ITKA203 – Käyttöjärjestelmät – tentti 10.7.2015

Kevään 2015 kurssin luennot, demot, esimerkkiohjelmat / Paavo Nieminen <paavo.j.nieminen@jyu.fi>

Yleisiä ohjeita: Muista merkitä vastauspaperiin oma **nimesi** ja **syntymäaikasi** sekä kurssin nimi. Lisäksi **vastauspaperisi tulee sisältää 48 peräkkäistä numeroitua kohtaa**, joissa on joko tehtävässä pyydetty vastaus tai viiva ”–” tyhjän vastauksen merkiksi. Mikäli tehtävässä ei mainita poikkeuksista, jokainen oikea vastaus tuo 0.5 pistettä, väärä vastaus -0.5 pistettä ja tyhjä vastaus 0.0 pistettä. Arvosanakaava on ilmoitettu mallitentin yhteydessä; kurssin läpäisy edellyttää tentistä vähintään 12 pistettä.

Esimerkkeihin perustuvissa tehtävissä oletetaan, että järjestelmässä ei ole yhtäaikaa muita käyttäjiä, prosesseja, vikoja tai muutakaan, jotka muuttaisivat toimintaa siitä, miltä se esimerkissä suoraviivaisesti näyttää.

Moniselitteisiä kysymyksiä ei ole laitettu mukaan tahallisesti. Mikäli jokin tehtävä on vahingossa sellainen, että vastaus ei olekaan yksikäsitteinen, laita vastauspaperiisi tehtävän kohdalle kommentti, jossa kerrot, miksi mielestäsi näin on. Virheellisiksi osoittautuvat kysymykset poistetaan tämän tenttikerran arvostelusta.

Numeroidut kysymyskohdat 1–48

1. **Väite:** Prosessin kaikilla säikeillä on aina sama sisältö kaikissa prosessorin rekistereissä. (A=kyllä; B=ei)

Ohje tehtävään 2: Järjestä seuraavat muistikomponentit niiden nopeuden mukaan: A=kovalevy, B=keskusmuisti, C=rekisteri, D=välimuisti.

2. Vastauksessasi on neljä järjestettyä kirjainta: nopein ensin, hitain viimeisenä

Tutkittava esimerkki tehtäviin 3–4: Kurssin luennoilta ja demoista tutussa ympäristössä (Linux,bash) tehty yksittäinen, POSIX-syntaksin mukainen komentorivi:

```
arg echo | arg | grep -v -i "arg echo" | arg > echo
```

3. Montako komentoa rivillä on yhteensä?
4. Montako argumenttia rivillä on yhteensä?

Ohje tehtäviin 5–8: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että kukin lause on totta. Alkuihin on yksikäsitteinen oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

Lauseiden alut:

5. Prosessielementti (PCB) …
6. Sivutaulu …
7. Ready-jono …
8. Blocked-jono …

Vaihtoehtoiset loput:

A. liittyy ensisijaisesti vuoronnukseen.
B. ilmoitetaan prosessorille muistiosoitteiden automaattista muuntamista varten.
C. sisältää kaikki tiedot yksittäisen prosessin tilasta.

Ohje tehtävään 9: Yhdistä lauseen loppua vastaavat kirjaimet (*vähintään* yksi, mutta *mahdollisesti* useita) lauseenalun perään siten, että muodostuvat lauseet vastaavat todellisuutta. Vastauksessa on oltava listattuna kaikki todellisuutta vastaavat vaihtoehdot.

Lauseen alku:

9. Kurssilla käsitelty käyttöjärjestelmän rajapintastandardi POSIX (vuoden 2008 versio) määrää, että …

Vaihtoehtoiset loput (mahdollisesti useita sopivia):

A. kellokeskeytyksen tulee tapahtua 50, 100 tai 1000 kertaa sekunnissa
B. järjestelmäkutsu write() tapahtuu sijoittamalla rekisteriin RAX luku 1 ennen syscall-käskyä
C. Tiedostovalikko (engl. *File*) on valikkopalkin vasemmanpuoleisin valikko
D. shell-komento *c99* käynnistää C99-standardin mukaisen C-kääntäjän

10. **Väite:** Tyypillisessä nykyprosessorissa (esim. AMD64) on käyttöjärjestelmätilassa (engl. *kernel mode*) käytettävissä useampia konekielikäskyjä kuin käyttäjätilassa (engl. *user mode*). (A=kyllä; B=ei)
11. **Väite:** Microkernel -tyyppinen käyttöjärjestelmä pyrkii suorittamaan palveluita mahdollisimman paljon käyttäjätilassa (engl. *user mode*). (A=kyllä; B=ei)

Ohje tehtäviin 12–15: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että kukin lause on totta. Alkuihin on yksikäsitteinen oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

Lauseiden alut:

12. Virtuaalimuistin hallinta (engl. *virtual memory management*) …
13. I/O -ohjelmisto (engl. *I/O subsystem*) …
14. IPC (engl. *inter-process communication*) …
15. Graafinen tiedostoselain (engl. *file browser*) …

Vaihtoehtoiset loput:

A. määrittelee nimet/osoitteet koneeseen liitetyille laitteille.
B. tarvitaan vain virtuaalikoneeseen asennetussa käyttöjärjestelmässä.
C. käsittelee sivukeskeytyksen (engl. *page fault*).
D. tarvitaan bittioperaatioiden, kuten AND ja OR, hoitamiseksi.
E. ei ole välttämätön osa nykyaikaista käyttöjärjestelmää.
F. toimittaa signaaleja (esim. apuohjelmalla *kill* lähetettyjä).

Ohje tehtäviin 16–17: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että kukin lause on totta. Alkuihin on yksikäsitteinen oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

Lauseiden alut:

16. Laiteriippuva I/O -ohjelmisto …
17. Laitteistoriippumaton I/O -ohjelmisto …

Vaihtoehtoiset loput:

A. tarvitaan vain silloin, kun tietokoneessa ei ole bittioperaatioita (esim. AND, OR) valmiiksi toteutettuna laitteistotasolla.
B. kääntää bittioperaatiot (esim. AND, OR) yhden laitteen konekielestä toisen laitteen konekielelle.
C. sisältää laiteohjainten ajurit.
D. määrittää järjestelmälle yhtenäisen lohkokoon.

Ohje tehtäviin 18–21: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että kukin lause on totta. Alkuihin on yksikäsitteinen oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

Lauseiden alut:

18. Keskinäinen poissulku (engl. *mutual exclusion, MutEx*) …
19. FLIH (*first-level interrupt handling*) …
20. Semafori (*semaphore*) …
21. käskyosoitin (*instruction pointer, IP*) …

Vaihtoehtoiset loput:

A. liittyy kilpa-ajotilanteiden (engl. *race condition*) ratkomiseen.
B. on osa prosessorilaitteiston toimintaa.
C. on tietoverkkoon liitetyn laitteen osoite.

Ohje tehtävään 22: Yhdistä lauseen loppua vastaavat kirjaimet (*vähintään* yksi, mutta *mahdollisesti* useita) lauseenalun perään siten, että muodostuvat lauseet vastaavat todellisuutta. Vastauksessa on oltava listattuna kaikki todellisuutta vastaavat vaihtoehdot.

Lauseen alku:

22. Prosessorin keskeytys …

Vaihtoehtoiset loput (mahdollisesti useita sopivia):

A. toimii vain moniydinprosessorissa.
B. voi aiheutua käyttäjätilassa toimivan prosessin toimenpiteiden johdosta.
C. voi aiheutua ulkoisen laitteen toiminnon johdosta.
D. voidaan estää sovellusohjelmassa käyttämällä synkronointia.

Tutkittava esimerkki tehtäviin 23–24: Kurssin luennoilta ja demoista tutussa ympäristössä (Linux,bash) tehtyjä tuttuja komentoja ja niiden tulosteita (merkistökoodaus on UTF-8):

```
[nieminen@halava esimerkki]$ whoami
nieminen
[nieminen@halava esimerkki]$ ls -la
total 600
drwx------. 2 nieminen nobody   1024 May 19 11:18 .
drwx------. 9 nieminen nobody   1024 May 19 11:16 ..
-rwxr--r--. 1 nieminen nobody 586405 May 19 11:20 kayttojarjestelmat.tex
[nieminen@halava esimerkki]$ echo Heippa >> kayttojarjestelmat.tex
[nieminen@halava esimerkki]$
```

23. **Väite:** komentojen jälkeen tiedoston `kayttojarjestelmat.tex` pituus on pienempi kuin 586405 tavua. (A=kyllä; B=ei)
24. **Väite:** komentojen jälkeen annettava komento `echo ./kayttojarjestelmat.tex` tulostaisi konsoliin "Heippa" (A=kyllä; B=ei)
25. **Väite:** P.J. Denningin ym. 1970-luvulla kehittelemä lokaalisuusperiaate (engl. *principle of locality*) on pohjana mm. nykyisen POSIXin kieliasetusten määrittelylle. (A=kyllä; B=ei)
26. **Väite:** Kurssilta tutun x86_64 linux-ABIn mukaan C-kielisen aliohjelmakutsun parametrit ja paikalliset muuttujat sijaitsevat prosessin koodialueella (engl. *code segment*). (A=kyllä; B=ei)
27. **Väite:** Kekoaluetta käytetään dynaamisesti luotavien tietorakenteiden/olioiden säilyttämiseen. (A=kyllä; B=ei)
28. **Väite:** Tällä kurssilla esitelty semafori on tietorakenne, joka sisältää kokonaisluvun ja jonon prosesseja tai säikeitä. (A=kyllä; B=ei)
29. **Väite:** Tällä kurssilla esitellyssä semaforikutsussa `wait()` semaforin arvo muuttuu aina. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 30–32: luentoesimerkeistä tuttua C-kielistä ohjelmanpätkää muistuttava koodi (POSIXin pthread-kirjastoa hyödyntävä; mahdollisesti ”rikottu” jollain selkeällä tavalla tai sitten ei)

```
#define N 1000
pthread_mutex_t mymutex = PTHREAD_MUTEX_INITIALIZER;
uint64_t summa = 0;
void * saikeen_koodi(void *v) {
    int i;
    for (i = 1; i <= N; i++){
        pthread_mutex_lock(&mymutex);
        summa++;
        pthread_mutex_unlock(&mymutex);
    }
    return NULL;
}
int main(int argc, char *argv[]) {
  pthread_t saieA, saieB;
  pthread_create(&saieA, NULL, saikeen_koodi, NULL);
  pthread_create(&saieB, NULL, saikeen_koodi, NULL);
  pthread_join(saieA, NULL);
  pthread_join(saieB, NULL);
  if (summa==2000){
    return 0; // homma toimi
  } else {
    return 1; // homma ei toiminut
  }
}
```

30. **Väite:** Esimerkin main() palauttaa aina 0:n, mikäli sen suoritus ylipäätään onnistuu loppuun saakka ilman esteitä ja kaikki sen sisältämät aliohjelmakutsut onnistuvat ilman virhekoodia (A=kyllä; B=ei)

31. **Väite:** Esimerkissä on periaatteessa mahdollista tilanne, jossa säikeessä A muuttuja i==123 ja säikeessä B muuttuja i==456 samaan aikaan. (A=kyllä; B=ei)
32. **Väite:** Esimerkin ohjelma voi aiheuttaa lukkiutumistilanteen (engl. *deadlock*), vaikka sen aliohjelmakutsut onnistuisivat ilman ongelmia (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 33–34: C-mäisenä pseudokoodina tehty ehdotus rengaspuskuria käyttävän Kuluttaja-tuottaja -ongelman ratkaisemiseksi semaforipalvelua käyttäen. Yhteistä muistia käytetään vain puskurioperaatioissa.

```
tuottaja(){
  while(true) { // tuotetaan loputtomiin
    tuota();    // tehdään yksi datapätkä
    wait(EMPTY);
    wait(MUTEX);
    siirra_puskuriin();
    signal(MUTEX);
    signal(FULL);
  }
}
kuluttaja(){
  while(true) { // kulutetaan loputtomiin
    wait(FULL);
    wait(MUTEX);
    lue_puskurista();
    signal(MUTEX);
    signal(EMPTY);
    kuluta();   // käytetään yksi datapätkä
  }
}
main(){
  EMPTY=tee_semafori( 0 );
  FULL=tee_semafori( 0 );
  MUTEX=tee_semafori( 1 );
  kaynnista_saie(tuottaja);
  kaynnista_saie(kuluttaja);
}
```

33. **Väite:** Esitetty koodi ratkaisee kuluttaja-tuottaja -ongelman oikeellisesti (A=kyllä; B=ei)
34. [Edellinen kohta on poikkeuksellisesti 1.0p arvoinen; tämä kohta jää tyhjäksi]

Tutkittava esimerkki tehtäviin 35–37: luennolla ja monisteessa esitetyn kaltainen minimalistinen shell-ohjelma (kommentit poistettu, rivit numeroitu, näytetty vain olennainen toimintopätkä, mahdollisesti ”rikottu” jollain selkeällä tavalla tai sitten ei):

```
1   while(true){
2     luekomento(komento, argumentit);
3     pid = fork();
4     if (pid > 0) {
5       status = wait();
6     } else if (pid == -1) {
7       ilmoita("fork() epäonnistui!");
8       exit(1);
9     } else {
10      exec(komento, argumentit);
11      ilmoita("Komentoa ei voitu suorittaa!");
12      exit(1);
13    }
14  }
```

35. **Väite:** Rivi 5 tapahtuu fork() -kutsun luomassa lapsiprosessissa. (A=kyllä; B=ei)
36. **Väite:** Rivin 8 suorittaminen lopettaa koko minimalistisen shellin suorituksen. (A=kyllä; B=ei)
37. **Väite:** Rivin 12 suorittaminen lopettaa koko minimalistisen shellin suorituksen. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 38–41: Kuvitellaan, että meillä on käytössä yksinkertainen tietokone, jonka muistiosoitteissa on 20 bittiä, joista ensimmäiset 8 ilmoittavat sivunumeron ja loput 12 ilmoittavat tavuindeksin sivun sisällä. Keskusmuistiin mahtuu vain 8 sivua prosessien käytössä olevaa muistia. Käyttöjärjestelmän täytyy heittovaihtaa loput kovalevylle. Tietorakenteiden tilanne tarkasteluhetkellä on seuraava.

Prosessin (PID=2) sivutaulu:

rivi#	virt. sivu	fyys. sivu	muist.	dirty	diskIndex
0	0x02	0x01	1	0	0x0010
1	0x03	0x05	0	0	0x0022
2	0x7f	0x06	1	1	0x00bc

Prosessin (PID=7) sivutaulu:

rivi#	virt. sivu	fyys. sivu	muist.	dirty	diskIndex
0	0x02	0x03	0	0	0x0004
1	0x03	0x05	1	0	0x0008
2	0x04	0x02	0	0	0x00f2
3	0x2e	0x07	1	1	0x006c
4	0x7f	0x04	1	1	0x0007

Järjestelmän kehystaulu:

fyys. sivu	omistajan PID	omistajan rivi#	aikayksiköt edellisen käyttökerran jälkeen
0x01	2	0	2
0x02	234	9	12
0x03	567	6	5
0x04	7	4	19
0x05	7	1	177
0x06	2	2	155
0x07	7	3	12
0x08	876	1	4

38. Mihin fyysiseen muistiosoitteeseen kohdistuisi prosessin 2 tekemä kirjoitus virtuaalimuistiosoitteeseen 0x7f7f7?
39. Mihin fyysiseen muistiosoitteeseen kohdistuisi prosessin 7 tekemä luku virtuaalimuistiosoitteesta 0x7f7f7?
40. Prosessi 2 suorittaa hyppykäskyn aliohjelmaan muistiosoitteessa 0x02345. Tapahtuuko prosessorissa sivunvaihtokeskeytys? (A=kyllä; B=ei)
41. Prosessissa 7 aiheutuu sivunvaihtokeskeytys. Korvausalgoritmi on LRU. Onko jonkin sivun sisältö tallennettava levylle? (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 42–44: Kurssin luennoilta tutulla symbolisella konekielellä (GNU assembler) kirjoitettu, rivi riviltä kommentoitu ohjelmanpätkä:

```
_start:
        movq    $3,%rcx   # luku 3 rekisteriin RCX
        movq    $5,%rdi   # luku 5 rekisteriin RDI
silimu:
        inc     %rdi      # lisää 1 rekisterin RDI arvoon
        dec     %rcx      # vähennä 1 rekisterin RCX arvosta
        jz      ulos      # hyppää, jos edellisen käskyn tulos oli 0
        jmp     silimu    # hyppää aina
ulos:
        dec     %rdi      # vähennä 1 rekisterin RDI arvosta
```

42. Kuinka monta konekielikäskyä ohjelmanpätkän jälki sisältää, ts. kuinka monta käskyä sen suorittaminen alusta loppuun vaatii?
43. Mikä on rekisterin RCX sisältö ohjelmanpätkän suorituksen lopuksi?
44. Mikä on rekisterin RDI sisältö ohjelmanpätkän suorituksen lopuksi?
45. **Väite:** Unixin i-solmuihin (engl. *inode*) perustuvassa tiedostojärjestelmässä tiedoston nimi sisältyy tiedoston i-solmuun. (A=kyllä; B=ei)
46. **Väite:** Unixin i-solmuihin (engl. *inode*) perustuvassa tiedostojärjestelmässä tieto tiedoston pituudesta sisältyy tiedoston i-solmuun. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 47–48: Kurssin esimerkeistä tuttua Linuxille käännettyä C-ohjelmaa ajetaan x86-64 -arkkitehtuurilla, ja debuggerilla on nähtävissä seuraavat hetkelliset tiedot

```
  Rekistereitä:
  RIP (käsky)  0x0000000000400504
  RSP (huippu) 0x00007fffffffdcc0
  RBP (kanta)  0x00007fffffffdcf0

  Muistin sisältöä (kaikki muuttujat 64-bittisiä eli 8-tavuisia):
                   0x7fffffffdd08: 0x0000000100000000
                   0x7fffffffdd00: 0x00007fffffffde38
                   0x7fffffffdcf8: 0x0000000000400647
          kanta --> 0x7fffffffdcf0: 0x00007fffffffdd50
                   0x7fffffffdce8: 0x0000000000000001
                   0x7fffffffdce0: 0x0000000000000000
                   0x7fffffffdcd8: 0x0000000000000000
                   0x7fffffffdcd0: 0x0000000000000000
                   0x7fffffffdcc8: 0x0000000000000000
        huippu --> 0x7fffffffdcc0: 0x0000000000000000
```

47. Mikä on kutsupinossa nykyistä aktivaatiota *edeltävän* aktivaation pinokehyksen kannan osoite? Ilmoita heksalukuna.
48. Mikä tulee olemaan RSP-rekisterin sisältö siinä vaiheessa, kun tämä meneillään oleva aliohjelma-aktivaatio on jossain vaiheessa loppumassa ja siihen sisältyvä käsky `ret` on juuri tulossa suoritukseen.

Vapaaehtoinen vapaa sana

Halutessasi voit antaa loppuun palautetta tai kommentoida muuten kurssia tai tenttiä. Vastauksen muoto on vapaa, eikä se vaikuta arvosteluun.