

## 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.
2. Was versteht man unter halben Multi-User-Betriebssystemen?
3. Was ist der Unterschied zwischen 8 Bit-, 16 Bit-, 32 Bit- und 64 Bit-Betriebssystemen?
4. Was sind die wesentlichen Kriterien von Echtzeitbetriebssystemen?
5. Es gibt zwei Arten von Echtzeitbetriebssystemen. Geben Sie deren Namen an.
6. Nennen Sie vier typische Einsatzgebiete von Echtzeitbetriebssystemen und ordnen Sie jedes Einsatzgebiet einer der Kategorien aus Teilaufgabe 5 zu.
7. Beschreiben Sie den Aufbau eines Thin-Kernels.
8. Beschreiben Sie den Aufbau eines Nano-Kernels.
9. Beschreiben Sie den Aufbau eines monolithischen Kerns.
10. Beschreiben Sie den Aufbau eines minimalen Kerns (Mikrokernels).
11. Beschreiben Sie den Aufbau eines hybriden Kerns.
12. GNU HURD verwendet einen...
  - monolithischen Kern
  - minimalen Kern (Mikrokern)
  - hybriden Kern
13. Linux verwendet einen...
  - monolithischen Kern
  - minimalen Kern (Mikrokern)
  - hybriden Kern
14. MacOS X verwendet einen...
  - monolithischen Kern
  - minimalen Kern (Mikrokern)
  - hybriden Kern

15. Windows NT4/Vista/XP/7/8/10 verwendet einen...
  - monolithischen Kern
  - minimalen Kern (Mikrokern)
  - hybriden Kern
16. Nennen Sie einen Vorteil und einen Nachteil von monolithischen Kernen.
17. Nennen Sie einen Vorteil und einen Nachteil von minimalen Kernen (Mikrokernen).
18. Nennen Sie einen Vorteil und einen Nachteil von hybriden Kernen.
19. 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.
20. Was versteht man unter Single System Image?

## Aufgabe 2 (Grundlegende Kommandos)

## Linux/UNIX-

Mit welchem Kommando können Sie...

1. Handbuchseiten („Man Pages“) öffnen
2. das aktuelle Verzeichnis in der Shell ausgeben?
3. ein neues Verzeichnis erzeugen?
4. in ein Verzeichnis wechseln?
5. den Inhalt eines Verzeichnisses in der Shell ausgeben?
6. eine leere Datei erzeugen?
7. versuchen den Inhalt einer Datei zu bestimmen?
8. den Inhalt verschiedener Dateien verknüpfen oder den Inhalt einer Datei ausgeben?
9. Zeilen vom Ende einer Datei in der Shell ausgeben?
10. Zeilen vom Anfang einer Datei in der Shell ausgeben?
11. Dateien oder Verzeichnisse an eine andere Stelle kopieren?
12. Dateien oder Verzeichnisse an eine andere Stelle verschieben?

13. Dateien oder Verzeichnisse löschen?
14. ein leeres Verzeichnis löschen?
15. eine Zeichenkette in der Shell ausgeben?
16. die Dateirechte von Dateien oder Verzeichnissen ändern?
17. Das Password eines Benutzers ändern?
18. die laufende Sitzung (und damit auch die Shell) zu beenden und den Rückgabewert eines Shell-Skripts festzulegen?
19. das System neu starten?
20. das System ausschalten?
21. einen neuen Benutzer erstellen?
22. einen Benutzer löschen?
23. einen Benutzer ändern?
24. die Gruppenzugehörigkeiten des Benutzers ausgeben?
25. eine neue Gruppe erstellen?
26. eine Gruppe löschen?
27. eine Gruppe ändern?
28. den Benutzer ( $\implies$  Besitzer) ändern, der einer Datei oder einem Verzeichnis zugeordnet ist
29. die Gruppe ändern, die einer Datei oder einem Verzeichnis zugeordnet ist
30. einen „Link“ erstellen?
31. eine Datei nach den Zeilen durchsuchen, die ein Suchmuster enthalten?
32. eine Liste der laufenden Prozesse in der Shell ausgeben?
33. einen im Hintergrund der Shell laufenden Prozess in den Vordergrund holen?
34. einen Prozess in den Hintergrund der Shell verschieben?
35. einen Prozess beenden?
36. eine Gruppe von Prozessen beenden?
37. die Priorität eines neuen Prozesses festlegen?
38. die Priorität eines existierenden Prozesses ändern?

39. eine Liste der existierenden Prozesse als Baumstruktur in der Shell ausgeben?

## Aufgabe 3 (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` imply that you are executing commands on remote equipment, or within another program.

### 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 equivalent | Description   |
|-------------|--------------------|---|
| -rw-----    | 600                | User has read & write permission.   |
| -rw-r--r--  | 644                | User has read & write permission.<br>Group and Other have read permission.  |
| -rw-rw-rw-  | 666                | Everyone (User, Group, Other) has<br>read & write permission (dangerous?)   |
| -rwx-----   | 700                | User has read, write, execute permission.   |
| -rwxr-xr-x  | 755                | User has read, write, execute permission.<br>Rest of the world (Other) has read & execute<br>permission (typical for web pages or 644). |
| -rwxrwxrwx  | 777                | Everyone has full access (read, write,<br>execute).   |
| -rwx--x--x  | 711                | User has read, write, execute permission.<br>Group and world have execute permission.   |
| drwx-----   | 700                | User only has access to this directory.<br>Directories require execute permission to<br>access.   |
| drwxr-xr-x  | 755                | User has full access to directory. Everyone<br>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
```