TENTAMEN
Operativsystem DVA315, 140611 kl. 08:10-11:30
Ansvarig lärare: Dag Nyström
Max poäng: 30
Betygsgränser: 3: 15p, 4: 21p, 5: 25p
Hjälpmedel: -
Påbörja varje uppgift på ett nytt papper! ***Lycka till!***

Begreppsdel

Uppgift 1 (4p) Processer

a) En av de mest fundamentala komponenterna i ett processbaserat operativsystem är processkontrollblock (*process control block, PCB)*. Beskriv dess syfte (särskilt hur det används vid multiprogrammerade system) samt ange minst 4 saker som brukar lagras i ett PCB. (3p)
b) Förklara skillnaderna mellan en *process* och en *tråd*. (1p)

Uppgift 2 (4p) Allmänt

Förklara kortfattat följande operativsystemsrelaterade begrepp:

a) Thrashing (0.5p)
b) Busy waiting (0.5p)
c) Supervisorläge (Supervisor mode) (0.5p)
d) Intern fragmentering (0.5p)
e) Strutsalgoritmen (Ostricht Algorithm) (0.5p)
f) Kvanta (Quantum) (0.5p)
g) Räknande semafor (till skillnad från binär semafor) (0.5p)
h) Asynkront meddelandesystem (0.5p)

Uppgift 3 (3p) Processer

Förklara hur en *klassisk 3-tillstånds processmodell* fungerar.
Ange och förklara noga tillståndmaskinen(rita upp), övergångar, köer med mer. (3p)

Uppgift 4 (4p) Baklås

a) Förklara och ge exempel på begreppet *osäkert tillstånd* i baklåssammanhang. (2p)
b) Förklara hur *bankers algorithm* fungerar och visa med ett eget konkret exempel hur den appliceras på processer som begär resurser. (2p)

Problemdel

Uppgift 5 (3p) Virtuellt minne

Ett virtuellt minne adresseras med 16-bitars adresser. Hur många *sidor* kan det virtuella minnet maximalt adressera samt hur många bitar används för *offset* respektive *sidindexering* vid följande sidstorlekar:

a) 2kb (1p)
b) 4kb (1p)
c) 16kb (1p)

Visa hur du kommer fram till dina svar!

Eventuella antaganden MÅSTE motiveras!

Uppgift 6 (5p) Filhantering

I filhantering finns många olika tekniker för att hålla reda på vilka block på disken som tillhör vilken fil. Två av dessa tekniker är **i-noder** och **länkade listor med index (t.ex. FAT-tabeller)**. Givet en tom disk med 16 lediga block och tre olika filer (*A*, *B* och *C*), visa hur dessa två tekniker hanterar följande förfrågningar efter diskutrymme (och hur de håller reda på de allokerade blocken):

1. Fil *A* begär 3 block
2. Fil *B* begär 2 block
3. Fil *C* begär 3 block
4. Fil *B* begär 1 block
5. Fil *A* begär 2 block

Kom ihåg att förklara, för **var och en** av teknikerna, vilken fil de allokerade blocken tillhör efter **varje** förfrågan. Om teknikerna använder några speciella strukturer för att hålla reda på blocken, beskriv också deras tillstånd efter varje förfrågan.

Eventuella antaganden MÅSTE motiveras!

Uppgift 7 (7p) Kritiska sektioner

Nedanstående del av ett multitrådat program hanterar ett antal olika konton (lagrade i arrayen `konto`). Två funktioner, `deposit` och `transfer`, används för att sätta in pengar, respektive föra över pengar mellan två kontonummer. Stödfunktionen `findIndex` returnerar ett index i arrayen för ett konto med ett visst kontonummer. Dessa tre funktioner kan användas samtidigt i flera trådar i applikationen. Du vill vidare att överföringar skall vara atomära, dvs. du skall inte kunna se ”halva” överföringar.
Notera: I detta förenklade system skall ni INTE ta hänsyn till att någon skulle lägga till, ta bort eller ändra kontonummer på ett konto. Notera vidare att VARJE konto har sin egen semafor, dvs. vid överföring behöver du låsa både avsändar- och mottagarkontot.

```
1  typedef struct{
2     int kontonummer;
3     int saldo;
4     semaphore konto_sem; //semafor för att låsa ett konto vid transaktioner
5  }konto_typ;
6
7  //Global array med alla konton
8  konto_typ konto[MAX_KONTON];
9
10 //Funktion för att hitta index i arrayen för visst konto
11 int findIndex(int kontonummer){
12    int index;
13    for(index=0;index<MAX_KONTON;index++){
14      if (konto[index].kontonummer==kontonummer)
15        return index;
16    }
17    return -1; //Kontot finns inte
18  }
19
20 //Funktion för att sätta in pengar
21 int deposit(int kontonummer, int summa){
22    int temp;
23    int index=findIndex(kontonummer);
24    if(index!=-1){
25      temp=konto[index].saldo;
26      temp=temp+summa;
27      konto[index].saldo=temp;
28      return TRUE;
29    }
30    return FALSE;
31  }
32
33 //Funktion för att föra över pengar mellan 2 konton
34 int transfer(int franKonto, int tillKonto, int summa){
35   int temp;
36   int fran=findIndex(franKonto);
37   int till=findIndex(tillKonto);
38   if(konto[fran].saldo<summa){
39     return FALSE; //Ej tillräckligt med pengar på kontot
40   }
41   temp=konto[fran].saldo; //Ta ut pengar
42   temp=temp-summa;
43   konto[fran].saldo=temp;
44   temp= konto[till].saldo; //Sätt in pengar
45   temp=temp+summa;
46   konto[till].saldo=temp;
47   return TRUE;
48 }
```

a) Visa på ett scenario där parallellt exekverande trådar ger ett felaktigt saldo. (2p)
b) Din uppgift är att skapa **minimala kritiska sektioner** i funktionerna `deposit` och `transfer`. Ange mellan vilka rader i koden `Wait` och `Signal` anrop skall skjutas in. Om du anser att det krävs ytterligare logik (kod) för att lösa detta (förutom wait och signal anropen) så skall du visa även detta med hjälp av c-kod, pseudokod eller tydligt formulerad text. (5p)
Psst: Det finns en hund begraven någonstans i denna uppgift. Kanske kan det i väldigt olyckliga fall bli tokigt om två eller flera överföringar sker samtidigt….;)

Eventuella antaganden MÅSTE motiveras!