

# ITKA203 -- Käyttöjärjestelmät -- Tentti 21.5.2014

Kevään 2014 kurssi (luennot, demot 1-4 ja oheismateriaali) / Paavo Nieminen <[paavo.j.nieminen@jyu.fi](mailto:paavo.j.nieminen@jyu.fi)>.

HUOM: Tarkastus jaetaan kolmen opettajan kesken. Palauta siksi **tasan kolme erillistä vastauspaperia** (ei siis yhtä eikä kahta), joissa jokaisessa on vähintään **nimesi, syntymäaikasi, kurssin tiedot** (koodi ja/tai nimi) sekä **vähintään tyhjä vastaus jokaista tehtävännumeroa kohden**. Jako on seuraava: Tehtävät 1, 2 ja 3 yhteen vastauspaperiin. Tehtävät 4 ja 5 yhteen paperiin. Tehtävä 6 yhteen paperiin.

## Kysymys 1/6 (vastauspaperille 1):

Selitä, mitä seuraavat termit tarkoittavat tietokonelaitteiston ja käyttöjärjestelmän yhteydessä:

- a) osoiteavaruus
- b) ohjelmalaskuri
- c) vuoronnus
- d) pre-emptiivisyys
- e) signaali

(1+1+1+1+1=5p)

## Kysymys 2/6 (vastauspaperille 1):

- a) Listaa tietokoneessa käytettävät erityyppiset muistit (ns. muistihierarkia) nopeasta hitaaseen.
- b) Miksi tarvitaan näin monia erilaisia muisteja eikä riitä vain yhdenlaiset?

(1+1=2p)

## Kysymys 3/6: (vastauspaperille 1)

Voit käsitellä luennoilla tai materiaalissa esitettyssä laajuudessa, oppikirjan esittelemänä konkreettisina esimerkkeinä, tai siinä laajuudessa kuin käsittääksesi on välttämätöntä järkevä moniajon toteutumiseksi:

- a) Missä eri tiloissa prosessi voi olla?
- b) Milloin tapahtuu mikäkin siirtymä tilojen välillä?

(1+2 = 3p)

**Kysymys 4/6 (vastauspaperille 2):**

- a) Selitä poissulkuongelma: millaisissa tilanteissa se voi tulla eteen, ja mitä se esimerkiksi voi käsittelemättömänä aiheuttaa.
- b) Esitä C-ohjelma tai pseudokoodi, jossa poissulku tehdään käyttäen käyttöjärjestelmän semaforipalveluita.
- c) Anna esimerkki tilanteesta, jossa voi tapahtua lukkiutuminen eli deadlock -tilanne.

(3p)

**Kysymys 5/6 (vastauspaperille 2):**

Merkin lukeminen päätteeltä on esimerkki I/O -operaatiosta. Kerro, miten luku tapahtuu: mitkä kaikki ohjelmisto- ja laitteisto-osat osallistuvat sen toteuttamiseen missäkin vaiheessa, ja mitä niiden kunkin vastuulla on? Aloita kertomus hetkestä, jolloin käyttäjän ohjelma tekee operaatiopyynnön, ja lopeta siihen, kun operaatio on valmis ja I/O:ta pyytänyt käyttäjän ohjelma jatkaa taas seuraavasta käskystä. (Tyyli vapaa - kaaviot, pseudokoodit, essee käyvät).

(3p)

**Kysymys 6/6 (vastauspaperille 3):**

- a) Mitä tekee tyypillinen unix-apuohjelma `grep`?
- b) Mitä tekee tyypillinen unix-apuohjelma `kill`?
- c) Mitä seuraava skripti tekee (hopsukat ovat "backtick"-merkkejä):

```
#!/bin/sh
echo '$0'
```

- d) Mikä on lopputulema edellisen kohdan skriptin ajamisesta?
- e) Seuraavan skriptin haluttaisiin toimivan kommenttien mukaisesti, mutta kaikkea ei ole vielä toteutettu (TODO-merkinnät) ja lisäksi tekijälle on tullut joitakin karkeita virheitä tähänastisessa. Kirjoita skripti uudelleen niin, että se toimii kommentteissa ilmoitetuin tavoin. (Kommentteja ei tarvitse toistaa vastauksessa vaan ainoastaan skripti niin että se toimii virheettömästi.)

Puolivalmis skripti nimeltä `skripti.sh`:

```
#!/bin/sh
# Prosessinhallinnan tilanteen tallentava skripti. Käyttöesimerkki:
#   skripti.sh logitiedot.txt

# Em. komento tallentaa 'logitiedot.txt' -nimiseen tiedostoon
# kellonajan ja suorituksessa olevat prosessit.
# Tiedosto ei saa olla ennestään olemassa.

# TODO: Pitäisi varmistaa, että käyttäjä on antanut tasan yhden
# argumentin ja lopettaa tähän, jos näin ei ole..

fname=$1

# TODO: Pitäisi tarkistaa, että tiedostoa ei vielä ole
# olemassa; jos on jo olemassa niin pitäisi lopettaa kesken
# (ettei vahingossa ylikirjoiteta käyttäjän aiempaa dataa)

# Laitetaan tiedoston ensimmäiselle riville otsikoksi
# tämänhetkinen kellonaika ja suorittajan käyttäjätunnus:

echo Käyttäjä whoami ajoi tämän hetkellä date > fname

# TODO: Pitäisi tarkistaa, että tiedosto on nyt syntynyt
# ja siihen voi kirjoittaa. Muuten pitäisi
# antaa käyttäjälle virheilmoitus ja lopettaa kesken.

# Jatketaan tiedostoa kirjoittamalla otsikon perään kaikki
# tällä hetkellä järjestelmässä suorituksessa olevat ohjelmat:
ps -ef > $fname
```

(1+1+1+1+4 = 8p)

## Liite: Yhden sivun sh-luntti

Joitakin ohjelmointirakenteita:

```
muuttuja=57          # tarkka syntaksista: ei välilyöntejä!
muuttuja="eki"; muuttuja="$muuttuja$muuttuja"
read muuttuja       # lukee muuttujaan syöttövirrasta

if EHTO              # Myös: if EHTO; then ...; \
then                 #     elif EHTO; then ...; fi
...                  #
fi                   #

for muuttuja in LISTA # muttujan "esittelyssä"/asetuksessa ei $
do
... jotakin $muuttuja jotakin ... # muuttujan käytössä $muuttuja
done

while EHTO; do ... ; ... ; done # käskyerotin rivinvaihto tai ;

# Tee jotain syötteen kaikille riveille:
while read rivi; do echo "luin: $rivi" ; done < rivit.txt

case $hanen_nimensa in
  kirsimarja)
    echo "Hei Kippe" ;;
  eskomatias)
    echo "Moi E.M." ;;
esac
```

Aliohjelmat mahdollisia&hyödyllisiä, mutta ei käsitellä ITKA203:lla.

Joidenkin ehtojen käyttöä (Välilyönnit merkityksellisiä!):

```
[[ -d TIEDOSTONIMI ]] # tosi, jos on hakemisto
[[ -f TIEDOSTONIMI ]] # tosi, jos on tavallinen tiedosto
[[ -w TIEDOSTONIMI ]] # tosi, jos on olemassa ja kirjoitettavissa
[[ ! -f TIEDOSTONIMI ]] # tosi, jos ei olemassa tai ei tavallinen
[[ "$muuttuja" -le "57" ]] # tosi, jos muuttuja <= 57 (myös lt,gt,ge)
```

Argumenttien käyttö: (yli 9 arg mahd., mutta ei käsitellä ITKA203:lla)

```
echo "Tämän skriptin nimi on $0. Eka argumentti $1, neljäs $4"
echo "Argumenttien määrä on $#"
```

Joitakin sisäänrakennettuja toimintoja:

```
exit VIRHEKOODI      # koodi 0 tarkoittaa onnistumista
cd                   # vaihtaa skriptin työhakemistoa
echo                 # kaiuttaa tekstiä
let muuttuja=$muuttuja+3 # (perusaritmetiikkaa)
```

Erikoismerkkejä (tuttuja interaktiivisesta käytöstä; toimii skripteissä):

```
< > >> | 'KOMENTO'
```