Übungsblatt 5

Übung zur Betriebssystemarchitektur
WS 2004/05

Dipl.-Inf. Bernhard Rabe
Betriebssysteme & Middleware

Prüfung Betriebssystemarchitektur

- Termine: **2.2. - 4.2** und **7.2.2005**
- Zeiten: **8:30-12:30** und **13:30-16:30** Uhr

- Anmeldung zur Prüfung:

  Ab **11.01.2005 9:00-11:00 Uhr**

  bei Sabine Wagner C-1.8
- Prüfungszeit: **1 bzw. ¾ Stunde pro Gruppe**

Inhalt

- Dateien Lesen/Schreiben/Informationen
- Dateisystemzugriff
- Sockets
- FTP Protokoll

I/O System

- Grundlegende Komponente eine Betriebssystems
- Zugriff auf I/O Geräte durch gemeinsame System API

Dateien Lesen/Schreiben

- “plattformübergreifend” mit POSIX Funktionen
- open, creat, read, write, lseek, close

open/creat, close

- `int open(const char *pathname, int flags [, mode_t mode])`
  - öffnet eine Datei/Device
- `int creat(const char *pathname, mode_t mode)`
  - erzeugt eine Datei
- `int close(int fd)`
  - schließt einen Descriptor

read/write

```
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
```

- lesen/schreiben maximal *count* Bytes in/von *buf* von/nach *fd*
- setzt den Positionszeiger bei regulären Dateien um den Rückgabewert vorwärts

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

- unter Windows

  #include <io.h> erforderlich

  stdio.h, stdlib.h

lseek

```
off_t lseek(int fildes, off_t offset, int whence);
```

- setzt den Positionszeiger relativ zur angebenen Position um `offset`-Bytes
- `whence`
  - SEEK_SET absolute Position
  - SEEK_CUR relativ zu aktueller Position
  - SEEK_END relativ zu Ende der Datei
- liefert neue Position zurück
  - aktuelle Position: offset=0, whence=SEEK_CUR

Windows

- CreateFile
- ReadFile
- WriteFile
- SetFilePointer
- CloseHandle

SetFilePointer

- **DWORD SetFilePointer(HANDLE** *hFile*, **LONG** *lDistanceToMove*, **PLONG** *lpDistanceToMoveHigh*, **DWORD** *dwMoveMethod* **);**
- *lDistanceToMove* : “relative” Byte Position
- *lpDistanceToMoveHigh:* obere 32 Bit einer 64 Bit Positon
  NULL für 32 Bit offset
- Postion oder Bewegungsrichtung
  - FILE_BEGIN
  - FILE_CURRENT
  - FILE_END
  - analog zu lseek

Verzeichnisse/Win32

HANDLE FindFirstFile(LPCTSTR *lpFileName*,
LPWIN32_FIND_DATA *lpFindFileData* **);**

- *lpFileName* Pfad und Filter
  - z.B. “*” für alle Dateien im lokalen Verzeichnis
- *lpFindFileData* Verzeichniseintrag
- liefert HANDLE für weitere Suchergebnisse mit FindNextFile

WIN32_FIND_DATA

```
typedef struct _WIN32_FIND_DATA {
    DWORD dwFileAttributes;          // Verzeichnis ?
    FILETIME ftCreationTime;
    FILETIME ftLastAccessTime;
    FILETIME ftLastWriteTime;        //letzte Änderung
    DWORD nFileSizeHigh;             //Größe  High Teil
    DWORD nFileSizeLow;              //Größe Low Teil
    DWORD dwReserved0;
    DWORD dwReserved1;
    TCHAR cFileName[MAX_PATH];  //langer Name
    TCHAR cAlternateFileName[14];   //8.3 Name
 } WIN32_FIND_DATA;
```

FindNextFile

BOOL FindNextFile(HANDLE *hFindFile*, **LPWIN32_FIND_DATA** *lpFindFileData* **);**

- liefert im Fehlerfall 0
  - GetLastError() liefert ERROR_NO_MORE_FILES, wenn keine passenden Einträge vorhanden sind

BOOL FindClose(HANDLE *hFindFile* **);**

  - schließt das HANDLE

FileTimeToSystemTime

- Wandelt FILETIME Struktur in eine SYSTEMTIME Struktur

```
typedef struct _SYSTEMTIME {
  WORD wYear;
  WORD wMonth;
  WORD wDayOfWeek;
  WORD wDay;
  WORD wHour;
  WORD wMinute;
  WORD wSecond;
  WORD wMilliseconds; } SYSTEMTIME;
```

Verzeichnislisting

DIR<TAB>Verzeichnisname

<TAB>Dateiname

list.c

Linux

#include <sys/types.h>

#include <dirent.h>

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

- liefert einen Pointer auf den Verzeichnis-Stream
- der Streamposition zeigt auf den 1. Eintrag

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

- liefert einen Zeiger auf eine `dirent` Struktur
- NULL wenn Fehler oder Ende des DIR-Streams

```
int closedir(DIR *dir);
```

- schließt den Verzeichnis-Stream

dirent

```
struct dirent {
  #if defined _FILE_OFFSET_BITS && _FILE_OFFSET_BITS == 64
    ino64_t            d_ino;         /* inode number */
    off64_t            d_off;         /* offset to the next dirent */
 #else
   ino_t               d_ino;         /* inode number */
    off_t              d_off;         /* offset to the next
   dirent */
 #endif
    unsigned short int d_reclen;      /* length of this record
   */
    unsigned char      d_type;        /* type of file */
    char               d_name[256];   /* filename */
};
```

stat

```
#include <sys/stat.h>

#include <unistd.h>
int stat(const char *file_name, struct stat *buf);
```

- liefert Informationen über die Datei *filename*

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

- liefert bei Links Informationen über den Link und nicht vom Ziel des Links

struct stat

```
struct stat {
     dev_t         st_dev;      /* device */
     ino_t         st_ino;      /* inode */
     mode_t        st_mode;     /* protection */
     nlink_t       st_nlink;    /* number of hard links */
     uid_t         st_uid;      /* user ID of owner */
     gid_t         st_gid;      /* group ID of owner */
     dev_t         st_rdev;     /* device type (if inode device) */
     off_t         st_size;     /* total size, in bytes */
     blksize_t     st_blksize;  /* blocksize for filesystem I/O */
     blkcnt_t      st_blocks;   /* number of blocks allocated */
     time_t        st_atime;    /* time of last access */
     time_t        st_mtime;    /* time of last modification */
     time_t        st_ctime;    /* time of last status change */
 };
```

- S_ISDIR(stat.st_mode) liefert >0 wenn ein Verzeichnis
- S_ISREG(stat.st_mode) liefert >0 wenn eine Datei

localtime

struct tm *localtime(**const** time_t *timep);

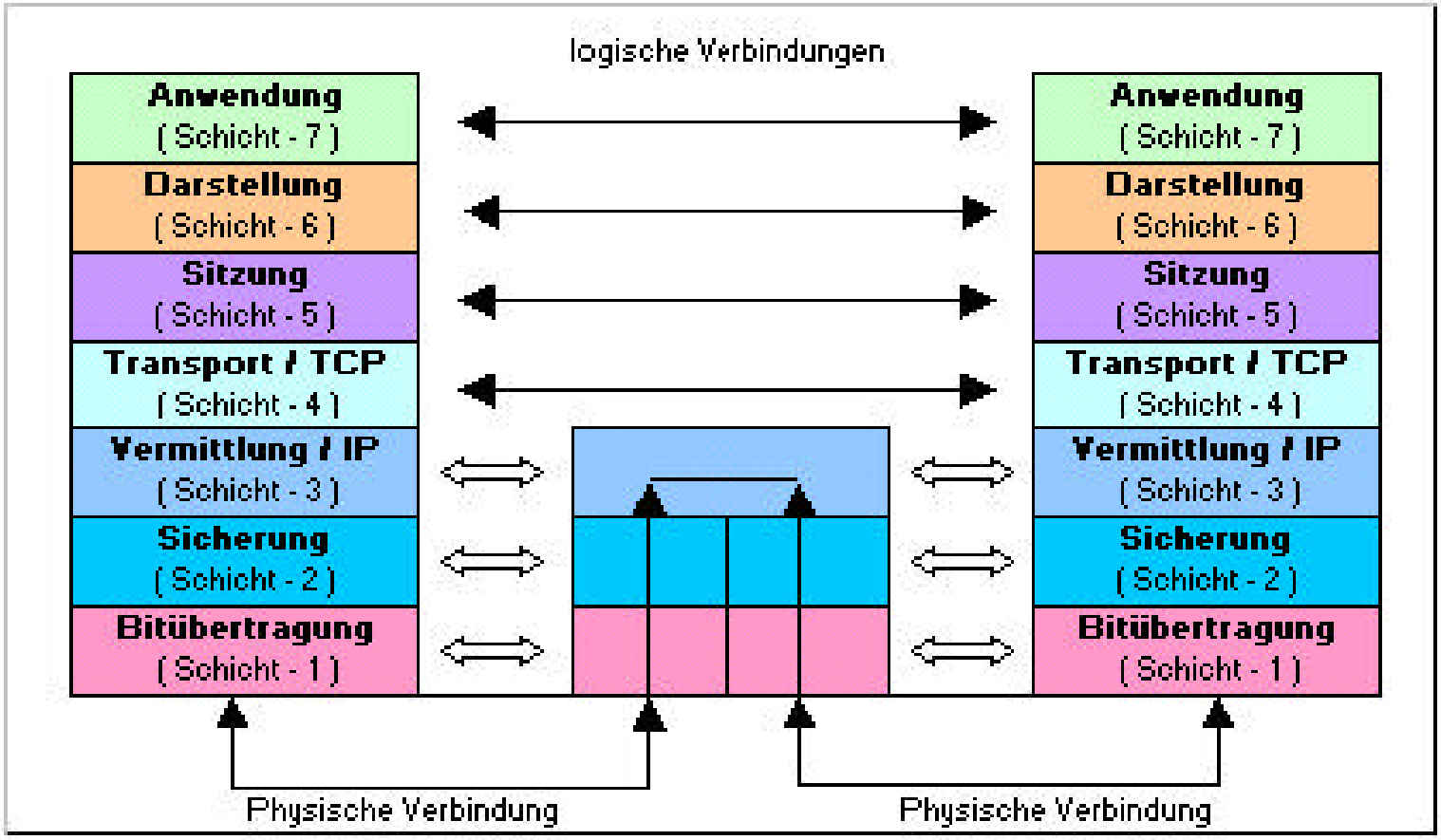
- wandelt eine `time_t` Zeitstempel in eine `tm`-Struktur um

```
struct tm {
    int     tm_sec;         /* seconds */
    int     tm_min;         /* minutes */
    int     tm_hour;        /* hours */
    int     tm_mday;        /* day of the month */
    int     tm_mon;         /* month */
    int     tm_year;        /* year */
    int     tm_wday;        /* day of the week */
    int     tm_yday;        /* day in the year */
    int     tm_isdst;       /* daylight saving time */
};
```

Neztwerkprogrammierung

- Rechnerübgreifende Kommunikation
  - Interprocesskommunikation
- Betriebssystemübergreifende Kommunikation
  - HTTP, FTP, SMB, NFS, ...
- Basistechnologie für unterschiedlichste Protokolle: TCP/IP UDP/IP

ISO/OSI Schichtenmodell



- **O**pen **S**ystems **I**nterconnection
  - idealisiertes Kommunikationsmodell
  - jedes Layer stellt Dienste für den darrüberliegenden Layer zur Verfügung und nutzt den darunter liegenden Layer
- **Application Layer (*Anwendungsschicht*):**
  - Datenaustausch zwischen Netzwerkanwendungen
- **Presentation Layer (*Darstellungsschicht*):**
  - Wahl einer gemeinsamen Syntax für den Datenaustausch

- **Session Layer (*Kommunikationssteuerungsschicht*):**
  - verwaltet Verbindungen (Sessions) zwischen kooperierenden Anwendungen
- **Transport Layer (*Transportschicht*):**
  - Abbilung zwischen Nachrichten und Packeten
  - TCP, UDP
- **Network Layer (*Vermittlungsschicht*):**
  - Transport von Paketen zwischen beliebigen Endpunkten, Adressierung, Routing
  - IP,

- **Link Layer (*Sicherungsschicht*):**
  - Realisierung eines zuverlässigen Übertragungskanals
- **Physical Layer (*Bitübertragungsschicht*):**
  - Transport von unformatierten Bit-Sequenzen

Sockets

- Interprozesskommunikation über Rechner- und Betriebssystemgrenzen
- Berkeley Socket API ist Quasistandard
- Tupel aus Quell- und Zieladresse, Quell- und Zielport und Transportprotokoll

Verbindungslos vs Verbindungsorientiert

- Verbindungslos
  - UDP
  - asynchrones Senden
  - keine Bestätigung für gesendete Nachrichten im Protokoll

- Verbindungsorientiert
  - TCP
  - synchrones Senden/Empfangen
  - Nachrichten werden durch das Protokoll bei erfolgreichem Empfang bestätigt
  - Timeout bei abgebrochenen Verbindungen
  - zuverlässige Kommunikation

Windows vs. Unix

#include <winsock2.h>

#include <sys/socket.h>

Link: ws2_32.lib

WSAStartup()

WSACleanup()

socket

- **SOCKET socket(int** *af*, **int** *type*, **int** *protocol* **);**
  - Windows
  - (unsigned) int32 = SOCKET

- **int** socket(**int** domain, **int** type, **int** protocol);
  - Unix

af (IPV4) **domain**

AF_INET PF_INET

#define PF_INET AF_INET

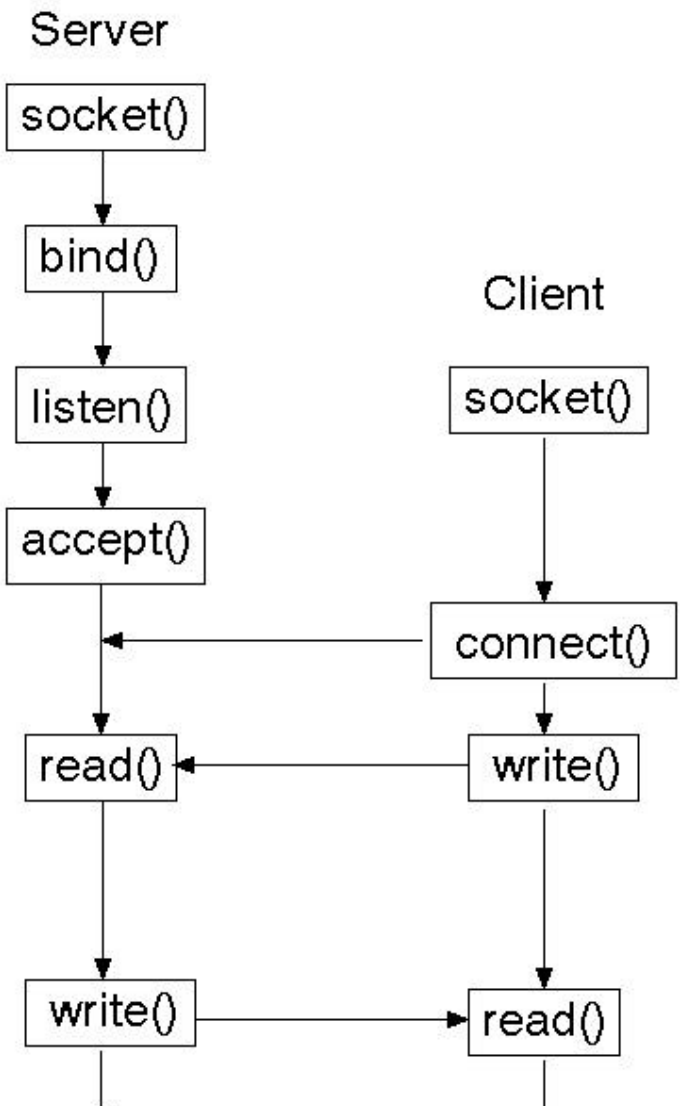
type

- SOCK_STREAM → TCP
- SOCK_DGRAM → UDP

protocol

- normalerweise ist *protocol* eindeutig für *type*
- 0 für IP (RFC 1700)

TCP Verbindung



Server Funktionen

- Server
  - `bind`
    - socket an eine IP und Port binden
  - `listen`
    - Anzahl der maximal wartenden Verbindung setzen
  - `accept`
    - auf einen Verbindung warten
    - blockiert
    - liefert einen neues Socket mit der Verbindung zum Klienten

Klienten Funktionen

- `connect`
  - socket mit einer gegebenen IP Adresse und Port verbinden

Gemeinsame Funktionen

- `send`
  - Bytestrom senden
- `recv`
  - Bytestrom empfangen
- `close/closesocket`
  - Verbindung beenden

Aufgabe 5.1

- Geburstagsdatenbank
- `hash.c` um die Eintragsnummer in der Datenbank zu finden
- jeder Eintrag hat eine feste Größe
- Lesen / Schreiben von Einträgen nach Positionierung mit `lseek`

Aufgabe 5.2

- Verzeichnisinhalt auslesen und formatiert ausgeben
- `char *list(const char* path)`
  - Funktion schreiben und in `list.c` speichern
  - Fehlerausschriften in `static char* errormsg;`
- Format ist ls -l

- Rechteblock kann statisch bis auf Eintragstyp sein (Datei vs Verzeichnis)
  - `drwx------`
  - `-rwx------`
- Hardlink kann Konstante sein
- Owner, Gruppe Konstanten
- Größe, Zeit der letzten Änderung (Schreiben) und Name müssen original sein

```
drwx------ 1  root None    0  Dec 4 2002 ..
-rwx------ 1  root None  123  Jan 6 2004 hallo.c
```

Aufgabe 5.3

- Daten einer Datei mit 49 Cluster in NTFS speichern
- Kompressionsarten
- nur bestimmte Cluster verfügbar auf dem Dateisystem → Fragmente
- Format:

VCN	LCN	Cluster
0	123	xxxx

Quellen

- http://new.linux-it.net/netzwerk.php?cat=osi
- http://pandonia.canberra.edu.au/ClientServer/socket/tcp_seq.gif