ITKA203 – Käyttöjärjestelmät – tentti 5.7.2019

Kevään 2019 kurssin luennot, demot, esimerkkiohjelmat (yhteensopiva vuosina 2018 ja 2017 pidettyjen kurssien 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 tehtäviin 1–4: 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.

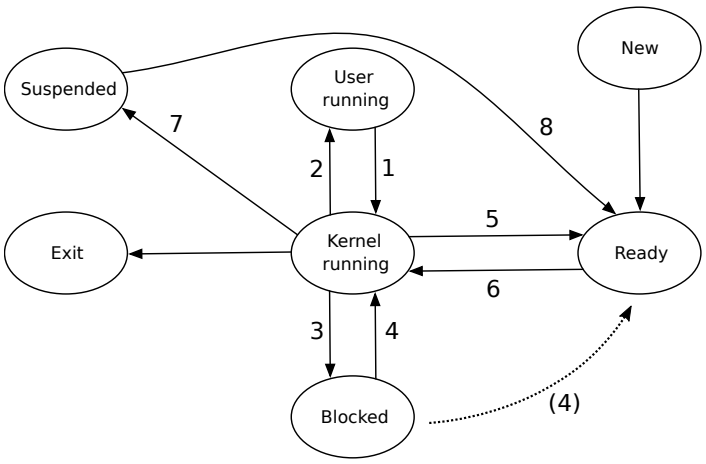
Lauseiden alut:

1. Käyttöjärjestelmän virtuaalimuistin hallintaosio (engl. *virtual memory management*) …
2. Käyttäjäkohtainen työpöytä (engl. *desktop manager*) …
3. IPC (engl. *inter-process communication*) …
4. Vuorontaja (engl. *scheduler*) …

Vaihtoehtoiset loput:

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

Ohje tehtäviin 5–7: 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. Tehtävässä tutkitaan prosessien tilasiirtymäkaaviota, jonka selitetekstit on korvattu numeroilla:



Lauseiden alut:

5. Tilasiirtymä nro 1 tapahtuu, ...
6. Tilasiirtymä nro 5 tapahtuu, ...
7. Tilasiirtymä nro 6 tapahtuu, ...

Vaihtoehtoiset loput:

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

Tutkittava esimerkki tehtäviin 8–9: 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]$
```

8. **Väite:** komentojen jälkeen tiedoston `kayttojarjestelmat.tex` pituus on pienempi kuin 12600 tavua. (A=kyllä; B=ei)
9. **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)

10. **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)

Tutkittava esimerkki tehtäviin 11–12: Järjestelmässä A kellokeskeytys tapahtuu kiinteästi 1000 kertaa sekunnissa. Järjestelmässä B kellokeskeytys tapahtuu kiinteästi 50 kertaa sekunnissa. Molemmissa vuoronnusmenettelynä on priorisoimaton kiertojono (engl. *round robin*) ja uudet prosessit liitetään jonon viimeiseksi. Ajatellaan tilannetta, jossa molemmissa käynnistetään peräjälkeen 100 intensiivisesti laskevaa sovellusta, joista jokainen tarvitsee prosessoriaikaa 10 sekuntia. Välittömästi sen jälkeen käynnistetään 10 sovellusta, joiden halutaan tulostavan tietty vastaus. Jokainen näistä viimeksi mainituista vaatii 0.002 sekuntia laskenta-aikaa. Järjestelmässä ei ole mainittujen 110 sovelluksen lisäksi mitään muuta kuormaa.

11. Kummassa järjestelmässä (A vai B) prosessorin käyttöaste (engl. *utilization*) on parempi?
12. Kummassa järjestelmässä (A vai B) on parempi vasteaika (engl. *response time*) prosesseilla, jotka tarvitsevat vain 0.002 sekuntia laskentaa?

Ohje tehtävään 13: 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:

13. Prosessorin keskeytys ...

Vaihtoehtoiset loput (mahdollisesti useita sopivia):

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

Ohje tehtävään 14: 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:

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

Vaihtoehtoiset loput (mahdollisesti useita sopivia):

A. järjestelmäkutsu exit() tapahtuu sijoittamalla rekisteriin RAX luku 60 ja suorittamalla syscall-konekielikäsky.
B. shell-komento *echo* tulostaa argumenttinsa.
C. Ikkunan oikeassa yläkulmassa tulee olla rastin muotoinen symboli, jolla ohjelman voi lopettaa.
D. käyttäjien kotihakemistot tulee sijoittaa hakemistoon nimeltä */nashomeN/*, missä N on kokonaisluku

15. **Väite:** Tyypillisessä nykyprosessorissa (esim. AMD64) on käyttöjärjestelmätilassa (engl. *kernel mode*) käytettävissä pienempi joukko konekielikäskyjä kuin käyttäjätilassa (engl. *user mode*). (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 16–17: Kurssin luennoilta ja demoista tutussa ympäristössä (Linux,bash) tehty yksittäinen, POSIX-syntaksin mukainen komentorivi:

```
kissa koira kissakoira argh kissa | arg | kissakoira grep echo > echo
```

16. Montako komentoa shell yrittää käynnistää koko rivin suorittamiseksi?
17. Montako argumenttia rivillä on yhteensä?

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

19. **Väite:** Päätavoite ohjelmistokerrosten ja niiden välisten rajapintastandardien laatimisessa on helpottaa saman toiminnallisuuden toteuttamista eri alustoille. (A=kyllä; B=ei)

Ohje tehtävään 20: Järjestä seuraavat muistikomponentit niiden kapasiteetin mukaan, eli mihin mahtuu eniten dataa kerralla: A=kovalevy, B=keskusmuisti, C=välimuisti, D=rekisteri.

20. Vastauksessasi on neljä järjestettyä kirjainta: isoin muisti ensin, pienin muisti viimeisenä.

Ohje tehtäviin 21–22: 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. Laiteriippuva I/O -ohjelmisto …
22. Laitteistoriippumaton I/O -ohjelmisto …

Vaihtoehtoiset loput:

A. sisältää laiteohjainten ajurit.
B. määrittää järjestelmälle yhtenäisen lohkokoon.
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.

23. **Väite:** Historiallisesti merkittävän ENIAC-tietokoneen (valmistettu vuonna 1946) mukana toimitettiin käyttöjärjestelmä nimeltä Multics, jossa oli jo mukana monia nykyisten käyttöjärjestelmien ominaisuuksia. (A=kyllä; B=ei)

Ohje tehtäviin 24–27: 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.

Lauseiden alut:

24. Säie (engl. *thread*) …
25. Kehystaulu (engl. *frame table*) …
26. Sovelluksen binäärirajapinta (*Application Binary Interface*) …
27. Signaali (esim. POSIXin komennolla kill lähetetty) …

Vaihtoehtoiset loput:

A. liittyy konekielisen aliohjelmakutsun muotoon.
B. liittyy muistinhallintaan.
C. liittyy vuoronnukseen.
D. liittyy prosessien väliseen kommunikointiin.

Ohje tehtäviin 28–31: 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:

28. Prosessitaulu …
29. Prosessielementti (PCB) …
30. Sivutaulu …
31. Ready-jono …

Vaihtoehtoiset loput:

A. liittyy useiden eri prosessien käsittelyyn.
B. on prosessikohtainen.
C. ei välttämättä liity prosessien tarpeisiin.

Tutkittava esimerkki tehtäviin 32–33: 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);
}
```

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

34. **Väite:** P.J. Denningin ym. 1970-luvulla kehittelemä lokaalisuusperiaate (engl. *principle of locality*) on pohjana sille, että konekieliset aliohjelmat voivat kutsua itseään rekursiivisesti. (A=kyllä; B=ei)

35. **Väite:** POSIXin semaforissa on ei-negatiivinen kokonaisluku ja mahdollisesti tyhjä joukko säikeitä. (A=kyllä; B=ei)
36. **Väite:** POSIXin semaforikutsun `sem_wait(s)` suorituksessa semaforin `s` arvo muuttuu aina. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 37–39: 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  }
```

37. **Väite:** Aina, kun rivi 4 tulee suoritukseen, se tapahtuu ainoastaan fork() -kutsun luomassa lapsiprosessissa. (A=kyllä; B=ei)
38. **Väite:** Aina, kun rivi 5 tulee suoritukseen, on olemassa sekä shell että sen luoma lapsiprosessi. (A=kyllä; B=ei)
39. **Väite:** Aina, kun rivi 7 tulee suoritukseen, se tapahtuu alkuperäisessä shell-prosessissa, ei siis fork():n luomassa lapsiprosessissa. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 40–43: Kuvitellaan, että meillä on käytössä yksinkertainen tietokone, jonka virtuaalimuistiosoitteissa on 20 bittiä, joista ensimmäiset 8 ilmoittavat sivunumeron ja loput 12 ilmoittavat tavuindeksin sivun sisällä. 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 2 ja 7) osalta.

Prosessin (PID=2) sivutaulun kartoitetut rivit:

virt. sivu	fyys. sivu	muist. (P)	dirty (D)	diskIndex
0x02	0x100	1	0	0x0010
0x03	0x000	0	0	0x0022
0x2e	0x106	1	0	0x00bb
0x7f	0x101	1	1	0x00bc

Prosessin (PID=7) sivutaulun kartoitetut rivit:

virt. sivu	fyys. sivu	muist. (P)	dirty (D)	diskIndex
0x02	0x000	0	0	0x0004
0x03	0x102	1	1	0x0008
0x04	0x000	0	0	0x00f2
0x2e	0x104	1	1	0x006c
0x7f	0x107	1	1	0x0007

Järjestelmän kehystaulu:

fyys. sivu	omistajan PID	omistajan PTE#	aikayksiköt edellisen käyttökerran jälkeen
0x100	2	0x02	19
0x101	2	0x7f	150
0x102	7	0x03	16
0x103	234	0x45	8
0x104	7	0x2e	12
0x105	876	0x45	4
0x106	2	0x2e	175
0x107	7	0x7f	9

40. Mihin fyysiseen muistiosoitteeseen kohdistuisi prosessin 7 tekemä kirjoitus virtuaalimuistiosoitteeseen 0x7f123?
41. Mistä virtuaaliosoitteesta prosessi 7 saisi käyttöönsä tavun, joka sijaitsee kovalevyllä indeksissä 0xf2345 heittovaihto-osion / -tiedoston (engl. *swap space*) alusta lukien?
42. Prosessi 2 suorittaa hyppykäskyn aliohjelmaan muistiosoitteessa 0x2e07f. Tapahtuuko prosessorissa sivuvirhe / sivunvaihtokeskeytys? (A=kyllä; B=ei)
43. Prosessissa 876 aiheutuu sivunvaihtokeskeytys. Korvausalgoritmi on LRU. Onko jonkin sivun sisältö tallennettava levylle? (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 44–46: 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    $2,%rcx   # vähennä 2 rekisterin RCX arvosta
       movq    %rcx,%rdi # kopioi rekisterin RCX sisältö rekisteriin RDI
       ret

_start:                  # Aloituspaikan symbolinen osoite on "_start"
       movq    $22,%rcx  # sijoita luku 22 rekisteriin RCX
       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
```

44. Kuinka monta konekielikäskyä ohjelmanpätkän jälki sisältää, ts. kuinka monta käskyä prosessi suorittaa siitä alken, kun käyttöjärjestelmä siirtää kontrollin nimettyyn aloituspisteeseen "_start"?
45. Mikä on rekisterin RCX sisältö ohjelmanpätkän suorituksen lopuksi?
46. Mikä on rekisterin RDI sisältö ohjelmanpätkän suorituksen lopuksi?

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 loppunut ja siihen sisältyvä käsky `ret` on viimeisimpänä suoritettu? Ilmoita *heksalukuna*.

Itsearvio ja vapaa sana: Laita loppuun itsearvio kurssista (asteikko 0-5). Itsearvio ei vaikuta varsinaiseen arvosteluun. Arvioiden korrelaatiota arvosanaan käytetään indikaattorina kurssikehityksessä. Halutessasi voit antaa myös palautetta tai kommentoida muuten. Vastauksen muoto on vapaa, eikä se vaikuta arvosteluun.