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 8

9. Juli 2003
Ausgabe: 2. Juli 2003

Aufgabe 1. *execlp*

Bringen Sie das folgende Programm zum Ablauf

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

#define MAXLINE 4096                    /* max line length */

int
main(void)
{
        char    buf[MAXLINE];
        pid_t   pid;
        int     status;

        printf("Input command: %% ");
        while (fgets(buf, MAXLINE, stdin) != NULL) {
                buf[strlen(buf) - 1] = 0; /* replace newline with null */

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

                else if (pid == 0) {            /* child */
                        execlp(buf, buf, (char *) 0);
                        fprintf(stderr, "couldn't execute: %s ", buf);
                        perror("execlp error");
```

```
                exit(127);
        }

        /* parent */
        if ( (pid = waitpid(pid, &status, 0)) < 0){
                perror("waitpid error");
                exit(1);
        }
        printf("%% ");
    }
    exit(0);
}
```

und erklären Sie seine Wirkungsweise Zeile für Zeile.

Aufgabe 2. *Ergänzung um Handler für SIGINT*

Ergänzen Sie das Programm aus Aufgabe 1 um einen Handler für das Signal SIGINT, der die Zeichenkette `"interrupt"` ausgibt und das Programm dann beendet.

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.