

U8 Verzeichnisse und Sortieren

- POSIX-Verzeichnis-Systemschnittstelle
- Datei-Attribute in Inodes
- Aufgabe 8

U8-1 Organisatorisches

- wsort-Abgabe verschoben auf **14.1.2009, 19:45**
- keine Übungen im Zeitraum 22.12.2008 - 11.1.2009

U8-2 POSIX-Verzeichnis-Systemschnittstelle

- Verzeichnisse öffnen: **opendir(3)**
- Verzeichnisse lesen: **readdir(3)**
- Verzeichnisse schliessen: **closedir(3)**

1 opendir / closedir

■ Funktions-Prototypen:

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

DIR *opendir(const char *dirname);

int closedir(DIR *dirp);
```

■ Argument von opendir

◆ **dirname**: Verzeichnisname

■ Rückgabewert: Zeiger auf Datenstruktur vom Typ **DIR** oder **NULL**

■ initialisiert einen internen Zeiger des directory-Funktionsmoduls auf den ersten Directory-Eintrag (für den ersten **readdir**-Aufruf)

■ **closedir** schliesst ein geöffnetes Verzeichnis nach Bearbeitungsende

2 readdir

- liefert einen Directory-Eintrag (interner Zeiger) und setzt den Zeiger auf den folgenden Eintrag

- Funktions-Prototyp:

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

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

- Argumente

- ◆ **dirp**: Zeiger auf **DIR**-Datenstruktur (von **opendir(3)**)

- Rückgabewert: Zeiger auf Datenstruktur vom Typ **struct dirent** oder **NULL**, wenn **EOF** erreicht wurde oder im Fehlerfall

- bei **EOF** bleibt **errno** unverändert (kritisch, kann vorher beliebigen Wert haben), im Fehlerfall wird **errno** entsprechend gesetzt
- **errno** vorher auf 0 setzen, sonst kann **EOF** nicht sicher erkannt werden!

2 ... readdir

- Problem: Der Speicher für die zurückgelieferte `struct dirent` wird von den dir-Bibliotheksfunktionen selbst angelegt und bei jedem Aufruf wieder verwendet!
 - ◆ werden Daten aus der dirent-Struktur länger benötigt, müssen sie vor dem nächsten readdir-Aufruf in Sicherheit gebracht (kopiert) werden
 - ◆ konzeptionell schlecht
 - aufrufende Funktion arbeitet mit Zeiger auf internen Speicher der readdir-Funktion
 - ◆ in nebenläufigen Programmen (mehrere Threads) nicht einsetzbar!
 - man weiss evtl. nicht, wann der nächste readdir-Aufruf stattfindet
- readdir ist ein klassisches Beispiel für schlecht konzipierte Schnittstellen in der C-Funktionsbibliothek

3 struct dirent

- Definition unter Linux (/usr/include/bits/dirent.h)

```
struct dirent {  
    __ino_t d_ino;  
    __off_t d_off;  
    unsigned short int d_reclen; /* tatsächl. Länge der Struktur */  
    unsigned char d_type;  
    char d_name[256];  
};
```

- POSIX: d_name ist ein Feld unbestimmter Länge, max. NAME_MAX Zeichen

U8-3 Datei-Attribute ermitteln: stat

- liefern Datei-Attribute aus dem Inode

- Funktions-Prototyp:

```
int stat(const char *path, struct stat *buf);  
int lstat(const char *path, struct stat *buf);
```

- Argumente:

- ◆ `path`: Dateiname

- ◆ `buf`: Zeiger auf Puffer, in den Inode-Informationen eingetragen werden

- Rückgabewert: 0 wenn OK, -1 wenn Fehler

- Beispiel:

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

1 stat: Ergebniserückgabe im Vergleich zur readdir

- problematische Rückgabe auf funktions-internen Speicher wie bei `readdir` gibt es bei `stat` nicht
- Grund: `stat` ist ein Systemaufruf - Vorgehensweise wie bei `readdir` wäre gar nicht möglich
- der logische Adressraum des Anwendungsprogramms ist nur eine Teilmenge (oder sogar komplett disjunkt) von dem logischen Adressraum des Betriebssystems
 - Betriebssystemspeicher ist für Anwendung nicht sichtbar/zugreifbar
 - Funktionen des Kernels (wie `stat`) können keine Zeiger auf ihre internen Datenstrukturen an Anwendungen zurückgeben

1 stat / lstat: stat-Struktur

- `dev_t st_dev`; Gerätenummer (des Dateisystems) = Partitions-Id
- `ino_t st_ino`; Inodenummer (Tupel `st_dev, st_ino` eindeutig im System)
- `mode_t st_mode`; **Dateimode, u.a. Zugriffs-Bits und Dateityp**
- `nlink_t st_nlink`; Anzahl der (Hard-) Links auf den Inode (Vorl. 7-32)
- `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

U8-4 Aufgabe 8

- einfaches find-Programm (SPIC Directory Evaluator Recursive)

`spider <path> [<minSize>]`

- durchläuft **rekursiv** den Verzeichnisbaum mit Wurzel `path`
- gibt die Namen aller gefundenen Verzeichniseinträge aus
 - ◆ den Pfad des Eintrags relativ zu `path` angehängt an `path`
 - ◆ optional: nur Einträge mit der Mindestgrösse (in Bytes) `minSize`
- Einträge, deren Namen mit einem `'.'` beginnt, werden ignoriert
- symbolische Links werden **nicht** verfolgt (Gefahr von Zyklen!)
 - ◆ welche **stat**-Funktion ist zu verwenden?