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 9

16. Juli 2003
Ausgabe: 9. Juli 2003

Aufgabe 1. *argv und environ*

Bringen Sie die folgenden Programme zum Ablauf

```
#include        "stdio.h"

int
main(int argc, char *argv[])
{
        int      i;

        for (i = 0; i < argc; i++)   /* echo all command-line args */
            printf("argv[%d]: %s\n", i, argv[i]);
        exit(0);
}
```

sowie

```
#include        "stdio.h"

int
main(int argc, char *argv[])
{
        int             i;
        char            **ptr;
        extern char     **environ;

        for (i = 0; i < argc; i++)    /* echo all command-line args */
```

```
        printf("argv[%d]: %s\n", i, argv[i]);

    for (ptr = environ; *ptr != 0; ptr++) /* and all env strings */
        printf("%s\n", *ptr);

    exit(0);
}
```

und erklären Sie ihre jeweilige Wirkungsweise Zeile für Zeile. Wann sollten die Werte der Environmentvariablen genutzt werden? Welche Probleme könnten entstehen?

Aufgabe 2. *tcsetattr und Passworteingabe*

Benutzen Sie `tcsetattr()` nach dem Muster von Aufgabe 2 / Übungsblatt 5, um eine Funktion `getpass()` ähnlich zu Aufgabe 1 / Übungsblatt 7 zu erstellen, die jedoch einen `raw`-Modus (vgl. Aufgabe 1 / Übungsblatt 5) realisiert.

Aufgabe 3. *vfork*

Bringen Sie das folgende Programm zum Ablauf

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

int     glob1 = 6;          /* external variable in initialized data */

int
main(void)
{
        int     var;        /* automatic variable on the stack */
        pid_t   pid;

        var = 88;
        printf("before vfork\n");       /* we don't flush stdio */

        if ( (pid = vfork()) < 0){
                perror("vfork error");
                exit(1);
        } else if (pid == 0) {          /* child */
                glob1++;                /* modify parent's variables */
                var++;
                printf("In child: pid = %d, glob1 = %d, var = %d\n",
                       getpid(), glob1, var);
                _exit(0);               /* child terminates */
        }

        /* parent */
        printf("In parent: pid = %d, glob1 = %d, var = %d\n",
               getpid(), glob1, var);
```

```
        exit(0);
}
```

und erklären Sie seine Wirkungsweise Zeile für Zeile.

Aufgabe 4. *fork*

Bringen Sie das folgende Programm zum Ablauf

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

int     glob1 = 6;         /* external variable in initialized data */
char    buf[] = "a write to stdout\n";

int
main(void)
{
        int     var;         /* automatic variable on the stack */
        pid_t   pid;

        var = 88;
        if (write(STDOUT_FILENO, buf, sizeof(buf)-1) != sizeof(buf)-1){
                perror("write error");
exit(1);
        }
        printf("before fork\n");        /* we don't flush stdout */

        if ( (pid = fork()) < 0){
                perror("fork error");
                exit(1);
        } else if (pid == 0) {          /* child */
                glob1++;                /* modify variables */
                var++;
        } else
                sleep(2);               /* parent */

        printf("pid = %d, glob1 = %d, var = %d\n", getpid(), glob1, var);
        exit(0);
}
```

und erklären Sie seine Wirkungsweise Zeile für Zeile.