10 Überblick über die 3. Übung

- UNIX-Kommandos
- Aufgabe 1: Warteschlange als verkettete Liste
- Infos zur Aufgabe 3: Verzeichnisse

11 UNIX-Kommandos

- Dateisystem
- Benutzer
- Prozesse
- Suchen

11.1 Dateisystem

- ls
  - `ls -l -a`
- chmod (chmod -R), chown, chgrp
- mkdir, rmdir
- rm
  - `rm -r <dir>`

11.2 Benutzer

- id, groups
- who
- last
- finger

11.3 Prozesse

- ps
  - `ps -ef`
  - `ps -ef --forest`
- top
- kill
  - kill -9 <pid>
  - kill -KILL <pid>
  - kill -INT <pid>

11.4 Suchen

- grep
  - `grep opendir *.c`
- find
  - `find . -name "*.c"`
  - `find . -name "*.c" -exec grep -l opendir {} \;`

12 Aufgabe 1

- 1. Include, Deklarationen

```
#include <stdio.h>
#include <stdlib.h>

void append_element(int value);
int remove_element(void);


struct listelement {
  int value;
  struct listelement *next;
};


struct listelement *first = NULL;
```


12 Aufgabe1

- Anfügen an die Liste

```
void append_element(int value) {
  struct listelement *e;

  if (value < 0) return;

  e = (struct listelement*) malloc(sizeof(struct listelement));
  if (e == NULL) {
    perror("Kann Listenelement nicht anlegen.");
    exit(EXIT_FAILURE);
  }

  e->value = value;
  e->next = NULL;

  if (first == NULL) {
    first = e;
  } else {
    /* Hinweis: man vermeidet das Durchlaufen der Liste, wenn man
       einen Zeiger auf das Listenende vorhaelt.
       Hier aber einfaches Durchlaufen.
    */
    struct listelement *p;
    for(p=first; p->next != NULL; p=p->next);
    p->next = e;
  }
}
```

12 Aufgabe1

- Entnehmen aus Liste

```
int remove_element() {
  struct listelement *e;
  int v;
  if (first == NULL) return -1;
  v = first->value;
  e = first;
  first = first->next;
  free(e);
  return v;
}
```


13 Aufgabe3

- opendir, readdir, closedir
- stat, lstat
- readlink
- getpwuid, getgrgid

13.1 opendir

- Funktions-Prototyp:

```
#include <sys/types.h>
#include <dirent.h>

DIR *opendir(const char *dirname);
```

- Argumente
  - ◆ **`dirname`**: Verzeichnisname
- Rückgabewert: Zeiger auf Datenstruktur vom Typ **`DIR`** oder **`NULL`**

13.2 readdir

- Funktions-Prototyp:

```
#include <sys/types.h>
#include <dirent.h>

struct dirent *readdir(DIR *dirp);
```

- Argumente
  - ◆ **`dirp`**: Zeiger auf **`DIR`**-Datenstruktur
- Rückgabewert: Zeiger auf Datenstruktur vom Typ **`struct dirent`** oder **`NULL`** wenn fertig oder Fehler (**`errno`** vorher auf 0 setzen!)
- Achtung: Unter Linux gibt es einen **`readdir`**-Systemcall mit anderen Aufrufparametern. (**`man 3 readdir`**)

13.3 stat / lstat

- Funktions-Prototyp:

```
#include <sys/types.h>
#include <sys/stat.h>
int stat(const char *path, struct stat *buf);
int lstat(const char *path, struct stat *buf);
```

- Argumente:
  - **path**: Dateiname
  - **buf**: Puffer für Inode-Informationen
- Rückgabewert: 0 wenn OK, -1 wenn Fehler
- Beispiel:

```
struct stat buf;
stat("/etc/passwd", &buf); /* Fehlerabfrage ... */
printf("Inode-Nummer: %d\n", buf.st_ino);
```


13.3 stat / lstat: stat-Struktur

- **dev_t st_dev;** Gerätenummer
- **ino_t st_ino;** Inodenummer
- **mode_t st_mode;** Dateimode, u.a. Zugriffs-Bits (siehe **chmod(1)**)
- **nlink_t st_nlink;** Anzahl der (Hard-) Links auf den Inode
- **uid_t st_uid;** UID des Besitzers
- **gid_t st_gid;** GID der Dateigruppe
- **dev_t st_rdev;** DeviceID, nur für Character oder Blockdevices
- **off_t st_size;** Dateigröße in Bytes
- **time_t st_atime;** Zeit des letzten Zugriffs (in Sekunden seit 1.1.1970)
- **time_t st_mtime;** Zeit der letzten Veränderung (in Sekunden ...)
- **time_t st_ctime;** Zeit der letzten Änderung der Inode-Information (...)
- **unsigned long st_blksize;** Blockgröße des Dateisystems
- **unsigned long st_blocks;** Anzahl der von der Datei belegten Blöcke

13.4 readlink

- Funktions-Prototyp:

```
#include <unistd.h>

int readlink(const char *path, char *buf, size_t bufsiz);
```

- Argumente
  - ◆ **path**: Dateiname
  - ◆ **buf**: Puffer für Link-Inhalt
  - ◆ **bufsiz**: Größe des Puffers
- Rückgabewert: Anzahl der Bytes oder -1

13.5 getpwuid

- Funktions-Prototyp:

```
#include <pwd.h>
struct passwd *getpwuid(uid_t uid);
```

- struct passwd:
  - ◆ char *pw_name; /* user's login name */
  - ◆ uid_t pw_uid; /* user's uid */
  - ◆ gid_t pw_gid; /* user's gid */
  - ◆ char *pw_gecos; /* typically user's full name */
  - ◆ char *pw_dir; /* user's home dir */
  - ◆ char *pw_shell; /* user's login shell */

13.6 getgrgid

- Prototyp:

```
#include <grp.h>
struct group *getgrgid(gid_t gid);
```

- struct group:
  - char *gr_name; /* the name of the group */
  - char *gr_passwd; /* the encrypted group password */
  - gid_t gr_gid; /* the numerical group ID */
  - char **gr_mem; /* vector of pointers to member names */