BERGISCHE UNIVERSITÄT
GESAMTHOCHSCHULE WUPPERTAL
GAUSS-STRASSE 20
42097 WUPPERTAL
(Korrespondenzanschrift)
42119 WUPPERTAL
(Lieferanschrift)
TELEX 8 592 262 bughw
TELEFAX (0202) 439-2901
TELEFON (0202) 439-1

Fachbereich 7

MATHEMATIK

Prof. Dr. Hans-Jürgen Buhl
Praktische Informatik / Numerik

e-mail: Juergen.Buhl@math.uni-wuppertal.de

Betriebssysteme: Konzepte, Dienste, Schnittstellen (Betriebssysteme und betriebssystemnahe Programmierung)

SS 2003 – Übungsblatt 3

28. Mai 2003
Ausgabe: 21. Mai 2003

Aufgabe 1. *file/record locking*

Geben Sie das folgende C-Programm ein und bringen Sie es zur Ausführung:

```
/* lock.c -- simple example of record locking */

#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>

/* displays the message, and waits for the user to press
   return */
void waitforuser(char * message) {
    char buf[10];

    printf("%s", message);
    fflush(stdout);

    fgets(buf, 9, stdin);
}

/* Gets a lock of the indicated type on the fd which is passed.
   The type should be either F_UNLCK, F_RDLCK, or F_WRLCK */
void getlock(int fd, int type) {
    struct flock lockinfo;
    char message[80];
```

```
    /* we'll lock the entire file */
    lockinfo.l_whence = SEEK_SET;
    lockinfo.l_start = 0;
    lockinfo.l_len = 0;

    /* keep trying until we succeed */
    while (1) {
        lockinfo.l_type = type;
/* if we get the lock, return immediately */
        if (!fcntl(fd, F_SETLK, &lockinfo)) return;

/* find out who holds the conflicting lock */
        fcntl(fd, F_GETLK, &lockinfo);

/* there's a chance the lock was freed between the F_SETLK
   and F_GETLK; make sure there's still a conflict before
   complaining about it */
        if (lockinfo.l_type != F_UNLCK) {
            sprintf(message, "conflict with process %d... press "
                    "<return> to retry:", lockinfo.l_pid);
            waitforuser(message);
        }
    }
}

int main(void) {
    int fd;

    /* set up a file to lock */
    fd = open("testlockfile", O_RDWR | O_CREAT, 0666);
    if (fd < 0) {
        perror("open");
        return 1;
    }

    printf("getting read lock\n");
    getlock(fd, F_RDLCK);
    printf("got read lock\n");

    waitforuser("\npress <return> to continue:");

    printf("releasing lock\n");
    getlock(fd, F_UNLCK);

    printf("getting write lock\n");
    getlock(fd, F_WRLCK);
    printf("got write lock\n");
```

```
    waitforuser("\npress <return> to exit:");

    /* locks are released when the file is closed */

    return 0;
}
```

Versuchen Sie in einem zweiten Terminalfenster das Programm ein zweites Mal zu starten und hier eine Schreibsperrung zu bekommen, während im anderen Prozess noch eine Lesesperrung aktiv ist. Was geschieht? Untersuchen Sie (in einem zweiten Terminalfenster) mittels `strace` bzw. `truss` die Systemaufrufe bei Ausführung von `lock`.

Erzeugen Sie eine statisch gelinkte Version von `lock` und untersuchen Sie mittels `readelf` bzw. `dump` den Inhalt der ELF-Datei im Vergleich zum Inhalt der dynamisch gelinkten Version. Wofür steht das Acronym ELF?

Aufgabe 2. *Zeitmessung im Anwenderprogramm*

Das Programm

```
////////////////////////////////
// Datei:   TakeTime.cc
// Version: 1.0
// Zweck:   Zeitmessung von Algorithmenteilen
// Autor:   Hans-Juergen Buhl
// Datum:   20.05.2003
////////////////////////////////


#include <iostream>
#include <iomanip>
#include <cmath>

#include <time.h>
#include <sys/times.h>
#include <sys/types.h>
#include <unistd.h>


using namespace std;

#ifndef CLK_TCK
        double CLK_TCK = sysconf(_SC_CLK_TCK);
#endif

double get_Time(void){

  double tim = -1.0;
```

```
  tms time_buff;

  if (times(&time_buff) < 0) {
cerr << " times() failed " << endl;
  } else {
   double total_t = time_buff.tms_utime +
 time_buff.tms_stime +
 time_buff.tms_cutime +
 time_buff.tms_cstime ;
tim = total_t /double(CLK_TCK);
  };
  return tim;

}

int main()
{
  double StartTime(get_Time());

  // ...
  double f(sin(1.4567));
  for (int i = 0; i<1000000; i++) f = sin(f);
  // ...

  double Time(get_Time()-StartTime);

  cout <<  "Der Algorithmus dauerte "
       <<  Time
       <<  " CPU-Sekunden"
       <<  " ( Auflösung: " << 1000.0 / double(CLK_TCK) << " ms )"
       <<  endl;

// cout << CLK_TCK << endl << CLOCKS_PER_SEC << endl;

}
```

mißt die Ausführungszeit von Codesegmenten. Bringen Sie es auf Ihrem Rechner zum Laufen. Warum werden hier vier verschiedene Zeitanteile aufaddiert?

Aufgabe 3. *system()*

In

```
#include <stdio.h>
#include <stdlib.h>

int main(){
```

```
   if (system("rm x.txt") == 0)
      printf("Datei x.txt wurde gelöscht\n");
   else
      printf("Datei x.txt konnte   n i c h t   gelöscht werden\n");
}
```

wird das UNIX-Shellkommando `rm` innerhalb eines Programms benutzt. Was genau wird beim Programmlauf ausgegeben, wenn eine Datei `x.txt` existiert bzw. nicht existiert?

Aufgabe 4. *curses/ncurses*

Textaus- und -eingabe mit Cursorsteuerung erlaubt Ihnen die Terminal-Bibliothek `curses` auf SOLARIS bzw. `ncurses` auf LINUX. Geben Sie bitte das folgende Programm als `curses.cc` ein, übersetzen Sie es dann mittels

```
CC -g -I/opt/local/include -o curses curses.cc -lcurses
```

auf Solaris beziehungsweise

```
g++ -g -I/opt/local/include -o curses curses.cc -lncurses
```

auf Linux-Systemen. (Ignorieren Sie eventuell auftretende Warnungen des Compilers). Lassen Sie es von einem `xterm`-Fenster aus ablaufen.

```
#include <iostream>
#include <cstdlib>

// #include <curses.h>                          für CC
// #include "/opt/local/gnu/include/curses.h"   für g++
//     (oder <curses.h> und g++ -g -I/opt/local/gnu/include ...

#ifdef __sparc
#ifdef  __SUNPRO_CC
#include <curses.h>
#endif
#ifdef __GNUC__
#include "/opt/local/gnu/include/curses.h"
#endif
#else
#include <curses.h>
#endif

#include <csignal>

using namespace std;

// Übersetze mittels:
//   CC  -g -o 8_1 8_1.cc -lcurses   (Sun)
```

```
//   g++ -g -o 8_1 8_1.cc -lcurses
// bzw.
//   g++ -g -o 8_1 8_1.cc -lncurses (LINUX)

extern "C" void finish(int sig);

int main(){

    int num = 0;

    /* initialize your non-curses data structures here */
    // ...

    (void) signal(SIGINT, finish);      /* arrange interrupts to terminate */

    (void) initscr();      /* initialize the curses library */
    keypad(stdscr, TRUE);  /* enable keyboard mapping */
    (void) nonl();         /* tell curses not to do NL->CR/NL on output */
    (void) cbreak();       /* take input chars one at a time, no wait for \n */
    (void) echo();         /* echo input - in color */

    if (has_colors())
    {
        start_color();

        /*
         * Simple color assignment, often all we need.  Color pair 0 cannot
         * be redefined.  This example uses the same value for the color
         * pair as for the foreground color, though of course that is not
         * necessary:
         */
        init_pair(1, COLOR_RED,     COLOR_BLACK);
        init_pair(2, COLOR_GREEN,   COLOR_BLACK);
        init_pair(3, COLOR_YELLOW,  COLOR_BLACK);
        init_pair(4, COLOR_BLUE,    COLOR_BLACK);
        init_pair(5, COLOR_CYAN,    COLOR_BLACK);
        init_pair(6, COLOR_MAGENTA, COLOR_BLACK);
        init_pair(7, COLOR_WHITE,   COLOR_BLACK);
    }

    mvaddstr( 3, 10, (char *)"Dies ist ein Programm mit Menu:");
    mvaddstr(10, 10, (char *)"Bitte geben sie einen der Buchstaben a,b oder c");
    mvaddstr(11, 10, (char *)"(oder q zum Programmabbruch) ein:");

    for (;;)
    {
        move(11, 44);
  // refresh();

        int c = getch();     /* refresh, accept single keystroke of input */
        attrset(COLOR_PAIR(num % 8));
        num++;
```

```
        /* process the command keystroke */

        mvaddstr(15,1, (char *)" ");
        clrtoeol();

        if ((c == 'a')||(c == 'b')||(c=='c')){
           mvaddstr(15, 20, (char *)"Sie drückten die Taste: ");
                addch(c);
        };
if (c == 'q') { beep(); finish(0);};
    }

    finish(0);               /* we're done */
}

extern "C" void finish(int sig)
{
    endwin();

    /* do your non-curses wrapup here */
    // ...
    cout << "Programm beendet!" << endl;

    exit(0);
}
```

Versuchen Sie, es Zeile für Zeile mit Hilfe der Kommentare, der Ergebnisse von `man curses`, `man mvaddstr`, ... beziehungsweise des Webs zu verstehen.

Schreiben Sie anschließend (in Aufsatzform) eine Beschreibung des Programms, die dessen Wirkungsweise Zeile für Zeile erklärt.

Aufgabe 5. *walltime*

Verfahren Sie analog wie in Aufgabe 4 mit dem Programm:

```
#include <stdio.h>
#include <sys/time.h>
#include <unistd.h>

int main() {
   struct timeval tv;
   struct timezone tz;
   time_t now;
   /* beginning_of_time is smallest time_t-sized value */
   time_t beginning_of_time = 1L<<(sizeof(time_t)*8 - 1);
   /* end_of_time is largest time_t-sized value */
   time_t end_of_time = ~beginning_of_time;
```

```
    printf("time_t is %d bits long\n\n", sizeof(time_t)*8);

    gettimeofday(&tv, &tz);
    now = tv.tv_sec;
    printf("Current time of day represented as a struct timeval:\n"
           "tv.tv_sec = 0x%08x, tv.tv_usec = 0x%08x\n"
           "tz.tz_minuteswest = 0x%08x, tz.tz_dsttime = 0x%08x\n\n",
           tv.tv_sec, tv.tv_usec, tz.tz_minuteswest, tz.tz_dsttime);

    printf("Demonstrating ctime()%s:\n", sizeof(time_t)*8 <= 32 ? "" :
           " (may hang after printing first line; press control-C)");
    printf("time is now %s", ctime(&now));
    printf("time begins %s", ctime(&beginning_of_time));
    printf("time ends %s", ctime(&end_of_time));

    exit (0);
}
```