UNIX-Schnittstelle

6. E/A-Geräte unter UNIX

Alle Beispiel-Quellen mittels SVN unter:

https://svn.informatik.hu-berlin.de/svn/unix-2014/File

6.1 Vorbemerkung

a) Hardware

E/A-Geräte

Blockorientierte Geräte
- Medium hat feste physische Blocklänge (128..4096 Byte, üblich 512/1024)
- Direktzugriff und sequentieller Zugriff
- Seekoperationen möglich

Beispiele: Festplatten, CD's, Floppy, Magnetband

Zeichenorientierte Geräte
- Ströme von Zeichen, sequentieller Zugriff
- keine feste Satzstruktur
- keine Seekoperationen

Beispiele: Terminal, Drucker, Modem

Sondergeräte
- nicht zum direkten Datenaustausch

Beispiele: Uhren, Hauptspeicher

Controller

Geräte haben Standardschnittstellen
(RS232, SCSI, SCSI2, IDE, Centronics,...)
Prozessoren realisieren unterschiedliche Busstrukturen.
Controller stellen die Verbindung zwischen den hersteller-spezifischen Buststrukturen und den standardisierten Geräte-schnittstellen her.

Aufgaben der Controller:
- Wandlung von Bitströmen in Byte-/Wortströme
- Geräteauswahl
- Prüfsummenberechnung
- Fehlerkorrektur
- Erzeugung von gerätespezifischen Steuersignalen (Synchronsignale)
- Interrupterzeugung
- Ausführung von Steueroperationen

Controller belegen in der Regel mehrere E/A-Adressen und ein Interruptvektor

Struktur eines Rechners (Busstruktur)

```
============================================================ Bus
  |         |           |             |             |
+---+  +---------+  +-----------+  +-----------+  +-----------+
|CPU|  |Speicher |  |Platten-   |  |Drucker-   |  |andere     |
|   |  |         |  |kontroller |  |kontroller |  |Kontroller |
+---+  +---------+  +-----------+  +-----------+  +-----------+
                       |   |             |          |      |
                       P1  P2         Drucker       G1     Gn
```

Struktur eines Rechners (Switch-Struktur)

```
+---+                                  +-----------+
|CPU|-------------+     +--------------|Platten-   |--P1
+---+             |     |              |kontroller |--P2
                  |     |              +-----------+
                +---------+
                | SWITCH  |
                +---------+
                  |     |
+---------+       |     |              +-----------+
|Speicher |-------+     +--------------|Drucker-   |--Drucker
|         |                            |kontroller |
+---------+                            +-----------+
```

b) Software

Grundforderungen:

- Gerätunabhängigkeit der Nutzersoftware
- Einheitliche Namensgebung von Geräten und Dateien
- Einheitliche Fehlerbehandlung für den Nutzer, d.h. E/A-Software muss gerätspezifische Fehlerbehandlung realisieren.
- synchrone und asynchrone E/A-Operationen
- Syncronisation und Organisation des Zugriffs zu E/A-Geräten
  - a) gleichzeitiger Zugriff mehrerer Nutzer auf ein Gerät (Platte - Filesystem)
  - b) Einzelzugriff auf ein Gerät (Magnetband)

Günstige Struktur des IO-Systems:

```
5. nutzerorientierte Software       (Nutzer)
4. gerätunabhängige Systemsoftware  (System - Kern)
3. Gerätetreiber                    (System - Hardwareentwickle
2. Interrupthandler                 (System - Kern)
1. Hardware
```



```
E/A-Anforderung         Schicht          E/A-Rückmeldung
      |         +-------------------------+ ^
      |         |                         | |
      +-------->| *   Benutzerprozess   * |-+ Spooling,  E/A-Aufrufe
                | |                     ^ |   E/A-Formatierung
                +-|---------------------|-+
                | |                     | |
                | V  Gräteunabhängige     |   Namensgebung, schützen
                |       Software        ^ |   puffern, blocken
                +-|---------------------|-+
                | |                     | |
                | V   Gerätetreiber       |   Gerätesteuerung
                |                       ^ |   Statusabfragen
                +-|---------------------|-+
                | |                     | |
                | V  Unterbrechungs-      |   Gerätetreiber aktivieren
                |    routine            ^ |   nach E/A-Unterbrechung
                +-|---------------------|-+
                | |                     | |
                | V      Hardware       | |   E/A-Operation
                | * .................   * |   ausführen
                +-------------------------+
```

Aufgaben der Schichten im Detail:

Hardware:

Realisierung der E/A-Operation

Interrupthandler:

Assemblerroutine des Kerns

- Retten des aktuellen Prozessorzustandes
- Memmorymanagment (ausblenden des aktuellen Prozesses, einblenden der Gerätetreiber)
- aktivieren des Gerätetreibers zur I/O-Ende-Behandlung
- aktivieren eines Prozesses nach Interruptbehandlung

Gerätetreiber

C-Routine zur Steuerung einer Gerätefamilie. Es werden folgende gerätunabhängige Funktionen zur Verfügung gestellt:

```
xx_open     - eröffnen eines Gerätes
xx_close    - abschliessen eines Gerätes
xx_read     - lesen (Zeichen)
xx_write    - schreiben (Zeichen)
xx_ioctl    - gerätspezifische Steueroperationen (Zeichen)
xx_intr     - Interruptroutine (Eintritt für Interrupthandler)
xx_strategy - lesen/schreiben  (blockorientiert)
```


Geräteunabhängige Systemsoftware

Einheitliche Schnittstellen für Gerätetreiber
Einheitliche Schnittstellen zum Nutzer
Einheitliche Namensgebung für Geräte und Files
Schutz der Geräte
Geräteunabhängige Blockgrösse
Pufferung
Speicherzuordnung auf blockorientierten Geräten (Filesystem)
Ver- und Freigabe von exklusiv genutzten Geräten
Fehlerstatistik

Nutzerorietierte Software

- basierend auf "low level I/O" : open, close, read, write, lseek, ioctl, ...
- basierend auf "high level I/O" : fopen, fclose, fprintf, fscanf, ...
- Spoolsysteme: lp, uucp

6.2 Systemrufe des Nutzers für E/A

Nutzbar für E/A-Operationen über Geräte und über Files (später).

```
  open    -  Eröffnen von (bestehenden) Files

  close   -  Abschliessen von Files

  ioctl   -  Ausführen von Steueroperationen für Geräte
              (nur bedingt für Files nutzbar)

  read    -  Lesen von Daten

  write   -  Schreiben von Daten
```



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

int open(const char *pathname, int oflag,.../*, mode_t mode */ );
```

open eröffnet ein File unter dem angegeben Namen *pathname für den angegeben Modus oflag . mode spezifiziert die Zugriffsrechte, falls das File neu erzeugt wird.

```
oflag:  O_RDONLY    - nur lesen
        O_WRONLY    - nur schreiben
        O_RDWR      - lesen und schreiben
        O_APPEND    - Anfügen
        O_CREAT     - Erzeugen eines neuen Files
        O_EXCL      - Erzeugt Fehler, wenn O_CREAT und File existiert
        O_TRUNC     - Wenn File zum Schreiben eröffnet, dann
                      kürze die Länge des Files auf 0
        O_NONBLOCK  - nonblocking Mode für FIFO, block special file
                      oder character special file
        O_SYNC      - warten auf physisches E/A-Ende bei
                      Schreiboperationen

Filenamenlänge:     alt:  14,     neu:  256
Pathnamelänge:      4095
Anzahl der eröffneten Files:  alt  20
                               neu: 256  (OPEN_MAX in sys/limits.h)
```

```
mode:   sys/stat.h
          S_IRUSR   user-read        S_IRGRP   group-read
          S_IWUSR   user-write       S_IWGRP   group-write
          S_IXUSR   user-execute     S_IXGRP   group-execute
          S_IROTH   other-read
          S_IWOTH   other-write
          S_IXOTH   other-execute
          S_ISUID   set-user-ID on execution
          S_ISGID   set-group-ID on execution
          S_ISVTX   sticky bit (textsegment in swap,
                       Directory: Schreibzugriff für Files nur für
                                  Eigentümer und su)
Rückkehrkode:
     >=0   - Filedescriptor für das eröffnete File
     < 0   - Fehler
             EACCESS  -   no search access, O_TRUNC and no
                          write-permission, no permission
             EDQUOT   -   file not exist, no quota
             EEXIST   -   O_EXCL and O_CREAT and file exist
             EFAULT   -   wrong address
             EINTR    -   signal
             EISDIR   -   file is directory and write
             ELOOP    -   too many symbolic links
             EMFILE   -   no filedescriptor
             ENAMETOOLONG- path to long (PAH_MAX)
             ENFILE   -   system file table full
             ENOENT   -   file not exist and no O_CREAT
             ENOSPC   -   no space in directory, no inode
             ENOTDIR  -   component in the path is not a directory
             EROFS    -   O_CREAT and read only file system
```


Kerneldatenstrukturen nach Open

ein Prozess mit Zugriff auf zwei Files
IO4

zwei unabhängige Prozesse mit Zugriff auf ein File
IO5

Vater- und Kindprozess mit Zugriff auf drei Files
IO6

```
#include <unistd.h>

int close(int filedes);
```

Schliessen eines Files. Freigabe aller Ressourcen einschliesslich "gelockter" Bereiche des Prozesses.

```
filedes   -  Filedescriptor

Rückkehrwert:
          0  - ok
         <0  - Fehler
               EBADF   - fieldes ist kein eröffnetes File
               EINTR   - Signal aufgetreten
```



```
#include <unistd.h>

ssize_t read(int filedes, void *buff, size_t nbytes);
```

read liest die Anzahl von nbytes Bytes des eröffneten Files filedes in den Puffer *buff. Der Rückkehrwert gibt die Anzahl der gelesenen Bytes an. Die Anzahl der gelesenen Bytes kann von der geforderten Zahl von Bytes abweichen wenn:

- Fileende
- wenn von einem Terminal gelesen wird
- wenn von einem Socket gelesen wird
- Blockorientierte Geräte können nur Blöcke liefern

```
Rückkehrwert:

     > 0  - Anzahl der gelesenen Bytes
       0  - Fileende  (EOF)
      <0  - Fehler
             EAGAIN  -  non-blocking File ohne Daten
             EBADF   -  Filedes ist kein eröffnetes File
             EFAULT  -  *buff nicht im Adressraum
             EINTR   -  Signal
             EINVAL  -  negativer Filedescriptor
             EIO     -  E/A-Fehler
```

```
#include <unistd.h>

ssiz_t write(int filedes, const void *buff, size_t nbytes);
```

write schreibt die in nbytes spezifizierte Anzahl von Bytes aus dem Puffer *buff in das File filedes.

Rückkehrwert:

```
>=0   - Anzahl der geschriebenen Bytes
< 0   - Fehler
         EBADF   - filedes ist kein eröffnetes File
         EDQUOT  - Quotafehler
         EFAULT  - *buff nicht im Adressraum
         EFBIG   - Filegrösse zu gross
         EINTR   - Signal
         EINVAL  - negativer Filedescriptor
         EIO     - E/A-Fehler
         ENOSPC  - Filesystem ist voll
```


Einfaches Beispiel:

Kopieren von Standardeingabe nach Standardausgabe:

```
./copy <Eingabefile >Ausgabefile
```

Kopieren mit unterschiedlichen Blockgrößen

```
./copy1 blocksize <Eingabefile >Ausgabefile
```

```
#include <unistd.h>
#include <sys/ioctl.h>

int ioctl(int filedes, int request, ...);
```

ioctl erlaubt die Ausführung von Steueroperationen uber dem File filedes. request spezifiziert die Art der Steueroperation. Ein weiterer Parameter (int-Wert oder Adresse eines Feldes) sind zulässig.

Rückkehrwert:

```
     >=0  - ok
      <0  - Fehler
            EBADF   -  filedes ist kein erööffnetes File
            EFAULT  -  Falsche Parameterkombination
            EINVAL  -  request nicht unterstützt
            ENOTTY  -  request passt nicht zum File
```



```
/*      @(#)mtio.h 2.24 91/06/26 SMI; from UCB 4.10 83/01/17    */
/*
 * Structures and definitions for mag tape io control commands
 */
/* structure for MTIOCTOP - mag tape op command */
struct  mtop    {
        short   mt_op;          /* operations defined below */
        daddr_t mt_count;       /* how many of them */
};
/* operations */
#define MTWEOF          0       /* write an end-of-file record */
#define MTFSF           1       /* forward space over file mark */
#define MTBSF           2       /* backward space over file mark (1/2"
#define MTFSR           3       /* forward space to inter-record gap *
#define MTBSR           4       /* backward space to inter-record gap
#define MTREW           5       /* rewind */
#define MTOFFL          6       /* rewind and put the drive offline */
#define MTNOP           7       /* no operation, sets status only */
#define MTRETEN         8       /* retension the tape (cartridge tape
#define MTERASE         9       /* erase the entire tape */
#define MTEOM           10      /* position to end of media */
#define MTNBSF          11      /* backward space file to BOF */
/*      @(#)ttycom.h 1.10 89/06/23 SMI  */
#define TIOCGWINSZ      _IOR(t, 104, struct winsize)    /* get window
#define TIOCSWINSZ      _IOW(t, 103, struct winsize)    /* set window
#define TIOCSSIZE       _IOW(t,37,struct ttysize)/* set tty size */
#define TIOCGSIZE       _IOR(t,38,struct ttysize)/* get tty size */
```

```
/*
 * 4.3BSD and SunOS terminal "ioctl"s with no "termios" equivalents.
 * This file is included by <sys/termios.h> and indirectly by <sys/ioc
 * so that programs that include either one have these "ioctl"s define
 */
#define TIOCSCTTY       _IO(t, 132)             /* get a ctty */
#define TIOCGPGRP       _IOR(t, 119, int)       /* get pgrp of tty */
#define TIOCGETPGRP     _IOR(t, 131, int)       /* get pgrp of tty (po
#define TIOCSPGRP       _IOW(t, 118, int)       /* set pgrp of tty */
#define TIOCSETPGRP     _IOW(t, 130, int)       /* set pgrp of tty (po
#define TIOCOUTQ        _IOR(t, 115, int)       /* output queue size *
#define TIOCSTI         _IOW(t, 114, char)      /* simulate terminal i
#define TIOCNOTTY       _IO(t, 113)             /* void tty associatio
#define TIOCPKT         _IOW(t, 112, int)       /* pty: set/clear pack
#define TIOCPKT_DATA            0x00    /* data packet */
#define TIOCPKT_FLUSHREAD       0x01    /* flush data not yet written
#define TIOCPKT_FLUSHWRITE      0x02    /* flush data read from contro
#define TIOCPKT_STOP            0x04    /* stop output */
#define TIOCPKT_START           0x08    /* start output */
#define TIOCPKT_NOSTOP          0x10    /* no more ^S, ^Q */
#define TIOCPKT_DOSTOP          0x20    /* now do ^S, ^Q */
#define TIOCPKT_IOCTL           0x40    /* "ioctl" packet */
#define TIOCMSET        _IOW(t, 109, int)       /* set all modem bits
#define TIOCMBIS        _IOW(t, 108, int)       /* bis modem bits */
#define TIOCMBIC        _IOW(t, 107, int)       /* bic modem bits */
#define TIOCMGET        _IOR(t, 106, int)       /* get all modem bits
#define TIOCM_LE        0001            /* line enable */
#define TIOCM_DTR       0002            /* data terminal ready */
#define TIOCM_RTS       0004            /* request to send */
#define TIOCM_ST        0010            /* secondary transmit */
```



```
#define TIOCM_SR        0020            /* secondary receive */
#define TIOCM_CTS       0040            /* clear to send */
#define TIOCM_CAR       0100            /* carrier detect */
#define TIOCM_CD        TIOCM_CAR
#define TIOCM_RNG       0200            /* ring */
#define TIOCM_RI        TIOCM_RNG
#define TIOCM_DSR       0400            /* data set ready */
#define TIOCREMOTE      _IOW(t, 105, int)       /* remote input editin
#define TIOCUCNTL       _IOW(t, 102, int)       /* pty: set/clr usr cn
```

```
/*
 * Sun-specific ioctls with no "termios" equivalents.
 */
#define TIOCTCNTL       _IOW(t, 32, int)        /* pty: set/clr interc
#define TIOCSIGNAL      _IOW(t, 33, int)        /* pty: send signal to
#define TIOCCONS        _IO(t, 36)              /* get console I/O */
#define TIOCSSOFTCAR    _IOW(t, 101, int)       /* set soft carrier fl
#define TIOCGSOFTCAR    _IOR(t, 100, int)       /* get soft carrier fl
#define TIOCISPACE      _IOR(t, 128, int)       /* space left in input
#define TIOCISIZE       _IOR(t, 129, int)       /* size of input queue
#define TIOCSINTR       _IOW(t, 99, int)        /* set DOS interrupt n
#define TCXONC          _IO(T, 6)
#define TCFLSH          _IO(T, 7)
#define TCGETS          _IOR(T, 8, struct termios)
#define TCSETS          _IOW(T, 9, struct termios)
#define TCSETSW         _IOW(T, 10, struct termios)
#define TCSETSF         _IOW(T, 11, struct termios)
#define TCOOFF          0               /* arg to TCXONC & tcflow() */
#define TCOON           1               /* arg to TCXONC & tcflow() */
#define TCIOFF          2               /* arg to TCXONC & tcflow() */
#define TCION           3               /* arg to TCXONC & tcflow() */
#define TCIFLUSH        0               /* arg to TCFLSH & tcflush() *
#define TCOFLUSH        1               /* arg to TCFLSH & tcflush() *
#define TCIOFLUSH       2               /* arg to TCFLSH & tcflush() *
#define TCSANOW         0               /* arg to tcsetattr() */
#define TCSADRAIN       1               /* arg to tcsetattr() */
#define TCSAFLUSH       2               /* arg to tcsetattr() */
#define TCGETA  _IOR(T, 1, struct termio)
#define TCSETA  _IOW(T, 2, struct termio)
#define TCSBRK  _IO(T, 5)
```



```
/*      @(#)sockio.h 1.7 88/12/06 SMI; from UCB ioctl.h 7.1 6/4/86
/*
 * General socket ioctl definitions.
 */
/* socket i/o controls */
#define SIOCSHIWAT      _IOW(s,  0, int)                /* set high wa
#define SIOCGHIWAT      _IOR(s,  1, int)                /* get high wa
#define SIOCSLOWAT      _IOW(s,  2, int)                /* set low wat
#define SIOCGLOWAT      _IOR(s,  3, int)                /* get low wat
#define SIOCATMARK      _IOR(s,  7, int)                /* at oob mark
#define SIOCSPGRP       _IOW(s,  8, int)                /* set process
#define SIOCGPGRP       _IOR(s,  9, int)                /* get process

#define SIOCADDRT       _IOW(r, 10, struct rtentry)     /* add route *
#define SIOCDELRT       _IOW(r, 11, struct rtentry)     /* delete rout

#define SIOCSIFADDR     _IOW(i, 12, struct ifreq)       /* set ifnet a
#define SIOCGIFADDR     _IOWR(i,13, struct ifreq)       /* get ifnet a
#define SIOCSIFDSTADDR  _IOW(i, 14, struct ifreq)       /* set p-p add
#define SIOCGIFDSTADDR  _IOWR(i,15, struct ifreq)       /* get p-p add
#define SIOCSIFFLAGS    _IOW(i, 16, struct ifreq)       /* set ifnet f
#define SIOCGIFFLAGS    _IOWR(i,17, struct ifreq)       /* get ifnet f
#define SIOCSIFMEM      _IOW(i, 18, struct ifreq)       /* set interfa
#define SIOCGIFMEM      _IOWR(i,19, struct ifreq)       /* get interfa
#define SIOCGIFCONF     _IOWR(i,20, struct ifconf)      /* get ifnet l
#define SIOCSIFMTU      _IOW(i, 21, struct ifreq)       /* set if_mtu
#define SIOCGIFMTU      _IOWR(i,22, struct ifreq)       /* get if_mtu
```

```
        /* from 4.3BSD */
#define SIOCGIFBRDADDR  _IOWR(i,23, struct ifreq)       /* get broadca
#define SIOCSIFBRDADDR  _IOW(i,24, struct ifreq)        /* set broadca
#define SIOCGIFNETMASK  _IOWR(i,25, struct ifreq)       /* get net add
#define SIOCSIFNETMASK  _IOW(i,26, struct ifreq)        /* set net add
#define SIOCGIFMETRIC   _IOWR(i,27, struct ifreq)       /* get IF metr
#define SIOCSIFMETRIC   _IOW(i,28, struct ifreq)        /* set IF metr
#define SIOCSARP        _IOW(i, 30, struct arpreq)      /* set arp ent
#define SIOCGARP        _IOWR(i,31, struct arpreq)      /* get arp ent
#define SIOCDARP        _IOW(i, 32, struct arpreq)      /* delete arp
#define SIOCUPPER       _IOW(i, 40, struct ifreq)       /* attach uppe
#define SIOCLOWER       _IOW(i, 41, struct ifreq)       /* attach lowe
#define SIOCSETSYNC     _IOW(i,  44, struct ifreq)      /* set syncmod
#define SIOCGETSYNC     _IOWR(i, 45, struct ifreq)      /* get syncmod
#define SIOCSSDSTATS    _IOWR(i, 46, struct ifreq)      /* sync data s
#define SIOCSSESTATS    _IOWR(i, 47, struct ifreq)      /* sync error
#define SIOCSPROMISC    _IOW(i, 48, int)                /* request pro
#define SIOCADDMULTI    _IOW(i, 49, struct ifreq)       /* set m/c add
#define SIOCDELMULTI    _IOW(i, 50, struct ifreq)       /* clr m/c add
/* FDDI controls */
#define SIOCFDRESET     _IOW(i, 51, struct ifreq)       /* Reset FDDI
#define SIOCFDSLEEP     _IOW(i, 52, struct ifreq)       /* Sleep until
#define SIOCSTRTFMWAR   _IOW(i, 53, struct ifreq)       /* Start FW at
#define SIOCLDNSTRTFW   _IOW(i, 54, struct ifreq)       /* Load the sh
#define SIOCGETFDSTAT   _IOW(i, 55, struct ifreq)       /* Get FDDI st
#define SIOCFDNMIINT    _IOW(i, 56, struct ifreq)        /* NMI to fdd
#define SIOCFDEXUSER    _IOW(i, 57, struct ifreq)        /* Exec in us
#define SIOCFDGNETMAP   _IOW(i, 58, struct ifreq)        /* Get a netm
#define SIOCFDGIOCTL    _IOW(i, 59, struct ifreq)        /* Generic io
```



```
/*      @(#)filio.h 1.5 91/06/18 SMI; from UCB ioctl.h 7.1 6/4/86
/*
 * General file ioctl definitions.
 */

#define FIOCLEX         _IO(f, 1)               /* set exclusive use o
#define FIONCLEX        _IO(f, 2)               /* remove exclusive us

/* another local */

#define FIONREAD        _IOR(f, 127, int)       /* get # bytes to read
#define FIONBIO         _IOW(f, 126, int)       /* set/clear non-block
#define FIOASYNC        _IOW(f, 125, int)       /* set/clear async i/o
#define FIOSETOWN       _IOW(f, 124, int)       /* set owner */
#define FIOGETOWN       _IOR(f, 123, int)       /* get owner */

/* file system locking */

#define FIOLFS          _IO(f, 64)              /* file system lock */
#define FIOLFSS         _IO(f, 65)              /* file system lock st
#define FIOFFS          _IO(f, 66)              /* file system flush *

/* short term backup */

#define FIOAI           _IO(f, 67)              /* allocation informat
#define FIODUTIMES      _IO(f, 68)              /* delay update access
#define FIODIO          _IO(f, 69)              /* delay write all dat
#define FIODIOS         _IO(f, 70)              /* status of FIODIO */
```