ITKA203 – Käyttöjärjestelmät – tentti 6.7.2018

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: 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. Prosessorin keskeytys …

Vaihtoehtoiset loput (mahdollisesti useita sopivia):

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

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. Prosessielementti (PCB) …
3. Sivutaulu …
4. Ready-jono …
5. Blocked-jono …

Vaihtoehtoiset loput:

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

Tutkittava esimerkki tehtäviin 6–7: Kurssin luennoilta ja demoista tutussa ympäristössä (Linux,bash) tehty yksittäinen, POSIX-syntaksin mukainen komentorivi:

```
arg arg | arg | grep -i "arg arg arg" | arg > arg
```

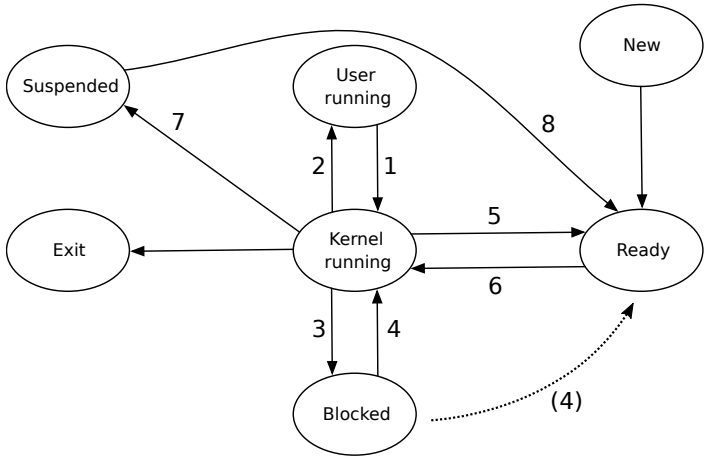
6. Montako komentoa shell yrittää käynnistää koko rivin suorittamiseksi?
7. Montako argumenttia rivillä on yhteensä?

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

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

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

Ohje tehtäviin 10–12: 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. Tehtävässä tutkitaan prosessien tilasiirtymäkaaviota, jonka selitetekstit on korvattu numeroilla:



Lauseiden alut:

10. Tilasiirtymä nro 1 tapahtuu, ...
11. Tilasiirtymä nro 2 tapahtuu, ...
12. Tilasiirtymä nro 3 tapahtuu, ...

Vaihtoehtoiset loput:

A. kun käyttöjärjestelmän vuorontaja siirtää suoritukseen toisen prosessin, vaikka nykyinenkin prosessi pystyisi jatkamaan laskemista jo seuraavassa konekielikäskyssään.
B. kun tapahtuu paluu käyttöjärjestelmäkutsun tai keskeytyksen käsittelijästä (esim. AMD64:n konekielikäsky *sysret* tai *iret*).
C. kun prosessi on tehnyt pyynnön esim. lukeakseen merkkejä tiedostosta, eikä luettava data ole vielä saapunut fyysiseltä laitteelta toimitettavaksi prosessille saakka.
D. kun prosessori siirtyy keskeytyksen tai käyttöjärjestelmäkutsun käsittelyyn.

13. **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 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. Avustus (engl. *Help*) on valikkopalkin oikeanpuoleisin valikko
B. komento *startx* käynnistää X-ikkunointijärjestelmän
C. kellokeskeytyksen tulee tapahtua 50, 100 tai 1000 kertaa sekunnissa
D. shell-komento *c99* käynnistää C99-standardin mukaisen C-kääntäjän

15. **Väite:** Prosessin konekieliset käskyt näkyvät samoissa virtuaalimuistiosoitteissa kaikissa prosessin säikeissä. (A=kyllä; B=ei)

Ohje tehtäviin 16–17: 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.

Lauseiden alut:

16. Laiteriippuva I/O -ohjelmisto …
17. Laitteistoriippumaton I/O -ohjelmisto …

Vaihtoehtoiset loput:

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

18. **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 tehtäviin 19–22: 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:

19. Käyttöjärjestelmän virtuaalimuistin hallintaosio (engl. *virtual memory management*) …
20. Käyttäjäkohtainen työpöytä (engl. *desktop manager*) …
21. IPC (engl. *inter-process communication*) …
22. Vuorontaja (engl. *scheduler*) …

Vaihtoehtoiset loput:

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

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

Vaihtoehtoiset loput:

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

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

28. **Väite:** Moderni prosessori (esim. AMD64) suorittaa keskeytyskäsittelijän ensimmäisen konekielikäskyn käyttäjätilassa (engl. *user mode*). (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( 0 );
  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)

31. **Väite:** P.J. Denningin ym. 1970-luvulla kehittelemä lokaalisuusperiaate (engl. *principle of locality*) on pohjana mm. heittovaihdolle (engl. *swap*). (A=kyllä; B=ei)

32. **Väite:** Kekoaluetta (engl. *heap*) käytetään dynaamisesti luotavien tietorakenteiden/olioiden säilyttämiseen. (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) {
    pthread_mutex_lock(&mymutex);
    for (int i = 1; i <= N; i++){
        summa++;
    }
    pthread_mutex_unlock(&mymutex);
    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 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)
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:** Rivin 5 suorittaminen tapahtuu shellin fork():lla luomassa lapsiprosessissa. (A=kyllä; B=ei)
37. **Väite:** Rivin 7 suorittaminen tapahtuu alkuperäisessä shell-prosessissa, ei siis fork():n luomassa lapsiprosessissa. (A=kyllä; B=ei)
38. **Väite:** Rivin 12 suorittamisen jälkeen kutsun exit(1) tehneen prosessin suoritus jatkuu riviltä 2, jossa luetaan käyttäjältä uusi komento. (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. 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
```

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. Kuinka monta tavua muistia on varattu nykyisen aktivaation väliaikaisten muuttujien (eli. lokaalit muuttujat ja paikalliset kopiot parametreista) säilyttämistä varten? Ilmoita *kymmenjärjestelmän* lukuna.
47. Jos tämä aliohjelma seuraavassa konekielikäskyssään kutsuisi jotakin toista aliohjelmaa, niin missä muistiosoitteessa tulisi sijaitsemaan paluuosoite?

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"*)

Itsearvio ja vapaaehtoinen vapaa sana: 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. Halutessasi voit antaa loppuun myös palautetta tai kommentoida muuten kurssia tai tenttiä. Vastauksen muoto on vapaa, eikä se vaikuta arvosteluun.