Chalmers Campus Lindholmen
Peter Lundin/ Data- och informationsteknik

2009-10-21
Tentamen realtidssystem
LET 624 0101

TENTAMEN

KURSNAMN	
PROGRAM: namn **åk / läsperiod**	**Realtidssystem , gk** **DAI2/EI3**
KURSBETECKNING	**LET624 0109 / LET 623 0100**
EXAMINATOR	**Peter Lundin**
TID FÖR TENTAMEN	**Onsdagen den 21/10 2009 kl. 14.00 – 18.00**
HJÄLPMEDEL	**Godkänd räknedosa**
ANSV LÄRARE: namn **telnr** **besöker tentamen kl**	**Peter Lundin** **772 5726** **Ca 15.00 samt 16.30**
DATUM FÖR ANSLAG av resultat samt av tid och plats för granskning	**Torsdagen den 12/11 2009 .** **Granskning : Onsdag 18/11 kl 12.15 – i sal Omega**
ÖVRIG INFORMATION	**Betygsgränser : Max 24 poäng (23p LET623)** **3:**10 – 15 **4:** 16 – 19,5 **, 5:** 20 -

Chalmers Campus Lindholmen
Peter Lundin/ Data- och informationsteknik

2009-10-21
Tentamen realtidssystem
LET 624 0101

FORMELSAMLING

RMSA enkel analys
***FÖRUTSÄTTNINGAR*:**

- Varje process i uppsättningen är periodisk
- Deadline och periodtid är samma
- 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 exact analys
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
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

Chalmers Campus Lindholmen
Peter Lundin/ Data- och informationsteknik

2009-10-21
Tentamen realtidssystem
LET 624 0101

Uppgift 1 (2p)

I realtidssammanhang brukar man organisera systemet i flera sk processer som utför väl avgränsade uppgifter.

a) Beskriv med egna ord och eventuella figurer vad man menar med en process i detta sammanhang.
b) En viktig del i hanteringen av processer är de sk. Processorkontrollblocket (PCB). Beskriv vad ett PCB vanligen kan består av.

Uppgift 2 (4x1p)

I ett realtidssystem med flera pseudoparallella processer använder man sig av ett sk realtidsoperativsystem för att hantera exekveringen av processerna. (RTOS). RTOS kan ha stöd för hantering av semaforer.

a) I systemet kan det förekomma variabler eller andra resurser som processerna delar. I samband med delade variablerna/resurserna inför man begreppen : 'Kritisk region' samt 'Odelbar region'. Beskriv vad man menar med dessa två begrepp.

b) En semafor beskrivs som en synkroniseringsprimitiv på låg nivå. Beskriv kort vad en semafor är samt vilka operationer man kan utföra på en semafor.

c) En semafor kan användas för några olika tillämpningar. Nämn två olika tillämpningar och beskriv hur man använder semaforen och tillhörande operationer på semaforen i dessa fall. Beskriv gärna med några korta programexempel.

d) Ange tre olika processtillstånd för en process i systemet samt beskriv speciellt vad som föranleder övergångar mellan de olika tillstånden. Använd gärna en figur.

Uppgift 3 (1,5 + 0,5 p)

Följande processer är givna. Skapa

Process	**p(ms)**	**c(ms)**	**d(ms)**
P1	10	5	8
P2	20	4	10
P3	32	3	15

a) Rita ett statiskt schema för processerna om de är av typen "preemtive" med prioritet enligt tabellen (P1 högst prioritet) .
b) Antag att vi strikt tolkar svarstiden för varje upprepning av processen som tiden mellan start och till det att processen exekverat klart. Är systemet då schemaläggningsbart.

Uppgift 4 (3p)

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

a) Avgör om systemet är schemaläggningsbart enligt RMSA om man bortser från semaforanvändningen.

b) Är systemet schemaläggningsbart om man tar hänsyn till inverkan från Semaforhanteringen och om operativsystemet använder prioritetsarv i samband med semaforhanteringen. Motivera svaret.

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 5 (3p)

I ett system använder man sig av ett antal olika processer med väl avgränsade uppgifter enligt följande:

Process A (PA) :
Läser ett värde från en givare , omvandlar värdet till en intern representation. Läsningen skall ske med en periodicitet om 100 Hz. Exekveringstiden för processen är 3 ms.

Process B (PB) :
Utför en beräkning baserad på inläst värde från PA. Periodiciteten för processen kan sättas till 100 Hz. Exekveringstiden är 3 ms.

Process C (PC) :
Omvandlar det beräknade värdet från process PB till ett styrvärde som matas ut till ett ställdon. Ställdonet kräver en uppdatering av styrvärde var 5:e ms (200 Hz) varför processens periodtid blir 5 ms. Exekveringstiden för PC är 1 ms.

Processerna PA och PC innehåller båda A/D och D/A- omvandlare varvid erhålles ett Jitter avseende exekveringstiden på motsvarande +/- 0,5 ms för vardera process.

Antag att man bestämmer att processerna skall prioriteras i följande ordning : PA, PB samt PC , där PC har lägst prioritet.
Beräkna den maximala svarstiden för PC enligt metoden för godtycklig deadline och fix prioritet. Det finns ingen semaforhantering att ta hänsyn till.

Uppgift 6 (2+1 p)

a) I CAN används CSMA/CR för bussarbitrering på bit-nivå. Redogör för hur denna arbitrering går till.

b) Själva grundprincipen för accessmetoden i CAN ger upphov till en begränsning av det maximala fysiska avståndet mellan två av noderna i en CAN-buss. Förklara varför det blir så.

Uppgift 7 (3p)

Antag att vi har ett system i med flera processer som exekveras ’samtidigt’ med hjälp av ett realtidsoperativsystem med stöd för semaforer av typen blockerande–kö. Systemet har **en** analog ingång där inspänningen kan läsas in via en 8 bitars AD-omvandlare. AD-omvandlaren har följande tre register:
Kontrollregister : Där man kan starta en omvandling genom att ettställa bit nr 2.
Statusregister: Med bland annat funktionen att bit nr 1 ettställs då omvandling är klar.
Dataregister: Där man kan läsa det 8 bitars digitala resultatet.

Man har i programmet deklarerat följande makron fö ratt underlätta programskrivandet.

```
#define ADPort_data      *( ( unsigned char* ) AD_adress_1 )
#define ADPort_status    *( ( unsigned char* ) AD_adress_2 )
#define ADPort_control   *( ( unsigned char* ) AD_adress_3 )
```

En inläsning via AD-omvandlaren görs enligt följande :

Starta en omvandling
Vänta tills omvandlingen är klar (Tar i detta fall maximalt ca 20 ms , långsam omvandlare)
Läs av det 8-bitars digitala värdet som finns i **dataregistret.**

Alla befintliga processer skall kunna läsa ett invärde via den gemensamma AD-omvandlaren. För detta ändamål skall man utveckla en gemensam funktion som varje process kan anropa. Anropet av funktionen ska ha följande funktionsprototyp:

Las_AD_port(int proc_nr) ;

Parametern **proc_nr** anger vilket process som anropar funktionen och funktionen kommer att skicka det inlästa värdet (8 bitars tal) till processen via en global minnesplats för respektive process som deklareras enligt följande:

```
unsigned char ad_Box[10];    // Avläst värde från funktionen Las_AD-port( proc_nr )
                             // placeras i ad_Box[proc_nr]
```

Funktionen skall med hjälp av en semafor utformas så att vi erhåller en säker hantering av den gemensamma AD-omvandlaren. När en process har anropat funktionen och startat en omvandling skall ej någon annan process kunna starta en omvandling utan anropande process skall hamna i en väntekö i väntan på att resursen AD-omvandlaren blir tillgänglig.

Den aktuella funktionen ska således kunna anropas av en varje enskild process enligt nedan programexempel :

```
Process Proc_1( void) {
   unsigned char analog_in;

   while ( 1 ) {
        ….
        ….
        Las_AD_port( 1 ) ;
        analog_in=ad_Box[1];
        …
        …
   }
}
```

a)

Skriv den aktuella funktion i vilken man använder sig av den deklarerade och initierade semaforen S1 för att säkerställa att enbart en process åt gången kommer använder AD-omvandlaren. Semaforoperationerna som kan användas är waitsem(S) respektive signalsem(S). ……(2p)

b)

För att snabba upp systemet skall funktionen återlämna återstående körtid för aktuell process om omvandlingen ej är klar genom att anropa en befintlig funktion **yield()** som omgående avslutar pågående process på rätt sätt och startar nästa process i Readykön.
Modifiera lösningen enligt a) så att man använder yield() i detta syfte. …….(1p)

Uppgift 8 (4 p) OBS !!!
(Denna uppgift görs enbart om du läser kursen för första gången och är registrerad på LET624. Om du läst kursen hösten 2008 eller tidigare gör du nästa uppgift)

Nedan finner du programlistan till ett C-program**.** Studera programmet och försök förstå vad programmet skapar

a) Vad skapar programmet om konstanten MAX sätts till 1 respektive 5.

b) Skriv funktionen **add_last(REGTYP* temp, int data)** som skall skapa ett nytt element , länkar in elementet sist i listan samt skriver in talet data till fältet tal i det nya elementet. Funktionen anropas från huvudprogrammet enligt: ***add_last(head, nydata)***

```
//#### Konstanter #####
#define MAX 5 // ( alternativt 1 )

// ##### Typdeklaretioner ####
typedef struct q{
       int tal;
       struct q *next;
       struct q *prev;
} REGTYP;

// ##### Funktionsprototyper #####

REGTYP* slumpa_lista();
void add_last(REGTYP* temp, int data);

//###### Huvudprogram #######
int main(int argc, char *argv[])
{
   int nydata, nr=0;
   REGTYP *akt_post , *head=NULL;

   srand(254);
   head=slumpa_lista();
   akt_post=head;
   while( akt_post!=NULL){
         printf("\n Post nr %d : %d" , nr++, akt_post->tal);
         akt_post=akt_post->next;
   }
  ……..
  ……

 //------------------
 system("PAUSE");
 return 0;
}
```

```
// ----------- Funktionen  slumpa lista ----------------------------------------

REGTYP* slumpa_lista(){
   int nr,i=0;
   REGTYP *top, *old, *item;
   item=(REGTYP*) malloc(sizeof(REGTYP));
   top=item;
   item->tal=0;
   item->next=NULL;
   item->prev=NULL;
   old=item;
   while(i<MAX){

            item=(REGTYP*) malloc(sizeof(REGTYP));
            nr=rand()%100;
            item->tal=nr;
            item->next=NULL;
            old->next=item;
            item->prev=old;
            old=item;
            i++;
   }
   return(top);
}

//-------- Funktionen add_last()-------------------------------------------

void add_last(REGTYP* temp, int data){
  // Funktion som lägger till ett element sist i den länkade listan samt
  // lägger in talet data i elementets fält tal.
……..
……..


}
```

Alternativ uppgift nr 8 för studenter som gör en omtentamen för kursen LET623 :
Uppgift 8_old (3p)

Semaforer kan beskrivas som starka respektive svaga beroende på hur de är implementerade.

a) Beskriv skillnaden mellan dessa två typer sett ur perspektivet hur processerna behandlas.

b) Antag att vi har ett system bestående av tre processer som utnyttjar en gemensam global variabel . För att säkerställa hanteringen av den globala variabeln använder man en semafor som är av den svaga typen. Beskriv en händelsekedja av semaforhantering och processbyten som leder till att en process hanteras orättvis till följd av olycklig turordning i processbyten. (dvs ej enbart p1-p2-p3-p1-..) .

Lycka till önskar Peter och Sven