Tarih: 26 Mayıs 2025

İŞLETİM SİSTEMLERİ DÖNEM PROJESİ RAPORU

Dönem: 2024–2025 Bahar

Yusuf Balmumcu 22360859001 - Yusuf Güney 22360859041
Grup No: 4

1.Giriş

Günümüz işletim sistemleri, aynı anda birden fazla işlemi ve alt iş parçacığını (thread) etkin ve düzenli şekilde yürütebilmek için çoklu işlem (multi-processing) ve çoklu iş parçacığı (multi-threading) yeteneklerini kullanmaktadır. Bu kavramların pratikte nasıl çalıştığını anlamak amacıyla, bu projede 10 katlı ve her katında 4 daire bulunan bir apartmanın inşa süreci üzerinden süreç (process), iş parçacığı (thread) ve senkronizasyon mekanizmaları modellenmiştir.

Bu yapı, işletim sistemlerinde farklı programların ve alt parçacıkların aynı anda çalışabilmesi ve kaynakları paylaşabilmesi konularını anlamaya yönelik pratik bir simülasyon sunmaktadır. Projede her kat bir process, her daire bir thread olarak tasarlanmış; temel atma, kat inşası, elektrik/su tesisatı, iç tasarım gibi işler sıralı ve paralel olarak modellenmiştir.

2. Modelleme Yaklaşımı

2.1 Apartman Yapısı

Yapı Öğesi	Sayı
Kat	10
Daire	4
Toplam	40

Her Kat = Bir process

Her Daire = Aynı process içinde çalışan bir thread

Bu yapı, işletim sistemlerindeki parent-child process ilişkisi ile aynı zamanda thread'lerin paylaşımlı kaynaklara erişimini modellemektedir.

2.2 Process Kavramı

Her kat, `fork()` fonksiyonu ile oluşturulan ayrı bir process olarak tanımlanmıştır. Üst katın inşâsına başlamadan önce alt katın tamamen bitmesi gerekmektedir. Bu yapı, `wait()` fonksiyonu ile gerçekleştirilmiştir.

```
for (int kat = 1; kat <= KAT_SAYISI; kat++) {
    pid_t pid = fork();

    if (pid < 0) {
        perror("fork hatası");
        exit(1);
    } else if (pid == 0) {
        // Çocuk process: kat inşaatı
        kat_insa_et(kat);
        exit(0);
    } else {
        // Ana process: sıradaki kata geçmeden önce bekler
        wait(NULL);
    }
}
```

2.3 Thread Kavramı

Her kat process'i içinde 4 daire pthread_create() ile oluşturulan iş parçacıkları (thread) olarak modellenmiştir:

```
for (int i = 0; i < DAIRE_SAYISI; i++) {
    daire_nolar[i] = kat_no * 10 + i + 1; // Örn: 11, 12, 13, 14
    pthread_create(&daireler[i], NULL, daire_insa_et, &daire_nolar[i]);
}

for (int i = 0; i < DAIRE_SAYISI; i++) {
    pthread_join(daireler[i], NULL);
}
```

Her bir thread şu işleri sırayla yapar:

- Asansör ile malzeme taşıma → semaphore ile senkronizasyon
- Elektrik/su tesisatı → mutex ile sırayla yapılır.
- İç tasarım → paralel yapılabilir.

3. Senkronizasyon Mekanizmaları

3.1 Process Senkronizasyonu

Senkronizasyonu, bir binanın katlarının inşa sürecine benzeterek temsil ettiğimiz bu yapıda, özellikle çoklu iş parçacığı (thread) kullanımıyla işlemlerin eşzamanlı ve uyumlu şekilde nasıl yürütüldüğü gösterilmiştir. Fonksiyon, belirli bir kat numarası alarak o kata ait dairelerin yapımını başlatır. Her daire için ayrı bir iş parçacığı oluşturulur, yani her daire aynı anda (paralel şekilde) inşa edilmeye başlanır. Örneğin, ikinci katta dört daire varsa bunlar 21, 22, 23 ve 24 numaralı daireler olarak adlandırılır. Her daire kendi numarasına göre inşa edilir ve bu işlemler bağımsız olarak gerçekleşir. Ancak katın tamamlanmış sayılabilmesi için, tüm dairelerin bitmesi beklenir. Bunun için `pthread_join()` komutu ile her iş parçacığının tamamlanması sağlanır. Tüm daireler bittikten sonra ise o kata ait inşaatın tamamlandığı ekrana yazdırılır. Bu kod sayesinde, gerçek hayattaki bir inşaat süreci örnek alınarak çoklu işlemlerin nasıl yönetildiği ve senkronizasyonun nasıl sağlandığı daha anlaşılır bir şekilde gösterilmiştir.

```
void kat_insa_et(int kat_no) {
    printf("[Kat %d] İnşaata başlandı.\n", kat_no);

    pthread_t daireler[DAIRE_SAYISI];
    int daire_nolar[DAIRE_SAYISI];

    for (int i = 0; i < DAIRE_SAYISI; i++) {
        daire_nolar[i] = kat_no * 10 + i + 1; // Örn: 11, 12, 13, 14
```

```
        pthread_create(&daireler[i], NULL, daire_insa_et,
&daire_nolar[i]);
    }

    for (int i = 0; i < DAIRE_SAYISI; i++) {
        pthread_join(daireler[i], NULL);
    }

    printf("[Kat %d] İnşaat tamamlandı.\n", kat_no);
}
```

3.2 Thread Senkronizasyonu

Bu yapıda, thread senkronizasyonu gerçek hayattaki bir bina inşaat süreciyle ilişkilendirilerek modellenmiştir. Her dairenin belirli ortak kaynaklara erişimi vardır ve bu kaynakların aynı anda birden fazla daire tarafından kullanılmaması gerekmektedir.

Asansör Kullanımı:

Asansör, bina inşaatında sınırlı sayıda kullanılabilen bir kaynak olarak ele alınmıştır ve bu nedenle senkronizasyonu semaphore (semafor) yapısıyla sağlanmıştır. `sem_wait()` komutu ile bir daire asansörü kullanmak istediğinde izin alır; eğer asansör meşgulse bekler. İşlem tamamlandığında `sem_post()` ile asansör serbest bırakılır ve sıradaki dairenin kullanımına sunulur.

Uygulamada asansör kullanımı üç farklı aşamada görülmektedir:

- Malzeme taşımada (inşaat öncesi hazırlık)
- Cam montajı (vinç benzeri kaynak gerektiren donanım kurulumu)
- Mobilya taşıma (inşaat sonrası son rötuşlar)

Bu yapı, sınırlı kaynakların kontrollü ve sıraya dayalı şekilde kullanımını sağlar ve gerçek hayattaki kaynak paylaşımını yazılımsal olarak doğru şekilde temsil eder.

Elektrik/Su/Doğalgaz Tesisatı:

Elektrik, su ve doğalgaz tesisatı işlemleri aynı anda yalnızca bir dairede yapılabilecek kritik işlemler olarak düşünülmüş ve mutex (karşılıklı dışlama) yapısıyla senkronize edilmiştir. `pthread_mutex_lock()` komutu ile tesisat işlemine başlayan daire, diğer dairelerin aynı işlemi yapmasını engeller. Bu sayede kaynak çakışması, güvenlik riski ve veri tutarsızlığı gibi senaryolar önlenmiş olur. İşlem tamamlandıktan sonra `pthread_mutex_unlock()` ile kaynak serbest bırakılır ve diğer dairelerin erişimine açılır.

Bu yaklaşım sayesinde altyapı işlemleri güvenli ve çakışmasız şekilde gerçekleştirilmiş olur.

```
// Asansör kullanımı
sem_wait(&asansor);
printf("[Daire %d] Asansör kullanılıyor\n", daire_no);
sleep(1);
sem_post(&asansor);

// Elektrik/su tesisatı
pthread_mutex_lock(&kaynak_mutex);
printf("[Daire %d] Elektrik/su tesisatı yapılıyor\n", daire_no);
sleep(1);
pthread_mutex_unlock(&kaynak_mutex);
```

4. Paralel ve Sıralı İşler

Sıralı Olarak Gerçekleştirilenler:

- Temel atılmadan hiçbir kat başlamaz. (Bu işlem yalnızca ilk kat öncesinde bir defa gerçekleştirilmiştir.)
- Kat 2, ancak Kat 1'in tamamlanmasından sonra başlar. Bu süreç `fork()` ile oluşturulan çocuk process'lerin `wait()` fonksiyonu ile senkronize edilmesiyle sağlanır.
- Elektrik, su ve doğalgaz tesisatı işlemleri aynı katta sırayla yapılır. Bu durum `mutex` ile sağlanmış ve her bir altyapı işlemi aynı anda yalnızca bir thread tarafından yapılabilecek şekilde düzenlenmiştir.
- Asansör ile yapılan taşıma işlemleri (malzeme, cam, mobilya), sınırlı kaynak olduğundan dolayı da sıralı yapılmaktadır. `semaphore` ile kontrol edilmiştir.

Paralel Olarak Gerçekleştirilenler:

- İskele kurulumu ve malzeme yerleştirme işlemleri birbirinden bağımsızdır ve her daire bunları aynı anda gerçekleştirebilir.
- Daire içi yapısal işlemler (duvar örme, kapı/pencere kasası yerleştirme, iç bölme duvar yapımı) paralel olarak yürütülür.
- İç kaplama işleri (sıva, fayans döşeme, boya ve parke uygulamaları) aynı anda birçok dairede yapılabilir.
- Donanım kurulumunun bazı parçaları (kapı montajı, mutfak dolabı yerleşimi, priz/anahtar montajı ve aydınlatma sistemleri) paralel yürütülebilecek işlemlerdir.
- Temizlik ve son kontroller gibi işler daireye özel olduğundan eş zamanlı yürütülebilir.

Bu paralel yapı sayesinde çok iş parçacıklı (multi-threaded) bir ortamda kaynak verimli kullanılarak dairelerin inşa süreci optimize edilmiştir.

5. Kaynak Paylaşımı ve Yarış Koşulları

Gerçek hayattaki gibi, burada da sınırlı kaynaklar paylaşılır:

Asansör: Bir daire malzeme taşırken diğerleri bekler. Eğer `sem_wait()` uygulanmazsa, race condition oluşur.
Tesisat: Aynı anda birden fazla thread elektrik/su tesisatı döşemeye çalışırsa veri bozulabilir. Mutex ile bu engellenir.

Bu durumlar işletim sistemlerinde karşılaşılan klasik senkronizasyon problemlerinin benzeridir.

6. Sonuç ve Değerlendirme
Bu projede, işletim sistemi kavramları olan process, thread, mutex, semafor ve senkronizasyon, apartman inşası örneği ile modellenmiştir. Gerçek hayattaki inşaat süreçlerinin sıralı ve eş zamanlı işleyişi, yazılım düzeyinde process ve thread'lerle uyumlu şekilde simüle edilmiştir.
7. Ekran Çıktısı Örneği