

# Übungen zu Systemnahe Programmierung in C (SPiC)

Sebastian Maier, Heiko Janker  
(Lehrstuhl Informatik 4)

## Übung 7



Wintersemester 2015/2016

## Inhalt

### POSIX Verzeichnisschnittstelle

Dateisystem: Dateien, Verzeichnisse und inodes  
opendir, closedir, readdir  
Fehlerbehandlung bei readdir  
Verwendung von stat

Aufgabe: printdir

### Debuggen

GDB & CGDB  
Valgrind

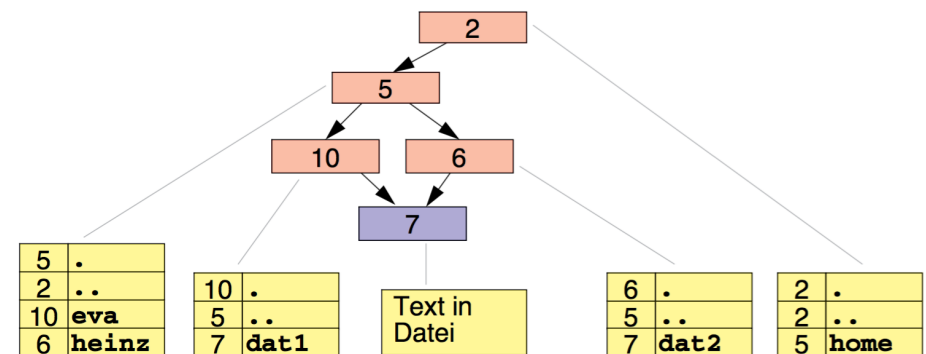
## Dateisystem: Dateien, Verzeichnisse und inodes

### POSIX Verzeichnisschnittstelle

Dateisystem: Dateien, Verzeichnisse und inodes  
opendir, closedir, readdir  
Fehlerbehandlung bei readdir  
Verwendung von stat

Aufgabe: printdir

Debuggen



**inode** enthält Dateiattribute & Verweise auf Datenblöcke

**Verzeichnis** spezielle Datei mit Paaren aus Namen & inode-Nummer

## opendir, closedir, readdir

### Funktions-Prototypen (Details siehe Vorlesung)

```
1 #include <sys/types.h>
2 #include <dirent.h>
3
4 DIR *opendir(const char *dirname);
5 int closedir(DIR *dirp);
6 struct dirent *readdir(DIR *dirp);
```

### Rückgabewert von readdir

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

~ bei EOF bleibt errno unverändert (auch wenn errno != 0),  
im Fehlerfall wird errno entsprechend gesetzt



## struct dirent

```
1 struct dirent {
2     ino_t      d_ino;          // inode number
3     off_t      d_off;          // not an offset; see NOTES
4     unsigned short d_reclen;    // length of this record
5     unsigned char d_type;       // type of file; not supported
6                                     // by all filesystem types
7     char        d_name[256];    // filename
8 };
```

### entnommen aus Manpage readdir(3)

- nur d\_name und d\_ino Teil des POSIX-Standards
- relevant für uns: Dateiname (d\_name)



## Einschub: Komma-Operator

### Funktionsweise:

1. Auswertung des ersten Ausdrucks (Verwerfen dieses Ergebnisses)
2. Auswertung des zweiten Ausdrucks (Rückgabe dieses Ergebnisses)

```
1 int c = (add(3,2), sub(3,2));
```

### Geeignet für Initialisierungen vor Überprüfung der Schleifenbedingung

⇒ cli/sei

```
1 while(cli(), event != 0){
2     sleep_enable();
3     sei();
4     sleep_cpu();
5     ...
6 }
```

### Elegant, aber keine Notwendigkeit!



## Fehlerbehandlung bei readdir

### Fehlerprüfung durch Setzen und Prüfen von errno:

```
1 #include <errno.h>
2 ...
3 struct dirent *ent;
4 while(1) {
5     errno = 0;
6     ent = readdir(...);
7     if(ent == NULL) break;
8     ... /* keine weiteren break-Statements in der Schleife */
9 }
10 /* EOF oder Fehler? */
11 if(errno != 0) {
12     /* Fehler */
13     ...
14 }
```

- errno=0 unmittelbar vor Aufruf der problematischen Funktion  
⇒ errno wird nur im Fehlerfall gesetzt und bleibt sonst evtl. unverändert
- Abfrage der errno unmittelbar nach Rückgabe des pot. Fehlerwerts  
⇒ errno könnte sonst durch andere Funktion verändert werden



## Fehlerbehandlung bei readdir

- Fehlerprüfung durch Setzen und Prüfen von errno:

```
1 #include <errno.h>
2 ...
3 struct dirent *ent;
4 while(errno=0, (ent=readdir()) != NULL) {
5     ... /* keine weiteren break-Statements in der Schleife */
6 }
7 /* EOF oder Fehler? */
8 if(errno != 0) {
9     /* Fehler */
10    ...
11 }
```

- errno=0 unmittelbar vor Aufruf der problematischen Funktion  
⇒ errno wird nur im Fehlerfall gesetzt und bleibt sonst evtl. unverändert
- Abfrage der errno unmittelbar nach Rückgabe des pot. Fehlerwerts  
⇒ errno könnte sonst durch andere Funktion verändert werden
- Zuweisungsausdruck hat nach Zuweisung Wert des **li. Operanden**
- Keine Hilfsfunktion hier (ferror() bei getchar())



## Datei-Attribute ermitteln: stat

- readdir(3) liefert **nur Name und Typ** eines Verzeichniseintrags
- Weitere Attribute stehen im **Inode**
- stat(2) Funktions-Prototyp:

```
1 #include <sys/types.h>
2 #include <sys/stat.h>
3 int stat(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:

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



## Das struct stat

- Ausgewählte Elemente
  - dev\_t st\_dev Gerätenummer (des Dateisystems) = Partitions-Id
  - ino\_t st\_ino Inodennummer (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
  - 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



## stat / lstat: st\_mode

- st\_mode enthält Informationen über den Typ des Eintrags:
  - S\_IFMT 0170000 bitmask for the file type bitfields
  - S\_IFSOCK 0140000 socket
  - S\_IFLNK 0120000 symbolic link
  - S\_IFREG 0100000 regular file
  - S\_IFBLK 0060000 block device
  - S\_IFDIR 0040000 directory
  - S\_IFCHR 0020000 character device
  - S\_IFIFO 0010000 FIFO
- Zur einfacheren Auswertung werden Makros zur Verfügung gestellt:
  - S\_ISREG(m) - is it a regular file?
  - S\_ISDIR(m) - directory?
  - S\_ISCHR(m) - character device?
  - S\_ISLNK(m) - symbolic link?

```
1 mode_t m = stat_buf.st_mode;
2 if( (m & S_IFMT) == S_IFREG) ...
```



## Inhalt

POSIX Verzeichnisschnittstelle

Aufgabe: printdir

Debuggen



## Aufgabe: printdir

- Iteration über alle via Parameter übergebene Verzeichnisse
- Ausgabe aller darin enthaltenen Einträge mit Größe und Name
- Anzeige der Anzahl von regulären Dateien und deren Gesamtgröße (pro Verzeichnis)
- Relevante Funktionen:
  - opendir(3) bekommt einen Pfad
  - readdir(3) liefert nur einen Dateinamen
  - stat(3) weiß nicht auf welchen Pfad sich dieser Dateiname bezieht
    - ⇒ stat(3) braucht einen vollständigen Pfad mit Datei
    - ⇒ strncpy(3), strncat(3), snprintf(3)
    - ⇒ Beim Kopieren von Zeichenketten muss man aufpassen, dass immer genug Speicher zur Verfügung steht.
- Fehlerbehandlung:
  - Jede falsche Benutzereingabe abfangen
    - ⇒ den DAU annehmen ☺
  - Aussagekräftige Fehlermeldungen



## Inhalt

POSIX Verzeichnisschnittstelle

Aufgabe: printdir

Debuggen  
GDB & CGDB  
Valgrind



## Debuggen

```
8 #define For(i, s, n) for(int i = (s); i < (n); ++ i)
9 #define Clr(a) memset(a, 0, sizeof a)
10 #define Ot(c, d) cout << "Case #" << c << ": " << (d) << endl
11
12 template<class T>
13 string str(T a){stringstream ss; ss << a; string ret; ss >> ret; return ret;}
14
15 int lr[41][2];
16 bool odd[41];
17
18 int main(){
19     int T;
20     cin >> T;
21     for(int c = 1; c <= T; ++ c){
22
23         ReadInt(N);
24         For(i, 1, N){
25             cin >> lr[i][0] >> lr[i][1];
26         }
27         Clr(odd);
28
29         int UPPER = (N - 1) * (1 << (N - 1));
30         int cur = 1; odd[1] = true;
31         int result = 0;
32
33         //Users/mirikle/Algorithm/e/5.C
34         cin >> lr[1][0] >> lr[1][1];
35     }
36     Clr(odd);
37
38     int UPPER = (N - 1) * (1 << (N - 1));
39     (gdb) r
```



- Übersetzen mit Debug-Symbolen (-g) & ohne Optimierungen (-O0)

```
1 gcc -g -pedantic -Wall -Werror -O0 -std=c99 -D_XOPEN_SOURCE=500
```

- Starten des Debuggers

```
1 gdb ./cworld
2 # alternativ ...
3 cgdb --args ./cworld arg0 arg1 ...
```

- Kommandos

- b(reak): Breakpoint setzen
- r(un): Programm bei main() starten
- n(ext): nächste Anweisung (nicht in Unterprogramme springen)
- s(tep): nächste Anweisung (in Unterprogramme springen)
- p(rint) <var>: Wert der Variablen var ausgeben

⇒ **Debuggen ist (fast immer) effizienter als Trial-and-Error!**



- Informationen über:

- Speicherlecks (malloc/free)
- Zugriffe auf nicht gültigen Speicher

- Ideal zum Lokalisieren von Segmentation Faults (SIGSEGV)

- Aufrufe:

- valgrind ./cworld
- valgrind --leak-check=full --show-reachable=yes --track-origins ✓  
↳ =yes ./cworld

