Probetestat

1 Aufgabe

1.1 Lösung mit Prozessen

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define size 8
#define max 10

int a[size][size], b[size][size], res[size][size];


int initMatrix()
{
  int i, j;
  for (i = 0; i < size; i++)
    for (j = 0; j < size; j++)
    {
      a[i][j] = rand() % max;
      b[i][j] = rand() % max;
    }
}

int printMatrix(int matrix[size][size])
{
  int i, j;
  for (i = 0; i < size; i++)
  {
    for (j = 0; j < size; j++)
      printf("%4d ", matrix[i][j]);
    printf("\n");
  }
  printf("\n");
}

int calcColumns(int j1, int j2)
{
  int h, i, k;
  if (j1 < size)
    for (i = 0; i < size; i++)
    {
      h = 0;
      for (k = 0; k < size; k++)
```



```
        h += a[i][k] * b[k][j1];
      res[i][j1] = h;
    }
  if (j2 < size)
    for (i = 0; i < size; i++)
    {
      h = 0;
      for (k = 0; k < size; k++)
        h += a[i][k]*b[k][j2];
      res[i][j2] = h;
    }
}

int main(int argc, char ** argv)
{
  double t;
  int i;
  clock_t start, stop;

  start = clock();
  srand(1024);
  initMatrix();

  switch (vfork())
  {
    case 0:
    {
      switch (vfork())
      {
        case 0: calcColumns(0, 2);
        exit(0);
      }
      calcColumns(1, 3);
      exit(0);
    }
    default:
    {
      switch(vfork())
      {
        case 0:
        {
          calcColumns(4, 6);
          exit(0);
        }
      }
    }
  }
```

```
  calcColumns(5, 7);
  printMatrix(a);
  printMatrix(b);
  printMatrix(res);
  stop = clock();
  t = (double) (stop - start) / (double) (CLOCKS_PER_SEC);
  printf("Zeit: %f Sekunden\n", t);
  exit(0);
}
```

1.2 Lösung mit Threads

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

#define size 8
#define max 10

int a[size][size], b[size][size], res[size][size];

int initMatrix()
{
  int i, j;
  for (i = 0; i < size; i++)
    for (j = 0; j < size; j++)
    {
      a[i][j] = rand() % max;
      b[i][j] = rand() % max;
    }
}

int printMatrix(int matrix[size][size])
{
  int i, j;
  for (i = 0; i < size; i++)
  {
    for (j = 0; j < size; j++)
      printf("%4d ", matrix[i][j]);
    printf("\n");
  }
  printf("\n");
}

void * calcColumns(void * arg)
{
```



```
  int h, i, k;
  int j = (int) arg;
  if (j < size)
    for (i = 0; i < size; i++)
    {
      h = 0;
      for (k = 0; k < size; k++)
        h += a[i][k] * b[k][j];
      res[i][j] = h;
    }
  if (j+1 < size)
    for (i = 0; i < size; i++)
    {
      h = 0;
      for (k = 0; k < size; k++)
        h += a[i][k]*b[k][j+1];
      res[i][j+1] = h;
    }
}

int main(int argc, char ** argv)
{
  double t;
  int i;
  clock_t start, stop;
  pthread_t tid[(size+1)/2];
  void * o;

  start = clock();
  srand(1024);
  initMatrix();

  for (i = 0; i < (size+1)/2; i++)
    pthread_create(&tid[i], NULL, calcColumns, (void*) (2*i));
  for (i = 0; i < (size+1)/2; i++)
    pthread_join(tid[i], &o);

  printMatrix(a);
  printMatrix(b);
  printMatrix(res);
  stop = clock();
  t = (double) (stop - start) / (double) CLOCKS_PER_SEC;
  printf("Zeit: %f Sekunden\n", t);
  exit(0);
}
```

2 Aufgabe

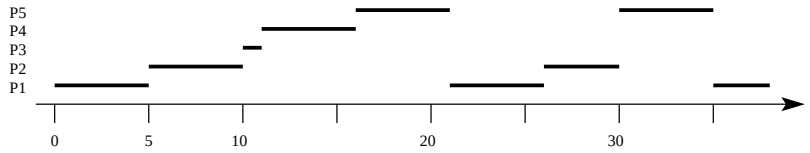

Abbildung 1: Round Robin mit Q=5

Prozess	P1	P2	P3	P4	P5
Antwortzeit	38	30	11	16	35

Es ergibt sich eine mittlere Antwortzeit von $\frac{38+30+11+16+35}{5} = \frac{130}{5} = 26$.

3 Aufgabe

R	1	2	3	4	5	6	7	2	3	3	4	7	2	3	8	9	1	6	5	7	2	10	4	3	8
	1	2	3	4	5	6	7	2	3	3	4	7	2	3	8	9	1	6	5	7	2	10	4	3	8
		1	2	3	4	5	6	7	2	2	3	4	7	2	3	8	9	1	6	5	7	2	10	4	3
			1	2	3	4	5	6	7	7	2	3	4	7	2	3	8	9	1	6	5	7	2	10	4
				1	2	3	4	5	6	6	7	2	3	4	7	2	3	8	9	1	6	5	7	2	10
					1	2	3	4	5	5	6	6	6	6	4	7	2	3	8	9	1	6	5	7	2
						1	2	3	4	4	5	5	5	5	6	4	7	2	3	8	9	1	6	5	7
							1	1	1	1	1	1	1	1	5	6	4	7	2	3	8	9	1	6	5
															1	5	6	4	7	2	3	8	9	1	6
																1	5	5	4	4	4	3	8	9	1
																					4	3	8	9	
4	F	F	F	F	F	F	F	F	F	-	F	-	-	-	F	F	F	F	F	F	F	F	F	F	F
7	F	F	F	F	F	F	F	-	-	-	-	-	-	-	F	F	F	F	F	F	F	F	F	F	F
D	∞	∞	∞	∞	∞	∞	∞	6	6	1	6	4	4	4	∞	∞	9	8	9	8	8	∞	10	10	10

Tabelle 1: LRU-Seitensetzungsalgorithmus

An Tabelle 1 kann man erkennen, dass mit 4 Seitenrahmen genau 21, mit 7 Seitenrahmen genau 18 Seitenfehler auftreten. Die Distanzzeichenkette ist ebenfalls Tabelle 1 zu entnehmen. Des weiteren ist die minimale Seitenfehlerrate 10:

4 Aufgabe

Der Zustand ist unabhängig von der Zahl x unsicher, da Prozess A noch bis zu 2 Ressourcen vom letzten Typ benötigt, jedoch nur maximal eine weitere dieser Ressourcen verfügbar ist.

5 Aufgabe

Ein Semaphor setzt sich aus einer Semaphor-Variable $s \in \mathbb{N}_0$ und zwei atomaren Operationen zusammen:

i	1	2	3	4	5	6	7	8	9	10	∞
C_i	1	0	0	3	0	3	0	3	2	2	10
F_i	24	24	24	21	21	18	18	15	13	10	10

```
down(s)
if s > 0
  then s--
  else gehe schlafen

up(s)
   s++
   wecke einen auf s wartenden Prozess auf
```

Bei der Umsetzung in Maschinensprache ist insbesondere darauf zu achten, dass die beiden Operationen atomar sind.

6 Aufgabe

```
gegeben: Sperrvariable fuer die Bruecke

Richtung Ost
  Auto kommt in Brueckennaehe

  Auto sperrt ab
  Solange Autos auf der Bruecke Richtung West fahren
  oder eine Warteschlange vor der Bruecke ist
    Sperre auf
    Warte
    Sperre ab
  Fahre ueber die Bruecke
  Sperre auf

Richtung West
  Auto kommt in Brueckennaehe
  Auto sperrt ab
  Solange Autos auf der Bruecke Richtung Ost fahren
  oder eine Warteschlange vor der Bruecke ist
    Sperre auf
    Warte
    Sperre ab
  Fahre ueber die Bruecke
  Sperre auf
```

7 Aufgabe

Es laufen zu viele Prozesse (die Prozesstabelle ist vermutlich voll!).