Practicum RtES – deel 1

1.Taskscheduling met pthreads.

Opgave 1a

Onderstaand is gegeven de code van 4 pthreads (4 taken). Schrijf een c-programma dat de 4 taken met de posix-api in round robin schedult.

Hint:

```
#include <pthread.h>
pthread_attr_t tattr;


pthread_attr_init(&tattr);  //tattr init met defaultwaarden

pthread_attr_setschedpolicy(&tattr, SCHED_RR); sched policy aanpassen
pthread_create(&tid, &tattr, taskOne, arg);
pthread_create(&tid, &tattr, taskTwo, arg);
```

```
void* taskOne()
{

 int i,j,m,n;

 while(1)
   {
       for(i=0;i<5;i++)
           {
              for(j=1;j<=8;j++)
                {
                  printf("taak 1 %i\n",j)
                   for (m=0;m<=1000;m++)
                       for(n=0;n<=10000;n++);
                       /* De for-lussen dienen om een vertraging te
                          realiseren, zodater een aantal keer een context
                          switch naar taak 2 optreedt */

                }

           }
    }
```

```
}
```

```
void* taskTwo()
{

int i,j,m,n;
while(1)
  {

      for(i=0;i<5;i++)
                  {
           for(j=1;j<=8;j++)
               {
                printf("taak 2 %i\n",j)

                for (m=0;m<=1000;m++)
                   for(n=0;n<=10000;n++);
                   /* De for-lussen dienen om een vertraging te
                      realiseren, zodat er een aantal keer een context
                      switch naar taak 4 optreedt */

                }

           }
   }

}
```

```
void* taskThree()
{

int i,j,m,n;

while(1)
   {
      for(i=0;i<5;i++)
          {
              for(j=1;j<=8;j++)
                {
                printf("taak 3 %i\n",j);
                for (m=0;m<=1000;m++)
                    for(n=0;n<=10000;n++);
                    /* De for-lussen dienen om een vertraging te
                       realiseren, zodat er een aantal keer een context
                       switch naar taak 4 optreedt */

                }
         }
   }
}
```

```
void* taskFour()
{

int i,j,m,n;

while(1)
    {
      for(i=0;i<5;i++)
              {

               for(j=1;j<=8;j++)
               {
                  printf("taak 4 %i\n",j);
                    for (m=0;m<=1000;m++)
                     for(n=0;n<=10000;n++);
                     /* De for-lussen dienen om een vertraging te
                        realiseren, zodat er een aantal keer een context
                        switch naar taak 3 optreedt */
                 }

              }
    }
}
```

Opgave 1b

Onderstaand is gegeven de code van 4 pthreads (4 taken). Thread 1 en thread 2 bevatten een oneindige while-loop en thread 3 en 4 zijn aflopend (geen oneindige while-loop). Schrijf nu een C-programma dat eerst thread 3 en thread 4 in round robin uitvoert en na het beeindigen van thread 3 en thread 4 de uitvoer vervolgt met round robin tussen thread 1 en thread 2.

Hint:

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

pthread_attr_t tattr;
pthread_t tid;
int ret;
int newprio = 20;
sched_param param;

/* initialized with default attributes */
ret = pthread_attr_init (&tattr);

/* safe to get existing scheduling param */
ret = pthread_attr_getschedparam (&tattr, &param);

/* set the priority; others are unchanged */
param.sched_priority = newprio;

/* setting the new scheduling param */
ret = pthread_attr_setschedparam (&tattr, &param);

/* with new priority specified */
ret = pthread_create (&tid, &tattr, func, arg);
```

```
void* taskOne()
{

 int i,j,m,n;

 while(1)
   {
       for(i=0;i<5;i++)
           {
              for(j=1;j<=8;j++)
                {
                  printf("taak 1 %i\n",j)
                   for (m=0;m<=1000;m++)
                       for(n=0;n<=10000;n++);
```

```
                    /* De for-lussen dienen om een vertraging te
                       realiseren, zodater een aantal keer een context
                       switch naar taak 2 optreedt */

                }

            }
        }

}
```

```
void* taskTwo()
{

int i,j,m,n;
while(1)
  {

      for(i=0;i<5;i++)
                 {
           for(j=1;j<=8;j++)
               {
                printf("taak 2 %i\n",j)

                for (m=0;m<=1000;m++)
                   for(n=0;n<=10000;n++);
                   /* De for-lussen dienen om een vertraging te
                      realiseren, zodat er een aantal keer een context
                      switch naar taak 4 optreedt */

                }

           }
   }

}
```

```
void* taskThree()
{

int i,j,m,n;

    for(i=0;i<5;i++)
          {
              for(j=1;j<=8;j++)
                {
                 printf("taak 3 %i\n",j);
                 for (m=0;m<=1000;m++)
                     for(n=0;n<=10000;n++);
                     /* De for-lussen dienen om een vertraging te
                        realiseren, zodat er een aantal keer een context
                        switch naar taak 4 optreedt */

                 }
          }

}
```

```
void* taskFour()
{

int i,j,m,n;

     for(i=0;i<5;i++)
            {

             for(j=1;j<=8;j++)
             {
                printf("taak 4 %i\n",j);
                  for (m=0;m<=1000;m++)
                   for(n=0;n<=10000;n++);
                   /* De for-lussen dienen om een vertraging te
                      realiseren, zodat er een aantal keer een context
                      switch naar taak 3 optreedt */
                }

            }

}
```

Opgave 1c

Schrijf een C-programma dat twee periodieke pthreads aanmaakt. Thread1 heeft een periodetijd van 0,5 sek en thread 2 heeft een periodetijd van 1,3 sek. Iedere thread drukt per periode de threadnaam en periodenummer af.