ITKA203 – Käyttöjärjestelmät – tentti 19.8.2016

Kevään 2016 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. Mikäli tehtävässä ei mainita poikkeuksista, jokainen 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

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

Ohje tehtävään 2: 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:

2. 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. shell-komennolla *kill* voi lähettää prosessille signaalin.

Ohje tehtävään 3: Järjestä seuraavat muistikomponentit niiden nopeuden mukaan: A=kovalevy, B=keskusmuisti, C=rekisteri, D=välimuisti.

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

4. **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)

5. **Väite:** Prosessin jaetut muistialueet näkyvät samoissa osoitteissa kaikissa prosessin säikeissä. (A=kyllä; B=ei)

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

7. **Väite:** Päätavoite ohjelmistokerrosten ja niiden välisten rajapintojen määrittelyssä on estää kilpailijoita toteuttamasta tuotteita, jotka toimivat samalla tavoin kuin valmistajan oma tuote. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 8–9: Kurssin luennoilta ja demoista tutussa ympäristössä (Linux,bash) tehty yksittäinen, POSIX-syntaksin mukainen komentorivi:

```
arg arg arg echo | hecho hecho | ostha --lecho | arg arg > echo
```

8. Montako komentoa rivillä on yhteensä?
9. Montako argumenttia rivillä on yhteensä?

Ohje tehtäviin 10–13: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että kukin lause on totta. Alkuihin on yksikäsitteinen oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

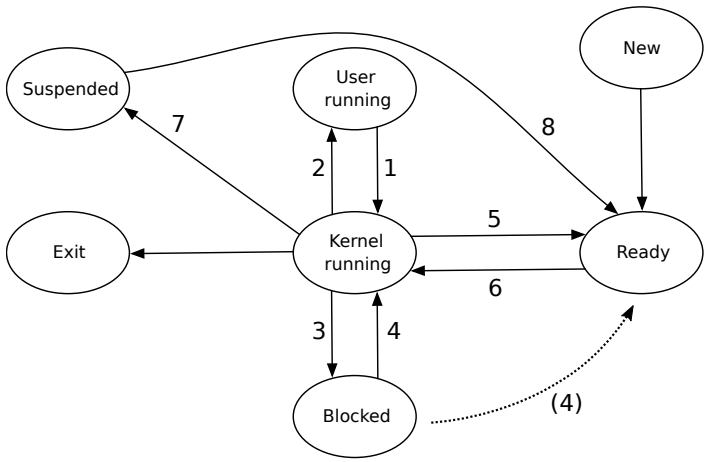
Lauseiden alut:

10. Prosessitaulu …
11. Prosessielementti (PCB) …
12. Sivutaulu …
13. Ready-jono …

Vaihtoehtoiset loput:

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

Ohje tehtäviin 14–16: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että kukin 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:

14. Tilasiirtymä nro 1 tapahtuu, …
15. Tilasiirtymä nro 5 tapahtuu, …
16. Tilasiirtymä nro 6 tapahtuu, …

Vaihtoehtoiset loput:

A. kun prosessori siirtyy keskeytyksen tai käyttöjärjestelmäkutsun käsittelyyn.
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 prosessille on lähetetty odottelusignaali esim. näppäilemällä pääteyhteydessä Ctrl-Z.

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

Lauseiden alut:

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

Vaihtoehtoiset loput:

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

Tutkittava esimerkki tehtäviin 21–22: 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 10740 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]$
```

21. **Väite:** komentojen jälkeen tiedoston `kayttojarjestelmat.tex` pituus on pienempi kuin 10740 tavua. (A=kyllä; B=ei)
22. **Väite:** komentojen jälkeen annettava komento `cat moi` tulostaisi konsoliin tekstin "kayttojarjestelmat.tex"ja rivinvaihdon. (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. toimii vain moniydinprosessorissa.
B. vaatii prosessorilta vähemmän toimenpiteitä kuin aliohjelmakutsuun siirtyminen (esim. AMD64:n käskyn `call` suorittaminen).
C. voidaan estää sovellusohjelmassa käyttämällä synkronointia.
D. voi aiheutua käyttäjätilassa toimivan prosessin toimenpiteiden johdosta.

Ohje tehtäviin 24–25: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että kukin 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. sisältää laiteohjainten ajurit.
B. määrittää järjestelmälle yhtenäisen lohkokoon.
C. tarvitaan vain silloin, kun tietokoneessa ei ole bittioperaatioita (esim. AND, OR) valmiiksi toteutettuna laitteistotasolla.
D. kääntää bittioperaatiot (esim. AND, OR) yhden laitteen konekielestä toisen laitteen konekielelle.

26. **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 27–30: Yhdistä lauseen loppua vastaava kirjain numeroituun alkuun siten, että kukin lause on totta. Alkuihin on yksikäsitteinen oikea loppu. Jokainen loppu voi sopia useampaan alkuun tai ei yhteenkään.

Lauseiden alut:

27. Keskinäinen poissulku (engl. *mutual exclusion, MutEx*) ...
28. RAID ...
29. Semafori ...
30. Lokaalisuusperiaate ...

Vaihtoehtoiset loput:

A. liittyy tietokoneiden muistihierarkian perusideaan.
B. liittyy tiedon tallennuksen vikasietoisuuteen.
C. liittyy kilpa-ajotilanteiden (engl. *race condition*) ratkomiseen.
D. liittyy käyttäjien automaattiseen tunnistamiseen.

Tutkittava esimerkki tehtäviin 31–32: 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);
}
```

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

33. **Väite:** POSIXin semafori on tietorakenne, joka sisältää etumerkittömän kokonaisluvun ja joukon prosesseja tai säikeitä. (A=kyllä; B=ei)
34. **Väite:** POSIXin semaforikutsun `sem_post(s)` suorituksessa semaforin `s` arvo muuttuu aina. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 35–37: 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) {
    int i;
    pthread_mutex_lock(&mymutex);
    for (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
  }
}
```

35. **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)
36. **Väite:** Esimerkissä on periaatteessa mahdollista tilanne, jossa säikeessä A muuttuja i==123 ja säikeessä B muuttuja i==456 samaan aikaan. (A=kyllä; B=ei)
37. **Väite:** Esimerkin ohjelma voi aiheuttaa lukkiutumistilanteen (engl. *deadlock*), jossa sen suoritus jää jumiin. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 38–40: 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  }
```

38. **Väite:** Rivi 10 tapahtuu fork() -kutsun luomassa lapsiprosessissa. (A=kyllä; B=ei)
39. **Väite:** Rivin 8 suorittaminen lopettaa koko minimalistisen shellin suorituksen. (A=kyllä; B=ei)

40. **Väite:** Rivin 12 suorittaminen lopettaa koko minimalistisen shellin suorituksen. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 41–44: 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. Virtuaalimuistin 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

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

Tutkittava esimerkki tehtäviin 45–47: Kurssin luennoilta tutulla symbolisella konekielellä (GNU assembler; AT&T -syntaksi) kirjoitettu, rivi riviltä kommentoitu ohjelmanpätkä:

```
 _start:
        movq    $2,%rcx   # luku 2 rekisteriin RCX
        movq    $10,%rdi  # luku 10 rekisteriin RDI
 silimu:
        add     %rcx,%rdi # rekisterien RCX ja RDI summa rekisteriin RDI
        dec     %rcx      # vähennä 1 rekisterin RCX arvosta
        jz      ulos      # hyppää, jos edellisen käskyn tulos oli 0
        jmp     silimu    # hyppää aina
 ulos:
        dec     %rdi      # vähennä 1 rekisterin RDI arvosta
```

45. Kuinka monta konekielikäskyä ohjelmanpätkän jälki sisältää, ts. kuinka monta käskyä sen suorittaminen alusta loppuun vaatii?
46. Mikä on rekisterin RCX sisältö ohjelmanpätkän suorituksen lopuksi?
47. Mikä on rekisterin RDI sisältö ohjelmanpätkän suorituksen lopuksi?

Tutkittava esimerkki tehtävään 48: Kaksi C-aliohjelmaa luentoesimerkistä (nimet ja järjestys arvottu tähän). Molemmat palauttavat alkioiden summan taulukosta, jossa on m riviä ja n saraketta.

```
double A(double *taulukko, int m, int n){
  double s=0.;
  for(int j=0;j<n;j++){
    for (int i=0;i<m;i++){
      s += taulukko[i*n+j];
    }
  }
  return s;
}

double B(double *taulukko, int m, int n){
  double s=0.;
  for (int i=0;i<m;i++){
    for(int j=0;j<n;j++){
      s += taulukko[i*n+j];
    }
  }
  return s;
}
```

48. **Väite:** Aliohjelma A() ruuhkauttaa prosessorin välimuistin todennäköisemmin kuin B(). (A=kyllä; B=ei)

Vapaaehtoinen vapaa sana

Halutessasi voit antaa loppuun palautetta tai kommentoida muuten kurssia tai tenttiä. Vastauksen muoto on vapaa, eikä se vaikuta arvosteluun.