grafik, yazı tipi, grafik tasarım, logo içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**PROJE ADI:** Çok Katlı Bir Apartmanın İnşası Üzerinden

Process, Thread ve Senkronizasyon Kavramlarının Modellenmesi

**HAZIRLAYANLAR :**

Levent Kutay SEZER 22360859013

Anıl SÜRMELİ 22360859018

**BÖLÜM:** Bilgisayar Mühendisliği

**DERS:** İşletim Sistemleri

**DANIŞMAN:** Mustafa Özgür CİNGİZ

**2024-2025 Bahar Dönemi**

**TARİH : 26.05.2025**

**İÇİNDEKİLER :**

1. **Giriş**
2. **Mimari Yapısı ve Fonksiyon Çeşitleri**
3. **Yarış Koşulları, Senkronizasyon ve Performans Analizi**
4. **Karşılaşılan Zorluklar**
5. **Geliştirilebilir Performans Analizi**
6. **Sonuç ve Değerlendirme**
7. **GİRİŞ :**

Günümüzde gelişen teknoloji yüzünden bilgisayar sistemleri gibi birçok yapı karmaşık olmasına rağmen hızlı çözümler üretmek amacıyla eşzamanlı ve paralel olarak donatılmaktadır. Kaynak yönetiminin zorlaşması ve gerçek zamanlı sistemlere entegre sürecinin zorluğu nedeniyle uygulamalarda bu yapılar kaçınılmaz hale gelmiştir. Bu proje vasıtasıyla söz konusu durumları uygulamalı örnek ile birleştirerek yapılandırmış ve apartman inşaatı örneğiyle bunun gibi süreçleri yazılımla modellenmiştir.

Projede 10 katlı bir bina ve her katında 4 dairesi bulunan bir apartman modeli oluşturulmuştur. Oluşturulan bu model bağımsız şekilde çalışan Thread’ler (iş parçaları) aracılığıyla yürütülmekte olup tesisatçı, tasarımcı, vinç ve asansör gibi kısıtlı kaynaklar kullanılmış olup mutex ve semafor yardımıyla kontrol edilmektedir. Her bir thread yapısı bağımsız bir daireyi inşa etmektedir ancak kaynaklara erişim sürecinde merkezi kontrol sistemlerine uymak zorundadır. Bu sayede çok iş parçacıklı (thread) programlarda en çok karşılaşılan problemlerden biri olan kaynak yarışı (race condition), verinin tutarsızlığı ve senkronizasyon eksikliği gibi birçok problemin önüne geçilmiştir.

Bu proje sayesinde sadece günümüzde değil aynı zamanda gelecekte karşılaşılabilecek karmaşık sistem senaryoları karşısında hazırlıklı olmamıza katkıda bulunur. Geliştirilen yapı, bu tarz sistemlerle kaynakların dinamik bir şekilde yönetilmesi ve kaynaklar arasındaki çakışmaların önlenmesi gibi sorunlar karşısında uygulanabilecek bir temel sunar.

1. **MİMARİ YAPISI VE FONKSİYON ÇEŞİTLERİ :**

* **Programlama Dili:** C
* **Kullanılan Kütüphaneler:**
  + **pthread.h:** İş parçacığı oluşturur ve yönetimi sağlar.
  + **semaphore.h:** Semaforlar ile kaynak kontrolü yapar.
  + **unistd.h, stdio.h, stdlib.h:** Sistem işlevleri ve Girdi Çıktı Kontrolü
* **Paralellik Yapısı:**
  + Her daire için bir iş parçacığı (thread) yaratılmaktadır.
  + Tüm katlar aynı anda inşa başlayabilmektedir ancak ortak kaynaklara erişim mutex'lerle denetlenmektedir.

**Tanımlar ve Parametreler:**

* **KAT\_SAYISI = 10:** Apartmanın toplam kat sayısını gösterir.
* **DAIRE\_SAYISI = 4:** Her katta bulunan daire sayısını gösterir.
* **MAX\_BUFFER = 512:** Pipe buffer büyüklüğünü gösterir. (işlemci)
* **DAIRE\_MALZEME = 2:** Her daire için gerekli malzeme birimini gösterir.

**Veri Yapıları ve Fonksiyonlar :**

1. **guvenli\_yazdir(const char\* mesaj) :**

**Amacı:** Thread-safe ile konsol çıktısı oluşturur.

**Ne yapar:** Mutex kullanarak konsola güvenli yazma işlemi gerçekleştirir.

**Neden gerekli:** Birden fazla thread aynı anda yazdırırsa karışıklık olur.

**Kullanım**: Her konsol mesajı bu fonksiyon ile yazılır.

1. **malzeme\_islem :**

**Amacı:** Malzeme talep ve kullanım yönetimini sağlar.

**Ne yapar:**

**islem\_turu** = 0: Malzeme kontrolü (başlangıç)

**islem\_turu** = 1: Malzeme kullanımı (bitiş)

**islem\_turu** = 2: Durum sorgulama

**Pipe kullanımı:** Parent process ile iletişim kurar.

**Dönüş:** 1 (başarılı), 0 (başarısız) olarak döndürür.

1. **kaynak\_kullan :**

**Amacı:** Ortak kaynakları güvenli kullanım sağlar.

**Ne yapar:** Vinç ve asansör gibi tek kaynaklı araçları mutex ile korur.

**Kaynaklar:** "vinç", "asansör" vb.

**Süreç:** Mutex’i al → Kaynak kullan → 1 saniye bekle → Mutex’i bırak.

1. **tesisati\_kur :**

**Amacı**: Su ve elektrik tesisatı kurulumunu sağlar.

**İki seviyeli senkronizasyon yapılır :**

**Kat seviyesi**: Aynı kattaki daireler sıralı çalışır. (mutex)

**İşçi seviyesi**: Sınırlı sayıda işçi oluşturur. (semaphore)

**Tesisatı türleri**: "su", "elektrik" vb.

**Süre**: 2 saniye kurulum sağlar.

1. **yangin\_alarm\_kur :**

**Amacı**: Yangın alarmı sistemi kurulumunu sağlar.

**Ne yapar**: 3 teknisyen ile paralel çalışmaya olanak tanır.

**Özellik**: Kat bazında sıralama olmaz. (bağımsız sistem)

**Süre:** 1 saniyedir. (hızlı kurulum)

1. **daire\_insa\_et :**

**Amacı**: Tek bir dairenin tam inşaat sürecini başlatır.

**Thread fonksiyonu**: Her daire için ayrı thread oluşturur.

1. **kat\_insa\_et :**

**Amacı**: Bir katın tüm dairelerini inşa etme görevini yapar.

**Ne yapar:**

* 4 thread oluşturur. (her daire için)
* Senkronizasyon araçlarını başlatır.
* Tüm thread'lerin bitmesini bekler. (pthread\_join)
* Senkronizasyon araçlarını temizler.

1. **process\_senkronizasyon\_baslat() :**

**Amacı:** Process içi senkronizasyon araçlarını başlatır.

**Başlattığı araçlar:**

* 6 adet mutex (vinç, asansör, kat tesisatları, konsol, pipe)
* 3 adet semaphore (2 elektrikçi, 2 tesisatçı, 3 yangın teknisyeni)

1. **process\_senkronizasyon\_temizle() :**

**Amacı**: Process içi senkronizasyon araçlarını temizler.

**Ne yapar**: Tüm mutex ve semaphore'ları destroy eder.

**Neden** **gerekli**: Bellek sızıntısını önler.

1. **malzeme\_sunucu\_calistir :**

**Amacı:** Merkezi malzeme deposu yönetir.

**Çalışma şekli**: Ayrı process'te çalışır.

**Görevleri:**

* Child process'lerden malzeme taleplerini dinler.
* Malzeme stok kontrolü yapar.
* Malzeme tüketimini kaydeder.
* Durum raporları üretir.

**Kritik:** Malzeme tükenirse tüm inşaatı durdurur.

1. **main() :**

**Amacı:** Ana program koordinatörü

**Görevleri**:

* Proje tanıtımı ve bilgilendirir.
* Pipe'ları oluşturur.
* Malzeme sunucu process'ini başlatır.
* Temel atma simülasyonu oluşturur.
* Her kat için process oluşturur.
* Yapısal istikrar kontrolü yapar. (wait() ile)
* Final raporu ve görsel çizim yapar.

**Mimari Yapısı :**

metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

1. **yarış koşulları , senkronizasyon ve performans analizi :**

Karşılaşılan temel zorluklardan birisi de aynı anda birçok threadın sınırlı sayıda kaynağa erişmeye çalışmasıdır. Bu durum dolayısıyla kaynaklar üzerinde belirli yarış koşulları (race condition) ve kilitleme (deadlock) gibi problemlere yol açması beklenir. Bu projede ise belirtilen sorunların çözülmesi amacıyla aşağıdaki senkronizasyon mekanizmaları kullanılmıştır :

* 1. **Mutex** **Kullanımı** :

* **vinc\_mutex:** Her seferinde yalnızca bir thread'in vincin kontrolünü almasına izin verilir. Bu, yük taşıma sürecinde çakışmaların önüne geçer.
* **asansor\_mutex**: Asansör kullanımı için sıraya dayalı bir yapı uygulanır. Böylece taşınan malzemelerin güvenli biçimde katlara ulaşması sağlanır.
* **konsol\_mutex**: Aynı anda birden fazla thread’in ekrana veri yazması engellenerek, çıktıların okunabilirliği ve anlam bütünlüğü korunur.
  1. **Semafor Kullanımı:**
* **tesisatci\_sem**: Simülasyonda yalnızca iki tesisatçının eş zamanlı çalışabilmesi tanımlanmıştır. Bu sayede gerçek hayattaki iş gücü kısıtları modellenmiş olur ve fazla thread’in aynı anda tesisat işlemi yaparak kaynak çakışmasına neden olması önlenir.
  1. **Yarış Koşulları Önleme:**
* Mutex’in kullanımı sayesinde aynı anda birden fazla thread aracılığıyla kaynağa erişim engellenir ve sistemdeki kontrol artırılır.
* Projede kullanılan mutex kilitlerinin edinilme sırası dikkatli bir şekilde belirlenmiş; olası deadlock senaryoları göz önünde bulundurularak kritik bölgelere öncelik verilmiştir. Özellikle vinç ve asansör gibi yüksek öneme sahip kaynaklar için tekil erişim zorunlu tutulmuştur.

d. **Performans Gözlemi :**

* gettimeofday() fonksiyonu ile her bir işlem ve thread için başlama ve bitiş zamanları alınmıştır. Bu veriler, süreçlerin ne kadar sürdüğünü ve hangi thread’in hangi kaynağı ne kadar süre kullandığını ölçmekte kullanılmıştır.
* Bu sayede tüm inşaat süreci boyunca elde edilen verilerle kaynak kullanım yoğunluğu analiz edilmiş ve darboğazların tespiti yapılabilmiştir.
* Gözlemler, paralel işlem avantajlarının performansa olan etkisini ortaya koymuş, sıralı yapılara göre zamandan ciddi tasarruf sağlandığını gösterilmiştir.

1. **KARŞILAŞILAN ZORLUKLAR** :

Projenin geliştirme sürecinde birçok sorunla karşılaşılmıştır fakat en önemli zorluklar şunlardır :

* **İş Parçacığı (thread) senkronizasyonu** : Aynı anda çalışan thread’lerin kaynaklara erişim sırasını güvenli bir şekilde organize etmek başlangıçta sorun yaratmıştır.
* **Kaynak Yarışı** : Özellikle belirtilen tesisatçı ve vinç gibi kaynaklar arasında deadlock oluşmaması için kod düzenlemeleri yapılmıştır.
* **Pipe İletişimi** : Processler arasındaki iletişimin sağlıklı olabilmesi için pipe yapılarını doğru ve sıralı bir şekilde yönetme biçimi büyük sorun yaratmıştır.
* **Çıktı** : Çıktı sırasının eksik veya düzensiz olması hata ayıklama sürecinde büyük sorunlara yol açmıştır

1. **GELİŞTİRİLEBİLİR YAPILAR :**

**Grafiksel Kullanıcı Arayüzü (GUI):** Simülasyonun işleyişini görselleştiren kullanıcı dostu bir arayüz, süreci daha anlaşılır kılar ve kullanıcıların sistem davranışlarını daha net gözlemlemesini sağlar. Thread durumları, kaynak kullanımı ve inşaat ilerleyişi gibi bilgiler gerçek zamanlı olarak arayüzde gösterilebilir.

**Kapsamlı Loglama Sistemi:** Sistemde gerçekleşen her olay, tarih ve kaynak bilgileriyle birlikte detaylı olarak kaydedilmelidir. Bu loglar daha sonra analiz edilerek darboğazlar, en yoğun kaynak kullanımı gibi verimlilik ölçümleri yapılabilir. Ayrıca log dosyaları yapay zeka tabanlı analiz sistemlerine veri sağlayarak sistem optimizasyonu yapılmasına imkân verir.

**Yapay Zeka Tabanlı Tahminleme:** Kaynak kullanımı ve olası kilitlenme durumları için sistemin geçmiş verilerine dayanarak kestirimde bulunabilen modüller eklenebilir.

1. **SONUÇ :**

Projede oluşturulan apartman inşaatı süreci paralel programlamayla başarılı bir şekilde oluşturulmuştur. Senkronizasyon ve eşzamanlılık yöntemleri ile birlikte ileride projenin geliştirilmesi sürecinde büyük bir temel atılmasına olanak sağlamıştır. Kaynakların verimli kullanımı ve sistem bütünlüğünün sağlanması gibi yazılım mühendisliğine özgü önemli konular somut bir senaryo üzerinden test edilmiştir.