ITKA203 – Käyttöjärjestelmät – tentti 15.5.2019

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

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

Ohje tehtäviin 2–5: 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:

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

Vaihtoehtoiset loput:

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

6. **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)

7. **Väite:** Mikä tahansa prosessin säikeistä voi käyttää prosessille avattuja tiedostoja tasa-arvoisesti, jos ei säikeitä erikseen synkronoida. (A=kyllä; B=ei)

Ohje tehtäviin 8–11: 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:

8. Säie (engl. *thread*) …
9. Kehystaulu (engl. *frame table*) …
10. Sovelluksen binäärirajapinta (*Application Binary Interface*) …
11. Signaali (esim. POSIXin komennolla kill lähetetty) …

Vaihtoehtoiset loput:

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

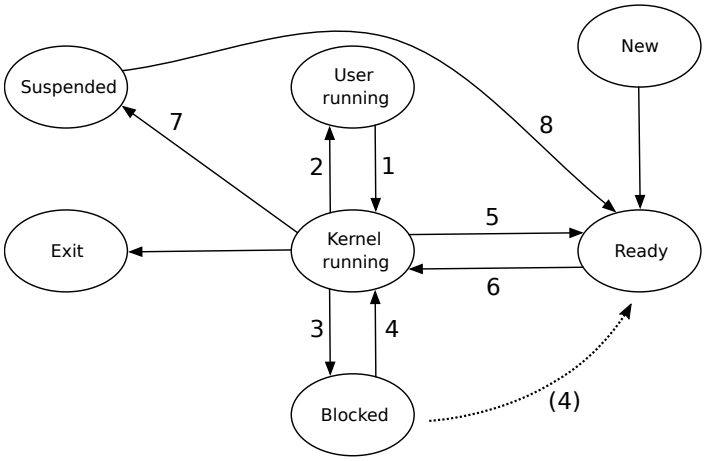
12. **Väite:** Moderni prosessori (esim. AMD64) suorittaa keskeytyskäsittelijän ensimmäisen konekielikäskyn käyttöjärjestelmätilassa (engl. *kernel mode*). (A=kyllä; B=ei)

Ohje tehtävään 13: Järjestä seuraavat muistikomponentit niiden nopeuden mukaan: A=kovalevy, B=keskusmuisti, C=välimuisti, D=rekisteri.

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

14. **Väite:** Nykyaikaisen käyttöjärjestelmän yksi päätehtävä on varmistaa, että jokainen käyttäjä pystyy tasa-arvoisesti hallitsemaan suoraan kaikkia tietokoneeseen liitettyjä laitteita ja muokkaamaan kaikkia tiedostoja (engl. *universal root access principle*). (A=kyllä; B=ei)

Ohje tehtäviin 15–17: 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:

15. Tilasiirtymä nro 4 tai (4) tapahtuu, …
16. Tilasiirtymä nro 5 tapahtuu, …
17. Tilasiirtymä nro 6 tapahtuu, …

Vaihtoehtoiset loput:

A. kun prosessi on tehnyt pyynnön esim. lukeakseen merkkejä tiedostosta, eikä luettava data ole vielä saapunut fyysiseltä laitteelta toimitettavaksi prosessille saakka.
B. kun käyttöjärjestelmän vuorontaja siirtää suoritukseen toisen prosessin, vaikka nykyinenkin prosessi pystyisi jatkamaan laskemista jo seuraavassa konekielikäskyssään.
C. kun prosessi aloittaa käyttöjärjestelmän vuorontajalta saamansa aikaikkunan käytön.
D. kun ohjelman odottama palvelu, esim. I/O-operaatio, on valmis.

Tutkittava esimerkki tehtäviin 18–19: 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]$
```

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

Ohje tehtävään 20: 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:

20. Prosessorin keskeytys ...

Vaihtoehtoiset loput (mahdollisesti useita sopivia):

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

Tutkittava esimerkki tehtäviin 21–22: Kurssin luennoilta ja demoista tutussa ympäristössä (Linux,bash) tehty yksittäinen, POSIX-syntaksin mukainen komentorivi:

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

21. Montako komentoa shell yrittää käynnistää koko rivin suorittamiseksi?
22. Montako argumenttia rivillä on yhteensä?

23. **Väite:** Laitteistoriippumaton I/O-ohjelmiston kerros määrittelee laitteille nimeämiskäytännön. (A=kyllä; B=ei)
24. **Väite:** I/O-operaation pyytäminen aiheuttaa aina pyytävän prosessin siirron suoraan ready-tilaan. (A=kyllä; B=ei)

Ohje tehtäviin 25–28: 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:

25. Prosessitaulu ...
26. Prosessielementti (PCB) ...
27. Sivutaulu ...
28. Ready-jono ...

Vaihtoehtoiset loput:

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

Ohje tehtävään 29: Kurssin esimerkeissä ja demoissa on tutkittu Linux-käyttöjärjestelmälle ja x86-64 -prosessoriarkkitehtuurille käännettyjä konekielisiä ohjelmia. Missä järjestyksessä seuraavat alueet/segmentit sijaitsevat muistiosoitteiden suuruusjärjestyksessä nykyään (2010-luku) käytössä olevan ABI:n mukaan? Luettele pienimmästä osoitealueesta suurimpaan: A=keko; B=pino; C=koodi; D=data.

29. Vastauksessasi on neljä järjestettyä kirjainta: pienimmät osoitteet ensin, suurimmat viimeisenä

30. **Väite:** P.J. Denningin ym. 1970-luvulla kehittelemä lokaalisuusperiaate (engl. *principle of locality*) on pohjana mm. nykyisen POSIXin kieliasetusten eli lokaalien (engl. *locale*) määrittelylle. (A=kyllä; B=ei)

31. **Väite:** Microkernel -tyyppinen käyttöjärjestelmä pyrkii suorittamaan palveluita mahdollisimman paljon käyttöjärjestelmätilassa (engl. *kernel mode*). (A=kyllä; B=ei)

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( 0 );
  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:** POSIXin semafori on binäärinen lippumuuttuja. (A=kyllä; B=ei)
35. **Väite:** POSIXin semaforikutsun `sem_wait(s)` suorituksessa semaforin `s` arvo muuttuu aina. (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 7 tulee suoritukseen, se tapahtuu alkuperäisessä shell-prosessissa, ei siis fork():n 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. 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

39. Mihin fyysiseen muistiosoitteeseen kohdistuisi prosessin 7 tekemä kirjoitus virtuaalimuistiosoitteeseen 0x7f123?
40. Mistä virtuaaliosoitteesta prosessi 7 saisi käyttöönsä tavun, joka sijaitsee kovalevyllä indeksissä 0xf2345 heittovaihto-osion / -tiedoston (engl. *swap space*) alusta lukien?
41. Prosessi 2 suorittaa hyppykäskyn aliohjelmaan muistiosoitteessa 0x2e07f. Tapahtuuko prosessorissa sivuvirhe / sivunvaihtokeskeytys? (A=kyllä; B=ei)
42. Prosessissa 876 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. Muista: suoritus alkaa osoitteesta "_start".

```
ali:
       addq    %rdi,%rdi # kerro RDI kahdella lisäämällä se itseensä
       dec     %rcx      # vähennä RCX:stä 1 ja jätä lopputulos RCX:ään
       jz      ohi       # hyppää, jos edellisen laskun tulos oli 0
       call    ali       # kutsu aliohjelmaa
ohi:
       ret               # palaa aliohjelmasta
_start:
       movq    $2,%rcx   # luku 2 rekisteriin RCX
       movq    $2,%rdi   # luku 2 rekisteriin RDI
       call    ali       # kutsu aliohjelmaa
       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 suorittaminen alusta loppuun vaatii?
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. Kuinka monta tavua muistia on varattu nykyisen aktivaation väliaikaisten muuttujien (eli. lokaalit muuttujat ja paikalliset kopiot parametreista) säilyttämistä varten? Ilmoita *kymmenjärjestelmän* lukuna.
47. Mikä tulee olemaan RIP-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", 0x11);
```

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

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.