CSE 344 Final Rapor

141044039 Doğa UYSAL

Server.c

Server başarılı bir çalıştırma ardından ilk SIGINT için signalHandle işlemini gerçekleştiriyor. Arguman olarak verilen log file açıp fileDescriptor'ünü global değişken olan logFileFD'ye veriyor. Ardından bağlanacak olan client'ler için hazırlanmış olan clientFDs array'ine dinamik olarak yer alıyor ve readFile() fonksyonu ile verilen data dosyasına girip tüm provider'ları satır satır okumaya başlıyor. Okunan her provider, provider_t türünde dinamik array olan providers içinde saklanıyor. Provider_t içerisinde thread ve provider ile ilgili tüm bilgilerin mevcut olmasının yanında, handlerThread ve providerThread arasında iletişim/senkronizasyonu sağlayacak tüm mutex ve condition variable'lar bulunmaktadır. Tüm provider'lar okunduktan sonra her provider için bir providerThread yaratılıp provider fonksyonuna gönderiliyor. Tüm bu ön hazırlıklar tamamlandıktan sonra Server Establish edilip bir döngü içerisinde Client beklemeye başlıyor. Bağlanan her client'i clientFD arrayıne koyup, ilgilenmesi için bir handleThread oluşturuyor.

handler thread

handlerThread ilk olarak client socket'inden okuma işlemini gerçekleştirip client'in bilgilerini alıyor ve parse ediyor. Ardından global olarak tanımlanmış handle_mutex 'ini kilitleyerek uygun bir provider aramasına başlıyor. Uygun bir provider bulunduktan sonra, provider bilgilerini sorunsuz şekilde güncelleyebilmek için provider'ın kendi içerisinde bulundurduğu ana mutex (provider_t.mutex_t.mutex)kilitlenir. Provider'ın sırada bekleyen client sayısı (provider_t.inQueue) arttırılır, index (provider_t.tracker) arttırılır, client socket bilgisi yazılır (provider_t.clientfd), hesaplanacak olan değer yazılır (provider_t.calc). Provider'ın gerekli tüm bilgileri güncellendikten sonra ana provider mutex'inin(provider_t.mutex_t.mutex) kilidi kaldırılır. Eğer provider client beklemesindeyse diye condition variable aracılığı ile sinyal gönderilir. Provider ile yapılması gereken her şey yapıldığı ve artık beklemeye geçileceği için handle_mutex'in kilidi kaldırılır. Provider'ının işinin bitmesini beklemek için provider'da yer alan done_mutex (provider_t.done_mutex) kilitlenir ve condition wati yapılır. Provider işini bitirdiği zaman sinyal gönderir ve handleThread done_mutex'in kilidini açarak, client socketini kapatır. Aynı zamanda clientFDs arrayı içerisinde de client socketini kapatılmış olarak işaretler.

Kısa şekilde handler thread iş örgüsü :

- Client verilerini oku
- handle_mutex'ini kilitle
- provider aramasını yap
- bulunan provider'ın temel mutex'ini kilitle
- provider'ın bilgilerini güncelleyip, client'i provider'a yönlendir.
- provider'In temel mutex'inin kilidini aç
- provider'a sinyal gönder.
- handle_mutex'inin kilidini aç
- provider için bekleme mutex'ini kilitle
- provider cond var ile beklemeye başla
- provider için bekleme mutex'inin kilidini aç
- client socketini kapat
- çıkış yap

providerThread

providerThread fonksyonuna giriş yapınca ilk olarak gettimeofday fonksoynu ile giriş zamanı kaydedilir. Ardından sonsuz bir döngü içerisine girilir. Döngü içerisinde ilk olarak provider'ın zamanının dolup dolmadığı kontrol edilir. Eğer dolmuş ise logOut(provider_t.logOut) değişkeni süresi dolmuş olarak işaretlenir ve çıkış yapılır. Dolmamışsa provider'ın ana mutex'i kilitlenelerek handlerThread'lerden sinyal ve client yönlendirmesi beklenir. Sinyal geldiği zaman, handlerThread'ın updatelediği providers arrayından işlenecek bilgileri çeker ve gerekli işlemleri yapar. Sonuçları client'a gönderir. Ardından onu beklemekte olan handlerThread'e sinyal göndererek ana mutex'inin kilidini kaldırır ve döngünün başına gelir.

Kısa şekilde provider thread iş örgüsü :

- temel mutexini kilitle
- handler'dan client yönlendirilene kadar, sıra boş olmayana kadar bekle
- cos() hesabını ve timerları hesapla
- providers üzerinde gerekli değişiklikleri yapıp, client'a sonuçları gönder
- handlerThread'e client ile işinin bittiğini belirten sinyali gönder
- temel mutex'inin kilitini kaldır

struct provider_t

data dosyasından okunan her bir provider'ın bilgilerini ve handleThread senkronize iş yapmaları için gerekli verileri barındıran struct yapısı.

- int inQueue -> sırada bekleyen client sayısı. Max 2 olabilir.
- int tracker -> işlenecek olan client için index.
- int logOut -> Provider'ın süresinin bitip bitmediğini belirtir.
- int clientfd[2] -> Bağlanmış ve sırada client'ların socket fd'lerini barındıran array. Hangi client ile ilgileneceği tracker değişkeni ile belirlenir.
- int calc[2] -> Bağlanmış ve sırada bekleyen client'ların hesaplanmasını istedikleri cos değerleri arrayi. Hangi değerin hesaplanacağı tracker değişkeni ile belirlenir.
- provider_mutex_t mutex_t -> Provider'ın ana mutex'i. Provider_t'nin içerisindeki değişkenler değiştireleceği zaman kilitlenen mutex.
- provider_mutex_t done_mutex[2] -> İşlerinin bitmesini bekleyen handleThread'ler için saklanan değişken. Sinyal gönderilirken tracker değişkeni ile hangi condition variable'a sinyal gönderileceği belirlenir.
- int clientServed -> Tamamlanmış iş sayısı.

struct provider_mutex_t

provider_t'nin derli toplu durup, tracker değişkeni kullanırak handleThread'lere sinyal gönderme işlemini kolaylaştırmak için oluşturulmuş struct. İçerisinde bir adet mutex ve bir adet condition variable bulundurur.

Client.c

İçerisinde özel bir döngü, kilitleme yoktur. Server'a port numarası ile bağlanıp bilgilerini gönderir. Ardından server'dan geri dönüş bekler. Geri dönüş geldikten sorna ilk kelimeyi kontrol ederek mesajın ne olduğuna bakılır. 3 olasılık vardır. SIGINT ile server kapatılıyordur, Provider yoktur veya beklenen işlem tamamlanmıştır ve cevaplar gelmiştir. Gelen cevaba göre ekrana çıktı yapılarak program sonlandırırlır.

Tespit Edilen Hatalar / Tam Olarak Çalışmayan veya Eksik Parçalar :

Cos() hesabı Taylor serileri ile değil, math.h kütüphanesi içinde ki cos() fonksyonu ile yapıldı.

Provider thread'leri zamanlarını doldukları zaman direkt olarak çıkış yapamıyorlar. Çıkış yapabilmeleri için sıralarında(queue) varsa en az bir client ile ilgilenmeleri gerekiyor. Böyle olduğu durumlarda bile bazen kilitlenip ne client'lara servis yapabiliyor ne de çıkış yapabiliyor.

Usage kontrolu sadece arguman sayısı olarak yapıldı. Yanlış veya doğru girilme durumları kontrol edilmedi.