

ITKA203 – Käyttöjärjestelmät – tentti 18.8.2017

Kevään 2017 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. Oikea vastaus tuo 0.5 pistettä. *Kyllä/ei -väittämissä sekä muissa kysymyksissä, joissa on kaksi vaihtoehtoa (A tai B), väärä vastaus tuo miinus pisteitä -0.375 pistettä*, jotta odotusarvoksi täydellä arvaamisella tulee selkeästi hylätty pistemäärä.

Esimerkkeihin perustuvissa tehtävissä oletetaan, että järjestelmässä ei ole yhtäaikaan muita käyttäjiä, prosesseja, vikoja tai muutakaan, jotka muuttaisivat toimintaa siitä, miltä se esimerkissä suoraviivaisesti näyttää. Oletetaan myös, että kaikki käyttöjärjestelmä- ja alustakirjastokutsut toimivat ilman poikkeuksia tai virheitä.

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 ja muokataan kysymyspankissa yksiselitteisempään muotoon tulevaisuutta varten.

Numeroidut kysymyskohdat 1–48

1. **Väite:** Moderni prosessori (esim. AMD64) suorittaa keskeytyskäsitelijän ensimmäisen konekielikäskyn käyttöjärjestelmätilassa (engl. *kernel mode*). (A=kyllä; B=ei)
2. **Väite:** Laitteistoriippumaton I/O-ohjelmiston kerros määrittelee laitteille nimeämiskäytännön. (A=kyllä; B=ei)
3. **Väite:** I/O-operaation pyytäminen aiheuttaa aina pyytävän prosessin siirron suoraan ready-tilaan. (A=kyllä; B=ei)
4. **Väite:** Tietokonejärjestelmän käyttöaste (engl. *utilization*) eli hyödylliseen laskentaan käytettävän ajan määrä kasvaa, kun lisätään kellokeskeytyksien määrää aikayksikössä. (A=kyllä; B=ei)

Ohje tehtävään 5: Järjestä seuraavat muistikomponentit niiden nopeuden mukaan: A=rekisteri, B=kovalevy, C=keskusmuisti, D=välimuisti.

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

Ohje tehtävään 6: 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:

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

Vaihtoehdot loput (mahdollisesti useita sopivia):

- A. shell-komento *c99* käynnistää C99-standardin mukaisen C-kääntäjän
- B. kellokeskeytyksen tulee tapahtua 50, 100 tai 1000 kertaa sekunnissa
- C. Avustus (engl. *Help*) on valikkopalkin oikeanpuoleisin valikko
- D. komento *startx* käynnistää X-ikkunointijärjestelmän

Tutkittava esimerkki tehtäviin 7–8: Kurssin luennoilta ja demoista tutussa ympäristössä (Linux,bash) tehtyjä tuttuja komentoja ja niiden tulosteita (merkistökoodaus on UTF-8):

```
[nieminen@halava tenttikys]$ whoami
nieminen
[nieminen@halava tenttikys]$ ls -l
total 16
-rw-r--r--. 1 nieminen users 12600 May 17 22:34 kayttojarjestelmat.tex
-rw-r--r--. 1 nieminen users    4 May 17 22:35 moi
[nieminen@halava tenttikys]$ echo Heippa > kayttojarjestelmat.tex
[nieminen@halava tenttikys]$
```

7. **Väite:** komentojen jälkeen tiedoston `kayttojarjestelmat.tex` pituus on pienempi kuin 12600 tavua. (A=kyllä; B=ei)
8. **Väite:** komentojen jälkeen annettava komento `./kayttojarjestelmat.tex` tulostaisi konsoliin "Heippa" ja rivinvaihdon. (A=kyllä; B=ei)

Ohje tehtäviin 9–12: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että lause on totta. Alkuihin on *yksi, yksikäsitteinen*, oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

Lauseiden alut:

Vaihtoehtoiset loput:

- | | |
|--|---|
| 9. Keskinäinen poissulku (engl. <i>mutual exclusion, Mutex</i>) ... | A. liittyy kilpa-ajotilanteiden (engl. <i>race condition</i>) ratkomiseen. |
| 10. Pinokehys ... | B. liittyy tietokoneiden muistihierarkian perusideaan. |
| 11. Semafori ... | C. liittyy aliohjelmakutsujen toteutukseen konekielessä. |
| 12. Lokaalisuusperiaate ... | D. liittyy käyttäjien sijaintitiedon hyödyntämiseen. |
13. **Väite:** Nykyaikaisen käyttöjärjestelmän yksi päätehtävä on varmistaa, että jokainen käyttäjä pystyy tasa-arvoisesti hallitsemaan suoraan kaikkia tietokoneeseen liitettyjä laitteita (engl. *universal root access principle*). (A=kyllä; B=ei)
 14. **Väite:** Päätaavoite ohjelmistokerrosten ja niiden välisten rajapintastandardien laatimisessa on estää kilpailijoita toteuttamasta tuotteita, jotka toimivat samalla tavoin kuin valmistajan oma tuote. (A=kyllä; B=ei)

Ohje tehtävään 15: 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:

Vaihtoehtoiset loput (mahdollisesti useita sopivia):

- | | |
|-------------------------------|---|
| 15. Prosessorin keskeytys ... | A. vaatii prosessorilta vähemmän toimenpiteitä kuin aliohjelmakutsuun siirtyminen (esim. AMD64:n käsky <code>call</code>). |
| | B. toimii vain moniydinprosessorissa. |
| | C. voi aiheutua käyttäjätilassa toimivan prosessin toimenpiteiden johdosta. |
| | D. voidaan estää sovellusohjelmassa käyttämällä synkronointia. |
16. **Väite:** POSIX-säikeitä käyttävä ohjelma täytyy kääntää erikseen yksiytimiselle ja moniytimiselle prosessorille, koska sama säikeitä käyttävä koodi ei voi toimia samanlaisena sekä yksiprosessori- että moniprosessorijärjestelmässä. (A=kyllä; B=ei)

Ohje tehtäviin 17–20: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että lause on totta. Alkuihin on *yksi, yksikäsitteinen*, oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

Lauseiden alut:

17. Prosessielementti (PCB) ...
18. Sivutaulu ...
19. Ready-jono ...
20. Blocked-jono ...

Vaihtoehdot loput:

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

Ohje tehtäviin 21–24: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että lause on totta. Alkuihin on *yksi, yksikäsitteinen*, oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

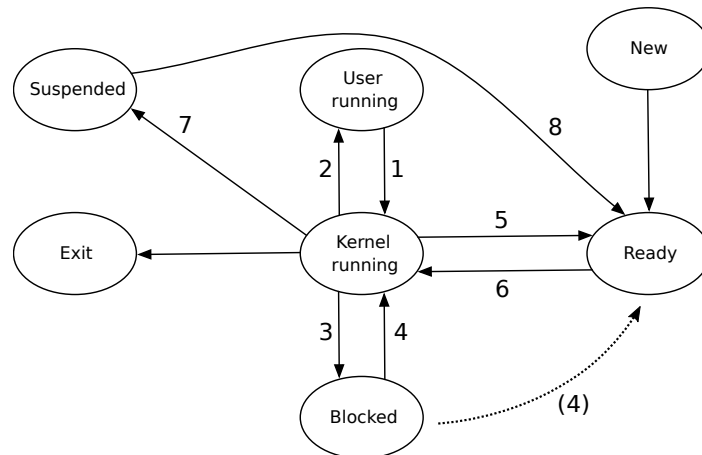
Lauseiden alut:

21. Käyttöjärjestelmän virtuaali-muistin hallintaosio (engl. *virtual memory management*) ...
22. Käyttäjakohtainen työpöytä (engl. *desktop manager*) ...
23. IPC (engl. *inter-process communication*) ...
24. Vuorontaja (engl. *scheduler*) ...

Vaihtoehdot loput:

- A. tekee toimenpiteitä sivuvirheen (engl. *page fault*) yhteydessä.
- B. tarjoaa palvelut mm. keskinäiseen poissulkuun.
- C. tarvitaan ainoastaan, kun suoritetaan virtuaalikoneita.
- D. tekee toimenpiteitä jokaisen kellokeskeytyksen yhteydessä.
- E. ei ole välttämätön osa nykyaikaista käyttöjärjestelmää.

Ohje tehtäviin 25–27: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että lause on totta. Alkuihin on *yksi, yksikäsitteinen*, oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään. Tehtävässä tutkitaan prosessien tilasiirtymäkaaviota, jonka selitetekstit on korvattu numeroilla:



Lauseiden alut:

25. Tilasiirtymä nro 1 tapahtuu, ...
26. Tilasiirtymä nro 5 tapahtuu, ...
27. Tilasiirtymä nro 6 tapahtuu, ...

Vaihtoehdot loput:

- A. kun käyttöjärjestelmän vuorontaja siirtää suoritukseen toisen prosessin, vaikka nykyinenkin prosessi pystyisi jatkamaan laskemista jo seuraavassa konekielikäskyssään.
- B. kun prosessi aloittaa käyttöjärjestelmän vuorontajalta saamansa aikaikkunan käytön.
- C. kun prosessori siirtyy keskeytyksen tai käyttöjärjestelmäkutsun käsittelyyn.
- D. kun prosessille on lähetetty odottelusignaali esim. näppäilemällä pääteyhteydessä Ctrl-Z.

Tutkittava esimerkki tehtäviin 28–29: Kurssin luennoilta ja demoista tutussa ympäristössä (Linux,bash) tehty yksittäinen, POSIX-syntaksin mukainen komentorivi:

```
arg arg cat grep | grep "a b c" kissa | sort -f -r | kissa > cat
```

28. Montako erillistä komentoa rivillä on yhteensä (rivin jäsentymisen kannalta, riippumatta siitä, toimisivatko kaikki komennot jossain järjestelmässä oikeasti)?
29. Montako erillistä argumenttia rivillä on yhteensä?
30. **Väite:** POSIXin semafori on binäärinen lippumuuttuja. (A=kyllä; B=ei)
31. **Väite:** POSIXin semaforikutsun `sem_wait(s)` suorituksessa semaforin `s` arvo muuttuu aina. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 32–34: Luento-esimerkeistä tuttua C-kielistä ohjelmanpätkeää muistuttava koodi (POSIXin `pthread`-kirjastoja hyödyntävä; mahdollisesti "rikottu" jollain selkeällä tavalla tai sitten ei). Oletetaan tehtävässä, että kaikki kutsut aina onnistuvat, eikä mitään ulkopuolisia vaikutuksia ole:

```
#define N 1000
pthread_mutex_t mymutex = PTHREAD_MUTEX_INITIALIZER;
uint64_t summa = 0;
void * saikeen_koodi(void *v) {
    pthread_mutex_lock(&mymutex);
    for (int i = 1; i <= N; i++){
        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
    }
}
```

32. **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)
33. **Väite:** Esimerkin suorituksessa on mahdollista syntyä tilanne, jossa säikeessä A muuttuja `i==123` ja säikeessä B muuttuja `i==456` samaan aikaan. (A=kyllä; B=ei)
34. **Väite:** Esimerkin ohjelma voi aiheuttaa lukkiutumistilanteen (engl. *deadlock*), jossa sen suoritus jää juumiin. (A=kyllä; B=ei)
35. **Väite:** P.J. Denningin ym. 1970-luvulla kehittänyt lokaalisuusperiaate (engl. *principle of locality*) on pohjana mm. nykyisen POSIXin kieliasetusten eli lokaalien (engl. *locale*) määrittelylle. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 36–38: luennolla ja monisteessa esitetyn kaltainen minimalistinen shell-ohjelma (kommentit poistettu, rivit numeroitu, näytetty vain olennainen toimintopätkä):

```
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 }
```

36. **Väite:** Rivi 5 tapahtuu fork() -kutsun luomassa lapsiprosessissa. (A=kyllä; B=ei)
37. **Väite:** Rivin 4 suorituksen aikana on aina olemassa sekä shell että sen luoma lapsiprosessi. (A=kyllä; B=ei)
38. **Väite:** Rivin 12 suorittaminen lopettaa koko minimalistisen shellin suorituksen. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 39–42: Kuvitellaan, että meillä on käytössä yksinkertainen tietokone, jonka virtuaalimuistiosoitteissa on 20 bittiä, joista ensimmäiset 8 ilmoittavat sivunumeron ja loput 12 ilmoittavat tavuindeksin sivun sisällä, eli käytössä on 4 kilotavun sivut. Fyysiset muistiosoitteet ovat 24-bittisiä. Keskumuistista on varattu prosessien käyttöön tasan 8 kehystä fyysistä muistia osoitteissa 0x100000-0x107fff. Heittovaihdon eli swapin avulla käytettävissä olevan muistin kokonaiskoko on 0x1000000 tavua (n. 16 megatavua). Tietorakenteiden tilanne tarkasteluhetkellä on seuraava. Prosessien sivutauluista näytetään vain kartoitetut osat kahden prosessin (PID:t 7 ja 14) osalta.

Prosessin (PID=7) sivutaulun kartoitetut rivit:

virt. sivu	fyys. sivu	muist. (P)	dirty (D)	diskIndex
0x02	0x000	0	0	0x0022
0x03	0x101	1	0	0x0008
0x04	0x000	0	0	0x0004
0x23	0x104	1	0	0x0abc
0x7f	0x107	1	0	0x0007

Prosessin (PID=14) sivutaulun kartoitetut rivit:

virt. sivu	fyys. sivu	muist. (P)	dirty (D)	diskIndex
0x02	0x100	1	0	0x0010
0x03	0x000	0	0	0x0023
0x23	0x106	1	1	0x00bb
0x7f	0x102	1	1	0x00bc

Järjestelmän kehystaulu:

fyys. sivu	omistajan PID	omistajan PTE#	aikayksiköt edellisen käyttökerran jälkeen
0x100	14	0x02	19
0x101	7	0x03	80
0x102	14	0x7f	75
0x103	234	0x45	8
0x104	7	0x2e	12
0x105	876	0x45	4
0x106	14	0x2e	1
0x107	7	0x7f	9

39. Prosessori suorittaa prosessia 14, ja käskyosoiterekisterin sisällöksi tulee 0x23232. Keskeytystä ei ole pyydetty. Mistä fyysisestä osoitteesta prosessori lähtee noutamaan seuraavaa konekielikäskyä?
40. Mistä virtuaaliosoitteesta prosessi 14 saisi käyttöönsä tavun, joka sijaitsee kovalevyllä indeksissä 0x23456 heittovaihto-osion / -tiedoston (engl. *swap space*) alusta lukien?
41. Prosessi 7 suorittaa hyppykäskyn aliohjelmaan muistiosoitteessa 0x04567. Tapahtuuko prosessorissa sivuvirhe / sivunvaihtokeskeytys? (A=kyllä; B=ei)
42. Prosessissa 234 aiheutuu sivunvaihtokeskeytys. Korvausalgoritmi on LRU. Onko jonkin sivun sisältö tallennettava levyllä? (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 43–45: Kurssin luennoilta tutulla symbolisella konekielellä (GNU assembler; AT&T -syntaksi; AMD64; Linux-järjestelmä) kirjoitettu, rivi riviltä kommentoitu kokonainen ohjelma. Muista: suoritus alkaa osoitteesta "_start".

```
ali:
    subq    $2,%rcx    # vähennä 2 rekisterin RCX arvosta
    movq    %rcx,%rdi  # kopioi rekisteristä RCX rekisteriin RDI
    ret

_start:
    # Aloituspaikan symbolinen osoite "_start"
    movq    $21,%rcx  # sijoita luku 21 rekisteriin RCX
    call    ali        # kutsu aliohjelmaa osoitteessa ali
    call    ali        # kutsu aliohjelmaa osoitteessa ali
    inc     %rdi       # lisää 1 rekisterin RDI arvoon
    inc     %rdi       # lisää 1 rekisterin RDI arvoon
    movq    $60,%rax  # Valmistele rajapinnan mukainen exit()-kutsu
    syscall    # Toteuta exit(KOODI), missä KOODI on RDIn arvo
```

43. Kuinka monta konekielikäskyä ohjelmanpätkän jälki sisältää, ts. kuinka monta käskyä sen suorittamisen alusta loppuun vaatii?
44. Mikä on rekisterin RCX sisältö ohjelmanpätkän suorituksen loppuksi?
45. Mikä on rekisterin RDI sisältö ohjelmanpätkän suorituksen loppuksi?

Tutkittava esimerkki tehtäviin 46–47: Kurssin esimerkeistä tuttua Linuxille käännettyä C-ohjelmaa ajetaan x86-64 -arkkitehtuurilla, ja debuggerilla on nähtävissä seuraavat hetkelliset tiedot

```
Rekistereitä:
RIP (käsky) 0x0000000000400504
RSP (huippu) 0x00007fffffffddcc0
RBP (kanta) 0x00007fffffffdcf0
```

Muistin sisältöä (kaikki muuttujat 64-bittisiä eli 8-tavuisia):

```
0x7fffffffdd08: 0x0000000100000000
0x7fffffffdd00: 0x00007fffffffde38
0x7fffffffdcf8: 0x0000000000400647
kanta --> 0x7fffffffdcf0: 0x00007fffffffdd50
0x7fffffffdcce8: 0x0000000000000001
0x7fffffffdcce0: 0x0000000000000000
0x7fffffffdcdd8: 0x0000000000000000
0x7fffffffdcdd0: 0x0000000000000000
0x7fffffffddcc8: 0x0000000000000000
huippu --> 0x7fffffffddcc0: 0x0000000000000000
```

46. Kuinka monta tavua muistia on varattu nykyisen aktivaation väliaikaisten muuttujien (eli. lokaalit muuttujat ja paikalliset kopiot parametreista) säilyttämistä varten? Ilmoita *kymmenjärjestelmän* lukuna.
47. Jos tämä aliohjelma seuraavassa konekielikäskyssään kutsuisi jotakin toista aliohjelmaa, niin missä muistiosoitteessa tulisi sijaitsemaan paluuosoite?

Tutkittava esimerkki tehtävään 48: Yksi vakiokirjasto (otsikkotiedosto "stdio.h") käyttävä C99-koodirivi:

```
printf("%d", 0xff);
```

48. Mitkä merkit koodirivin suoritus tulostaa? (speksistä: "*argument is converted to signed decimal notation*")

Vapaaehtoinen vapaa sana

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