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 11

Ausgabe: 4. Juli 2005

Abgabe: bis spätestens 11. Juli 2005
im Fachschaftsraum Mathematik
oder per email an `c.markmann@uni-wuppertal.de`

Aufgabe 1. *Passworteingabe: Ergänzung um Handler für SIGINT*

Ergänzen Sie das Programm aus Aufgabe 4 / Übungsblatt 10 um einen Handler für das Signal SIGINT, der die Zeichenkette `"interrupt"` ausgibt und das Programm dann beendet.

Hinweise: http://www.gnu.org/software/libc/manual/html_node/Signal-Handling.html

Aufgabe 2. *tcsetattr und Passworteingabe*

Modifizieren Sie die Funktion `getpass()` in Aufgabe 1 so, dass der `raw`-Modus der tty-Schnittstelle benutzt wird (kein Inline-Editieren des einzugebenden Passworts mehr möglich).

Experimentieren Sie mit Timeouts (`VTIME`) für die Passworteingabe.

Aufgabe 3. *termination code*

Bringen Sie das folgende Programm zum Ablauf

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

void pr_exit(int);
```

```
int main(void)
{
        int             status;

        if ( (status = system("date")) < 0){
                perror("system() error");
                exit(1);
        }
        pr_exit(status);
        if ( (status = system("nosuchcommand")) < 0){
                perror("system() error");
                exit(1);
        }
        pr_exit(status);

        if ( (status = system("who; exit 44")) < 0){
                perror("system() error");
                exit(1);
        }
        pr_exit(status);

        exit(0);
}

void pr_exit(int status){

        if (WIFEXITED(status))
                printf("normal termination, exit status = %d\n",
                        WEXITSTATUS(status));
        else if (WIFSIGNALED(status))
                printf("abnormal termination, signal number = %d%s\n",
                        WTERMSIG(status),
#ifdef WCOREDUMP
                    WCOREDUMP(status) ? " (core file generated)" : "");
#else
                        "");
#endif
        else if (WIFSTOPPED(status))
                printf("child stopped, signal number = %d\n",
                        WSTOPSIG(status));
}
```

und erklären Sie seine Wirkungsweise Zeile für Zeile.

Aufgabe 4. *abnormal termination*

Bringen Sie das folgende Programm zum Ablauf

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

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

void pr_exit(int);

int main(void)
{
        pid_t   pid;
        int             status;
        int zero = 0;

        if ( (pid = fork()) < 0){
                perror("fork error");
                exit(1);
        }

        else if (pid == 0)                      /* child */
                exit(7);

        if (wait(&status) != pid){              /* wait for child */
                perror("wait error");
                exit(1);
        }

        pr_exit(status);                /* and print its status */

        if ( (pid = fork()) < 0){
                perror("fork error");
                exit(1);
        }

        else if (pid == 0)              /* child */
                abort();                /* generates SIGABRT */

        if (wait(&status) != pid){      /* wait for child */
                perror("wait error");
                exit(1);
        }

        pr_exit(status);                /* and print its status */

        if ( (pid = fork()) < 0){
                perror("fork error");
                exit(1);
        }

        else if (pid == 0){             /* child */
                status /= zero;         /* divide by 0 generates SIGFPE */
        }
```

```
        if (wait(&status) != pid){      /* wait for child */
                perror("wait error");
                exit(1);
        }

        pr_exit(status);                /* and print its status */

        exit(0);
}


void pr_exit(int status){

        if (WIFEXITED(status))
                printf("normal termination, exit status = %d\n",
                        WEXITSTATUS(status));
        else if (WIFSIGNALED(status))
                printf("abnormal termination, signal number = %d%s\n",
                        WTERMSIG(status),
#ifdef WCOREDUMP
                  WCOREDUMP(status) ? " (core file generated)" : "");
#else
                        "");
#endif
        else if (WIFSTOPPED(status))
                printf("child stopped, signal number = %d\n",
                        WSTOPSIG(status));
}
```

und erklären Sie seine Wirkungsweise Zeile für Zeile.

Aufgabe 5. *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);
}
```

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