Deadlock Empire peli

Henri Laakso, 240062, henri.m.laakso@student.tut.fi

Niko Lappalainen, 253002, niko.lappalainen@student.tut.fi

Antti Tolonen, 247589, antti.tolonen@student.tut.fi

Tutoriaali 1

Ei mitään rinnakkaiste ongelmien estokeinoja. Molemmat säikeet pääsevät käsittelemään kriittistä aluetta samaa aikaan.

Tutoriaali 2

Molemmat säikeet pääsevät käsittelemään muuttujaa a ilman estoja. Kyseessä kilpailutilanne. Tilanteeseen päästään lukemalla vanha a molemmissa säikeissä ennen niiden kasvattamista.

Boolean Flags Are Enough For Everyone

Lukkoina toimivat boolean liput eivät toimi, sillä lipun tarkistus ja sen varaus eivät toimi atomisena operaationa. Molemmat säikeet pääsevät käsittelemään kriittistä aluetta, kun molemmat säikeet ohittavat lipun tarkastuksen ennen lipun asettamista ja kriittisen alueen tarkastelua. Kyseessä jälleen kilpailutilanne.

Simple Counter

Molemmat säikeet pääsevät kriittiselle alueelle, kun oikeanpuoleinen säie käy looppinsa läpi kolmasti ja sen jälkeen vasen kahdesta, jolloin molempien counterin if tarkastelu päästää säikeet käsittelemään kriittistä aluetta samaan aikaan. Kyseessä kilpailutilanne.

Confused Counter

Kilpailutilanne molmpien muuttujien kanssa. Ohjelma epäonnistuu, jos vain first muuttujaa inkrementoidaan samanaikaisesti, mutta second muuttujan inkrementointi tapahtuu molemmissa eri aikaan. Muuttujien inkrementointi ei ole atominen operaatio.

Insufficient Lock

Ohjelma epäonnistuu, jos säikeiden looppeja käydään läpi sopivasti. Esimerkiksi toisen säie käy kerran läpi ja sen jälkeen ensimmäinen säie menee if ehdost läpi kolmannella kierroksella. Kyseessä ei ole suoranaisesti mikään rinnakkaisen ohjelmoinnin ongelma, vaan vain epäpätevä ohjelman rakenne.

Deadlock

Ongelman nimen mukaan kyseessä deadlock tilanne, joka saavutetaan varaamalla ensin mutex1 ensimmäisessä säikeessä ja sen jälkeen mutex2 toisessa säikeessä. Tämän jälkeen molemmat säikeet vaativat toisen säikeen varaaman mutexin eikä kumpikaan pääse jatkamaan eteenpäin.

A More Complex Thread

Ohjelma saatiin lukkoon ensin suorittamalla säiettä 1 kunnes mentiin mutex sisälle ja asetettiin lippu epätodeksi. Nyt säie 0 ei pääse mutexin sisälle, vaan valitaan else haara, jossa asetetaan lippu todeksi. Jatketaan säikeen 1 suoritsta takaisin loopun alkuun. Säie 0 eteenpäin if lauseen sisään lukiten mutexin ja jatketaan aina mutexin vapautukseen saakka. Tämän jälkeen säie 1 etenee mutexin lukitukseen saakka, jolloin havaitaan ohjelman lukkiutuminen. Mutex on lukittu rekursiivisesti kahdesti, muttei sitä vapauteta yhtä monta kertaa, vaan jää mutex säikeen 0 hallintaan. Jos säikeen 0 looppia ajetaan useasti tulee mutexille joka rekursiolla yksi lukituskerta lisää. Säie 1 ei pääse ikinä mutexin sisälle, jos säie 0 on sen jossain välissä aktivoinut.

Manual Reset Event

Eensin säie 0 pääsee odottamaan signaalia. Säie 1 resetoi signaalin, inkrementoi counteria kahdesti ja signaloi toiselle säikeelle. Säie 0 etenee vain yhden askeleen odottamaan if lauseen suorittamista. Säie 1 etenee lopin toiselle kierrokselle, ja suorittaa signaalin reseting sekä vain toisen inkrementoinnin, jonka jälkeen säie 0 pääsee if lauseen sisään ja ohjelma epäonnistuu. Tässä esimerkissä ongelmana oli toimintojen epäatomisuus, jonka johdosta säie 1 inkrementoi counter muuttujaa, säikeen 0 ollessa odotustilasta poissa. Oletuksena if lause olisi pitänyt tutkia heti odotustilasta poitumisen jälkeen, mitä ei tapahtunut.

Countdown Event

Säikeessä 0 otetaan progress muuttuja väliaikaismuuttujaan. Säie 1 kasvattaa progress muuttujaa kolmellakymmenellä. Säie 0 kasvattaa väliaikaismuuttujaansa ja asettaa sen progressin arvoksi. Progress arvo 20 ja säie 0 suorittaa toimintansa loppuun signaloiden kerran. Säikeessä 1 progressin ollessa 20 ei se pääsekään ensimmäisen if lauseen sisälle ja jää signaloiminen tekemättä. Kasvatetaan progressin arvoa viidellekymmenellä, muttei se silti riitä tisen if lauseen sisälle pääsyyn ja toinenkin signalointi jää tekemättä. Ohjelma on lukossa, sillä se ei saanut tarpeeksi montaa signaalia suorituksen aikana. Ongelmana oli muuttujaan lisäysoperaation epäatomisuss ja siitä johtuva kilpailutilanne.

Countdown Event Revisited

Ohjelma odottaa kolmea signaalia. Säie 0 ottaa progress muuttujan väliaikaiseen muuttujaan. Säie 1 suorittaa eventin odotukseen asti. Säie 0 asettaa progress muuttujan arvoon 20 ja jatkaa suoritustaan eteenpäin. Ohjelma on saanut kaikki haluamansa kolme signaalia, mutta if ehto ei täyty kummassakaan säikeessä. Looppi alkaa alusta ja jompikumpi säie signaloi ohjelmalle signaalin, vaikka on se jo vastaanottanut haluamansa 3 signaalia. Event laskuri menee tällöin negatiiviselle, josta ei seuraa hyviä asioita.

Barrier

Säie 0 suorittaa barrierille saapumiseen saakka. Säie 2 suorittaa barrierille saakka ja vapauttaa säikeen 0. Säie 2 jatkaa seuraavalle barrierille saakka. Tulipallojen määrä tällä hetkellä 2. Säie 0 jatkaa suoriustaan, eikä pääse if lauseen sisälle ja tämän jälkeen kuluttaa yhden tulipallon. Säikeen 0 looppi alkaa alusta ja tuottaa se yhden tulipallon ja tämän jälkeen saapuu barrierille. Barrierilla oli jo säie odottamassa, jolloin toinen säie vapauttaa kaikki barrierilla olleet säikeet. Säie 2 suorittaa seuraavan komentonsa ja asettaa tulipallojen määrän nollaan. Säie 0 jatkaa if lauseeseen ja pääsee sen sisään saaden ohjelman epäonnistumaan. Sopiva säikeiden vuoronnus, johtuen barrierin/koodin epäpätevästä toteutuksesta, johtaa ohjelman epäonnistumiseen.

Semaphores

Säie 0 semaforille odottamaan. Säie 1 suorittaa looppinsa kerran läpi vapauttaen semaforin, jolloin säie 0 pääsee jatkamaan kriittiselle alueelle. Tämän jälkeen säie 1 käy looppinsa uudestaan läpi vapauttaen semaforin uudestaan, jolloin säie 1 pääsee myös kriittiselle alueelle samaan aikaan säikeen 0 kanssa. Ongelmana tässä oli semaforin virheellinen vapauttaminen ja yleisesti semaforin virheellinen käyttö.

Producer-Consumer

Säie 1 vapauttaa semaforin, jolloin säie 0 pääsee if lauseen sisälle. Säie 0 luulee queuessa olevan luettavaa, vaikka säie 1 on vasta lisäämässä asiaansa queueen. Kyseessä ongelman nimen mukaan tuottaja kuluttaja ongelma, sekä yleinen semaforin väärinkäyttö, joka mahdollistaa queuesta lukemisen kesken kirjoittamisen.

Producer-Consumer (variant)

Säie 0 käy loopin läpi kerran lisäten yhden golemin queueen. Säie 0 jatkaa toiselle kierrokselle aloittaen uuden golemin lisäämisen queueen. Säie 1 pääsee if lauseen sisään ja aloittaa queuen lukemisen. Säie 0 kirjoittaa samalla, kun säie 1 yrittää lukea queuesta. Tuottaja-kuluttaja ongelma, koska queueta ei lukittu kirjoittamisen ajaksi.

Condition Variables

Säie 0 mutexiin. Säie 0 if lauseen sisään, mutexin vapautus ja plssin odotus. Samat operaatiot säikeelle 1. Säie 2 käy koko looppinsa kerran läpi varaten mutexin lisäten yhden queueen, signaloiden muille säikeille ja lopulta vapauttaen mutexin. Säie 0 saa signaalin ja jatkaa suoritustaan poistaen yhden queuesta ja sitten vapauttaen mutexin. Säie 1 tekee saman, mutta säikeen 1 yrittäessä poistaa yhtä queuesta ei se onnistu, koska queue on jo tyhjä ja ohjelma päätyy määrittelemättömään tilaan. Ongelmana esimerkissä oli queuen tyhjuuden tarkastelu ennen lukkoa, jolloin tarkastelua ei tehty uudestaan queuen tilan muutoksen jälkeen, joka johti tyhjästä queuesta poistamiseen.

Dragonfire

Säie 0 ei pääse if lauseen sisään. Säie 0 ottaa c muuttujan arvon tilapäismuuttujaan. Säie 1 käy looppinsa läpi kertaalleen kasvattaen c muutujan arvoa yhdellä ja lisäten yhden tulipallon. Säie 0 vähentää tilapäis muuttujaansa yhdellä ja asettaa c muutujan arvoksi -1. Säie 0 jatkaa looppinsa alkuun ja nostaen c muuttujan arvoa yhdellä. Nyt säie 1 käy läpi looppinsa kahdesti lisäten kaksi tulipalloa lisää ja kolmannella kerralla c muuttujan arvo on 2, jolloin se menee else haaraan käsittelemään kriittistä aluetta. Nyt tulipaloja on kokonaisuudessaan 3 ja säie 0 pääsee koko if rakenteensa jokaisesta sisään, sillä tulipalloja on tarpeeksi monta varastossa. Säie 0 pääsee käsittelemenään kriittistä aluetta. Molemmat säikeet käsittelevät kriittistä aluetta ja ohjelma päätyy määrittelemättömään tilaan. Tässä esimerkissä ongelmana oli kilpailutilanne muuttujan c suhteen, joka mahdollisti ylimääräisten tulipallojen lisäyksen ja siten säikeen 0 pääsyn kriittiselle alueelle.

Triple Danger

Energybursteja alussa valmiina 3. Säie 1 käy looppinsa läpi kahdesti vähentäen energybursteja kahdella. Energybursteja 1. Säie 1 pääsee if lauseen sisään ja varaa mutexin. Säie 2 pääsee if lauseen sisään ja jää odottamaan mutexin vapautumista. Säie 1 suorittaa looppinsa loppuun ja vähentää yhden energyburstin, jolloin energybursteja ei jää enää jäljelle. Säie 2 pääsee mutexin sisään ja ottaa yhden energyurstin tyhjästä queuesta. Ohjelma päätyy määrittelemättömän tilaan. Tässä esimerkissäongelmana oli if lauseen tarkastelu ennen mutexin varaamista, joka johti tuottaja-kuluttaja ongelmaan, jossa kaksi kuluttajaa pääsi kuluttamaan yhtä resurssia peräkkäin ennen uuden resurssin lisäämistä.

Boss Fight

Säie 1 käy looppinsa läpi kahdesti saaden fortressin arvoksi 2. Säie 0 ottaa darkness muuttujan tilapäismuuttujaan. Säie 1 lisää darkness arvoa yhdellä. Säie 0 lisää tilapäismuuttujan arvoa yhdellä ja asettaa sen darkness arvoksi. Säie 0 lisää yhden evil muuttujaan ja menee if rakenteen sisälle. Säie 1 lisää yhden evil muuttujaan ja menee if lauseen sisälle. Säie 0 pääsee if lauseen sisään vähentäen yhden fortress counterista. Säie 0 varaa sanctum monitorin. Säie 0 vapauttaa sanctum monitorin ja jää odottamaan signaalia. Säie 1 pääsee sanctum monitorin sisään ja signaloi pulssin säikeelle 0. Säie 1 vapauttaa sanctum monitorin ja pääsee käsittelemään kriittistä aluetta. Säie 0 pääsee eteenpäin odotustilastaan saatuaan signaalin ja varaa sanctum monitorin ja pääsee käsittelemään kriittistä aluetta. Molemmat säikeet käsittelevät kriittistä aluetta samanaikaisesti ja ohjelma päätyy määrittelemättömään tilaan. Tässä esmerkissä ongelmina olivat kilapilutilanne darkness muuttujaa inkrementoitaessa ja monitorin epäpätevä käyttö. Säikeen 1 koodi vapauttaa monitorin ennen kriittisen alueen tarkastelua.

Yleisesti koko pelistä

Mielestämme peli oli varsin mielenkiintoinen ja erittäin hyvä idea rinnakkaisen ohjelmoinnin opettamiseksi. Tosin suurin osa tehtävistä käsitteli vain huonosti toteutettua koodia, joka saatiin jumiin vuorontamalla säikeitä sopivasti saaden aikaan kilpailutilanteita suojaamattomille muuttujille, jotka johtivat virhetilanteisiin. Monissa tehtävissä käytettiin mutexeja/monitoreja, mutta niiden implementointi oli täysin väärä. Mielestämme missään tehtävässä ei ilmennyt nälkiintymistä ja deadlock tilanteetkin olivat vain muutamassa hyvin simppelissä deadlock esimerkissä. Parhaana esimerkkinä tehtäväkoodien epäpätevyydestä oli barrier tehtävä, jonka koodissa ei ollut käytännössä mitään järkeä.