ITK145 Käyttöjärjestelmät

Tenttikysymyksiä ja vastauksia

Aliohjelman suoritusperiaate, ts. selvitä pinon käyttö ja paluuarvon välittäminen (3p)

1. Ennen aliohjelman aloittamista kutsuja tallettaa pinoon parametrien arvot.
2. Aliohjelmaan siirtyminen tapahtuu call-käskyllä.
    a. IP rekisterin arvoksi aliohjelman ensimmäisen käskyn osoite
    b. Pinon huipulle paluuosoite
3. Ollaan aliohjelmassa. aliohjelma laskee paluuarvonsa rekisteriin ax
4. Poistuttaessa aliohjelmasta ret-käskyllä, on pinon huipulla paluuosoite joka laitetaan IP:n arvoksi ja poistetaan pinosta
5. Suoritus jatkuu call-käskyä seuraavasta käskystä, joka yleensä on add-käsky, tehtävänään lisätä pino-osoitetta parametrien tavumäärällä jolloin pino palaa tilaan jossa se oli ennen parametrien pinoamista. (Huom. pino ”kasvaa” negatiivisilla arvoilla -> käskyllä add ”tyhjennetään” pino ja palataan nollatilaan)

Selvitä keskeytyskäskyn perusmekanismi, ts. keskeytyksenhoitoon siirtyminen ja sieltä paluu (3p)

1. Signaali prosessorin toimenpiteitä edellyttävästä tapahtumasta tulee prosessorin keskeytysnastaan (intr)
2. Tarkistetaan onko keskeytykset kielletty. (Jos on, niin keskeytystä ei huomioida)
3. Laitetaan rekisterin arvot Kernel-pinoon (ainakin IP ja PSW- eli flags-rekisteri), jota keskeytyskäsittelijä käyttää suorituspinonaan
4. Siirrytään keskeytyskäskyn laitenumeron perusteella hyppykäskyyn jonka suoritus ohjaa toiminnan oikeaan keskeytyskäsittelijään.
5. Keskeytyskäsittelijän lopetettua toimintansa palataan takaisin keskeytettyyn ohjelmaan iret-erikoiskäskyllä. (Palauttaa pinoon laitetut rekisterin arvot)

Sivuttavan virtuaalimuistin periaate ja merkitys. Selvitä myös millaisia tauluja virtuaalimuisti edellyttää ja miten se niitä käyttää (4p)

Koska keskusmuisti on nopeampi kuin oheismuisti, olisi pyrittävä käyttämään sitä. Kapasiteetti ei kuitenkaan normaalisti mahdollista sitä. Virtuaalimuisti siirtelee prosesseja tai niiden osia keskusmuistin ja levymuistin välillä.

Sivutettu virtuaalimuisti jakautuu kahteen osaan. Toinen keskusmuistissa ja toinen levyllä. Ne on jaettu samankokoisiin osiin.

Keskusmuistiosoitteen bitit ilmoittavat monesko sivu on kyseessä ja monesko sivun muistipaikka on kyseessä.

Oheismuistissa ohjelmat ja tietoalueet ovat täydellisinä. Keskusmuistissa pyritään pitämään vain tarvittavia osia.

Kun prosessi tarvitsee ohjelman osaa joka ei ole keskusmuistissa, tapahtuu sivuviittausvirhe ja sen seurauksena sivunvaihto. Haetaan keskusmuistista sivu jota ei ole hetkeen tarvittu. Varmistetaan että sen muutokset on päivitetty levylle ja kirjoitetaan tarvittava ohjelman osa sen paikalle keskusmuistiin.

Prosessit viittaavat muistiin ”virtuaaliosoitteilla”. Keskusmuistin sivujen varausta hoidetaan kehystaululla, joka sisältää sivua kohti seuraavat tiedot

- Minkä prosessin käytössä sivu on ja mitä prosessin sivua se vastaa.
- Kirjoitusbitti kertoo onko sivu talletettava levylle sivunvaihdon yhteydessä
- Käyttölaskuri kertoo sivun käyttöajan
- Mahdollista suojaustietoa

Jokaista prosessia kohti on myös sivutaulu:

- Muistibitti ilmoittaa onko sivu keskusmuistissa

- Fyysinen sivun numero, jos se on keskusmuistissa
- Paikka levyllä

Mikä on prosessitaulu ja mitä se sisältää? (2p)

Prosessitaulu on käyttöjärjestelmän tärkein tietorakenne. Se kertoo tiedot kaikkien prosessien tilanteesta.

Taulussa jokaisella prosessilla on oma PCB (program control block), joka sisältää:

1. rekisterien arvot
2. prosessin tila (running/ready/blocked)
3. sanomanvälitysjono
4. signaalijono
5. resurssien hallinta (avoimet tiedostot, suojaus, muistialueet)
6. prosessin omistajan tunnus
7. vanhempiprosessin tunnus

Semafori ja sen käyttö poissulkemisongelman ratkaisussa (4p)

Poissulkemisongelma: Kahden prosessin pääsy päivittämään yhteistä tietorakennetta kriittisellä alueella samanaikaisesti.

Semafori on periaate joka pitää huolen, että vain yksi prosessi kerrallaan toimii kriittisellä alueella ja muut odottavat.

Semaforilla on kaksi osaa

1. Arvo (val). Sen perusteella tiedetään voiko prosessi edetä vai täytyykö sen odottaa pääsyä kriittiselle alueelle
2. Semaforijono (queue). Prosessit jotka ovat pyrkineet kriittiselle alueelle, mutta eivät ole päässeet.

Semaforiin liittyy kaksi käyttöjärjestelmäfunktiota

1. Wait. Prosessi pyrkii wait-kutsulla kriittiselle alueelle. Jos arvo (val) = 1, se pääsee sinne ja arvo muutetaan nollaksi. Muuten se laitetaan semaforijonoon (blocked-tilaan)
2. Signal. Jos semaforijono on tyhjä, muutetaan arvo (val) ykköseksi. Muuten siirretään semaforijonon ensimmäinen prosessi ready-jonoon.

Viestinvälitys send- ja receive-kutsuilla (2p)

Viestin lähettäminen: send(destination,message). Eli viesti sekä kohde. Kohde voi olla prosessitunnus tai se voi sijaita vaikka toisessa koneessa.

Vastaanottaminen: receive(source,message).

1. Prosessit antavat send-kutsuja palvelijalle ja Micro-kernel asettaa viestejä jonoon (blocked-tilaan).
2. Kun palvelija antaa receive-kutsun, vapauttaa Micro-kernel viestin palvelijalle.
3. Toteutettueen palvelun palvelija lähettää send-kutsun asiakasprosessille (kuittauksena toteutetusta palvelusta) jolloin micro-kernel siirtää käyttäjäprosessin ready-jonoon.
4. Mikäli viestijono on tyhjä, siirtyy palvelija blocked-tilaan josta sen siirtää ready-jonoon micro-kernel saatuaan uuden send-kutsun.

UNIX-tiedostojärjestelmän toteutus. Selvitä myös miten suoritetaan tiedoston haku nimen perusteella. (2p)

Tiedostojärjestelmässä ylläpidetään taulukkoa i-solmuista.

Jokaisella tiedostolla on i-numero (indeksi i-solmutaulukkoon)

Jokainen i-solmu sisältää seuraavat tiedostokohtaiset tiedot:

1. mode-kenttä (tavallinen/hakemisto/linkki/erikois)
2. suojaus
3. omistajatunnukset (UID = user identifier ja GID = group ID
4. aikaleimat
5. koko
6. käytettävissä olevien lohkojen määrä
7. osoittimet levylohkoihin

Hakemisto-tiedoston avulla löydetään muita tiedostoja. Se on taulukko jolla on kaksikenttäisiä rivejä (tiedoston nimi ja i-numero)

Kun haetaan tiedostoa X, etsitään aluksi nykyisestä hakemistosta riviä jonka tiedostonimenä on X. Kyseisen rivin toisesta sarakkeesta löytyy i-numero jonka avulla löydetään i-solmu ja osoitin oikeaan levylohkoon.

Mitä seuraava komentorivi tekee (1p)

*ls *.r?? > a.r*

ls – tulostaa hakemiston tiedoston luettelon

* - mikä tahansa merkkijono

? - mikä tahansa merkki

Komentorivi tulostaa tiedostoon a.r kaikkien niiden tiedostojen nimet, joiden määrite-osa alkaa r-kirjaimella ja koostuu kolmesta merkistä.

Tee komentotiedosto(rakenne), joka testaa onko parametrin (argumentin) nimeämä hakemisto olemassa (argumenttitaulukko on argv) ja ilmoittaa löytymisestä. Ilmoituksena riittää pelkkä kyllä/ei eri tapauksissa. Jos komennon nimi on loytyyko, niin sen kutsu voisi olla

loytyyko hak

Ja tarkoituksena olisi tutkia hak-nimistä hakemistoa (1p)

```
#!/bin/csh
if (-e –d $argv[1]) then
    echo ‘kyllä’
else
    echo ‘ei’
endif
```

tämä rivi aina komentotied. alussa

-e testaa onko olemassa

-d testaa onko hakemisto

argv viittaa tuotuun parametriin (hak)

Yleistä ylempi komentotiedosto useamman parametrin tapaukseen. Tällöin kutsu voisi olla vaikkapa loytyyko hak1 hak2, ja tarkoituksena olisi testata olemassaolo hakemistoille hak1 ja hak2 (argumenttien määrä on muuttujassa argc. (2p)

```
#!/bin/csh
$apu = $argc  // $argc on parametrien lukumäärä
while ($apu != 0)
  if (-e –d $argv[$apu]) then
    echo “kyllä”
  else
    echo “ei”
  endif
  $apu = $apu – 1
end
```

```
while ($argc != 0)
     if (-e –d $argv[$argc]) then
          echo ‘kyllä’
     else
          echo ‘ei’
     endif
     $argc = $argc-1
end
```

Prosessien tilat. Milloin prosessi on missäkin tilassa ja milloin tilat voivat muuttua? (2p)

Prosessilla on kolme eri perustilaa, running, ready ja blocked

1. Running tilassa prosessia suoritetaan
2. Jos suoritus ei ulkoisen tekijän, esim lukukäskyn, takia voi jatkua, se asetetaan blocked-tilaan.
3. Kun ulkoinen operaatio on suoritettu tai blocked kumottu, voidaan prosessi siirtää ready-tilaan
4. Kun running-tilassa oleva prosessi on kuluttanut aika-annoksensa loppuun se siirretään ready-tilaan ja uusi prosessi otetaan suoritukseen ready-jonosta.

Selvitä path ja cwd muuttujien merkitys csh-kuoressa

Path-muuttuja määrittelee hakemistot joista komentoja haetaan.

CWD ilmoittaa oletushakemiston

Semaforin käyttö synkronoinnissa? (2p)

Synkronointiongelma: Kun kaksi prosessia eivät toimi oikea-aikaisesti toisiinsa nähden. (ns. kuluttaja-tuottaja ongelma)

Synkronointiongelman ratkaisemisessa semaforin arvo (val) voi saada muitakin arvoja kuin 0 tai 1.

Binäärinen semafori ”MUTEX” saa arvoja 0 tai 1. Se kertoo onko puskurialue vapaa.

Semafori ”FREE” kertoo vapaiden puskuripaikkojen määrän. Tuottajaprosessi tuottaa puskuriin kokonaisuuksia odottamaan kuluttajaprosessia. Jos FREE-semaforin arvo on 0, tuottajaprosessi nukahtaa FREE-semaforin jonoon. Tämä tapahtuu FREE:n wait-kutsulla.

Semafori ”FULL” kertoo täysien puskuripaikkojen määrän. Kuluttajaprosessi katsoo, onko FULL-semaforin arvo > 0 eli onko puskurissa kokonaisuuksia. Jos ei ole, kuluttajaprosessi nukahtaa FULL-semaforin jonoon.

Puskurialueen muuttamisen jälkeen testataan Signal-kutsulla, onko kyseiselle alueelle jonottamassa nukkuvia prosesseja.