Solution of Exercise Sheet 2

Exercise 1 (Classifications of Operating Systems)

1. At any given moment, only a single program can be executed. Give the technical term for this operation mode.

   Singletasking.

2. Name one singletasking operating system. *MS-DOS, Palm OS*

3. Name one multitasking operating system. *Linux/UNIX, MacOS X, Server editions of the Windows NT family, MacOS 8x/9x, AmigaOS, Risc OS, OS/2, Windows 3x/95/98, BeOS*

4. Name one single-user operating system. *MS-DOS, Palm OS*

5. Name one multi-user operating system. *Linux/UNIX, MacOS X, Server editions of the Windows NT family*

6. Describe what half multi-user operating systems are.

   Different users can work with the system only one after the other, but the data and processes of the different users are protected from each other.

7. Describe the difference between 8 bit, 16 bit, 32 bit, and 64 bit operating systems.

   The bit number indicates the memory address length, with which the operating system works internally.

8. Name two essential criteria of real-time operating systems.

   Response time (short latency) and deadlines must be met.

9. Name the two types of real-time operating systems.

   Hard real-time operating systems and soft real-time operating systems.

10. Name four typical application areas of hard real-time operating systems and assign each application area to one of the categories of subtask 9.

    Typical application areas of real-time operating systems: Welding robot, reactor control, Anti-lock braking system (ABS), aircraft flight control, monitoring systems of an intensive care unit,...

    Typical applications of soft real-time operating systems: Telephone system, parking ticket vending machine, ticket machine, multimedia applications such as audio/video on demand,...

11. Describe the structure of a thin kernel.

    The operating system kernel itself runs as a process with lowest priority. The real-time kernel does the scheduling. Real-time processes have the highest priority $\Longrightarrow$ minimum reaction time (latency).

12. Describe the structure of a nano kernel.

    In addition to the real-time kernel, any number of kernels of other operating systems may be executed.

13. Describe the structure of a monolithic kernel.

    Monolithic kernels contain functions for...

    - memory management
    - process management
    - interprocess communication
    - hardware management (drivers)
    - file systems

    Outside the kernel are the user processes.

14. Describe the structure of a microkernel.

    The kernel contains only...

    - essential functions for memory management and process management
    - functions for process synchronization and interprocess communication
    - essential drivers (e.g. for system start)

    Device drivers, file systems, and services (servers) are located outside the kernel and run equal to the user applications in user mode.

15. Describe the structure of a hybrid kernel.

    Hybrid kernels are a tradeoff between monolithic kernels and microkernels. They contain some components for performance reasons, which are never located inside microkernels. It is not specified which additional components are located inside hybrid kernels.

16. GNU HURD implements a...

    ☐ monolithic kernel ☒ microkernel ☐ hybrid kernel

17. Linux implements a...

    ☒ monolithic kernel ☐ microkernel ☐ hybrid kernel

18. MacOS X implements a...

    ☐ monolithic kernel ☐ microkernel ☒ hybrid kernel

19. Windows NT4/Vista/XP/7/8/10 implements a...

    ☐ monolithic kernel ☐ microkernel ☒ hybrid kernel

20. Name one advantage and one drawback of monolithic kernels.

    - *Advantages:*
      - *Fewer context switching $\Longrightarrow$ better performance*
      - *Grown stability*
    - *Drawbacks:*
      - *Crashed components cannot be started separately in the kernel and may cause the entire system to crash*
      - *Kernel extensions cause a high development effort, because for each compilation of the extension, the complete kernel needs to be recompiled*

21. Name one advantage and one drawback of microkernels.

    - *Advantages:*
      - *Components can be exchanged easily*
      - *Best stability and security in theory, because fewer functions run in kernel mode*
    - *Drawbacks:*
      - *Slower because of more context switches*
      - *Development of a new (micro)kernel is a complex task*

22. Name one advantage and one drawback of hybrid kernels.

    - *Advantages:*
      - *Better performance as with microkernels (because fewer context switching)*
      - *The stability is (theoretically) better as with monolithic kernels*
    - *Drawbacks:*
      - *Development of a new (hybrid)kernel is a complex task*

23. Your colleague recommends you to relocate frequently used server daemons, such as web server, email server, SSH server and FTP server, from user mode to kernel mode. How do you feel about this idea? Give reasons for your answer. Explain a benefit and one drawback.

    One advantage would be that the operating system and the server daemons overall perform better, because in the scenario described, fewer context switches between user mode and kernel mode are required.

    More serious, however, is the resulting drawback. There is a security risk. Complex software such as server daemons should not run in kernel mode. Bugs in the server daemons may cause system crashes or allow attackers to gain complete control of the system.

24. Describe what Single System Image means.

    Users and their applications do not know that the services they use run on multiple computers.

Exercise 2 (Starting the Operating System)

1. Name one advantage and one drawback of the autonomous subsystems in modern PCs. Examples include the Intel Management Engine and AMD Platform Security Processor.

   Advantage: They allow a computer to be monitored and woken up over the network (Wake-on-LAN) and enable remote administration (remote management).

   Drawback: These subsystems are not fully documented (quasi-secret). These subsystems always run when sufficient energy is available. They can access all hardware resources, including main memory, I/O interfaces, interfaces and bus systems, and network interfaces. It is an uncontrollable computer within the computer whose exact range of functions is unclear. Such a system is a major potential security risk.

2. Describe the purpose of the firmware in the computer.

   The firmware performs the Power-On Self-Test (POST). Among other things, this checks the correct functioning of the processor, the buffer memory (cache), and the main memory. After the computer has started and the self-test has been completed, the firmware searches for the first boot device (boot drive) and starts the boot loader.

3. Give the name of the firmware in classic computers from the early 1980s to the late 2000s.

   BIOS (Basic Input/Output System)

4. Give the name of the firmware in modern computers.

   UEFI (Unified Extensible Firmware Interface)

5. Explain what the boot loader is.

   The boot loader is a program that loads the operating system kernel into the main memory when the operating system is started. It also loads the initial RAM disk (`initrd`) or the initial RAM file system (`initramfs`).

6. Explain where the boot loader is stored.

   When using a classic PC partition table, the boot loader is stored in the 512-

byte large master boot record (MBR) at the very beginning of the drive. When using a GUID partition table (GPT), the boot loader is stored in the ESP (EFI System Partition).

7. Explain why an initial RAM disk (`initrd`) or than initial RAM file system (`initramfs`) are used.

   The temporary root file system loaded by `initrd` or `initramfs` implements a minimum Linux environment in the main memory. Its primary purpose is to provide the kernel with additional device drivers, file system drivers, and programs to load the operating system's real root file system into memory.

8. Describe the task of a `getty` process.

   A `getty` process allows text-based user login via a (virtual) console.

9. Specify how many `getty` processes the operating system starts.

   The operating system starts a separate instance of the `getty` process for each virtual console (TTY1 to TTY6).

Exercise 3 (Basic Linux/UNIX commands)

Which command is used to...

1. check the man pages?

   man

2. print out the present working directory in the shell?

   pwd

3. create a new directory?

   mkdir

4. navigate to a directory?

   cd

5. print out the content of a directory in the shell?

   ls

6. create an empty file?

   touch

7. try to determine the content of a file?

file

8. concatenate the content of files with other files and can also be used to print out the content of a file?

   cat

9. print out lines from the end of a file in the shell?

   tail

10. print out lines from the beginning of a file in the shell?

    head

11. copy files or directories to a different location?

    cp

12. move files or directories to a different location?

    mv

13. delete files or directories?

    rm

14. delete an empty directory?

    rmdir

15. place a string in the shell?

    echo

16. modify the permissions of the file or directory?

    chmod

17. change the password of a user?

    passwd

18. terminate a session (and thus shell) and allows to specify the return value of the shell script?

    exit

19. reboot the system?

    reboot or alternatively shutdown

20. shut the system down?

halt or alternatively shutdown

21. create a new user?

    adduser or alternatively useradd

22. delete a user?

    deluser or alternatively userdel

23. modify a user?

    usermod

24. print out the group memberships of a user?

    groups

25. create a new group?

    groupadd

26. delete a group?

    groupdel

27. change a group?

    groupmod

28. change the user ($\Longrightarrow$ ownership) which is associated with a file or directory?

    chown

29. change the group which is associated with a file or directory?

    chgrp

30. create a link?

    ln

31. search a file for lines which contain a search pattern?

    grep

32. print out a list of running processes in the shell?

    ps

33. bring a process running in the background of the shell, into foreground?

    fg

34. bring a process into the background of the shell?

    bg

35. kill (terminate) a process?

    kill

36. kill (terminate) a group of processes?

    killall

37. specify the priority of a new process?

    nice

38. modify the priority of an existing process?

    renice

39. print out the process tree in the shell?

    pstree

Exercise 4 (Permissions / Access Rights)

The source of this tutorial is:
`http://www.ws.afnog.org/afnog2012/unix-intro/presos/permissions-exercises.pdf`

Notes

- Commands preceded with `$` imply that you should execute the command as a general user and not as root.
- Commands preceded with `#` imply that you should be working as root with `sudo`

REFERENCE

If you look at files in a directory using `ls -al` you will see the permissions for each file and directory. Here is an example:

```
drwxrwxr-x    3 bnc   bnc      4096 Feb 25 09:49 directory
-rwxr--r--   12 bnc   bnc      4096 Feb 16 05:02 file
```

The left column is important. You can view it like this:

```
Type User Group Other Links Owner Group  Size  Date   Hour  Name
```

```
=================================================================
d    rwx  rwx   r-x   3     bnc   bnc    4096  Feb 25 09:49 directory
-    rwx  r     r     12    bnc   bnc    4096  Feb 16 05:02 file
```

The directory has `r` (read), `w` (write), `x` (execute) access for the User (= Owner) and Group. For Other it has `r` (read) and `x` (execute) access.

The file has `r` (read), `w` (write), `x` (execute) access for User and `r` (read) only access for everyone else (Group and Other).

You can change permissions with the `chmod` command. `chmod` uses a base eight (octal) system to configure permissions. Or, you can use an alternate form to specify permissions by column (User/Group/Other) at a time.

Permissions have values like this:

```
Letter   Permission   Value
============================
r        read         4
w        write        2
x        execute      1
-        none         0
```

Thus you can give permissions to a file using the sum of the values for each permission you wish to give for each column. Here is an example:

```
Letter   Permission                 Value
=======================================
---      none                      0
--x      execute                   1
-w-      write only (rarely used)  2
-wx      write and execute (rare)  3
r--      read only                 4
r-x      read and execute          5
rw-      read and write            6
rwx      read, write, and execute  7
```

This is just one column. Since we have three areas of permissions (User, Group, Other), it looks like this, if you want to specify all 3 sets:

```
Permissions  Numeric     Description
             equivalent

-rw-------   600         User has read & write permission.
```

```
-rw-r--r--   644        User has read & write permission.
                        Group and Other have read permission.
-rw-rw-rw-   666        Everyone (User, Group, Other) has
                        read & write permission (dangerous?)
-rwx------   700        User has read, write, execute permission.
-rwxr-xr-x   755        User has read, write, execute permission.
                        Rest of the world (Other) has read & execute
                        permission (typical for web pages or 644).
-rwxrwxrwx   777        Everyone has full access (read, write,
                        execute).
-rwx--x--x   711        User has read, write, execute permission.
                        Group and world have execute permission.
drwx------   700        User only has access to this directory.
                        Directories require execute permission to
                        access.
drwxr-xr-x   755        User has full access to directory. Everyone
                        else can see the directory.
drwx--x--x   711        Everyone can list files in the directory,
                        but Group and Other need to know a filename
                        to do this.
```

1.) CHANGING FILE PERMISSIONS

If you are logged in as the root user on your machine please do the following to become a normal user.

```
# exit
```

Your prompt should change and now include a `$` sign.

```
$
```

Please check your username with the command `whoami`:

```
$ whoami
```

Please create a file and set permissions of the file in various ways.

```
$ cd
$ echo "test file" > working.txt
$ chmod 444 working.txt
```

What does that look like?

```
$ ls -lah working.txt
```

Because the file has no write permission for the owner, the owner can still change the file's permissions. This way, the owner can change the permissions of the file at any time to have write access again.

```
$ chmod 644 working.txt
```

Or, you can do this by using this form of `chmod`:

```
$ chmod u+w working.txt
```

Note: When you type these commands you should be able to use the tab key for command completion once you've typed the `w` in the file name `working.txt`. This will save you a lot of time. It's highly recommended!

To remove the read permission of a file for the user you would do

```
$ chmod u-r working.txt
```

Or, you can do something like this:

```
$ chmod 344 working.txt
```

You probably noticed that you can use the `-` (minus) sign to remove permissions from a file. Try reading your file:

```
$ cat working.txt
```

Now you cannot read your own file. Please make the file readable again by you with the `chmod` command. Look at your reference at the start of these tutorial to figure out what permissions are required.

2.) PROGRAM EXECUTION, PRIVILEGES AND SUDO

As a general user you can see that there is a file called `/etc/shadow`:

```
$ ls /etc/shadow
```

But, you cannot see its contents:

```
$ less /etc/shadow
```

Figure out the permissions of this file.

As a general user, however, you can see the content of the `/etc/shadow` file if you do the following:

```
$ sudo less /etc/shadow
```

What is `sudo`? Read about it:

```
$ man sudo
```