BERGISCHE UNIVERSITÄT
WUPPERTAL
GAUSS-STRASSE 20
42119 WUPPERTAL
TELEFAX (0202) 439-2901
TELEFON (0202) 439-0
WWW www.uni-wuppertal.de

Fachbereich C

MATHEMATIK UND NATURWISSENSCHAFTEN
Fachgruppe Mathematik und Informatik

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

e-mail: Hans-Juergen.Buhl@math.uni-wuppertal.de
WWW: http://www.math.uni-wuppertal.de/∼buhl

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

SS 2005 – Übungsblatt 9

Ausgabe: 20. Juni 2005

Abgabe: bis spätestens 27. Juni 2005
im Fachschaftsraum Mathematik
oder per email an `c.markmann@uni-wuppertal.de`

Aufgabe 1. *net-tools*

Besorgen Sie sich analog zum letzten Übungsblatt die `net-tools`. Entpacken und studieren Sie die Datei `hostname.c`.

Versuchen Sie den Inhalt von `hostname.c` in groben Zügen funktional zu beschreiben.

Aufgabe 2. *setjmp*

Bringen Sie das folgende Programm zum Ablauf

```
#include        <setjmp.h>

static void     f1(int, int, int);
static void     f2(void);

static jmp_buf  jmpbuffer;

int
main(void)
{
        int             count;
        register int    val;
```

```
        volatile int    sum;

        count = 2; val = 3; sum = 4;
        if (setjmp(jmpbuffer) != 0) {
                printf("after longjmp: count = %d, val = %d, sum = %d\n",
                                count, val, sum);
                exit(0);
        }
        count = 97; val = 98; sum = 99;
                                /* changed after setjmp, before longjmp */
        f1(count, val, sum);    /* never returns */
}

static void
f1(int i, int j, int k)
{
        printf("in f1(): count = %d, val = %d, sum = %d\n", i, j, k);
        f2();
}

static void
f2(void)
{
        longjmp(jmpbuffer, 1);
}
```

und erklären Sie seine Wirkungsweise Zeile für Zeile.

Warum sollte diese Funktionalität so wenig wie möglich eingesetzt werden?

Aufgabe 3. *atexit*

Bringen Sie das folgende Programm zum Ablauf

```
#include        <stdio.h>
static void     my_exit1(void), my_exit2(void);
int main(void)
{
        if (atexit(my_exit2) != 0){
                fprintf(stderr, "%s\n", "can't register my_exit2");
                exit(1);
        }
        if (atexit(my_exit1) != 0){
                fprintf(stderr, "%s\n", "can't register my_exit1");
                exit(1);
        }
        if (atexit(my_exit1) != 0){
                fprintf(stderr, "%s\n", "can't register my_exit1");
```

```
                exit(1);
        }
        if (atexit(my_exit1) != 0){
                fprintf(stderr, "%s\n", "can't register my_exit1");
                exit(1);
        }
        if (atexit(my_exit2) != 0){
                fprintf(stderr, "%s\n", "can't register my_exit2");
                exit(1);
        }
        if (atexit(my_exit1) != 0){
                fprintf(stderr, "%s\n", "can't register my_exit1");
                exit(1);
        }
        printf("main is done\n");
        return(0);
}

static void
my_exit1(void)
{
        printf("first exit handler\n");
}
static void
my_exit2(void)
{
        printf("second exit handler\n");
}
```

und erklären Sie seine Wirkungsweise Zeile für Zeile.

In welcher Form sollte `atexit()` in Aufgabe 1 von Übungsblatt 7 eingesetzt werden?

Aufgabe 4. *getrlimit*

Bringen Sie das folgende Programm zum Ablauf

```
#include        <sys/types.h>
#include        <sys/time.h>
#include        <sys/resource.h>
#include        <stdio.h>

#define doit(name)      pr_limits(#name, name)

static void     pr_limits(char *, int);

int
main(void)
```

```
{
        doit(RLIMIT_CORE);
        doit(RLIMIT_CPU);
        doit(RLIMIT_DATA);
        doit(RLIMIT_FSIZE);
#ifdef  RLIMIT_MEMLOCK
        doit(RLIMIT_MEMLOCK);
#endif
#ifdef  RLIMIT_NOFILE   /* SVR4 name */
        doit(RLIMIT_NOFILE);
#endif
#ifdef  RLIMIT_OFILE    /* 44BSD name */
        doit(RLIMIT_OFILE);
#endif
#ifdef  RLIMIT_NPROC
        doit(RLIMIT_NPROC);
#endif
#ifdef  RLIMIT_RSS
        doit(RLIMIT_RSS);
#endif
        doit(RLIMIT_STACK);
#ifdef  RLIMIT_VMEM
        doit(RLIMIT_VMEM);
#endif
        exit(0);
}

static void
pr_limits(char *name, int resource)
{
        struct rlimit   limit;

        if (getrlimit(resource, &limit) < 0){
                fprintf(stderr, "getrlimit error for %s", name);
                exit(1);
        }

        printf("%-14s  ", name);
        if (limit.rlim_cur == RLIM_INFINITY)
                printf("(infinite)  ");
        else
                printf("%10ld  ", limit.rlim_cur);
        if (limit.rlim_max == RLIM_INFINITY)
                printf("(infinite)\n");
        else
                printf("%10ld\n", limit.rlim_max);
}
```

und erklären Sie seine Wirkungsweise Zeile für Zeile.

Erstellen Sie eine Tabelle, die die verfügbaren limitierbaren Ressourcen übersichtlich darstellt.

Kann man Prozessressourcen-Limits auch in einer UNIX-Shell für diese und alle Kindprozesse limitieren (wenn ja: wie?)?

Aufgabe 5. *Terminal character size*

Bringen Sie das folgende Programm zum Ablauf

```
#include        <stdio.h>
#include        <termios.h>
#include        <unistd.h>

int
main(void)
{
        struct termios  term;
        int             size;

        if (tcgetattr(STDIN_FILENO, &term) < 0){
                fprintf(stderr, "%s\n", "tcgetattr error");
                exit(1);
        }
        size = term.c_cflag & CSIZE;
        if      (size == CS5)   printf("5 bits/byte\n");
        else if (size == CS6)   printf("6 bits/byte\n");
        else if (size == CS7)   printf("7 bits/byte\n");
        else if (size == CS8)   printf("8 bits/byte\n");
        else                    printf("unknown bits/byte\n");

        term.c_cflag &= ~CSIZE;         /* zero out the bits */
        term.c_cflag |= CS8;            /* set 8 bits/byte */

        if (tcsetattr(STDIN_FILENO, TCSANOW, &term) < 0){
                fprintf(stderr, "%s\n", "tcsetattr error");
                exit(1);
        }
        exit(0);
}
```

und erklären Sie seine Wirkungsweise Zeile für Zeile.

In welchen Zusammenhängen ist es wichtig, zwischen 5..8-Bit/Byte zu unterscheiden?