ITKA203 – Käyttöjärjestelmät – tentti 16.5.2018

Kevään 2018 kurssin luennot, demot, esimerkkiohjelmat (yhteensopiva kevään 2017 kurssin kanssa) / 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 miinuspisteitä -0.375 pistettä*, jotta odotusarvoksi täydellä arvaamisella tulee selkeästi hylätty pistemäärä.

Esimerkkeihin perustuvissa tehtävissä oletetaan, että järjestelmässä ei ole yhtäaikaa 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 huomioidaan tämän tenttikerran arvostelussa ja muokataan kysymyspankissa yksiselitteisempään muotoon tulevaisuutta varten.

Numeroidut kysymyskohdat 1–48

Ohje tehtävään 1: Järjestä seuraavat muistikomponentit niiden nopeuden mukaan: A=kovalevy, B=rekisteri, C=välimuisti, D=keskusmuisti.

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

Tutkittava esimerkki tehtäviin 2–3: Kurssin luennoilta ja demoista tutussa ympäristössä (Linux,bash) tehty yksittäinen, POSIX-syntaksin mukainen komentorivi:

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

2. Montako erillistä komentoa rivillä on yhteensä (rivin jäsentymisen kannalta, riippumatta siitä, toimisivatko kaikki komennot jossain järjestelmässä oikeasti)?
3. Montako erillistä argumenttia rivillä on yhteensä?

Ohje tehtäviin 4–7: 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:

4. Keskinäinen poissulku (engl. *mutual exclusion, MutEx*) …
5. FLIH (*first-level interrupt handling*) …
6. Semafori (*semaphore*) …
7. IP-rekisteri (*instruction pointer*) …

Vaihtoehtoiset loput:

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

8. **Väite:** Prosessin virtuaalimuistin koodialue (eli ”koodisegmentti”) on sen kaikille säikeille yhteinen. (A=kyllä; B=ei)

9. **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)

10. **Väite:** Päätavoite ohjelmistokerrosten ja niiden välisten rajapintastandardien laatimisessa on estää kilpailijoita toteuttamasta tuotteita, jotka toimivat samalla tavoin kuin valmistajan oma tuote. (A=kyllä; B=ei)

Ohje tehtäviin 11–12: 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:

11. Laiteriippuva I/O -ohjelmisto …
12. Laitteistoriippumaton I/O -ohjelmisto …

Vaihtoehtoiset loput:

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

Ohje tehtäviin 13–16: 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.

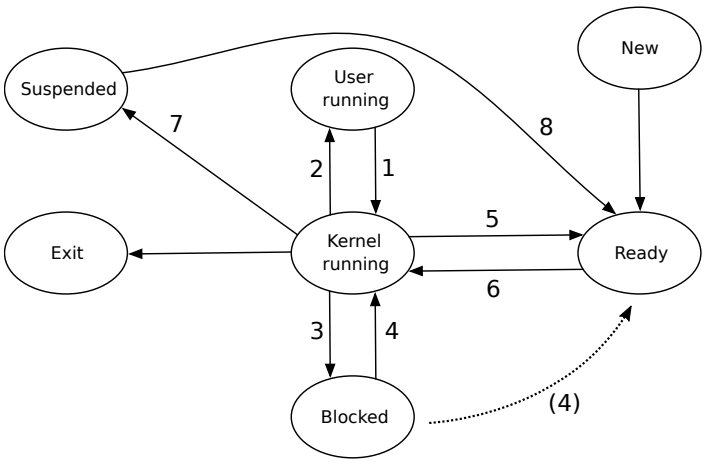
Lauseiden alut:

13. Käyttöjärjestelmän virtuaalimuistin hallintaosio (engl. *virtual memory management*) …
14. Käyttäjäkohtainen työpöytä (engl. *desktop manager*) …
15. IPC (engl. *inter-process communication*) …
16. Vuorontaja (engl. *scheduler*) …

Vaihtoehtoiset loput:

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

Ohje tehtäviin 17–19: 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:

17. Tilasiirtymä nro 1 tapahtuu, ...
18. Tilasiirtymä nro 5 tapahtuu, ...
19. Tilasiirtymä nro 6 tapahtuu, ...

Vaihtoehtoiset 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 prosessille on lähetetty odottelusignaali esim. näppäilemällä pääteyhteydessä Ctrl-Z.
D. kun prosessori siirtyy keskeytyksen tai käyttöjärjestelmäkutsun käsittelyyn.

20. **Väite:** Nykyaikaisen käyttöjärjestelmän yksi päätehtävä on eristää sovellusohjelmat laitteistosta siten, että laitteistoresurssien ohjaus kulkee aina yksinomaan ns. käyttöjärjestelmäkutsurajapinnan (*system call interface*) kautta. (A=kyllä; B=ei)

21. **Väite:** Tietokonejärjestelmän käyttöaste (engl. *utilization*) eli hyödylliseen laskentaan käytettävän ajan määrä kasvaa, kun lisätään kellokeskeytyksien määrää aikayksikössä. (A=kyllä; B=ei)

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. voi aiheutua käyttäjätilassa toimivan prosessin toimenpiteiden johdosta.
B. voidaan estää sovellusohjelmassa käyttämällä synkronointia.
C. vaatii prosessorilta useampia toimenpiteitä kuin aliohjelmakutsuun siirtyminen (esim. AMD64:n käsky `call`).
D. toimii vain moniydinprosessorissa.

Ohje tehtäviin 23–26: 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:

23. Prosessielementti (PCB) ...
24. Sivutaulu ...
25. Ready-jono ...
26. Blocked-jono ...

Vaihtoehtoiset loput:

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

Tutkittava esimerkki tehtäviin 27–28: 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]$ cat moi > kayttojarjestelmat.tex
[nieminen@halava tenttikys]$
```

27. **Väite:** komentojen jälkeen tiedoston `kayttojarjestelmat.tex` pituus on pienempi kuin 12600 tavua. (A=kyllä; B=ei)
28. **Väite:** komentojen jälkeen annettava komento `cat kayttojarjestelmat.tex` tulostaisi konsoliin tekstin "kayttojarjestelmat.tex" ja rivinvaihdon. (Muistinvirkistys: manuaalin antama synopsis komennolle `cat` on "concatenate files and print on the standard output") (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 29–30: C-mäisenä pseudokoodina tehty ehdotus rengaspuskuria käyttävän tuottaja-kuluttaja -ongelman ratkaisemiseksi POSIX-tyyppistä semaforipalvelua käyttäen. Yhteistä muistia käytetään puskurioperaatioissa. Semafori EMPTY mallintaa rengaspuskurin vapaata, kirjoitettavissa olevaa tilaa ja FULL mallintaa kirjoitettua mutta toistaiseksi käsittelemätöntä tilaa.

```
tuottaja(){
  while(true) { // tuotetaan loputtomiin
    tuota();    // tehdään yksi datapätkä
    sem_wait(EMPTY);
    sem_wait(MUTEX);
    siirra_puskuriin();
    sem_post(MUTEX);
    sem_post(FULL);
  }
}

kuluttaja(){
  while(true) { // kulutetaan loputtomiin
    sem_wait(FULL);
    sem_wait(MUTEX);
    lue_puskurista();
    sem_post(MUTEX);
    sem_post(EMPTY);
    kuluta();   // käytetään yksi datapätkä
  }
}

main(){
  EMPTY=tee_semafori( PUSKURIN_KOKO );
  FULL=tee_semafori( 0 );
  MUTEX=tee_semafori( 1 );
  kaynnista_saie(tuottaja);
  kaynnista_saie(kuluttaja);
}
```

29. **Väite:** Semaforien alustus on oikeellinen ongelman ratkaisemiseksi. (A=kyllä; B=ei)
30. **Väite:** Semaforien käyttö säikeissä on oikeellinen ongelman ratkaisemiseksi. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 31–33: Luentoesimerkeistä tuttua C-kielistä ohjelmanpätkää muistuttava koodi (POSIXin pthread-kirjastoa 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) {
    for (int i = 0; i < N; i++){
        pthread_mutex_lock(&mymutex);
        summa++;
    }
    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
  }
}
```

31. **Väite:** Esimerkin ohjelmalla on jossain vaiheessa suoritusta varmasti kolme säiettä. (A=kyllä; B=ei)
32. **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)
33. **Väite:** Esimerkin ohjelma voi aiheuttaa lukkiutumistilanteen (engl. *deadlock*), jossa sen suoritus jää jumiin. (A=kyllä; B=ei)

34. **Väite:** Kurssilta tutun x86_64 linux-ABIn mukaan C-kielisen aliohjelmakutsun parametrit ja paikalliset muuttujat sijaitsevat prosessin data-alueella (engl. *data segment*). (A=kyllä; B=ei)

35. **Väite:** Microkernel -tyyppinen käyttöjärjestelmä pyrkii suorittamaan palveluita mahdollisimman paljon käyttöjärjestelmätilassa (engl. *kernel mode*). (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:** Aina, kun rivi 4 tulee suoritukseen, se tapahtuu ainoastaan fork() -kutsun luomassa lapsiprosessissa. (A=kyllä; B=ei)
37. **Väite:** Aina, kun rivi 5 tulee suoritukseen, on olemassa sekä shell että sen luoma lapsiprosessi. (A=kyllä; B=ei)
38. **Väite:** Aina, kun rivi 11 tulee suoritukseen, se tapahtuu fork() -kutsun luomassa lapsiprosessissa. (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. Keskusmuistista 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	75
0x102	14	0x7f	80
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 levylle? (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. HUOM: suoritus alkaa osoitteesta "_start", kuten oikeassa GNU-työkaluilla tuotetussa ohjelmassa.

```
ali:
       subq    $3,%rcx   # vähennä 3 rekisterin RCX arvosta
       movq    %rcx,%rdi # kopioi rekisterin RCX sisältö rekisteriin RDI
       ret

_start:                  # Aloituspaikan symbolinen osoite on "_start"
       movq    $20,%rcx  # sijoita luku 20 rekisteriin RCX
       call    ali       # kutsu aliohjelmaa osoitteessa "ali"
       call    ali       # kutsu aliohjelmaa osoitteessa "ali"
       call    ali       # kutsu aliohjelmaa osoitteessa "ali"
       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 prosessi suorittaa siitä alken, kun käyttöjärjestelmä siirtää kontrollin nimettyyn aloituspisteeseen "_start"?
44. Mikä on rekisterin RCX sisältö ohjelmanpätkän suorituksen lopuksi?
45. Mikä on rekisterin RDI sisältö ohjelmanpätkän suorituksen lopuksi?

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) 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
```

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

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

```
printf("%d", 0x101);
```

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

Itsearvio kurssista

Laita loppuun vielä itsearvio kurssista skaalalla 0-5. Itsearvio ei vaikuta varsinaiseen arvosteluun, joka perustuu kysymyksiin 1-48. Itsearvioiden korrelaatiota todelliseen arvosanaan käytetään indikaattorina kurssin kehittämiseksi.

Vapaaehtoinen vapaa sana

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