Sovellusohjelmointi assignment 2 oppimispäiväkirja

Elias Kemppi

20.11.2021

Tänään aloitin projektin suunnittelun. Katselin tehtävien aiheita, ja haluaisin tehdä tehtävän ‘data transfer’ -aiheesta. Tein edellisen tehtävän myös tästä aiheesta, mikä varmasti helpottaisi tehtävän tekoa. Katselin hieman luentovideoita aiheista ‘I/O Multiplexing’ ja ‘Inter-Process Communication’. Näissä puhuttiin esimerkiksi fifo-putkista, joita tässä tehtävässä on tarkoitus käyttää.

21.11.2021

Tänään katsoin lisää luentovideoita Inter-Process Communcationista ja hieman YouTube videolta löytyviä koodiesimerkkejä. Tein myös oman yksinkertaisen testiohjelman, jossa kokeilin fifojen käyttöä. En vielä tässä vaiheessa tiedä, miten tehtävänanto olisi tarkoitus toteuttaa. Jos teen useamman fifo-putken, en tiedä, miten daemon-ohjelma osaa lukea jokaista. Jos taas teen yhden fifon, en tiedä miten saan muutettua sen kirjoitusoikeuksia jokaiselle kirjoittajalle eri aikaan.

22.11.2021

Mietin vielä tänäänkin, miten saisin ‘Data transfer’ aiheen suoritettua tehtävänannon mukaisesti, mutta parin tunnin pähkäilyn jälkeen, aloin miettiä, olisiko jokin muu aihe kuitenkin helpompi ja parempi. Heti aluksi helpoimmalta vaikutti ‘Printout capturing’.

Löysin stdout-virran kaappaavan koodin suoraan netistä. Sitä hieman muutamalla sain sen toimimaan suoraan tehtävässä halutulla tavalla. Tämä toteutus kuitenkin käyttää FILE:ja ja virtoja. Haluaisin siis tehdä sen enemmän kurssin rajapintoja käyttäen, mutta en vielä ole varma miten se onnistuu, joten teen sen nyt näin. Sunnitelmanani olisi, että tekemässäni kirjastossa olisi yksi funktio, joka aloittaa kaappauksen, ja yksi funktio joka lopettaa kaappauksen. Kaikki printf-funktiot näiden välissä kaapattaisiin.

23.11.2021

Tein tänään vain pienen testiohjelman, jossa testasin, että tekemäni kaappauskirjasto todella toimii haluamallani tavalla. Minun oma capture-funktio ottaa parametriksi tidoston nimen ja tällä hetkellä kirjoittaa file-virtaa ‘stdout’ suoraan tiedostoon. Tein myös funktion end_capture, joka lopettaa capturen. End_capturen toteutukseni ei varmaan ole kovin hyvää koodia, sillä ainoa keino, jolla saan capturen lopetettua on, että avaamalla tiedoston ensin uudestaan.

Testiohjelmassani tein forkilla kaksi prosessia, joista toisen output kaapataan parametrina annettuun tiedostoon ja toisen ei. Tämä toimi odotetulla tavalla. Voin myös esimerkiksi aloittaa kaappauksen ennen forkia, jolloin kaikki ouputit kaapataan. Voin myös lopettaa kaappauksen missä tahansa kohdassa ja sen jälkeen outputit tulostuvat normaalisti. Toteutuksessani funktio on helppo lisätä ohjelmiin. Vain yksi helppo funktiokutsu aloittaa kaappauksen tai lopettaa kaappauksen.

25.11.2021

Löysin nyt tavan, kaapata stdout ilman file-viroja. Käyttämällä dup2-funktiota voin kopioida tiedoston file-descriptorin stdoutin default file-descriptorin päälle. Ainoa ongelma tässä totutuksessa on, että en tiedä miten lopetan kaappauksen. En siis tiedä miten saisin kopioitua stdoutin file desciptorin takaisin sen alkuperäiselle paikalle. Tätä ominaisuutta ei taidettu pyytää tehtävänannossa, joten jätän sen nyt ainakin toistaiseksi tekemättä.

Seuraavaksi sunnittelen minkälainen lopullinen testiohjelmani tulee olemaan. Siinä täytyy käyttää ainakin säikeitä, ja prosessien välistä kommunikointia. Voisin tässä kohdassa hyödyntää fifoja, koska niiden käyttö minulle tuli jo opeteltua.

26.11.2021

Olen nyt suunnitellut minkälaisen ohjelman teen. Tarkoitus on, että ohjelmalle voi antaa parametriksi pienen numeron ja tiedoston nimen. Sitten se tekee niin monta säiettä, joista jokainen palauttaa satunnaisen numeron välillä 0-9. Nämä numerot lähetetään fifo-putkea pitkin toiseen prosessiin, joka tulostaa numerot yhdelle riville. Nämä tulostukset kaapataan annettuun tiedostoon kirjoittamallani capture-funktiolla.

Koodin kirjoittaminen sujui säikeitten lisäämiseen asti helposti, sillä olin opetellut fifojen käyttöä. Myös prosessit ja tiedostojen lukeminen ja kirjoittaminen olivat tulleet edellisestä tehtävästä tutuksi. Katsoin luentovideoita ja YouTube videoita säikeistä, jonka jälkeen nekin olivat helppo lisätä. Sain siis periaatteessa koko tehtävän tehtyä ilman, että jäin mihinkään kohtaan jumiin. Kaiken tämän koodin lisäämisen jälkeen toiminnallisuus alkaa olla halutunlainen. Nyt minun pitää vielä lisätä mutex-synkronisointi, jolloin kaikki tehtäväkohtaiset vaatimukset tulevat täyteen. Ohjelman toiminnallisuuden kannalta synkronisoinnilla ei ole väliä, joten ehkä vielä muutan ohjelmaa jotenkin, jolloin synkronisointi näkyy paremmin.

29.11.2021

Muutin ohjelmaa siten, että nyt se tulostaa parametrinä annetun numeron verran satunnaisia numeroita niin, että ensimmäinen numero on välillä 0-9, toinen välillä 10-19 jne. Näin voin testata mutexin toimimista. Lisäsin lyhyen sleepin, jonka takia ilman mutexia kaikki numerot olisivat välillä 0-9, mutta mutex-synkronisoinnin ansiosta numerot tulostuvat oikein.

Mutexien lisääminen onnistui helposti mallikoodien avulla. Koodin muuttamisen takia huomasin kuitenkin, että numerot tulostuvat vähän väärässä järjestyksessä. Jätän koodin nyt kuitenkin tällaiseksi, sillä se kuitenkin toimii mielestäni hyvin.

Tämä tehtävä oli mielestäni aika hyvä. Tehtävänannossa on painotettu aika paljon varsinaista capture ominaisuutta, mikä oikeasti oli vain pieni osa tehtävää ja sen sai tehtyä nopeasti. Säikeitä taas ei ehkä painotettu tarpeeksi, vaikka ne olivat käsittääkseni tämän tehtävän tärkein asia.