sheet 10

Exercise 1 (Virtualization and Emulation)

1. Explain the difference between emulation and virtualization.

   Emulation simulates the entire hardware of a computer system, for running an unmodified operating system designed for a different hardware architecture (CPU).

2. Name a drawback of emulation against virtualization.

   The development is very expensive and challenging. The Performance is low compared with virtualization.

3. Explain how partitioning works.

   When partitioning is used, the total amount of resources can be split to create subsystems of a computer system. Each subsystem may contain an executable operating system instance. Each subsystem can be used like an independent computer system.

4. Name the component of a computer that distributes the physical resources to the virtual machines in the partitioning virtualization concept.

   The resources (CPU, main memory, storage,. . .) are managed by the firmware of the computer and assigned to the VMs.

5. Mark the sort of computer systems that usually implement the partitioning virtualization concept.

   ☐ Mobiltelefone ☐ Desktop PCs ☐ Mainframes ☒ Workstations

6. Explain how application virtualization works.

   Applications are executed inside a virtual environment, which uses local resources and provides all the components the application needs.

7. Name an example for application virtualization.

   Java Virtual Machine or VMware ThinApp.

8. Explain how full virtualization works.

   Full virtualization software solutions offer each VM a complete virtual PC environment, including an own BIOS. Each guest operating system gets its own VM with virtual resources (e.g. CPU, main memory, storage devices, network adapters).

9. Explain the function of the Virtual Machine Monitor (VMM).

   The VMM distributes hardware resources to VMs.

10. Mark the correct answer that explains where the Virtual Machine Monitor (VMM) runs.

    ☒ The VMM runs *hosted* as an application in the host operating system.
    ☐ The VMM runs *bare metal* and replaces the host operating system.

11. Can all physical hardware resources be virtualized when full virtualization is used? If this is not possible, give an example where it does not work and explain your answer.

    This is not possible. Some hardware components like network adapters are emulated, because they are not designed for the concurrent access by multiple operating systems.

12. Give the number of privilege levels a x86-compatible CPU contains.

    x86-compatible CPUs implement 4 privilege levels.
    In ring 0 (= kernel mode) runs the kernel.
    In ring 3 (= user mode) run the applications.

13. Give the privilege level number where the VMM runs.

    The VMM runs in privilege level 3.

14. Give the privilege level number where the VMs run.

    The VMs run in privilege level 1.

15. Explain how VMs can access hardware resources when using full virtualization.

    Access to hardware resources is only possible via the VMM.

16. Name an example of a full virtualization implementation.

    - *VirtualBox*
    - *VMware Server, VMware Workstation, and VMware Fusion*
    - *Parallels Desktop and Parallels Workstation*
    - *Kernel-based Virtual Machine (KVM)*

17. Explain how paravirtualization works.

    No hardware is virtualized or emulated. Guest operating systems use an abstract management layer ($\Longrightarrow$ hypervisor) to access the physical resources. The hypervisor is a meta operating system, which is reduced to a minimum. The hypervisor distributes hardware resources among the guest systems, the same way, an operating system would distribute hardware resources among running processes.

18. Explain where the hypervisor runs when using paravirtualization.

    ☐ The hypervisor runs *hosted* as an application in the host operating system.
    ☒ The hypervisor runs *bare metal* and replaces the host operating system.

19. Give the privilege level number where the hypervisor runs when using paravirtualization.

    The hypervisor in paravirtualization runs in privilege level 0 (= kernel mode).

20. Explain why a host operating system is required for using paravirtualization.

    The hypervisor must include all required device drivers or as alternative, a guest operating system instance that includes the required device drivers is mandatory.

21. Explain what an unprivileged domain (DomU) is when using Xen.

    VMs are called unprivileged domain (DomU).

22. Explain what a Domain 0 (Dom0) is when using Xen.

    The hypervisor replaces the host operating system. But the developers cannot develop all drivers from scratch and maintain them. Therefore, the hypervisor launches an (Linux) instance with its drivers and borrows them. This instance is called Domain0 (Dom0).

23. Name a drawback of paravirtualization.

    Kernels of guest operating systems must be modified (adapted) for operation in the paravirtualized context.

24. Explain how the privilege levels of x86-compatible CPUs have been modified for providing hardware virtualization.

    A new privilege level ($\Longrightarrow$ privilege level -1, called root mode) for the hypervisor is added.

25. Name an advantage of hardware virtualization.

    Unmodified operating systems can be used as guest systems The solutions from Intel and AMD are similar but incompatible.

26. Explain how operating system-level virtualization (containers/jails) works.

    Under a single kernel, multiple identical, isolated system environments are executed. No additional operating system is started. Isolated runtime environments are created. All running applications use the same kernel.

27. Name a drawback of operating system-level virtualization (containers/jails).

    All virtual environments use the same kernel. Only independent instances of

the same operating system are started. It is impossible to start different operating systems at the same time.

28. Name an example of an operating system-level virtualization (containers/jails) implementation.

    - *SUN/Oracle Solaris*
    - *OpenVZ for Linux*
    - *Linux-VServer*
    - *FreeBSD Jails*
    - *Parallels Virtuozzo (commercial version of OpenVZ)*
    - *FreeVPS*
    - *Docker*
    - *chroot*

29. Explain how storage virtualization works.

    Storage is provided to users in form of virtual drives (volumes). Logical storage is separated from physical storage.

30. Explain how network virtualization via Virtual Local Area Networks (VLAN) works.

    Distributed devices can be combined via VLAN in a single virtual (logical) network. VLANs separate physical networks into logical subnets (overlay networks). VLAN-capable switches do not forward packets of a VLAN into other VLANs. A VLAN is a network, over existing networks, which is isolated from the outside.

Exercise 2 (Shell Scripts, Loops)

1. Program a shell script, which generates with loops this output:

```
1
22
333
4444
55555
```

```
1  #!/bin/bash
2  #
3  # Skript: schleifen_beispiel1.bat
4  #
5  # Aufruf zum Beispiel: .schleifen_beispiel1.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 $j + 1`
17     echo -n "$i"
18   done
19
20   echo ""              # Newline am Zeilenende
21 done
```

2. Program a shell script, which generates with loops this output:

```
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. Program a shell script, which generates with loops this output:

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

```
1  #!/bin/bash
2  #
3  # Skript: schleifen_beispiel3.bat
```

```
#
# Aufruf zum Beispiel: .schleifen_beispiel3.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 " |"        # Senkrechten Strich ausgeben
  done

  echo "_"             # Horizontalen Strich ausgeben
done
```

4. Program a shell script, which generates with loops this output:

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

```
#!/bin/bash
#
# Skript: schleifen_beispiel4.bat
#
# Aufruf zum Beispiel: .schleifen_beispiel4.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
```

5. Program a shell script, which generates with loops this output:

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

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

```
#!/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. Program a shell script, which generates with loops this output:

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

```
#!/bin/bash
#
```

```
3  # 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
```