ITKA203 – Käyttöjärjestelmät – tentti 7.7.2017

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 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 poistetaan tämän tenttikerran arvostelusta 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 yksikäsitteinen oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

Lauseiden alut:

1. Prosessitaulu …
2. Prosessielementti (PCB) …
3. Sivutaulu …
4. Ready-jono …

Vaihtoehtoiset loput:

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

Tutkittava esimerkki tehtäviin 5–6: 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.

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

Ohje tehtäviin 7–10: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että lause on totta. Alkuihin on yksikäsitteinen oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

Lauseiden alut:

7. I/O -ohjelmisto (engl. *I/O subsystem*) ...
8. Käyttöjärjestelmän virtuaalimuistin hallintaosio (engl. *virtual memory management*) ...
9. IPC (engl. *inter-process communication*) ...
10. Graafinen tiedostoselain (engl. *file browser*) ...

Vaihtoehtoiset loput:

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

Tutkittava esimerkki tehtäviin 11–12: 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]$
```

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

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. 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. käyttäjien kotihakemistot tulee sijoittaa hakemistoon nimeltä */nashomeN/*, missä N on kokonaisluku
C. shell-komento *echo* tulostaa argumenttinsa.
D. Ikkunan oikeassa yläkulmassa tulee olla rastin muotoinen symboli, jolla ohjelman voi lopettaa.

Tutkittava esimerkki tehtäviin 14–15: 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
```

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

Ohje tehtävään 16: Järjestä seuraavat muistikomponentit niiden nopeuden mukaan: A=kovalevy, B=keskusmuisti, C=rekisteri, D=välimuisti.

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

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

Lauseiden alut:

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

Vaihtoehtoiset loput:

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

21. **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)

22. **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 tehtävään 23: 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:

23. Prosessorin keskeytys ...

Vaihtoehtoiset loput (mahdollisesti useita sopivia):

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

Ohje tehtäviin 24–25: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että lause on totta. Alkuihin on yksikäsitteinen oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

Lauseiden alut:

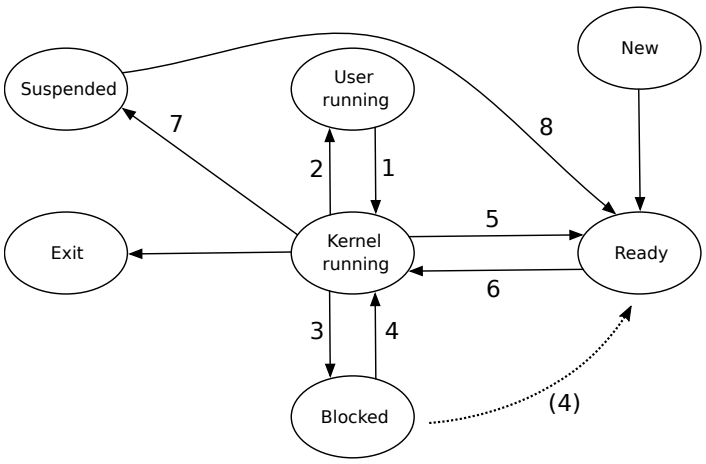
24. Laiteriippuva I/O -ohjelmisto ...
25. Laitteistoriippumaton I/O -ohjelmisto ...

Vaihtoehtoiset loput:

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

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

Ohje tehtäviin 27–29: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että lause on totta. Alkuihin on yksikäsitteinen oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään. Tehtävässä tutkitaan prosessien tilasiirtymäkaaviota, jonka selitetekstit on korvattu numeroilla:



Lauseiden alut:

27. Tilasiirtymä nro 1 tapahtuu, ...
28. Tilasiirtymä nro 5 tapahtuu, ...
29. 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.

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

32. **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)

Tutkittava esimerkki tehtäviin 33–35: 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
  }
}
```

33. **Väite:** Esimerkin ohjelmalla on jossain vaiheessa suoritusta varmasti kolme säiettä. (A=kyllä; B=ei)
34. **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)
35. **Väite:** Esimerkin ohjelma voi aiheuttaa lukkiutumistilanteen (engl. *deadlock*), jossa sen suoritus jää jumiin. (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, 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	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 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", 0x7f);
```

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.