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 12

Ausgabe: 11. Juli 2005

Abgabe: bis spätestens 18. Juli 2005
im Fachschaftsraum Mathematik
oder per email an `c.markmann@uni-wuppertal.de`

Aufgabe 1. *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 2. *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.

Aufgabe 3. *create and join threads*

Bringen Sie das folgende Programm zum Ablauf

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

void print_message_function( void *ptr );

main(){
           pthread_t thread1, thread2;
           char *message1 = "Thread 1";
           char *message2 = "Thread 2";
           int  iret1, iret2;

    /* Create independant threads each of which will execute function */

           iret1 = pthread_create( &thread1, NULL,
                   (void*)&print_message_function, (void*) message1);
           iret2 = pthread_create( &thread2, NULL,
                   (void*)&print_message_function, (void*) message2);

   /* Wait till threads are complete before main continues. Unless we  */
   /* wait we run the risk of executing an exit which will terminate   */
   /* the process and all threads before the threads have completed.   */

           pthread_join( thread1, NULL);
           pthread_join( thread2, NULL);

           printf("Thread 1 returns: %d\n",iret1);
           printf("Thread 2 returns: %d\n",iret2);
           exit(0);
}

void print_message_function( void *ptr ){
           char *message;
           message = (char *) ptr;
           printf("%s \n", message);
}
```

und erklären Sie seine Wirkungsweise in Form eines Struktogramms mit Pseudocode.

Aufgabe 4. *mutex for synchronization*

Bringen Sie das folgende Programm zum Ablauf

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

      void *functionC();
      pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;
      int  counter = 0;

      main()
      {
         int rc1, rc2;
         pthread_t thread1, thread2;

   /* Create independant threads each of which will execute functionC */

         if( (rc1=pthread_create( &thread1, NULL, &functionC, NULL)) )
         {
            printf("Thread creation failed: %d\n", rc1);
         }

         if( (rc2=pthread_create( &thread2, NULL, &functionC, NULL)) )
         {
            printf("Thread creation failed: %d\n", rc2);
         }

   /* Wait till threads are complete before main continues. Unless we  */
   /* wait we run the risk of executing an exit which will terminate   */
   /* the process and all threads before the threads have completed.   */

         pthread_join( thread1, NULL);
         pthread_join( thread2, NULL);

         exit(0);
      }

      void *functionC()
      {
         pthread_mutex_lock( &mutex1 );
         counter++;
         printf("Counter value: %d\n",counter);
         pthread_mutex_unlock( &mutex1 );
      }
```

und erklären Sie seine Wirkungsweise Zeile für Zeile.

Aufgabe 5. *wait for 10 threads*

Bringen Sie das folgende Programm zum Ablauf

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

```
#define NTHREADS 10
void *thread_function();
pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;
int  counter = 0;

main()
{
   pthread_t thread_id[NTHREADS];
   int i, j;

   for(i=0; i < NTHREADS; i++)
   {
      pthread_create( &thread_id[i], NULL, &thread_function, NULL );
   }

   for(j=0; j < NTHREADS; j++)
   {
      pthread_join( thread_id[j], NULL);
   }

/* Now that all threads are complete I can print the final result.     */
/* Without the join I could be printing a value before all the threads */
/* have been completed.                                                */

   printf("Final counter value: %d\n", counter);
}

void *thread_function()
{
   printf("Thread number %ld\n", pthread_self());
   pthread_mutex_lock( &mutex1 );
   counter++;
   pthread_mutex_unlock( &mutex1 );
}
```

und erklären Sie seine Wirkungsweise in Form eines Struktogramms mit Pseudocode.