Chalmers Campus Lindholmen
D&IT / Peter Lundin

2008-10-23
Tentamen realtidssystem
LET 623 0101

TENTAMEN

KURSNAMN	
PROGRAM: namn **åk / läsperiod**	**Realtidssystem , gk** **DAI2/EI3**
KURSBETECKNING	**LET 623 0101 för DAI2/EI3**
EXAMINATOR	**Peter Lundin**
TID FÖR TENTAMEN	**Torsdagen den 23/10 2008 kl. 13:30 – 18:00**
HJÄLPMEDEL	**Godkänd räknedosa**
ANSV LÄRARE: namn **telnr** **besöker tentamen kl**	**Peter Lundin** **772 5726** **Ca 14.30 samt 16.00**
DATUM FÖR ANSLAG av resultat samt av tid och plats för granskning	**Granskning vid Peter Lundins arbetsrum, Jupiter plan 4 Tisdagen den 11/11 kl 15-16.**
ÖVRIG INFORM.	**Betygsgränser : max är 25 poäng** **Betyg 3 : 11 – 15 , Betyg 4 :15,5 – 20 , Betyg 5:20,5 -**

Chalmers Campus Lindholmen
D&IT / Peter Lundin

2008-10-23
Tentamen realtidssystem
LET 623 0101

Uppgift 1 (3p)

I realtidssammanhang kan man dela in ett systems mjukvara i flera processer som utför väl avgränsade uppgifter.

a) Beskriv med egna ord och eventuella figurer vad man menar med en process i detta sammanhang.
b) Vilka fördelar finns att dela in mjukvaran i processer.
c) I samband med processer talar man om samtidig exekvering av processer. Beskriv vad man menar med det.
d) En viktig del i hanteringen av processer är de sk. Processorkontrollblocket (PCB). Beskriv vad PCB vanligen kan består av. Ange de absolut nödvändiga innehållet i PCB för att man ska kunna göra processbyten mellan processerna vid godtyckliga tidpunkter samt att man ska kunna skapa köer av exempelvis körklara processer (ReadyQ)

Uppgift 2: (3 p)
Det finns ett flertal algoritmer som är till för att lösa problemet med att garantera ömsesidig uteslutning utan att ha stöd av en underliggande realtidskärna eller speciella implementeringar i språket. Boken beskriver Dekkers algoritm. För att leda in till själva algoritmen beskriver läroboken fyra olika enklare förslag till lösningar för två processer som inte fungerar utan de ger upphov till svält, deadlock eller att den icke garanterar ömsesidig uteslutning. Slutligen beskrivs en fungerande lösning för två processer. Den slutliga lösningen har trots att den fungerar en del begränsningar som gör den olämplig att använda.

Nedan visas en annan lösning på samma problem (Petersen´s algoritm) för två processer P0 och P1 som skall exekveras samtidigt i ett system. Vi antar att programdelarna ingår i en process som ständigt återupprepas. Initieringen av flaggorna görs endast en gång.

```
flag[0]   = 0
flag[1]   = 0
turn      = 0
```

```
P0: flag[0] = 1
    turn = 1
    while( flag[1] && turn == 1 );
            // do nothing
    // critical section
    ...
    // end of critical section
    flag[0] = 0
```

```
P1: flag[1] = 1
    turn = 0
    while( flag[0] && turn == 0 );
            // do nothing
    // critical section
    ...
    // end of critical section
    flag[1] = 0
```

2 a)

Den kritiska regionen föregås av tre rader med hantering av variabler (flaggor) vars avsikt är att garantera ömsesidig uteslutning av de två kritiska regionerna.

Visa att lösningen fungerar för följande händelsescenario :
P0: startar exekveringen och vi får efter ett antal loopa av P0 ett processbyte till P1 efter raden turn=1; i P0.
P1 exekverar och efter ett tag får vi ett nytt processbyte.
Beskriv händelserna (flagvärden och vad som exekveras) därefter för några ytterligare processbyten varav ett byte när man befinner sig i en kritisk region.

2b)

Lösningen enligt denna algoritm och Dekkers algoritm har två problem/nackdelar som gör att man inte använder sig av algoritmerna. Ange dessa två problem.

2c)

De icke fungerande enkla lösningarna i boken ger upphov till svält, deadlock eller att lösningen ej garanterar ömsesidig uteslutning av de kritiska regionerna.
Beskriv vad man menar med deadlock, kritisk region samt ömsesidig uteslutning av kritiska regioner.

Uppgift 3 (3 p)

a) För att garantera ömsesidig uteslutning finns olika metoder för process-synkronisering bland annat 'Semaforer'. Beskriv vad en Semafor är och vilka operationer man kan utföra på Semaforen.
b) Exemplifiera hur man kan använda en Semafor för att garantera ömsesidig uteslutning samt hur man kan synkronisera processer med hjälp av flera semaforer.
c) Följande programkod är hämtad från implementeringen av den realtidskärna som har använts under kursens laborationer.
Beskriv i detalj hur en processen 'Process_1 ' kommer att hanteras då den anropar den aktuella funktionen signalsem(SEM1). Vid det aktuella tillfället för anropet finns en annan process ' Process_2' i väntekön som hör till den aktuella semaforen SEM1.

```
static int __signal(int semaphoreID)
{
    PCB   *awaked;


    if (WaitQ[(short)(semaphoreID-1)].proc != (PCB *) 0 ) {
        awaked = remque(&(WaitQ[(short)(semaphoreID-1)].proc));
        awaked->stat = PROC_RUNNABLE;
        insert_first(awaked, &ReadyQ);
        insert_last(Running, &ReadyQ);
        return 0;
    }
    (WaitQ[(short)(semaphoreID-1)].cnt) ++;
    return 1;
}


void signalsem(int semaphoreID)
{
    DISABLE;                        // Stänger av avbrottsmöjlighet
    if( __signal(semaphoreID) ){
        ENABLE;                     // Aktiverar avbrottmöjlighet
        return;
    }

    suspend(Running);
    dispatch();
}
```

Uppgift 4 (3p)

I ett realtidssystem med stöd av en realtidskärna kan man exekvera flera processer 'samtidigt'. I ett sådant system är principerna för schemaläggning av central betydelse. I samband med detta finns ett antal olika begrepp enligt nedan. Försök så gott det går att kort förklara innebörden av begreppen.

a) Pre-emptive
b) Round robin
c) Pseudoparallell respective seriell exekvering
d) Statisk schemaläggning respektive dynamisk scemaläggning

Uppgift 5 (4 p)

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) Rita ett statiskt schema för processerna. ANm: P3 kommer ej att kunna starta exakt enligt periodiciteten i vid den tredje starten iunder LCM-tiden. Låt P3 starta då det finns tid efter den önskade.
c) Bestäm ur ditt ritade schema den maximala svarstiden för samtliga tre processer om man antar att alla processer är startklara vid tiden t=0.
d) Beräkna enligt metoden för RMSA exakt analys svarstiden för processerna om man låter dem ha prioritet enligt tabellens ordning där P1 har högst prioritet.
Jämför svaren med värdet från ditt ritade schema.
e) Är systemet schemaläggningsbart om deadline för processerna är enligt tabellen. Basera ditt svar utifrån de svar du fått i deluppgift c och d.

Process	p	c	d
P1	6	3	4
P2	12	2	5
P3	16	4	8

Uppgift 6 (2p)

Ett system består av två givare som avläses via AD-ingångar med hjälp av två processer P1 och P2. De inlästa värdena behandlas sedan av ytterligare en process P3. Samtliga processers data har upskattats till de som anges i nedan tabell. Processerna skall vid schemaläggningen prioriteras enligt den i tabellen angivna ordningen.
Processerna har ingen uppskattat deadline. Beräkna den maximala svarstiden för P3 enligt metoden för godtycklig deadline och fix prioritet.

Processdata:

Process	Prioritet	p	c
P1	0	120	40
P2	1	100	30
P3	2	80	20

Uppgift 7 (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 50
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 & n< MAX){
        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:

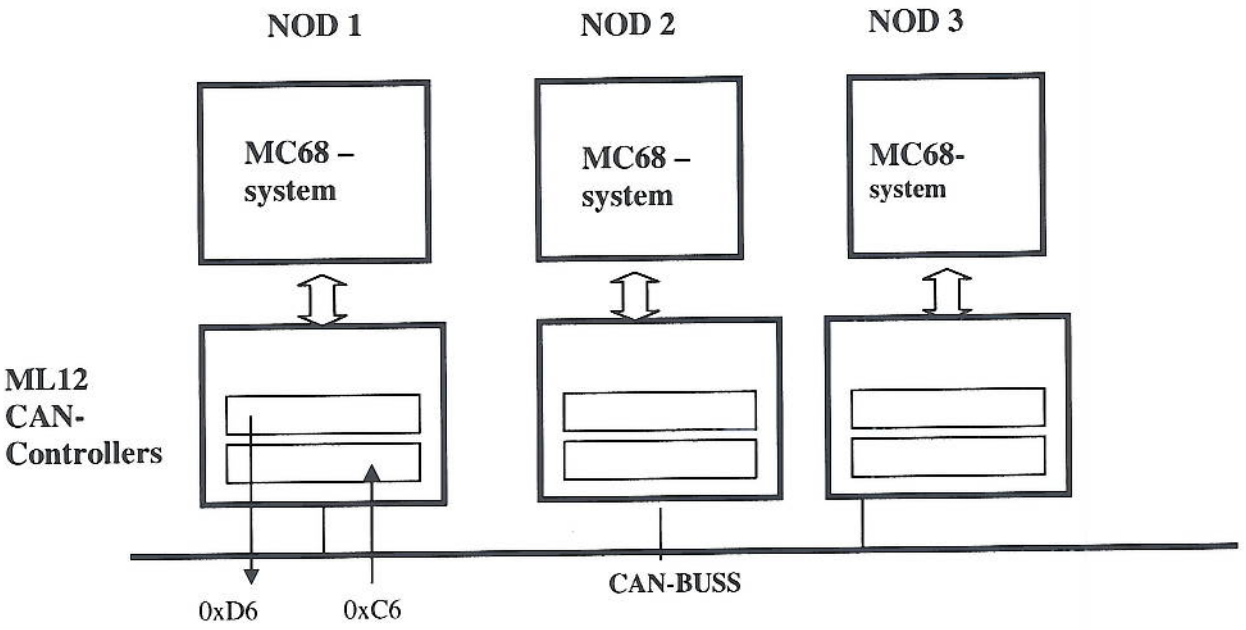
BB1= [5 ,5]
BB2=[5,5]
BB3= [10,10]
BB4= [25,30]
BB5= [10 ,10]
BB6= [5,5]
BB7= [5,5]
BB8= [10,10]

BB10= [15 ,15]

Chalmers Campus Lindholmen
D&IT / Peter Lundin

Uppgift 8 (2 p)

Antag att vi har ett system bestående av tre datornoder anslutna till varandra via en CAN-buss enligt figuren nedan.



Till programutvecklingsmiljön finns följande funktioner avsedda för CAN-noderna:

CANInit() :
Initierar noden till viss kommunikationsstandard vilket bland annat innefattar att alla arbitreringsbitar måste vara lika för att ett meddelandet skall läsas in.

CANSetupRec(int *arbitrering*, int *msgobject*) :
Aktiverar nodens meddelandeobjekt nr ***msgobjekt*** för mottagning av meddelande med arbitreringsinnehåll enligt ***arbitrering***.

int CANRec(int *msgobject*, char **buffer*) :
Läser eventuellt inkommit meddelande object . Parametern ***msgobjekt*** anger aktuellt objekt nr i noden medan pekaren ****buffer*** överför en pekare till mottagande sträng.
Rutinen returnerar '1' om nytt objekt finna att läsa samt '0' i annat fall.

CANSend(int *arbitrering*, int *msgobject*, char **data*):
Läser in data (8 tecken) till angivet meddelandeobjekt i noden som sedan sändes med angiven ***arbitrering*** då bussen ledig.

NOD nr 1 i systemet skall fungera på följande sätt:

Ska initieras så att ett av meddelandeobjekten är uppsatta för att ta emot meddelanden med arbitreringskoden 0xC6. Man kan välja godtycklig meddelandeobjekt nr av nodens möjliga 1 – 14.

Noden ska sedan återupprepat med jämna mellanrum om 100 ms (tidsloop via funktionen Delay(int antal_ms)) kontrollerar om det inkommit eventuellt meddelande med arbitreringskoden 0xC6 samt sända

ut en datasträng UtData i ett meddelande med arbitreringskoden 0xD6. (Se programskal) . Eventuell indata läses in till strängen InData . Därefter skall tecken nr 3 (dvs position 4) i InData skrivas in i fältet Buffert[n] där n är en räknare från 0 till 19 som räknas upp för varje nytt tecken. När bufferten är full börjar räknaren om från 0 igen.

Programskal

```
#include      <stdio.h>
#include      " can_drivers_me_05.h "  // Funktionsprototyper till befintliga funktionerna.
char  InData[9];
char  UtData[9]
char  Buffert[50]

void   main(){
       int n=0;
       strcpy(InData,"0000000");
       strcpy(UtData,"1223334");
       .....
-----------------------
```

Uppgift : Skriv ett C-program för NOD1 som ger önskad funktion.

Uppgift 9 (3 p)

Tre standard-CAN-noder (A,B,C) anslutna till samma buss börjar sända vid exakt samma tidpunkt. Nod A's meddelande har identifierare 257 (hexadecimalt), nod B's meddelande har identifierare 360 (hexadecimalt) och nod C's meddelande har identifierare 25F (hexadecimalt).

a) Beskriv i detalj hur arbitrering och adressering tillgår i CAN-protokollet

b) Redogör, i detalj för vad som händer och visa vilket meddelande som vinner arbitreringen.

c) Kan flera CAN-noder sända meddelanden med samma identifierare i ett fungerande CAN-bussystem ? Motivering krävs.

.

Lycka till
Peter Lundin