BURSA TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

İŞLETİM SİSTEMLERİ PROJE RAPORU

AHMET FURKAN ÖCEL 23360859729

LÜTFÜ BEDEL 21360859030

1. GİRİŞ :

Modern işletim sistemlerinde süreç (process) ve iş parçacığı (thread) kavramları, karmaşık görevlerin eşzamanlı ve verimli bir şekilde gerçekleştirilmesini sağlayan temel yapı taşlarıdır. Bu proje kapsamında, çok katlı bir apartmanın inşa sürecini temel alarak process'lerin ve thread'lerin birbirleri ile senkron bir şekilde çalışması modellenmiştir. Her bir kat bağımsız bir process olarak ele alınırken, her katta yer alan dört daire ayrı thread'lerle temsil edilmiştir. Bu yapı sayesinde hem çoklu işlem hem de çoklu iş parçacığı kullanımı simüle edilmiş; sınırlı sayıda bulunan vinç gibi ortak kaynakların paylaşımı ise semaphore kullanılarak senkronize edilmiştir. Bu yaklaşım, işletim sistemlerinin kaynak yönetimi, eşzamanlılık kontrolü ve yarış durumu gibi temel sorunlarını somut bir model üzerinden inceleme imkânı sunmaktadır.

2. TEKNİK DETAYLAR :

Gerekli kütüphanelerin dahil edilmesi :

- **#include <stdio.h>** **:** Girdi çıktı işlemleri için kullanılır.
- **#include <stdlib.h>** **:** Genel amaçlı fonksiyonlar için kullanılır.
- **#include <unistd.h>** **:** fork() ve sleep() gibi POSIX fonksiyonlarını kullanmak için gereklidir.
- **#include <pthread.h** **:** Thread'lerin oluşturulması ve yönetimi için gerekli olan kütüphanedir.
- **#include <semaphore.h>** **:** Semaphore işlemleri için kullanılır.
- **#include <fcntl.h** **:** Semaphore flag'leri için kullanılır.
- **#include <sys/wait.h>** **:** Process senkronizasyonu için wait() fonksiyonuna erişimi sağlar.
- **#include <string.h>** **:** String işlemleri yapılabilmesi için dahil edilmiştir.

Kullanılacak sabitlerin tanımlanması :

- **#define KAT_SAYISI 10** **:** Apartmanın toplamda 10 katlı olduğunu belirtir.
- **#define DAIRE_SAYISI 4** **:** Her katta 4 daire bulunduğunu ifade eder.
- **#define SEM_VINC "/vinc"** **:** Vinc isimli ortak kaynak için sistem genelinde tanımlanan semaphore'un adı. Tüm thread'ler bu semaforu kullanır.

```c
#include <stdio.h>      // Girdi cikti islemleri icin
#include <stdlib.h>     // Genel amacli fonksiyonlar icin (malloc, exit, vb.)
#include <unistd.h>     // POSIX fonksiyonlari icin (fork, sleep, vb.)
#include <pthread.h>    // Thread olusturma ve yonetimi icin
#include <semaphore.h>  // Semaphore islemleri icin
#include <fcntl.h>      // Semaphore flag'lari icin (O_CREAT, O_EXCL, vb.)
#include <sys/wait.h>   // Process bekleme islemleri icin
#include <string.h>     // String islemleri icin (gerekirse)

#define KAT_SAYISI 10       // Toplam kat sayisi
#define DAIRE_SAYISI 4      // Her katta 4 daire var
#define SEM_VINC "/vinc"    // Ortak kaynak: vinc icin semaphore adi
```

Daire Thread Fonksiyonu

Her daire işlemi ayrı bir thread olarak tanımlanmıştır. Fonksiyonun içeriği sırasıyla aşağıdaki gibidir .

Öncelikle parametre olarak verilen arguman pointeri int'e çevrilerek "daire_no" olarak tutulur. Ardından bellek sızıntısını önlemek için free() fonksiyonu ile argüman olarak verilen pointerın hafızada gösterdiği yer serbet bırakılır.

Ardından daha önceden global olarak tanımlanmış olan "vinç" semaforu açılır. Semaforun açılması sırasında bir hata oluşursa konsola hata mesajı yazdırılır ve thread'den çıkış yapılır.

Hata oluşmazsa eğer sem_wait() fonksiyonu ile ortak kaynak olarak tanımlanmış olan "vinç" kilitlenir ve kullanılır. 1 saniyelik bir bekleme süresinden sonrada sem_post() fonksiyonu ile kaynak (vinç) serbest bırakılır.

İşlem tamamlandıktan sonra sem_close() fonksiyonu ile semaphore bağlantısı kapatılır ve thread'den çıkılır.

```c
// -------------------- DAIRE THREAD FONKSIYONU --------------------
void* daire_islemi(void* arg) {
    int daire_no = *((int*)arg);         // Gelen argumani int olarak al
    free(arg);                           // Bellek sizintisi olmamasi icin heap'ten alinan bellegi bosalt

    // Vinc semaforunu ac
    sem_t* vinc = sem_open(SEM_VINC, 0);
    if (vinc == SEM_FAILED) {            // Acilamazsa hata ver
        perror("[HATA] Vinc semaforu acilamadi");
        pthread_exit(NULL);              // Thread cikisi
    }

    printf("[DAIRE %02d] Siva ve tesisat icin vinc bekleniyor...\n", daire_no);

    sem_wait(vinc);                      // Vinc kaynaginin kilidini al

    // Vinc kullanimi simulasyonu
    printf("[DAIRE %02d] Vinci kullaniyor...\n", daire_no);
    sleep(1);                            // 1 saniye bekleyerek islem simulasyonu yap

    printf("[DAIRE %02d] Vinci kullanmayi bitirdi.\n", daire_no);
    sem_post(vinc);                      // Vinc kaynagini serbest birak

    sem_close(vinc);                     // Semaphore baglantisini kapat

    pthread_exit(NULL);                  // Thread basariyla bitti
}
```

KAT PROCESS FONKSIYONU :

Bu fonksiyon process'lerin yani katların inşa edildiği kısımdır.

Başlangıçda daire sayısı kadar yani 4 adet thread oluşturulur ve her biri bir daireyi temsil eder. Thread'ler oluşturulurken her biri için bulunduğu kata ve sırasına göre bir daire numarası oluşturulur ve malloc() fonksiyonu ile kaydedilir.

pthread_create() fonksiyonu ile oluşturulan her bir thread başlatılır. Başlatılma sırasında hata oluşursa konsola hata mesajı yazdırılır ve free() fonksiyonu ile daireler için bellekten ayrılmış kısımlar serbest bırakılır.

Hata alınmaması durumunda pthread_join() fonksiyonu ile tüm thread'lerin tamamlanması beklenir. Kat işlemleri tamamlandıktan sonra process kendini exit() ile sonlandır.

```c
// -------------------- KAT PROCESS FONKSIYONU --------------------
void kat_insa_et(int kat_no) {
    pthread_t threads[DAIRE_SAYISI];     // Her daire icin thread dizisi

    printf("\n[KAT %d] Insaat basladi.\n", kat_no);

    // Her daire icin bir thread olustur
    for (int i = 0; i < DAIRE_SAYISI; i++) {
        int* daire_no = malloc(sizeof(int)); // Her daire icin dinamik arguman olustur
        if (daire_no == NULL) {
            perror("[HATA] malloc basarisiz");
            exit(1);
        }

        *daire_no = kat_no * 10 + (i + 1);   // Daire numarasi ornegi: 21, 22, 23, 24

        if (pthread_create(&threads[i], NULL, daire_islemi, daire_no) != 0) {
            perror("[HATA] Thread olusturulamadi");
            free(daire_no); // malloc basarisiz degil ama thread olusturulamadiysa da serbest birak
            exit(1);
        }
    }

    // Tum thread'lerin bitmesini bekle
    for (int i = 0; i < DAIRE_SAYISI; i++) {
        pthread_join(threads[i], NULL);     // Thread'leri bekle
    }

    printf("[KAT %d] Tum daireler tamamlandi.\n", kat_no);

    exit(0);                                // Process basariyla bitti
}
```

ANA PROGRAM :

Programın çalışmasını başlatan ana fonksiyon şu işlemleri gerçekleştirir:

Başlangıçda önceden var olan semaphore varsa set_unlink() ile temizlenir.

O_CREAT | O_EXCL ile sistemde yalnızca bir tane olacak şekilde bir vinc semaforu oluşturulur. Semafor oluşturma sırasında bir hata oluşursa konsola hata mesajı yazdırılır ve programdan çıkılır.

Hata yoksa for döngüsü ve fork() fonksiyonu kullanılarak her kat için ayrı bir process başlatılır. Çocuk process kat_insa_et() fonksiyonunu çağırarak kat inşasını başlatır. Ebeveyn process, çocuğun bitmesini wait() fonksiyonu ile bekler, böylece katlar sıralı bir şekilde inşa edilir.

Tüm katlar inşa edildikten sonra vinç semaforu kapatılır ve sistemden silinir. Program “Apartman insaati basariyla tamamlandi.” mesajı ile bitirilir.

```c
// -------------------- ANA PROGRAM --------------------
int main() {
    sem_unlink(SEM_VINC);                  // Onceki semaphore varsa sil

    // Vinc semaforu 1 kaynakla olustur
    sem_t* vinc = sem_open(SEM_VINC, O_CREAT | O_EXCL, 0644, 1);
    if (vinc == SEM_FAILED) {
        perror("[HATA] Vinc semaforu olusturulamadi");
        exit(1);
    }

    printf("Apartman insaati basliyor...\n");

    // Her kat icin process olustur
    for (int kat = 1; kat <= KAT_SAYISI; kat++) {
        pid_t pid = fork();                // Yeni process olustur

        if (pid < 0) {
            perror("[HATA] fork hatasi");  // Fork hatasi kontrolu
            exit(1);
        }

        if (pid == 0) {
            // Cocuk process: kendi katini insa eder
            kat_insa_et(kat);
        } else {
            // Ana process: cocuk bitene kadar bekler
            wait(NULL);
        }
    }

    sem_close(vinc);                       // Vinc semaforu kapat
    sem_unlink(SEM_VINC);                  // Vinc semaforu sistemden sil

    printf("\nApartman insaati basariyla tamamlandi.\n");
    return 0;                              // Program sonlandi
}
```

PROGRAN ÇIKTILARI :

```
btu59030@Ubuntu:~/İndirilenler$ ./code
Apartman insaati basliyor...

[KAT 1] Insaat basladi.
[DAIRE 11] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 11] Vinci kullaniyor...
[DAIRE 14] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 13] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 12] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 11] Vinci kullanmayi bitirdi.
[DAIRE 14] Vinci kullaniyor...
[DAIRE 14] Vinci kullanmayi bitirdi.
[DAIRE 13] Vinci kullaniyor...
[DAIRE 13] Vinci kullanmayi bitirdi.
[DAIRE 12] Vinci kullaniyor...
[DAIRE 12] Vinci kullanmayi bitirdi.
[KAT 1] Tum daireler tamamlandi.

[KAT 2] Insaat basladi.
[DAIRE 22] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 22] Vinci kullaniyor...
[DAIRE 21] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 23] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 24] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 22] Vinci kullanmayi bitirdi.
[DAIRE 21] Vinci kullaniyor...
[DAIRE 21] Vinci kullanmayi bitirdi.
[DAIRE 23] Vinci kullaniyor...
[DAIRE 23] Vinci kullanmayi bitirdi.
[DAIRE 24] Vinci kullaniyor...
[DAIRE 24] Vinci kullanmayi bitirdi.
[KAT 2] Tum daireler tamamlandi.

[KAT 3] Insaat basladi.
[DAIRE 33] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 33] Vinci kullaniyor...
[DAIRE 31] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 34] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 32] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 33] Vinci kullanmayi bitirdi.
[DAIRE 31] Vinci kullaniyor...
[DAIRE 31] Vinci kullanmayi bitirdi.
[DAIRE 34] Vinci kullaniyor...
[DAIRE 34] Vinci kullanmayi bitirdi.
[DAIRE 32] Vinci kullaniyor...
[DAIRE 32] Vinci kullanmayi bitirdi.
[KAT 3] Tum daireler tamamlandi.
```

```
[DAIRE 73] Vinci kullaniyor...
[DAIRE 73] Vinci kullanmayi bitirdi.
[DAIRE 72] Vinci kullaniyor...
[DAIRE 72] Vinci kullanmayi bitirdi.
[DAIRE 74] Vinci kullaniyor...
[DAIRE 74] Vinci kullanmayi bitirdi.
[KAT 7] Tum daireler tamamlandi.

[KAT 8] Insaat basladi.
[DAIRE 81] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 81] Vinci kullaniyor...
[DAIRE 82] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 83] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 84] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 81] Vinci kullanmayi bitirdi.
[DAIRE 82] Vinci kullaniyor...
[DAIRE 82] Vinci kullanmayi bitirdi.
[DAIRE 83] Vinci kullaniyor...
[DAIRE 83] Vinci kullanmayi bitirdi.
[DAIRE 84] Vinci kullaniyor...
[DAIRE 84] Vinci kullanmayi bitirdi.
[KAT 8] Tum daireler tamamlandi.

[KAT 9] Insaat basladi.
[DAIRE 91] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 91] Vinci kullaniyor...
[DAIRE 93] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 94] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 92] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 91] Vinci kullanmayi bitirdi.
[DAIRE 93] Vinci kullaniyor...
[DAIRE 93] Vinci kullanmayi bitirdi.
[DAIRE 94] Vinci kullaniyor...
[DAIRE 94] Vinci kullanmayi bitirdi.
[DAIRE 92] Vinci kullaniyor...
[DAIRE 92] Vinci kullanmayi bitirdi.
[KAT 9] Tum daireler tamamlandi.

[KAT 10] Insaat basladi.
[DAIRE 102] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 102] Vinci kullaniyor...
[DAIRE 104] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 101] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 103] Siva ve tesisat icin vinc bekleniyor...
[DAIRE 102] Vinci kullanmayi bitirdi.
[DAIRE 104] Vinci kullaniyor...
[DAIRE 104] Vinci kullanmayi bitirdi.
[DAIRE 101] Vinci kullaniyor...
[DAIRE 101] Vinci kullanmayi bitirdi.
[DAIRE 103] Vinci kullaniyor...
[DAIRE 103] Vinci kullanmayi bitirdi.
[KAT 10] Tum daireler tamamlandi.

Apartman insaati basariyla tamamlandi.
btu59030@Ubuntu:~/İndirilenler$
```

3. SONUÇ :

Bu projede, işletim sistemlerinin temel konularından olan process, thread ve senkronizasyon kavramları, çok katlı bir apartman inşaatı senaryosu üzerinden modellenmiştir. Her kat ayrı bir process olarak ele alınmış, her katta yer alan daireler ise thread'lerle temsil edilmiştir. Katlar sıralı şekilde inşa edilirken, aynı kattaki dairelerin işlemleri eş zamanlı olarak gerçekleştirilmiştir. Bu yapıda, sınırlı bir kaynak olan vincin tüm daireler arasında yarış durumu oluşturmadan paylaşılabilmesi için semaphore mekanizmasından yararlanılmıştır.

Gerçekleştirilen bu model sayesinde, çok işlemcili sistemlerde kaynak paylaşımı, görevlerin eş zamanlı yürütülmesi ve süreçler arası iletişim gibi temel işletim sistemi konuları somut bir şekilde gözlemlenmiştir. Geliştirilen C kodu, hem çoklu process hem de çoklu thread kullanımını başarılı bir şekilde birleştirmekte; aynı zamanda yarış durumlarını önleyerek doğru ve güvenli bir eşzamanlı çalışma ortamı sağlamaktadır.

Sonuç olarak, bu çalışma sayesinde, işletim sistemlerinin çekirdek prensipleri olan paralellik, senkronizasyon ve kaynak yönetimi gerçek hayattaki bir inşaat analojisi üzerinden simüle edilerek hem teorik hem de uygulamalı bir öğrenme sağlanmıştır.

4. PROJENİN GİTHUB LİNKİ :

https://github.com/lutfubedel/Isletim_Sistemleri_Donem_Projesi

5. KAYNAKÇA :

https://www.geeksforgeeks.org/multithreading-in-c/

https://www.elektrikport.com/makale-detay/thread-nedir-thread-cde-nasil-kullanilir-ve-nasil-yazilir/23510#ad-image-0

https://www.geeksforgeeks.org/fork-system-call/

https://www.gnu.org/software/libc/manual/html_node/Processes.html