Lösung von Übungsblatt 2

Aufgabe 1 (Klassifikationen von Betriebssystemen)

1. Zu jedem Zeitpunkt kann nur ein einziges Programm laufen. Nennen Sie den passenden Fachbegriff für diese Betriebsart.

   Einzelprogrammbetrieb (Singletasking).

2. Nennen Sie ein Beispiel für ein Singletasking-Betriebssystem.

   MS-DOS, Palm OS

3. Nennen Sie ein Beispiel für ein Multitasking-Betriebssystem.

   Linux/UNIX, MacOS X, Server-Versionen der Windows NT-Familie, MacOS 8x/9x, AmigaOS, Risc OS, OS/2, Windows 3x/95/98, BeOS

4. Nennen Sie ein Beispiel für ein Single-User-Betriebssystem.

   MS-DOS, Palm OS

5. Nennen Sie ein Beispiel für ein Multi-User-Betriebssystem.

   Linux/UNIX, MacOS X, Server-Versionen der Windows NT-Familie

6. Beschreiben Sie, was man man unter halben Multi-User-Betriebssystemen versteht.

   Verschiedene Benutzer können nur nacheinander am System arbeiten, aber die Daten und Prozesse der Benutzer sind voreinander geschützt.

7. Beschreiben Sie den Unterschied zwischen 8 Bit-, 16 Bit-, 32 Bit- und 64 Bit-Betriebssystemen.

   Die Bit-Zahl gibt die Länge der Speicheradressen an, mit denen das Betriebssystem intern arbeitet.

8. Nennen Sie die wesentlichen Kriterien von Echtzeitbetriebssystemen.

   Reaktionszeit (geringe Latenzzeit) und das Einhalten von Deadlines.

9. Es gibt zwei Arten von Echtzeitbetriebssystemen. Geben Sie deren Namen an.

   Harte Echtzeitsysteme und weiche Echtzeitsysteme.

10. Nennen Sie vier typische Einsatzgebiete von Echtzeitbetriebssystemen und ordnen Sie jedes Einsatzgebiet einer der Kategorien aus Teilaufgabe 9 zu.

Typische Einsatzgebiete harter Echtzeitbetriebssysteme: Schweißroboter, Reaktorsteuerung, ABS, Flugzeugsteuerung, Überwachungssysteme auf der Intensivstation,...

Typische Einsatzgebiete weicher Echtzeitbetriebssysteme: Telefonanlage, Parkschein- oder Fahrkartenautomat, Multimedia-Anwendungen wie Audio/Video on Demand,...

11. Beschreiben Sie den Aufbau eines Thin-Kernels.

    Der Betriebssystemkern selbst läuft als Prozess mit niedrigster Priorität. Der Echtzeit-Kernel übernimmt das Scheduling. Echtzeit-Prozesse haben die höchste Priorität $\Longrightarrow$ minimale Latenzzeiten.

12. Beschreiben Sie den Aufbau eines Nano-Kernels.

    Neben dem Echtzeit-Kernel kann eine beliebige Anzahl anderer Betriebssystem-Kernel laufen.

13. Beschreiben Sie den Aufbau eines monolithischen Kernels.

    Monolithische Kerne enthalten...

    - *Funktionen für Speicherverwaltung*
    - *Funktionen für Prozessverwaltung*
    - *Funktionen für Prozesskommunikation*
    - *Gerätetreiber*
    - *Dateisysteme-Treiber*

    Außerhalb des Kerns befinden sich die Benutzerprozesse.

14. Beschreiben Sie den Aufbau eines minimalen Kerns (Mikrokernels).

    Im Kern befinden sich üblicherweise nur:

    - *Notwendigste Funktionen zur Speicher- und Prozessverwaltung*
    - *Funktionen zur Synchronisation und Interprozesskommunikation*
    - *Notwendigste Treiber (z.B. zum Systemstart)*

    Gerätetreiber, Dateisysteme und Dienste (Server), befinden sich außerhalb des Kerns und laufen wie die Anwendungsprogramme im Benutzermodus

15. Beschreiben Sie den Aufbau eines hybriden Kernels.

    Hybride Kerne sind ein Kompromiss zwischen monolithischen Kernen und minimalen Kernen. Die enthalten aus Geschwindigkeitsgründen Komponenten, die bei minimalen Kernen außerhalb des Kerns liegen. Es ist nicht festgelegt, welche Komponenten bei Systemen mit hybriden Kernen zusätzlich in den Kernel einkompiliert sind.

16. GNU HURD verwendet einen...

☐ monolithischen Kern
☒ minimalen Kern (Mikrokernel)
☐ hybriden Kern

17. Linux verwendet einen...

☒ monolithischen Kern
☐ minimalen Kern (Mikrokernel)
☐ hybriden Kern

18. MacOS X verwendet einen...

☐ monolithischen Kern
☐ minimalen Kern (Mikrokernel)
☒ hybriden Kern

19. Windows NT4/Vista/XP/7/8/10 verwendet einen...

☐ monolithischen Kern
☐ minimalen Kern (Mikrokernel)
☒ hybriden Kern

20. Nennen Sie einen Vorteil und einen Nachteil von monolithischen Kernen.

- *Vorteile:*
  - *Weniger Kontextwechsel als Mikrokernel $\Longrightarrow$ höhere Geschwindigkeit*
  - *Gewachsene Stabilität*
- *Nachteile:*
  - *Abgestürzte Komponenten können im Kernel nicht separat neu gestartet werden und das gesamte System nach sich ziehen*
  - *Hoher Entwicklungsaufwand für Erweiterungen am Kern, da dieser bei jedem Kompilieren komplett neu übersetzt werden muss*

21. Nennen Sie einen Vorteil und einen Nachteil von minimalen Kernen (Mikrokerneln).

- *Vorteile:*
  - *Einfache Austauschbarkeit der Komponenten*
  - *Beste Stabilität und Sicherheit (in der Theorie!), weil weniger Funktionen im Kernelmodus laufen*
- *Nachteile:*
  - *Langsamer wegen der größeren Zahl von Kontextwechseln*
  - *Entwicklung eines neuen (Mikro-)kernels ist eine komplexe Aufgabe*

22. Nennen Sie einen Vorteil und einen Nachteil von hybriden Kernen.

- *Vorteile:*
  - *Höhere Geschwindigkeit als minimale Kerne (da weniger Kontextwechsel)*
  - *Bessere Stabilität (in der Theorie!) als monolithische Kerne*

- *Nachteile:*
  - *Entwicklung eines neuen (Hybrid-)kernels ist eine komplexe Aufgabe*

23. Ein Kollege empfiehlt Ihnen häufig verwendete Server-Dienste wie z.B. Web-Server, Email-Server, SSH-Server und FTP-Server vom Benutzermodus in den Kernelmodus zu verlagern. Wie stehen Sie zu dieser Idee? Begründen Sie Ihre Antwort. Nennen Sie hierfür einen Vorteil und einen Nachteil.

    Von Vorteil wäre, das das Betriebssystem und die Server-Dienste insgesamt schneller arbeiten, weil im beschriebenen Szenario weniger Moduswechsel zwischen Benutzermodus und Kernelmodus nötig sind.

    Gravierender ist aber der entstehende Nachteil. Es liegt ein Sicherheitsrisiko vor. Komplexe Software wie Server-Dienste sollten nicht im Kernelmodus laufen. Softwarefehler in den Server-Diensten könnten zu Systemabstürzen oder zur vollständigen Kontrollübernahme durch Angreifer führen.

24. Beschreiben Sie was ein Single System Image ist.

    Die Benutzer und deren Anwendungen wissen nicht, dass die von ihnen in Anspruch genommenen Dienste auf mehreren Rechnern laufen.

Aufgabe 2 (Start des Betriebssystems)

1. Nennen Sie einen Vorteil und einen Nachteil der autonomen Subsysteme in modernen PCs. Beispiele hierfür sind: Intel Management Engine und AMD Platform Security Processor.

    Vorteil: Sie ermöglichen ein Überwachen und Aufwecken eines Rechners über das Netzwerk (Wake-on-LAN) und realisieren Möglichkeiten zur Fernadministration (Remote-Management).

    Nachteil: Diese Subsysteme sind nicht vollständig dokumentiert (quasi geheim). Diese Subsysteme laufen immer dann, wenn ausreichend Energie zur Verfügung steht. Sie haben Zugriff auf alle Hardwareressourcen inkl. Hauptspeicher, I/O-Schnittstellen, Schnittstellen und Bussysteme sowie auf die Netzwerkschnittstellen. Es handelt sich um einen unkontrollierbaren Computer im Computer deren genauer Funktionsumfang unklar ist. Ein solches System ist ein großes potenzielles Sicherheitsrisiko.

2. Beschreiben Sie die Aufgabe der Firmware im Computer.

    Die Firmware führt den Selbsttest POST (Power-On Self-Test) durch. Dabei werden u.a. die korrekte Funktion des Prozessors, des Pufferspeichers (Cache) und das Hauptspeichers überprüft. Nach dem Start des Computers und dem erfolgreichen Selbsttest sucht die Firmware nach dem ersten Bootgerät (Bootlaufwerk) und startet den Bootloader.

3. Nennen Sie den Namen der Firmware in klassischen Computern von Anfang der 1980er Jahre bis Ende der 2000er Jahre.

   BIOS (Basic Input/Output System)

4. Nennen Sie den Namen der Firmware in modernen Computern.

   UEFI (Unified Extensible Firmware Interface)

5. Geben Sie an, was der Bootloaders ist.

   Der Bootloader ist ein Programm, das beim Start des Betriebssystems den Betriebssystemkern in den Hauptspeicher lädt. Zudem lädt es die initiale RAM-Disk (`initrd`) oder das initiale RAM-Dateisystem (`initramfs`).

6. Geben Sie an, wo der Bootloaders abgespeichert ist.

   Bei Nutzung einer klassischen PC-Partitionstabelle liegt der Bootloader innerhalb des 512 Bytes großen Master Boot Record (MBR) am Anfang des Laufwerks. Bei Nutzung einer GUID Partition Table (GPT) liegt der Bootloader in der ESP (EFI System Partition).

7. Geben Sie den Nutzen der initialen RAM-Disk (`initrd`) bzw. des initialen RAM-Dateisystem (`initramfs`) an.

   Das durch `initrd` oder `initramfs` geladene temporäre Root-Dateisystem realisiert eine minimale Linux-Umgebung im Arbeitsspeicher Sie dient in erster Linie dazu, dem Kern weitere Gerätetreiber, Treiber für Dateisysteme und Programme bereitzustellen, um das echte Root-Dateisystem des Betriebssystems in den Arbeitsspeicher zu laden.

8. Beschreiben Sie die Aufgabe eines `getty`-Prozesses.

   Ein `getty`-Prozess ermöglicht eine textbasierte Benutzeranmeldung über eine (virtuelle) Konsole.

9. Geben Sie an, wie viele `getty`-Prozesse das Betriebssystem startet.

   Für jede virtuellen Konsole (TTY1 bis TTY6) startet das Betriebssystem jeweils eine eigene Instanz des Prozesses `getty`.

Aufgabe 3 (Grundlegende Linux/UNIX-Kommandos)

Geben Sie ein Kommando an, um...

1. Handbuchseiten („Man Pages“) zu öffnen.

man

2. das aktuelle Verzeichnis in der Shell auszugeben.

   pwd

3. ein neues Verzeichnis zu erzeugen.

   mkdir

4. in ein Verzeichnis zu wechseln.

   cd

5. den Inhalt eines Verzeichnisses in der Shell auszugeben.

   ls

6. eine leere Datei zu erzeugen.

   touch

7. versuchen können den Inhalt einer Datei zu bestimmen.

   file

8. den Inhalt verschiedener Dateien zu verknüpfen oder den Inhalt einer Datei auszugeben.

   cat

9. Zeilen vom Ende einer Datei in der Shell auszugeben.

   tail

10. Zeilen vom Anfang einer Datei in der Shell auszugeben.

    head

11. Dateien oder Verzeichnisse an eine andere Stelle zu kopieren.

    cp

12. Dateien oder Verzeichnisse an eine andere Stelle zu verschieben.

    mv

13. Dateien oder Verzeichnisse zu löschen.

    rm

14. ein leeres Verzeichnis zu löschen.

    rmdir

15. eine Zeichenkette in der Shell auszugeben.

    echo

16. die Dateirechte von Dateien oder Verzeichnissen zu ändern.

    chmod

17. Das Password eines Benutzers zu ändern.

    passwd

18. die laufende Sitzung (und damit auch die Shell) zu beenden und den Rückgabewert eines Shell-Skripts festzulegen.

    exit

19. das System neu zu starten.

    reboot oder alternativ shutdown

20. das System auszuschalten.

    halt oder alternativ shutdown

21. einen neuen Benutzer zu erstellen.

    adduser oder alternativ useradd

22. einen Benutzer zu löschen.

    deluser

23. einen Benutzer zu ändern.

    usermod

24. die Gruppenzugehörigkeiten des Benutzers auszugeben.

    groups

25. eine neue Gruppe zu erstellen.

    groupadd or alternatively useradd

26. eine Gruppe löschen.

    groupdel or alternatively userdel

27. eine Gruppe ändern.

    groupmod

28. den Benutzer ($\Longrightarrow$ Besitzer) zu ändern, der einer Datei oder einem Verzeichnis zugeordnet ist.

    chown

29. die Gruppe ändern, die einer Datei oder einem Verzeichnis zugeordnet ist.

    chgrp

30. einen „Link“ zu erstellen.

    ln

31. eine Datei nach den Zeilen zu durchsuchen, die ein Suchmuster enthalten.

    grep

32. eine Liste der laufenden Prozesse in der Shell auszugeben.

    ps

33. einen im Hintergrund der Shell laufenden Prozess in den Vordergrund zu holen.

    fg

34. einen Prozess in den Hintergrund der Shell zu verschieben.

    bg

35. einen Prozess zu beenden.

    kill

36. eine Gruppe von Prozessen zu beenden.

    killall

37. die Priorität eines neuen Prozesses festzulegen.

    nice

38. die Priorität eines existierenden Prozesses zu ändern.

    renice

39. eine Liste der existierenden Prozesse als Baumstruktur in der Shell auszugeben.

    pstree

Aufgabe 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
```