Exercise Sheet 3

Exercise 1 (Computer Architecture)

1. Which are the two esential components of any CPU?

2. Name two other (optional) components a CPU may possess.

3. Which three digital bus systems contains each computer system according to the Von Neumann architecture?

4. Which tasks are carried out by the three digital bus systems of subtask 3?

5. What is the Front Side Bus (FSB)?

6. Which two components contains the chipset?

7. Name the tasks of the components of the chipset.

Exercise 2 (Input/Output Devices)

1. What is the fundamental distinction between character and block I/O devices?

2. Name two examples for character and block devices.

3. Name three possible ways for processes to read data from I/O devices.
   -
   -
   -

4. Name a benefit and a drawback for each possible way from subtask 3.
   -
     -
     -

-
  -
  -
-
  -
  -

Exercise 3 (Digital Data Storage)

1. Name one mechanic digital data storage.

2. Name two rotating magnetic digital data storages.

3. Name four benefits of data storage without moving parts compared with data storage with moving parts.

4. What is random access?

5. Name one non-persistent data storage.

6. The storage of computer systems is distinguished into the categories primary storage, secondary storage and tertiary storage. Which category or categories can the CPU access directly?

7. Which category or categories of subtask 6 can the CPU only access via a controller?

Exercise 4 (Write policies)

1. Name the two basic cache write policies.

2. With which cache write policy of subtask 1 may inconsistencies occur?

3. With which cache write policy of subtask 1 is the system performance lower?

4. With which cache write policy of subtask 1 are so called dirty bits used?

5. For what reason are dirty bits used?

Exercise 5 (Time-based Command Execution, Control Structures, Archiving)

1. Program a shell script, which reads two numbers as command line arguments. The script should check whether the numbers are identical, and print out the result of the check.

2. Extend the shell script in a way that if the numbers are not identical, it is checked, which one of the two numbers is the larger one. The result of the check should be printed out.

3. Program a shell script, which creates a backup of a directory of your choice. The script should create an archive file with the file extension `.tar.bz2` from the directory. The archive file should be stored in the directory `/tmp`. The name of the archive file should correspond to the following naming scheme:

   `Backup_<USERNAME>_<YEAR>_<MONTH>_<DAY>.tar.bz2`

   The fields `<USERNAME>`, `<YEAR>`, `<MONTH>` and `<DAY>` should be replaced by the current values.

4. Program a shell script, which checks if already today an archive file was created according to the naming scheme of subtask 3. The result of the check should be printed out in the shell.

5. Write two `cron` jobs. The first `cron` job should execute at 6:15 am on every day (except on weekends) the shell script from subtask 3, which creates the archive file with the backup. The second `cron` job should execute at 11:45 am on every day (except on weekends) the shell script from subtask 4, which checks, whether already today an archive file was created. The output from the shell scripts should be appended to a file `/tmp/backup.log`. If the archive file `Backup...tar.bz2` has been created successfully, this should be noted in the log file `/tmp/backup.log`. Before each new entry in the file, lines according to the following pattern (with current values) should be inserted into the log file `/tmp/backup.log`.

```
*****************************
20.11.2013 --- Time: 21:39:51
```

Exercise 6 (Shell Scripts)

1. Program a shell script, which checks for a file, which is specified as an argument, whether it exists and if it is a file, a directory, a symbolic link, a socket or a named pipe.
   - The script should print out the result of the check.

2. Extend the shell script from subtask 1 in a way that if the file, which is specified as an argument, exists, it is checked, if the file could be executed and if write access would be possible.

3. Program a shell script, which reads so long text on the command line, until it is terminated by typing `END`.
    - The script should convert the text, which is read in from the command line, to uppercase.

4. Program a shell script, which prints out the number of running processes for all logged in users.

5. Extend the shell script from subtask 4 in a way that that the output is sorted.
    - The user with most processes should stand at the beginning.

6. Program a shell script, which checks after start every 10 seconds, if a file `/tmp/lock.txt` exists.

   - Each time after the script has checked the existence of the file, it should output an appropriate message on the shell.
   - Once the file `/tmp/lock.txt` exists, the script should terminate itself.