TENTAMEN

KURSNAMN	
PROGRAM: namn **åk / läsperiod**	**REALTIDSSYSTEM , GK** **DAI2 samt EI3**
KURSBETECKNING	**LET 624 0109 (6p)**
EXAMINATOR	**Peter Lundin**
TID FÖR TENTAMEN	**Tisdagen den 23/10 2012 kl. 08.30 – 12.30**
HJÄLPMEDEL	**Godkänd räknedosa**
ANSV LÄRARE: namn **telnr** **besöker tentamen kl**	**Peter Lundin** **772 5726** **Peter Lundin Ca 9.30 samt 11.00**
DATUM FÖR ANSLAG av resultat samt av tid och plats för granskning	**Granskning: Måndagen den 5/11 12.15 – 12:45** **Sal J 121. Därefter finns tentorna vid D&IT:s studieexpedition plan 4 hus Jupiter.**
ÖVRIG INFORM.	**Betygsgränser : Max 25 poäng** **3:**11– 15**,5** **4:** 16 – 20,5 **, 5:** 21 -

FORMELSAMLING

RMSA enkel analys
***FÖRUTSÄTTNINGAR*:**

- Varje process i uppsättningen är periodisk
- Deadline (d) och periodtid (p) är lika stora.
- Konstant exekveringstid för processer
- Ingen processkommunikation förekommer
- Alla processer är avbrytbara

n	$n(2^{1/n}-1)$
1	1,000
2	0,828
3	0,780
4	0,757
5	0,743

En uppsättning processer (P_1,P_2,...P_n) är schemaläggningsbar om:

$$\sum_{i=1}^{n}\frac{c_i}{p_i} \le n\,(2^{1/n}-1)$$

där c_i är är exekveringstiden för P_i och p_i är periodtiden för P_i.

RMSA exakt analys (krav som ovan men här räcker det med att d <= p)
Beräkning av svarstiden för process ***i*** under interferens från högre prioriterade processer ***j*** ges av sambandet

$$R_i^{n+1} = c_i + b_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{p_j} \right\rceil c_j$$

Där b är blockeringsfaktorn, interferens från lägre prioriterade processer.

Svarstidsanalys vid processuppsättningar med fix prioritet och godtycklig deadline.
För en uppsättning processer gäller att maximal svarstid *R*, inom ett fönster *w*, kan beräknas. För blockeringsfaktorn *b* gäller att prioritetstaksprotokoll (max en blockering) är implementerat.

$$w_i^{n+1}(q) = (q+1)c_i + b_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^n(q) + J_j}{p_j} \right\rceil c_j$$

Iterationen utförs tills : $w_{i,q} <= (q+1)\,p_i$

Svarstiden R ges av:

$$R_i = \left(\max_{q=0,1,2,\ldots} \left(w_i(q) - qp_i \right) \right) + J_i$$

och där:

- *w* är det ”fönster” inom vilket vi betraktar svarstider
- *q* är den start, numrerad 0,1,2..., inom fönstret vi betraktar
- *c* är processens exekveringstid
- *b* är blockeringsfaktor för processen
- *p* är processens periodtid
- *J* är maximalt jitter för processen

Uppgift 1. (2p)
I CAN-protokollet används CSMA/CR för bussaccess.

a) Beskriv vad metoden innebär generellt för de olika delarna av förkortningen dvs CS, MA respektive CR
b I samband med bussaccess talar man om begreppet bussarbitrering. Beskriv vad begreppet står för.

Uppgift 2 (4p)

I ett tidsdelat system bestående av flera processer som exekveras ”pseudoparallellt” via tidsdelningen har man oftast ett realtidsoperativsystem som sköter tidsdelning genom processbyten som normalt genereras av klockavbrott. För att ge stöd för ömsesidig uteslutning av kritiska regioner samt för synkronisering av processerna har vanligen ett realtidssystem stöd för s.k. semaforer.

a)
En process i ett sådant system kan befinna sig i ett av tre olika tillstånd, vilka är dessa.
Beskriv även vilken händelse som initierar **samtliga möjliga** byten mellan de olika tillstånden.

b)
Beskriv vad en Semafor är samt vilka operationer man kan utföra på Semaforen.

c)
Visa med korta programexempel hur man kan använda sig av Semaforer för att garantera ömsesidig uteslutning samt synkronisering av händelser/processer.

d) Beskriv vad man menar med **kritisk region** samt **ömsesidig** uteslutning av kritiska regioner.

Uppgift 3 (3 p)

Man har implementerat en synkron meddelandekanal genom vilken man kan skicka data av typen integer mellan processer. För att använda kanalen har man skrivit två funktioner enligt nedan. Tanken är att man från en godtycklig process skall kunna anropa önskad funktion av dessa två för att lämna ett tecken (ChannelWrite) alternativt att hämta ett tal (ChannelRead). Programmet har stöd av realtidskärnan RTK12 , dvs samma som vi har använt i denna kurs.
För att synkronisera och garantera ömsesidig uteslutning har man använt sig av fyra semaforer enligt nedan.

```
//----- Global dataarea --------
int data;

main(){

        ……
        initsem ( WRITE, 1);
        initsem(READ,1);
        initsem(WRITE_OK,0);
        initsem(READ_OK,0);
        ……

void channelWrite( int tempdata){
        waitsem(WRITE);
        data=tempdata;
        signalsem(WRITE_OK);
        waitsem(READ_OK);
        signalsem(WRITE);
}
```

```
void channelRead( int *t_data){
        waitsem(READ);
        waitsem(WRITE_OK);
        *t_data=data;
        signalsem(READ_OK);
        signalsem(READ);
}
```

Antag att vi hart tre processer som exekveras pseudoparallellt och som anropar funktionerna enligt nedan tidsschema där $t_1 < t_2$ osv. Mellan tiderna finns det relativt stora tidsluckor.

Före nedan händelsekedja har ingen process anropat någon av funktionerna.

t_1 : Process 1 anropar funktion enligt : channelRead(&tal);

t_2 : Process 2 anropar funktion enligt : channelRead(&nr);

t_3 : Process 3 anropar funktionen enligt : channelWrite(3);

…….

Beskriv den händelsekedja som uppstår med avseende på semaforhantering och vilken process som exekverar med hänsyn till anropen enligt ovan.. Beskriv även hantering av variablerna.

Uppgift 4 (4p)

Antag att vi har en uppsättning av processer med data enligt tabellen nedan.
Processerna skall schemaläggas med ett statiskt schema. Processerna får avbrytas.

a) Bestäm LCM för periodtiderna utan att göra någon periodtidsjustering.
b) Föreslå en lämplig periodjustering för att minska den sk LCM-tiden för periodtiderna.
c) Rita ett statiskt schema för processerna efter periodjusteringen.
d) Antag att samma processer schemalägges dynamiskt . Beräkna i så fall svarstiderna enligt metoden för RMSA exakt analys om man låter processerna ha prioritet enligt tabellens ordning (P1 högsta prioritet) och periodtider enligt de som gäller efter justeringen.
Jämför resultaten med det ritade statiska schemat enligt c. Kan man förklara de uträknade svarstiderna med stöd av detta.
e) Är systemet schemaläggningsbart enligt en eller båda schemaläggningsmetoderna. Motivera ditt svar.

Process	**p**	**c**	**d**
P1	6	2	4
P2	10	2	6
P3	16	5	12

Uppgift 5 (3p)

I ett system med tre processer har man uppskattat processdata enligt nedan tabell.

Avgör om systemet är schemaläggningsbart enligt RMSA.

Proc	Pri	p (ms)	d(ms)	c(ms)	Semafor S
P1	0	6	6	2	2 ms
P2	1	13	8	3	
P3	2	25	15	5	3 ms

Uppgift 6 (2p)

I ett styrsystem läser en process in data i form av 8 bitars binära tal 1 – 255. Talen sparas i en globlal minnesbuffert indata[ ]. Den binära koden 0 används som sluttecken för den senaste inlästa följden av tal. Maximalt kan de ingå MAX antal tal i bufferten utöver slutttecknet 0. Om bufferten är tom är första tecknet lika med slutttecknet.

En funktion ***int soekAntalTal (unsigned char soektal)*** utvecklas för att söka efter antalet förekomster av ett visst tal i bufferten enligt följande :

```
#define MAX 10
unsigned char indata[MAX+1];

int soekAntalTal ( unsigned char soektal ){
            int antal, n;
            unsigned char temp;
      1     antal=0;
      2     n=0;
      3     temp=indata[n];
      4     while(temp!=0x00){
      5           if(temp==soektal)
      6                 antal++;
      7           n++;
      8           temp=indata[n];
      9     }
      10    return(antal);
}
```

a) Rita en programflödesgraf för raderna 1 – 10

b) Beräkna exekveringstiden [min,max] för programdelen 1 – 10 om följande uppskattningar av exekveringstiderna gäller:

BB1= [5 ,5]
BB2=[5,5]
BB3= [10,10]
BB4= [20,25]
BB5= [10 ,10]
BB6= [5,5]
BB7= [5,5]
BB8= [10,10]
BB10= [15 ,15]

Uppgift 7 (3p)

Antag att vi har ett system med tre processer med struktur enligt nedan. Process två och tre delar en semafor för att garantera omsesidig uteslutning för exekveringen av funktionerna comput2() respektive compute3() .
Process 1 har prioritet 6, Process 2 har prioritet 10 och Process 3 har prioritet 2. Högsta prioritet ges för lägsta prioritetsnummer.
Exekveringstider är :
compute1() : 5 sek, compute2() : 4 sek samt compute3() 2 sek.

Funktionerna : parkForEvent_x() innebär att processen avvakter en viss händelse nr x och ges ej körtid under denna väntan. När händelse inträffar ändrar processen tillstånd till Ready och ges körtid utifrån den fixa prioriteringen den har vilket kan innebära att den får vänta tills högre prioriterade processer körts klart..

Antag att händelserna nr 1 – 3 inträffar i följande ordning , händelse nr 2 , 3 och sist 1 , med en sekund mellan händelserna. Vi får anta att inga andra processer påverkar körningen av de ter processerna.

```
Process 1() {
   while(1) {
      parkForEvent_1();
      compute1();
  }
}
//-----------------------------------------
process2(){
  while(1) {
     parkForEvent_2();

     waitsem(S1);
     comput2();
     signalsem(S1)
  }
}
//-----------------------------------------
Process3(){
  while(1) {
    parkForEvent_3();
    waitsem(S1);
    compute3();
    signalsem(S1)
  }
}
//-----------------------------------------------
```

a) Rita ett tidsdiagram over exekveringen av processerna från det att första händelsen inträffar till att samtliga tre processer genomfört en iteration av sin exekvering, dvs att funktionen computex() genomförts.
Rita tidsdiagrammet så att processernas tillstånd (**waiting samt running**) framgår.

b) På grund av semaforhanteringen kommer man att få en oönskad effekt vid exekveringen av processerna. Beskriv denna effekt och ange vad den kallas.

c) Det finns en metod att minimera denna effekts inverkan. Beskriv kort om denna metod och vad de man inför kallas.

Uppgift 8 (2p)

Två datornoder NOD 1 och NOD 2 byggda kring microcontrollern Motorola HCS12. Controllern har två inbyggda CAN moduler CAN0 respektive CAN4. Nodernas båda CAN-moduler är anslutna till en gemensam datorbuss.

Till systemen finns en antal drivrutiner enligt nedan. I drivrutiner och det exempel på ett huvudprogram som ges används variabler av typen struct samt pekare varav några definieras nedan.
Den direkta hanteringen av kretsen register är dold i drivrutinerna.

```
struct CAN
{
        unsigned short CANx;                    //CAN interface
        unsigned char CAN_IDAM;                 //Identifier Acceptence Mode
        unsigned long CAN_AR_MR_ID[5];          //Acceptance Code 1st Bank
                                                //Acceptance Code 2nd Bank
                                                //Acceptance Mask 1st Bank
                                                //Acceptance Mask 2nd Bank
                                                //Message Identifier
        unsigned char CAN_IDE;                  //ID Extended ( 12/29 ARB)
        unsigned char CAN_RTR;          //Remote Transmission Request
        unsigned char CAN_DLR;                  //Data Length
        unsigned char CAN_PRIO;             //Transmit Buffer Priority
};
struct CAN_message
{
      unsigned char length;       //number of bytes (0-8)
      char byte[8];               //eight data bytes
};
```

Pekare :
ptrCAN0_init : en pekare till en struct av typen CAN_init för CAN0.
prtCAN0 : en pekare till en struct av typen CAN för CAN0.
ptrCAN_trans_message: en pekare till en struct av typen message.
ptrCAN_rec_message : en pekare till en struct av typen message.
void init_CAN_pointers(void) : Initierar ett antal pekare som används av programmet
void default_CAN(struct CAN_init *,struct CAN *); : Intierar en CAN struct till defaultvärden.
void init_CAN(struct CAN_init *,struct CAN *,unsigned char recive);
:Initierar CAN modulen för sand/mottag.
void transmit_CAN(struct CAN *,struct CAN_message *,unsigned char); : Sänder ett meddelande.
int receive_CAN(struct CAN *,struct CAN_message *);
:Läser ett mottaget meddelande. Returnerar 1 om meddelande finns mottaget.
//---

Nedan visas ett exempel på ett program för systemets NOD 1 som sänder ett meddelande avsett för NOD 2 samt läser in ett eventuellt meddelande från NOD 2 med arbitreringskoden 0x0AAAAAAC eller =0x0AAAAAAA.

```
//  ------- Huvudprogram NOD 1 ------------
void main(void) {
     int length=0, i, n;
     char sendtxt[]={"Text_1\0"};
     char intxt[8];
     unsigned long identifier;
      ……..


   ptrCAN0->CAN_AR_MR_ID[4]=0x0AAAAABA;
   //default initialize CAN4
   default_CAN(ptrCAN4_init,ptrCAN4);
  //Acceptence Filter 1 & 2
   ptrCAN4->CAN_AR_MR_ID[0]=0x0AAAAAAC;   // ID acc reg
   ptrCAN4->CAN_AR_MR_ID[1]=0x0AAAAAAA;  // ID acc reg
   ptrCAN4->CAN_AR_MR_ID[2]=0xF0000000;       // ID mask reg
   ptrCAN4->CAN_AR_MR_ID[3]=0xF0000000;       // ID mask reg
  //initialize CAN0 for transmission
   init_CAN(ptrCAN0_init,ptrCAN0,CANtransmit);
  //initialize CAN4 for reception//
  init_CAN(ptrCAN4_init,ptrCAN4,CANreceive);

  while(1){
        //message to send
         i=0;
         for(i=0; i<7;i++)
                ptrCAN_trans_message->byte[i]=sendtxt[i];

          ptrCAN_trans_message->length=7;
          transmit_CAN(ptrCAN0,ptrCAN_trans_message,TXE0);     / /send message

         if(receive_CAN(ptrCAN4,ptrCAN_rec_message)==1){       //if received message
                length=ptrCAN_rec_message->length;                      // antal databytes
                identifier = read_ID_CAN(ptrCAN4);                      // mottaget ID
                 if (identifier==0x0AAAAAAC) puts("ID ok \n\r");          // Kolla ID för mottagen data

                 for ( i=0;i<length;i++){
                    intxt[i]=(ptrCAN_rec_message->byte[i]);
                 }
                  intxt[i]='\0';                                                // fixa strängslut
                  puts(intxt);
         }else
                  puts("no message\n\r");
}
}
```

Uppgiften: Beskriv vilka ändringar man behöver göra i det redovisade programmet för att programmet skall ge NOD 2 önskad funktion dvs sända ett meddelande till NOD 1 samt läsa meddelandet från NOD 1. Ingen av noderna skall läsa sitt eget utsänt meddelande.

Uppgift 9 (2p)

Nedan finns en inledande del till en programlista av ett C-program . Programmet hanterar bland annat en länkad lista bestående av poster av typen REGTYP. I main-funktionen finns en pekare head av typen REGTYP som pekar på listans första element. Sista elementets next-pekare är NULL. Vi kan utgå från att det alltid finns minst ett element i listan.

Skriv en funktion som söker om det finns en post med ett visst angivet enamn. Funktionen skall returnera en pekare till den första posten med angivet enamn alternativt NULL om det ej finns en post med angivet enamn.
Funktione har deklarationen:

REGTYP* find_person(char *name, REGTYP *top);

Not : Biblioteksfunktionen int strcmp(const char *s1, const char s2) får användas. Funktionen returnerar talet 0 om strängarna är identiskt lika.

```
#include <stdio.h>
#include <stlib.h>
#include <string.h>

// ##### Typdeklaretioner ####
typedef struct q{
      char enamn[20];
      char fnamn[20];
      struct q *next;
} REGTYP;

// ##### Funktionsdeklarationer #####
REGTYP* find_person( char *name, REGTYP *top);
…….

//###### Huvudprogram #######
int main(int argc, char *argv[])
{
   int antal_poster;
   REGTYP *head;
   REGTYP *aktuellpost;
   char s_enamn[20];     // sökt pesons enamn.
   …….
   // Anrop av funktionen:
  aktuellpost=  find_person( s_enamn, head);


   .........
   .........
     system("PAUSE");
  return 0;
}
```

Lycka till önskar Peter Lundin