ITKA203 – Käyttöjärjestelmät – tentti 13.5.2017 (MALLITENTTI 2017)

Kevään 2017 kurssin luennot, demot, esimerkkiohjelmat / Paavo Nieminen <paavo.j.nieminen@jyu.fi>

Tentin ja kurssin arvosteluperusteet 2017: Tentissä on 48 puolen pisteen arvoista kysymyskohtaa. Ne arvotaan kysymyspankista tenttikohtaisesti siten, että (1) jokainen kysymys mittaa yhtä tai useampaa kurssille määriteltyä osaamistavoitetta, (2) puolet kysymyksistä liittyvät kurssin ydinainekseen eli välttävään perustietouteen, (3) luennoilla mainitut "perustärpit"edistyneemmistä aiheista ovat mukana joka tentissä ja (4) kaikkien kysymysten yksityiskohdat riippuvat arvonnasta. Joka tentissä on siis osittain erilaiset, mutta mahdollisesti osittain samat kysymykset, arvonnasta riippuen. Kysymyspankkia laajennetaan jatkuvasti aivan uusilla kysymyksillä. Uusienkin kysymysten yleinen formaatti tulee olemaan jokin nykyisistä (esim. ”väittämiä esimerkkikoodin toiminnasta” tai ”yhdistä lauseet”). Kukin kysymys pyrkii vastaamaan binäärisesti kysymykseen ”osaako opiskelija kurssin käytyään todella . . . [+osaamistavoitteen teksti]”.

Kysymykset pyrkivät olemaan soveltavia, joten niissä täytyy olla tarkkaavainen ja miettiä asiat läpi. Koodiesimerkit on luettava läpi ja ymmärrettävä, väittämät eivät ole suoria lainauksia materiaalista, jne. Väittämissä ja monivalinnoissa on suunnitelmallisia ”ansoja”, joihin on tarkoituskin harhauttaa vastaamaan väärin ne, jotka eivät ole kurssin asioita omaksuneet. Varo siis vaaraa. Tarkoitus on kuitenkin, että oikeasta vastauksesta ei ole mitään epäselvää silloin, kun *tiedot löytyvät* aivojen sopukoista ja *niitä osataan soveltaa*.

Mikäli tehtävässä ei mainita poikkeuksista, jokainen oikea vastaus tuo 0.5 pistettä. Kyllä/ei -väittämissä sekä muissa *fifty-fifty -kysymyksissä väärä vastaus tuo miinuspisteitä -0.375 pistettä*. Tällä tavoin odotusarvoksi arvaamalla tulee asymptoottisesti 25% maksimipisteistä, aivan kuin normaalissa monivalinnassa, jossa joka neljäs arvaus neljän vaihtoehdon välillä menee säkällä oikein. Tyhjä vastaus tarkoittaa 0.0 pistettä. Monipuolisemmissa kysymyksissä vääriä vastauskombinaatioita on huomattavasti enemmän kuin 50%, joten niissä ei sakkopisteitä tarvita, koska odotusarvo pisteille on arvaamalla joka tapauksessa lähempänä nollaa kuin edes 25% maksimista.

Arvosana läpäisyrajalla eli 12 pisteellä on 1 ja täysillä pisteillä 5. Pistealue 12-24 jaetaan tasaväleihin neljällä väliarvolla, jotka pyöristetään lähimpään 0.5 pisteen granulariteetilla ilmaistavaan pistemäärään:

Pisteväli	Välin pituus	Arvosana
[0.0, 12.0)	12p	hylätty
[12.0, 14.5)	2.5p	1
[14.5, 16.5)	2.0p	2
[16.5, 19.0)	2.5p	3
[19.0, 21.5)	2.5p	4
[21.5, 24.0]	2.5p	5

Jos pistemäärä osuu täsmälleen rajalle, luetaan arvosana rajan paremmalta puolelta, esim. 19.0 pistettä riittää arvosanaan 4.

Kurssin läpäisy arvosanalla 1 edellyttää **tentistä** vähintään 12 pistettä. Mahdolliset bonuspisteet lasketaan vasta läpäistyn tentin jälkeen. **Bonukset lasketaan mukaan vain kevään 2017 tenttiyrityksiin, ja vain tentin arvostelun yhteydessä**. Arvosanaa ei siis voi hinkuttaa ylöspäin jälkikäteen bonustehtävien avulla. Läpäistyn tentin saa kylläkin halutessaan uusia arvosanan korottamiseksi.

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:** 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)

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

Tutkittava esimerkki tehtäviin 3–4: Kurssin luennoilta ja demoista tutussa ympäristössä (Linux,bash) tehty yksittäinen, POSIX-syntaksin mukainen komentorivi:

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

3. Montako komentoa rivillä on yhteensä?
4. Montako argumenttia rivillä on yhteensä?

Ohje tehtäviin 5–8: 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.

Lauseiden alut:

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

Vaihtoehtoiset loput:

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

Ohje tehtävään 9: 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:

9. Prosessorin keskeytys …

Vaihtoehtoiset loput (mahdollisesti useita sopivia):

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

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 yhteen-

kään.

Lauseiden alut:

10. Käyttöjärjestelmän virtuaalimuistin hallintaosio (engl. *virtual memory management*) …
11. Käyttäjäkohtainen työpöytä (engl. *desktop manager*) …
12. IPC (*inter-process communication*) …
13. Vuorontaja (engl. *scheduler*) …

Vaihtoehtoiset loput:

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

14. **Väite:** Prosessin kaikilla säikeillä on aina sama sisältö jokaisessa prosessorin rekisterissä (esim. AMD64:n RIP:ssä ja RAX:ssä) (A=kyllä; B=ei)

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

17. **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 18: 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.

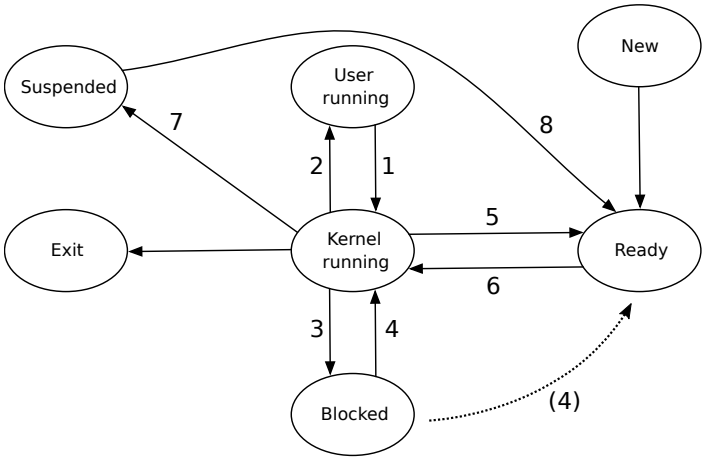
Lauseen alku:

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

Vaihtoehtoiset loput (mahdollisesti useita sopivia):

A. shell-komento *echo* tulostaa argumenttinsa.
B. käyttäjien kotihakemistot tulee sijoittaa hakemistoon nimeltä */nashomeN/*, missä N on kokonaisluku
C. järjestelmäkutsu exit() tapahtuu sijoittamalla rekisteriin RAX luku 60 ja suorittamalla syscall-konekielikäsky.
D. Ikkunan oikeassa yläkulmassa tulee olla rastin muotoinen symboli, jolla ohjelman voi lopettaa.

Ohje tehtäviin 19–21: 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:

19. Tilasiirtymä nro 1 tapahtuu, ...
20. Tilasiirtymä nro 5 tapahtuu, ...
21. 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 prosessori siirtyy keskeytyksen tai käyttöjärjestelmäkutsun käsittelyyn.
C. kun prosessille on lähetetty odottelusignaali esim. näppäilemällä pääteyhteydessä Ctrl-Z.
D. kun prosessi aloittaa käyttöjärjestelmän vuorontajalta saamansa aikaikkunan käytön.

Ohje tehtäviin 22–25: 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:

22. Säie (engl. *thread*) ...
23. Kehystaulu (engl. *frame table*) ...
24. Deskriptoritaulu (engl. *descriptor table*) ...
25. Ajurirajapinta (engl. *driver interface*) ...

Vaihtoehtoiset loput:

A. liittyy I/O -laitteiden ohjaukseen.
B. liittyy vuoronnukseen.
C. liittyy muistinhallintaan.
D. liittyy tiedostonhallintaan.

Ohje tehtävään 26: Järjestä seuraavat muistikomponentit niiden nopeuden mukaan: A=kovalevy, B=keskusmuisti, C=rekisteri, D=välimuisti.

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

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. heittovaihdolle (engl. *swap*). (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_post(s)` suorituksessa semaforin `s` arvo muuttuu aina. (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) {
    int i;
    for (i = 1; i <= N; i++){
        pthread_mutex_lock(&mymutex);
        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
  }
}
```

32. **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)
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 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( PUSKURIN_KOKO );
  FULL=tee_semafori( 0 );
  MUTEX=tee_semafori( 1 );
  kaynnista_saie(tuottaja);
  kaynnista_saie(kuluttaja);
}
```

35. **Väite:** Semaforien alustus on oikeellinen ongelman ratkaisemiseksi. (A=kyllä; B=ei)
36. **Väite:** Semaforien käyttö säikeissä on oikeellinen 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:** Rivi 5 tapahtuu fork() -kutsun luomassa lapsiprosessissa. (A=kyllä; B=ei)
38. **Väite:** Rivin 4 suorituksen aikana on aina olemassa sekä shell että sen luoma lapsiprosessi. (A=kyllä; B=ei)
39. **Väite:** Rivin 12 suorittaminen lopettaa koko minimalistisen shellin suorituksen. (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 ilmoitta-

vat 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

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

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?

Tutkittava esimerkki tehtäviin 47–48: 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
```

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

Vapaaehtoinen vapaa sana

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