TIE-02500 Rinnakkaisuus

Tentti 9.5.2016

Tentin vastuuhenkilö: jyke.savia@tut.fi
Laskimen käyttö on kiellettyä.
Muista kirjoittaa kaikkiin vastauspapereihin nimesi ja opiskelijanumerosi.
Vastauksessa oleta vastaavasi sellaisen ihmisen esittämään kysymykseen, joka tuntee kohtalaisen hyvin ohjelmistotekniikan aihealuetta muutoin paitsi juuri tämän kysymyksen osalta. Mieti etukäteen vastauksesi pääkohdat ja lajittele ne johdonmukaiseen järjestykseen — älä kirjoita yhteen pötköön kaikkea mieleen tulevaa. Muista vastata kaikkiin tehtävän kysymyslauseisiin, sillä täysiä pisteitä ei voi saada jos kaikkiin kysyttyihin asioihin ei ole vastattu. Jos vastaus vaatii ohjelmakoodin kirjoitamista, sen ei tarvitse olla pilkulleen syntaksiltaan oikein. Mikä tahansa johdonmukaisesti käytetty ja yleisessä käytössä olevia ohjelmointirakenteita sisältävä koodin esitysmuoto käy.

1. Ovatko seuraavat väittämät **oikein** vai **väärin**?
   Jos väärin, niin kerro mitä väittämässä on pielessä (vaikka kohdan väite olisi väärin, niin siitä ei saa pisteitä jos selitystä ei ole olemassa). *9p.*

   (a) [1 piste] Ohjelman käynnistyessä käyttöjärjestelmä luo aina automaattisesti yhden suoritussäikeen.

   (b) [1 piste] Ohjelmoija saa päättää ajossa olevista säikeistä koska kukin niistä saa suoritusaikaa prosessorilla.

   (c) [1 piste] Neljän suorittimen (quad core) prosessorissa ajettava ohjelma voi luoda maksimissaan neljä säiettä.

   (d) [1 piste] Ohjelma 1 toteuttaa poissulkemisen kunhan säikeet muistavat aina ennen kriittistä aluetta kutsua rutiinia `enter()` ja poistuessaan rutiinia `exit()`.

```
1  bool varattuna = false;
2
3  void enter()
4  {
5      while( varattuna == true )
6          ;
7      varattuna = true;
8  }
9
10 void exit()
11 {
12     varattuna = false;
13 }
```

Ohjelma 1: Poissulkeminen

(e) [1 piste] Ohjelmoija ei voi mitenkään tietää ohjelman 2 muuttujista ovatko ne säikeille yhteistä muistia vai pelkästään säikeen omassa käytössä.

```
1  namespace {
2      int A = 42;
3
4      void funktio( double param ) {
5          int i = 0;
6          // ...
7      }
8  }
```

Ohjelma 2: Muisti

(f) [1 piste] Säieturvallinen ohjelmakirjasto lupaa, että kirjaston rutiineja (funktioita) voi kutsua useammasta säikeestä ilman lukituksia (kirjasto itse pitää huolen siitä, että rutiinit toimivat luvatulla tavalla riippumatta yhtäaikaisten kutsujen määrästä).

(g) [1 piste] C++-ohjelman lauseke `i++` (kasvata muuttujan arvoa yhdellä) on säieturvallinen.

(h) [1 piste] Säikeitä käyttävässä ohjelmassa voi olla käytössä vain yksi lukko (mutex).

(i) [1 piste] Rinnakkaisessa ohjelmoinnissa monitori on rakenne, joka tutkii ovatko säikeet menneet jumiin (deadlock).

2. Kerro lyhyesti mitä seuraavat asiat ovat? *6p.*

   (a) [2 pistettä] Atominen konekäsky. Anna esimerkki tällaisesta käskystä (selosta myös mitä käsky tekee -- pelkkä nimi ei riitä)

   (b) [2 pistettä] rw-lukko (read-write lock). Miten toimii? Missä tilanteissa sitä tyypillisesti voi käyttää rinnakkaisessa ohjelmassa?

   (c) [1 piste] Käänteisprioriteettiongelma (priority inversion problem)?

   (d) [1 piste] Minkälainen rinnakkaisuuden hallintamekanismi on futuuri (future) ohjelmointikielissä?

3. Semafori rinnakkaisuuden hallintamekanismina. *6p.*

   (a) [1 piste] Mikä on semaforin käyttörajapinta? (miten ohjelmoija semaforia voi käyttää)

   (b) [1 piste] Mihin voi käyttää semaforia, jonka alkuarvo on nolla (0)?

   (c) [1 piste] Mihin voi käyttää semaforia, jonka alkuarvo on yksi (1)?

   (d) [1 piste] Mihin voi käyttää semaforia, jonka alkuarvo on suurempi kuin 1 (N)?

   (e) [2 pistettä] Miten semafori toteutetaan käyttöjärjestelmän sisällä?

4. Barrier-synkronointi. *6p.*

   (a) [2 pistettä] Mikä on Barrier-rakenteen käyttötarkoitus rinnakkaisessa ohjelmoinnissa?

   (b) [1 piste] Minkälainen on Barrierin käyttörajapinta? (Miten ohjelmoija käyttää rakennetta)

   (c) [3 pistettä] Hahmottele ohjelmakoodi, joka toteuttaa Barrier-rakenteen