ITKA203 – Käyttöjärjestelmät – tentti 20.5.2016

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

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. on yhtä nopea toimenpide kuin aliohjelmakutsuun siirtyminen (esim. AMD64:n käskyn `call` suorittaminen).

Ohje tehtäviin 2–5: 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.

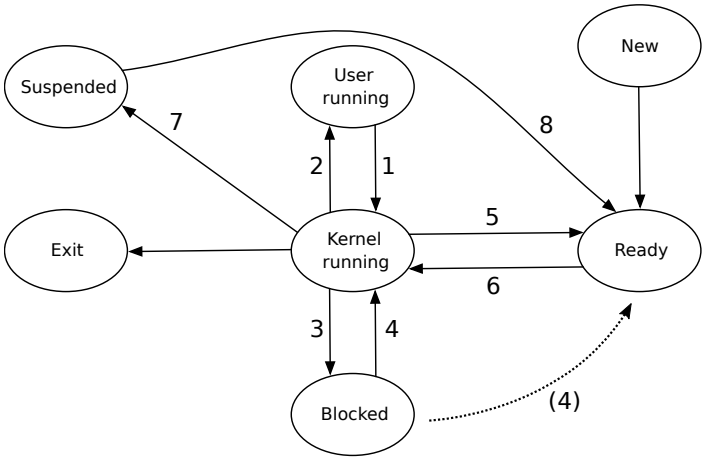
Lauseiden alut:

2. Keskinäinen poissulku (engl. *mutual exclusion*, *MutEx*) …
3. FLIH (*first-level interrupt handling*) …
4. Semafori (*semaphore*) …
5. IP-rekisteri (*instruction pointer*) …

Vaihtoehtoiset loput:

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

Ohje tehtäviin 6–8: 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:

6. Tilasiirtymä nro 1 tapahtuu, ...
7. Tilasiirtymä nro 2 tapahtuu, ...
8. Tilasiirtymä nro 3 tapahtuu, ...

Vaihtoehtoiset loput:

A. kun tapahtuu keskeytys.
B. kun prosessi tekee pyynnön esim. lukeakseen merkkejä tiedostosta, eikä tulos ole telos ole vielä valmiina puskureissa.
C. kun välillä on palveltava muita prosesseja, vaikka nykyisessä on laskeminen kesken.
D. kun tapahtuu paluu käyttöjärjestelmäkutsun tai keskeytyksen käsittelijästä (esim. AMD64:n konekielikäsky *sysret* tai *iret*).

Ohje tehtävään 9: Järjestä seuraavat muistikomponentit niiden kapasiteetin mukaan, eli mihin mahtuu eniten dataa kerralla: A=kovalevy, B=keskusmuisti, C=välimuisti, D=rekisteri.

9. Vastauksessasi on neljä järjestettyä kirjainta: isoin ensin, pienin viimeisenä.

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. Virtuaalimuistin hallinta (engl. *virtual memory management*) ...
11. Käyttäjäkohtainen työpöytä (engl. *desktop manager*) ...
12. IPC (*inter-process communication*) ...
13. Vuorontaja (engl. *scheduler*) ...

Vaihtoehtoiset loput:

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

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

15. **Väite:** Nykyaikaisen käyttöjärjestelmän yksi päätehtävä on eristää sovellusohjelmat laitteistosta siten, että laitteistoresurssien ohjaus kulkee aina yksinomaan ns. käyttöjärjestelmäkutsurajapinnan (*system call interface*) kautta. (A=kyllä; B=ei)

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

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

18. **Väite:** Tietokonejärjestelmän tuottavuutta (engl. *throughput*) eli aikayksikössä loppuun saatujen tehtävien määrää voidaan kasvattaa lisäämällä kellokeskeytyksien määrää aikayksikössä. (A=kyllä; B=ei)

Ohje tehtäviin 19–22: 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:

19. Prosessitaulu ...
20. Prosessielementti (PCB) ...
21. Sivutaulu ...
22. Ready-jono ...

Vaihtoehtoiset loput:

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

23. **Väite:** Laiteriippuva I/O-ohjelmiston kerros määrittelee laitteille nimeämiskäytännön. (A=kyllä; B=ei)
24. **Väite:** I/O-operaation pyytäminen voi johtaa pyytävän prosessin siirtämiseen blocked-tilaan. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 25–26: Kurssin luennoilta ja demoista tutussa ympäristössä (Linux,bash) tehty yksittäinen, POSIX-syntaksin mukainen komentorivi:

```
arg arg | grep -i "a b c" | sort -f -r | kissa > cat
```

25. Montako komentoa rivillä on yhteensä?
26. Montako argumenttia rivillä on yhteensä?

Tutkittava esimerkki tehtäviin 27–28: 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]$ echo Heippa >> kayttojarjestelmat.tex
[nieminen@halava tenttikys]$
```

27. **Väite:** komentojen jälkeen tiedoston `kayttojarjestelmat.tex` pituus on pienempi kuin 10740 tavua. (A=kyllä; B=ei)
28. **Väite:** komentojen jälkeen annettava komento `cat moi` tulostaisi konsoliin "Heippa"ja rivinvaihdon. (A=kyllä; B=ei)

29. **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)

30. **Väite:** POSIXin semafori on tietorakenne, joka sisältää etumerkittömän kokonaisluvun ja joukon prosesseja tai säikeitä. (A=kyllä; B=ei)
31. **Väite:** POSIXin semaforikutsun `sem_wait(s)` suorituksessa semaforin `s` arvo ei koskaan pysy samana. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 32–34: 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
  }
}
```

32. **Väite:** Esimerkin ohjelma ratkaisee tuottaja-kuluttaja -ongelman kahdella säikeellä (A=kyllä; B=ei)
33. **Väite:** Esimerkissä on periaatteessa mahdollista tilanne, jossa säikeessä A muuttuja i==123 ja säikeessä B muuttuja i==456 samaan aikaan. (A=kyllä; B=ei)
34. **Väite:** Esimerkin ohjelma voi aiheuttaa lukkiutumistilanteen (engl. *deadlock*), jossa sen suoritus jää jumiin. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 35–36: C-mäisenä pseudokoodina tehty ehdotus rengaspuskuria käyttävän tuottaja-kuluttaja -ongelman ratkaisemiseksi POSIX-tyyppistä semaforipalvelua käyttäen. Yhteistä muistia käytetään vain puskurioperaatioissa.

```
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);
}
```

35. **Väite:** Semaforien alustus on järkevä ongelman ratkaisemiseksi. (A=kyllä; B=ei)
36. **Väite:** Semaforien käyttö säikeissä on järkevä ongelman ratkaisemiseksi. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 37–39: 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  }
```

37. **Väite:** Rivin 5 suorittaminen tapahtuu shellin fork():lla luomassa lapsiprosessissa. (A=kyllä; B=ei)
38. **Väite:** Rivin 7 suorittaminen tapahtuu alkuperäisessä shell-prosessissa, ei siis fork():n luomassa lapsiprosessissa. (A=kyllä; B=ei)
39. **Väite:** Rivin 12 suorittamisen jälkeen suoritus jatkuu riviltä 2, jossa luetaan käyttäjältä uusi komento. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 40–43: 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.

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	1	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	155
0x102	7	0x03	199
0x103	234	0x45	8
0x104	7	0x2e	12
0x105	876	0x45	4
0x106	2	0x2e	8
0x107	7	0x7f	9

40. Mihin fyysiseen muistiosoitteeseen kohdistuisi prosessin 7 tekemä kirjoitus virtuaalimuistiosoitteeseen 0x2e123?
41. Mistä virtuaaliosoitteesta prosessi 2 saisi käyttöönsä tavun, joka sijaitsee kovalevyllä indeksissä 0x22345 heittovaihto-osion / -tiedoston (engl. *swap space*) alusta lukien?
42. Prosessi 2 suorittaa hyppykäskyn aliohjelmaan muistiosoitteessa 0x03456. Tapahtuuko prosessorissa sivuvirhe / sivunvaihtokeskeytys? (A=kyllä; B=ei)
43. Prosessissa 234 aiheutuu sivunvaihtokeskeytys. Korvausalgoritmi on LRU. Onko jonkin sivun sisältö tallennettava levylle? (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 44–46: Kurssin luennoilta tutulla symbolisella konekielellä (GNU assembler; AT&T -syntaksi) kirjoitettu, rivi riviltä kommentoitu ohjelmanpätkä:

```
_start:
       movq    $3,%rcx   # luku 3 rekisteriin RCX
       movq    $4,%rdi   # luku 5 rekisteriin RDI
silimu:
       inc     %rdi      # lisää 1 rekisterin RDI arvoon
       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
```

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

47. **Väite:** Unixin i-solmuihin (engl. *inode*) perustuvassa tiedostojärjestelmässä tiedoston nimi määräytyy hakemistotiedostossa sijaitsevan linkin perusteella. (A=kyllä; B=ei)

Tutkittava esimerkki tehtäviin 48–48: RAID 1 -järjestelmä perustuu tiedon peilaamiseen identtisenä useammalle kovalevylle, mikä parantaa toimintavarmuutta (varma fakta; varsinainen väite seuraa).

48. **Väite:** Tiedon lukeminen RAID 1 -levyjärjestelmästä on väistämättä hitaampaa kuin yksittäistä kovalevyä käyttämällä. (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.