1. Aufgabe

1. rm:
   (a) löscht Dateien, aber Standardmäßig keine Directories
   (b) /bin/rm
   (c) rm test.txt
   (d) unlink()
2. mv:
   (a) verschiebt Datein, kann auch zum umbennen genutzt werden
   (b) /bin/mv
   (c) mv test.txt /Documents/test.txt
   (d) access() und rename()
3. chmod:
   (a) aendert die Zugriffsrecht von Dateien
   (b) /bin/chmod
   (c) symbolisch: chmod +rwx file, binaer: chmod 777 file
   (d) pathconf()
4. chown:
   (a) aendert den Besitzer einer Datei
   (b) Debian: /bin/chown, FreeBSD: /usr/bin/chown
   (c) chown [user] file
   (d) munmap(), exit()
5. mkdir:
   (a) erstellt Ordner
   (b) /bin/mkdir
   (c) mkdir name
   (d) sigprocmask()
6. rmdir:
   (a) loescht Ordner
   (b) /bin/rmdir
   (c) rmdir name
   (d) execve(), mmap()
7. kill:
   (a) sendet ueber die PID Signale an Prozesse
   (b) /bin/kill
   (c) kill [PID]
   (d) lstat()
8. ln:

(a) steht fuer link, erzeugt Verknuepfung zu Datei oder Ordner

(b) /bin/ln

(c) ln

$$option$$

ZIEL

$$name_der_verknuepfung$$

(d) openat(), fstat()

9. sleep:

(a) laesst den Prozess fuer eine angegebene Zeit warten

(b) /bin/sleep

(c) sleep 0.1h

(d) fstatfs()

10. wget:

(a) Programm um Dateien aus dem Terminal von ftp- oder http-Servern zu laden

(b) /usr/local/bin/wget

(c) wget http://example.com/folder/file

(d) read(), write()

2. Aufgabe

lstat()? int lstat(const char *path, struct stat *sb);

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]) {
       struct stat st;

       if (argc != 2) {
              printf(stderr, "Usage: %s <pathname>\n", argv[0]);
              exit(EXIT_FAILURE);
       }

       if (stat(argv[1], &st) == -1) {
              perror("stat");
              exit(EXIT_FAILURE);
       }

       printf("File type:             ");

       switch (st.st_mode & S_IFMT) {
              case S_IFBLK: printf("block device\n");        break;
              case S_IFCHR: printf("character device\n");    break;
              case S_IFDIR: printf("directory\n");           break;
              case S_IFIFO: printf("FIFO/pipe\n");           break;
              case S_IFLNK: printf("symlink\n");             break;
              case S_IFREG: printf("regular file\n");        break;
              case S_IFSOCK: printf("socket\n");             break;
```

```
            default:      printf("unknown?\n");            break;
        }

        printf("I-node number:         %ld\n", (long) st.st_ino);

        printf("Mode:                  %lo (octal)\n", (unsigned long) st.st_mode);

        printf("Link count:            %ld\n", (long) st.st_nlink);
        printf("Ownership:             UID=%ld  GID=%ld\n", (long) st.st_uid, (long)
            st.st_gid);

        printf("Preferred I/O block size: %ld bytes\n", (long) st.st_blksize);
        printf("File size:             %lld bytes\n", (long long) st.st_size);
        printf("Blocks allocated:      %lld\n", (long long) st.st_blocks);

        printf("Last status change:    %s", ctime(&st.st_ctime));
        printf("Last file access:      %s", ctime(&st.st_atime));
        printf("Last file modification: %s", ctime(&st.st_mtime));

        exit(EXIT_SUCCESS);
}
```