

ITKA203 – Käyttöjärjestelmät – tentti 17.8.2018

Kevään 2018 kurssin luennot, demot, esimerkkiohjelmat (yhteensopiva kevään 2017 kurssin kanssa) / Paavo Nieminen <paavo.j.nieminen@ju.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 huomioidaan tämän tenttikerran arvostelussa ja muokataan kysymyspankissa yksiselitteisempään muotoon tulevaisuutta varten.

Numeroidut kysymyskohdat 1–48

Ohje tehtävään 1: 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:

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

Vaihtoehdot loput (mahdollisesti useita sopivia):

- A. kellokeskeytyksen tulee tapahtua 50, 100 tai 1000 kertaa sekunnissa
- B. shell-komento *c99* käynnistää C99-standardin mukaisen C-kääntäjän
- C. Tiedostovalikko (engl. *File*) on valikkopalkin vasemmanpuoleisin valikko
- D. järjestelmäkutsu *exit()* tapahtuu sijoittamalla rekisteriin RAX luku 60 ja suorittamalla *syscall*-konekielikäskey

Ohje tehtäviin 2–3: 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.

Lauseiden alut:

2. Laiteriippuva I/O -ohjelmisto ...
3. Laitteistoriippumaton I/O -ohjelmisto ...

Vaihtoehdot loput (mahdollisesti useita sopivia):

- A. voi tulla suoritukseen prosessorin keskeytyksen seurauksena.
- B. voi tulla suoritukseen sovellusohjelman pyynnön seurauksena.
- C. sisältää laiteohjainten ajurit.
- D. määrittää järjestelmälle yhtenäisen lohkokoon.

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. Prosessitaulu ...
5. Prosessielementti (PCB) ...
6. Sivutaulu ...
7. Ready-jono ...

Vaihtoehdot loput:

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

Ohje tehtävään 8: Järjestä seuraavat muistikomponentit niiden nopeuden mukaan: A=kovalevy, B=rekisteri, C=välimuisti, D=keskusmuisti.

8. Vastauksessasi on neljä järjestettyä kirjainta: nopein ensin, hitain viimeisenä.
9. **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)
10. **Väite:** Pää tavoite 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 11: 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):

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

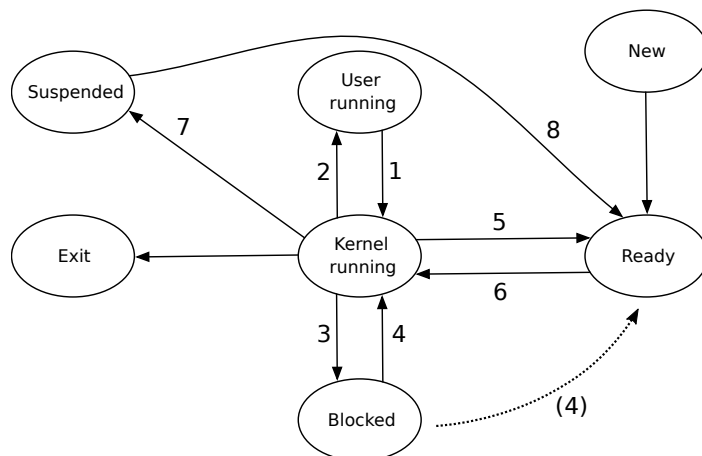
Ohje tehtäviin 13–16: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että lause on totta. Alkuihin on *yksi, yksikäsitteinen*, oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

Lauseiden alut:

Vaihtoehtoiset loput:

13. Käyttöjärjestelmän virtuaali-
muistin hallintaosio (engl. *virtual
memory management*) ...
 - A. tarjoaa palvelut mm. keskinäiseen poissulkuun.
 - B. ei ole välttämätön osa nykyaikaista käyttöjärjestelmää.
 - C. tarvitaan ainoastaan, kun suoritetaan virtuaalikoneita.
 - D. tekee toimenpiteitä sivuvirheen (engl. *page fault*) yhteydessä.
 - E. tekee toimenpiteitä jokaisen kellokeskeytyksen yhteydessä.
14. Käyttäjakohtainen työpöytä
(engl. *desktop manager*) ...
15. IPC (engl. *inter-process commu-
nication*) ...
16. Vuorontaja (engl. *scheduler*) ...
17. **Väite:** Tyypillisessä nykyprosessorissa (esim. AMD64) on käyttöjärjestelmätilassa (engl. *kernel mode*) käytettävissä pienempi joukko konekielikäskyjä kuin käyttäjätilassa (engl. *user mode*). (A=kyllä; B=ei)

Ohje tehtäviin 18–20: 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. Tehtävässä tutkitaan prosessien tilasiirtymäkaaviota, jonka selitetekstit on korvattu numeroilla:



Lauseiden alut:

18. Tilasiirtymä nro 1 tapahtuu, ...
19. Tilasiirtymä nro 2 tapahtuu, ...
20. Tilasiirtymä nro 3 tapahtuu, ...

Vaihtoehtoiset loput:

- A. kun tapahtuu paluu käyttöjärjestelmäkutsun tai keskeytyksen käsittelijästä (esim. AMD64:n konekielikäsky *sysret* tai *iret*).
 - B. kun prosessi on tehnyt pyynnön esim. lukeakseen merkkejä tiedostosta, eikä luettava data ole vielä saapunut fyysiseltä laitteelta toimitettavaksi prosessille saakka.
 - C. kun käyttöjärjestelmän vuorontaja siirtää suoritukseen toisen prosessin, vaikka nykyinenkin prosessi pystyisi jatkamaan laskemista jo seuraavassa konekielikäskyssään.
 - D. kun prosessori siirtyy keskeytyksen tai käyttöjärjestelmäkutsun käsittelyyn.
21. **Väite:** Prosessin kaikilla säikeillä on aina sama sisältö jokaisessa prosessorin rekisterissä (esim. AMD64:n RIP:ssä ja RAX:ssä) (A=kyllä; B=ei)

Ohje tehtäviin 22–25: 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:

22. Keskinäinen poissulku (engl. *mutual exclusion, Mutex*) ...
23. FLIH (*first-level interrupt handling*) ...
24. Semafori (*semaphore*) ...
25. IP-rekisteri (*instruction pointer*) ...

Vaihtoehtoiset loput:

- A. on tietoverkkoon liitetyn laitteen osoite.
 - B. on osa prosessorilaitteiston toimintaa.
 - C. liittyy tietokoneiden muistihierarkian perusideaan.
 - D. liittyy kilpa-ajotilanteiden (engl. *race condition*) ratkomiseen.
26. **Väite:** Tietokonejärjestelmän käyttöaste (engl. *utilization*) eli hyödylliseen laskentaan käytettävän ajan määrä kasvaa, kun lisätään kellokeskeytyksien määrää aikayksikössä. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 27–28: 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
```

27. Montako erillistä komentoa rivillä on yhteensä (rivin jäsentymisen kannalta, riippumatta siitä, toimivatko kaikki komennot jossain järjestelmässä oikeasti)?
28. Montako erillistä argumenttia rivillä on yhteensä?

Tutkittava esimerkki tehtäviin 29–31: Luento-esimerkeistä tuttua C-kielistä ohjelmanpätkeä 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 = 1; i <= N; i++){
        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
    }
}
```

29. **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)
30. **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)
31. **Väite:** Esimerkin ohjelma voi aiheuttaa lukkiutumistilanteen (engl. *deadlock*), jossa sen suoritus jää jumiin. (A=kyllä; B=ei)
32. **Väite:** Kurssilta tutun x86_64 linux-ABI:n mukaan C-kielisen aliohjelmakutsun parametrit ja paikalliset muuttujat sijaitsevat prosessin data-alueella (engl. *data segment*). (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 33–34: C-mäisenä pseudokoodina tehty ehdotus rengaspuskuriä 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(EMPTY);
    }
}

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

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

33. **Väite:** Semaforien alustus on oikeellinen ongelman ratkaisemiseksi. (A=kyllä; B=ei)
34. **Väite:** Semaforien käyttö säikeissä on oikeellinen ongelman ratkaisemiseksi. (A=kyllä; B=ei)
35. **Väite:** Microkernel -tyyppinen käyttöjärjestelmä pyrkii suorittamaan palveluita mahdollisimman paljon käyttäjätilassa (engl. *user 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 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 käytävissä 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 hyppykäslyn aliohjelmaan muistiosoitteessa 0x2e07f. Tapahtuuko prosessorissa sivuvirhe / sivunvaihtokeskeytys? (A=kyllä; B=ei)
42. Prosessissa 876 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. 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 RDI:n 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 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) 0x00007fffffffddcf0
```

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

```
0x7fffffffdd08: 0x0000000100000000
0x7fffffffdd00: 0x00007fffffffde38
0x7fffffffddcf8: 0x0000000000400647
kanta --> 0x7fffffffddcf0: 0x00007fffffffdd50
0x7fffffffddce8: 0x0000000000000001
0x7fffffffddce0: 0x0000000000000000
0x7fffffffddcd8: 0x0000000000000000
0x7fffffffddcd0: 0x0000000000000000
0x7fffffffddcc8: 0x0000000000000000
huippu --> 0x7fffffffddcc0: 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 loppumassa ja siihen sisältyvä käsky `ret` on juuri tulossa suoritukseen.

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

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

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.