# Betriebssysteme

## Verzeichnisse

Praktikum 7

Fachhochschule Bielefeld
Campus Minden
Studiengang Informatik

---

Beteiligte Personen:



| Name | Matrikelnummer |
|---|---|
| Peter Dick | 1050185 |



---

8. Juni 2016

# Inhaltsverzeichnis

# Aufgabe 7 - Verzeichnisse

## 1 Aufgabenstellung

### 1.1 Aufgabe 1: Typische Verzeichnisstrukturen

- Geben Sie die typische Verzeichnisstruktur einer aktuellen Linuxdistribution an. Was ist in den Verzeichnissen enthalten? Nennen Sie mindenstens fünf Beispiele (wie /etc, /usr usw.). Finden Sie zusätzlich heraus, wo Log-Dateien gespeichert werden. Wo liegen ausgelagerte Inhalte des Hauptspeichers?

- Geben Sie die typische Ordnerstruktur von Microsoft Windows an. Nennen Sie analog zur vor vorherigen Aufgabe einige beispielhafte Inhalte der jeweiligen Verzeichnisse (wie z.B, C:\Windows\System32).

### 1.2 Aufgabe 2: Verzeichnisse

Entwickeln Sie ein Programm myls, das den Inhalt von Verzeichnissen ausgibt. Die grundlegende Funktion ist in etwa vergleichbar mit dem Shell-Kommando ls.

Randbedingungen:

- Der Name des auszulesenden Verzeichnisses soll dem Programm als Argument übergeben werden. Wird kein Verzeichnis angegeben, so wird das lokale Verzeichnis ausgegeben.

- Sie sollen die Funktionen opendir(), readdir() und closedir() verwenden, um die Einträge des Verzeichnisses abzufragen.

- Für den Zugriff auf Informationen zu den Einträgen kann die Funktion lstat() verwendet werden. Diese lifert deterillierte Informationen zu jedem Verzeichniseintrag. Sie sollen mindestens die Attribute abragen und ausgeben, die bei Eingabe des Befehls ls -algo auf UNIX / Linux-System ausgegeben werden.

- Ihr Programm soll hinsichtlich der beiden Optionen -a und -l parametrisierbar sein die beiden Parameter können auch zusammen angewendet werden, also myls -a -l oder myls -al. Hinsichtlich der Bedeutung der Parameter können Sie sich an dem Standard-UNIX/Linux-Kommando orientieren.

- Verwenden Sie getopt(), um die Ausgabeparameter von der Kommandozeile einzulesen.

- Ihr Programm braucht keine weiteren Argumente oder Parameter zu unterstützen.

- Geben Sie bei gesetzter Option -l den Namen ausführbarer Dareien in rot und den Namen von C-Dateien (Dateiendung: .c) in grün aus. Nutzen Sie für die Einfärbung von Verzeichniseinträgen Escape Sequenzen, die Sie beispielsweise unter http://www.linupedia.org/opensuse/Farbe_in_der_Konsole finden.

# 2 Aufgabe 7.1

## 2.1 Vorbereitung

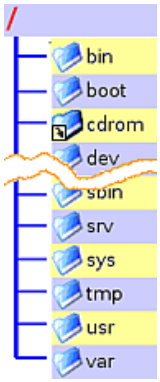
Den Eintrag Verzeichsstuktur aus
”https://wiki.ubuntuusers.de/Verzeichnisstruktur/” lesen

## 2.2 Durchführung

Ergebnisse recherchieren und protokollieren.

## 2.3 Fazit

Aufgabe 7.1.1




/bin:
Enthält für Linux unverzichtbare Programme.

/boot:
Enthält zum Booten benötigte Dateien.

/dev:
Enthält alle Gerätedateien, über die die Hardware im Betrieb angesprochen wird.

/etc:
Enthält Konfigurations- und Informationsdateien des Basissystems.

/lib:
Enthält unverzichtbare Bibliotheken fürs Booten und die dynamisch gelinkten Programme des Basissystems.

/var/log:
Enthält alle Log-Dateien der Systemprogramme.

/var/cache ist vorgesehen als Datenpuffer für Anwendungsproramme.

Aufgabe 7.1.2

C:\Users:
Enthält Verzeichnisse der Benutzer.

C:\Files:
Enthält installierte Programme.

C:\Temp:
Enthält temporäre Dateien.

C:\Windows\Help:
Enthält Hilfe-Dateien.

C:\Windows\Fonts:
Enthält Schrifttyp-Dateien

C:\Windows\Logs:
Enthält alle Log-Dateien der Systemprogramme.

C:\pagefile.sys:
Speichert ausgelagerte Inhalte des Hauptspeichers.

# 3 Aufgabe 7.2

## 3.1 Vorbereitung

C-Projekt anlegen.
Makefile schreiben.

## 3.2 Durchführung

Code schreiben und dann testen bzw debuggen.

## 3.3 Fazit

Randbedingung 1:

```
int main(int argc, char **argv) {
        int aflag = 0;
```

```
int lflag = 0;
int index;
int c;
struct dirent *dirptr;

opterr = 0;
while ((c = getopt (argc, argv, "lal::")) != -1)
        switch (c) {
        case 'a':
                aflag = 1;
                break;
        case 'l':
                lflag = 1;
                break;
        case '?':
                if (isprint (optopt)) {
                        fprintf (stderr, "Unknown option `-%c'.\n", optopt);
                } else {
                        fprintf (stderr, "Unknown option character `\\x%x'.\n", optopt);
                }
                return 1;
        default:
                abort ();
        }
index = optind;
if( index < argc) {
        path = argv[index];
} else {
        path = getenv("PWD");
}
```

In der Variable "argv" sind die Kommandozeilenparameter die übergeben sind gespeichert. Dann wird mit "getopt" die Flags abgearbeitet. In der Variable "optind" (Zeile 48) ist der Index für das Element nach dem letzten Flag gespeichert. Ist der index kleiner als die in "argc" übergebene Anzahl dann wird das entsprechende Argument aus "argv" in path gespeichert. Ansonsten wird ein Standard-Pfad mit "getenv("PWD")" gesetzt.

Randbedingung 2:

```
DIR *dir = opendir(path);
if(dir == NULL) {
        fprintf(stderr,"Fehler beim oeffnen der Datei.");
        return EXIT_FAILURE;
}
dirptr = readdir(dir);
while(dirptr != NULL) {
        char *filename = (*dirptr).d_name;
        if(aflag == 1 && lflag == 1) {
                printFile(filename);
        }
        else if(lflag == 1) {
                if(filename[0] != '.') {
                        printFile(filename);
                }
        }
        else if(aflag == 1) {
                printf ("%s\n", filename);
        } else {
                if(filename[0] != '.') {
                        printf ("%s\n", filename);
                }
        }
        dirptr = readdir(dir);
}
if (closedir(dir) == -1) {
        fprintf(stderr,"Fehler beim Schliessen von %s\n", path);
}
```

Mit "DIR *dir = opendir(path);" wird das Verzeichniss deöffnet. Dann wird mit "dirptr = readdir(dir);" ein Element des Verzeichnisses eingelesen und dann ausgegeben. Das wird solange wiederholt bis "dirptr" gleich NULL ist. Dann wird mit "closedir(dir)" das Verzeichnis geschlossen.

Randbedingung 3:

```
void printFile(char *name) {
        char *dateiendung = strrchr(name, '.');
        struct stat sb;
        char l_rwx[10];
        char rwx[] = "rwxrwxrwx";
        int bits[] = {
                        S_IRUSR,S_IWUSR,S_IXUSR,    /* Zugriffsrechte User     */
```

```
                        S_IRGRP,S_IWGRP,S_IXGRP,    /* Zugriffsrechte Gruppe  */
                        S_IROTH,S_IWOTH,S_IXOTH     /* Zugriffsrechte der Rest */
        };
        if (lstat(name, &sb) == -1) {
                perror("Fehler beim Dateiinformationen auslesen.");
        }
        if(sb.st_mode & S_IEXEC) {
                printf ("\033[0;0;31m%s \033[0m\t\t", name);
        } else if (dateiendung != 0 && !strcmp(dateiendung, ".c")) {
                printf ("\033[0;0;32m%s \033[0m\t\t", name);
        } else {
                printf ("%s\t\t", name);
        }
        if(S_ISBLK(sb.st_mode)) {
                printf("b");
        } else if(S_ISCHR(sb.st_mode)) {
                printf("c");
        } else if(S_ISDIR(sb.st_mode)) {
                printf("d");
        } else if(S_ISFIFO(sb.st_mode)) {
                printf("p");
        } else if(S_ISLNK(sb.st_mode)) {
                printf("l");
        } else if(S_ISREG(sb.st_mode)) {
                printf("-");
        } else if(S_ISSOCK(sb.st_mode)) {
                printf("s");
        } else {
                printf("-");
        }
        for(int i = 0; i < 9; i++) {
                /* wenn nicht 0, dann gesetzt */
                if(sb.st_mode & bits[i]) {
                        l_rwx[i]=rwx[i];  /*r,w oder x*/
                }
                else {
                        l_rwx[i] = '-';
                }
        }
        l_rwx[9]= '\0';
        printf("%s ",l_rwx);
        printf("%ld ", (long) sb.st_nlink);
        printf("%lld bytes ",(long long) sb.st_size);
        printf("%s", ctime(&sb.st_mtime));
}
```

Mit "lstat(name, &sb)" werden die Dateiinformationen in eine struct sb die den Typ stat hat geschrieben. Mit den Makros:

- S_ISBLK(sb.st_mode) fr Block Device
- S_ISCHR(sb.st_mode) fr Character Device
- S_ISDIR(sb.st_mode) fr Verzeichnis/Ordner
- S_ISFIFO(sb.st_mode) fr Pipe
- S_ISLNK(sb.st_mode) fr Link
- S_ISREG(sb.st_mode) fr reguläre Datei
- S_ISSOCK(sb.st_mode) fr Socket

wird der Dateityp ermittelt. Dann werden die Berechtigungen ermittelt dabei wird "sb.st_mode" mit den Bitmasken die in "bits" stehen und-Verknpft.

```
        char l_rwx[10];
        char rwx[] = "rwxrwxrwx";
        int bits[] = {
                        S_IRUSR,S_IWUSR,S_IXUSR,    /* Zugriffsrechte User    */
                        S_IRGRP,S_IWGRP,S_IXGRP,    /* Zugriffsrechte Gruppe  */
                        S_IROTH,S_IWOTH,S_IXOTH     /* Zugriffsrechte der Rest */
        };
```

Wenn das Ergebnis nicht null ist wird r, w oder x gesetzt. Die Anzahl der ist in "sb.st_nlink" gespeichert. In "sb.st_size" die Dateigrösse und in "sb.st_mtime" das Änderungsdatum gespeichert. Alle Daten werden mit printf() ausgegeben.

Randbedingung 4-6:

```
int aflag = 0;
int lflag = 0;
int index;
int c;
struct dirent *dirptr;

opterr = 0;
while ((c = getopt (argc, argv, "lal::")) != -1)
        switch (c) {
        case 'a':
                aflag = 1;
                break;
        case 'l':
                lflag = 1;
                break;
        case '?':
                if (isprint (optopt)) {
                        fprintf (stderr, "Unknown option `-%c'.\n", optopt);
                } else {
                        fprintf (stderr, "Unknown option character `\\x%x'.\n", optopt);
                }
                return 1;
        default:
                abort ();
        }
```

Zuerst wird solange wie "c" ungleich -1 ist mit "c = getopt (argc, argv, "lal::")" die Flags eingelesen. Der String "lal::" erlaubt z. B. folgende eingaben:

- myls
- myls <file>
- myls -a
- myls -l
- myls -al
- myls -la
- myls -a -l
- myls -l -a
- myls -a <file>
- myls -l <file>
- myls -al <file>
- myls -la <file>
- myls -a -l <file>
- myls -l -a <file>

Mit einen switch csae Konstrukt werden dann die Flags "aflag" und "lflag" gesetzt bzw nicht gesetzt.

Randbedingung 7:

```
if(sb.st_mode & S_IEXEC) {
        printf ("\033[0;0;31m%s\033[0m\t\t", name);
} else if (dateiendung != 0 && !strcmp(dateiendung, ".c")) {
        printf ("\033[0;0;32m%s\033[0m\t\t", name);
} else {
        printf ("%s\t\t", name);
}
```

Mit "sb.st_mode & S_IEXEC" wird ermittelt ob die Datei ausührbar ist. Mit "char *dateiendung = strrchr(name, '.');" wird die Dateiendung ermittelt. Die wird mit dem String ".c" verglichen(strcmp). Die Mit einer Escape-Sequenz die Farbe bestimmt. Sie fängt mit "\033[" an und endet mit "m" Dazwischen ist die Hintergrundfarbe, Codes für Textattribute und die Vordergrundfarbe. Die Codes sind mit Semikolon getrennt. 31 steht für rot und 32 für grün. Nach dem der Name eingefärbt wurde wird mit der Escape-Sequenz "\033[0m" die Standard Einstellungen wiederhergestellt.

# 4 Quellen


- https://wiki.ubuntuusers.de/Verzeichnisstruktur/ Aufruf: 08.06.16
- http://www.gnu.org/software/libc/manual/html_node/Using-Getopt.html#Using-Getopt Aufruf: 08.06.16
- http://www.gnu.org/software/libc/manual/html_node/Example-of-Getopt.html#Example-of-Getopt Aufruf: 08.06.16
- https://wiki.ubuntuusers.de/ls/ Aufruf: 08.06.16
- http://openbook.rheinwerk-verlag.de/c_von_a_bis_z/017_c_dateien_verzeichnisse_001.htm Aufruf: 08.06.16
- http://linux.die.net/man/2/lstat Aufruf: 08.06.16
- http://www.linupedia.org/opensuse/Farbe_in_der_Konsole Aufruf: 08.06.16