T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

GÖREVLENDİRİCİ (DISPATCHER) KABUĞU

GRUP ÜYELERİ:

- **AYŞEGÜL BİLİCİ (B211210100)**
- **MUSTAFA BİÇER (B201210078)**
- **SEVDA FARSHİDFAR (B201210602)**
- **VEDAT ÖZTÜRK (B211210085)**
- **MAHDİ HOJJATİ (B211210574)**

SAKARYA, 2023-2024

Özet

Projede dinamik bellek tahsisinin kullanılması tercih edilmiştir çünkü gerçek zamanlı işlemler de dâhil olmak üzere çeşitli öncelik seviyelerine sahip görevlerin yönetilmesi ve kaynakların etkin bir şekilde kullanılması önemlidir. Dinamik bellek tahsisi, sistemdeki işlemlere daha fazla esneklik sağlayarak kaynakları optimize etmeye yardımcı olacaktır. Proje, temel bir çok düzeyli görevlendirme sistemi tasarımını anlamak ve uygulamak için bir temel sunmaktadır.

Bellek Tahsis Algoritmaları

Projede dinamik bellek tahsisinin kullanılması tercih edilmiştir çünkü gerçek zamanlı işlemler de dâhil olmak üzere çeşitli öncelik seviyelerine sahip görevlerin yönetilmesi ve kaynakların etkin bir şekilde kullanılması önemlidir. Dinamik bellek tahsisi, sistemdeki işlemlere daha fazla esneklik sağlayarak kaynakları optimize etmeye yardımcı olacaktır.

Bellek ve diğer kaynakların Yönetimi

Bellek ve diğer kaynakların yönetimi için kullanılan yapılar şunlardır:

Bellek Yönetimi: Bellek tahsisini kontrol etmek için "allocateMemory "ve " releaseResources" metodları kullanılmıştır. Bu metotlar, işlemlere bellek sağlama ve bellek iade etme işlemlerini gerçekleştirir.

Kaynak Yönetimi: "allocateResources" metodunda, yazıcılar, tarayıcılar, modemler ve CD sürücülerinin tahsis edilmesi kontrol edilir. Bu yöntem, ilgili kaynakları işleme tahsis etme ve işlem tamamlandığında bu kaynakları serbest bırakma işlevselliğini içerir.

Program Yapısı ve Modüller

Dispatcher Sınıfı: Dispatcher sınıfı, işlemleri önceliklerine göre yönlendirme ve kaynakları tahsis etme görevini üstlenir. dispatchProcesses ve printStatus metotları bu sınıfta bulunur.

Process Sınıfı: Process sınıfı, bir işlemin özelliklerini (varış zamanı, öncelik, işlem süresi, bellek gereksinimi, yazıcılar, tarayıcılar, modemler, CD sürücüleri) temsil eder.

Artılar:

- Gerçek zamanlı işlemlere ve farklı öncelik seviyelerine uyum sağlar.
- Dinamik bellek tahsisi, kaynakları etkin bir şekilde yönetir.

- Öncelik kuyrukları, işlemlerin öncelik düzeyine göre sıralı bir şekilde işlenmesini sağlar.

İyileştirmeler ve Eksiklikler:

- Gerçek zamanlı işlemler için daha kesin bir garanti sağlamak için özel bir gerçek zamanlı işletim sistemi tasarımı düşünülebilir.
- Performans artışı için düşük seviyeli optimizasyonlar ve donanım özelliklerine daha fazla odaklanma.
- Etkinlik ve güvenilirlik için hata yönetimi mekanizmalarını güçlendirmek.

Dispatcher

Bu modül, bir işletim sistemi simülasyonu veya gerçek zamanlı görev yönetim sistemini modelleyen bir programın bir parçasını içerir. Temel olarak, "**Dispatcher**" adlı bu sınıf, işlemleri önceliklerine göre yönlendirme ve gerekli kaynakları tahsis etme görevini üstlenir.

Bileşenler

1. **UserJobQueue ve RealTimeQueueScheduler**: Kullanıcı işlemlerini ve gerçek zamanlı işlemleri yönetmek için kullanılan kuyruk yapılarıdır. Gerçek zamanlı işlemler öncelikle RealTimeQueueScheduler kullanılarak işlenir.
2. **DeviceManager:** Bu sınıf, tüm statik G/Ç (Input/Output) cihazlarını içerir ve işlemlerin bu cihazlara erişimini kontrol eder.
3. RAM (Bellek): Sistem belleğini temsil eder ve işlemlerin bellek ihtiyaçlarını kontrol eder.
4. **Chronometer:** Zamanı ölçen bir kronometre sınıfıdır. İşlemlerin varış zamanları ve işlemlerin süreleri ile ilgili hesaplamalarda kullanılır.
5. **dispatchProcesses** Metodu: Bu metod, işlemleri yönlendirme ve kaynakları tahsis etme işlevselliğini gerçekleştirir. Özellikle, işlemlerin varış zamanlarına ve önceliklerine göre işlemleri uygun kuyruklara ekler. Ayrıca, gerekli kaynakların mevcut olup olmadığını kontrol eder ve uygun durumdaysa işlemi ilgili kuyruğa ekler.
6. **Eksiklikler ve İyileştirmeler:** Kodun son kısmında, gerçek zamanlı işlemler için daha kesin bir garanti, performans artışı için düşük seviyeli optimizasyonlar ve donanım özelliklerine odaklanma, hata yönetimi mekanizmalarını güçlendirme gibi iyileştirmeler ve eksiklikler önerilmiştir.

Sistem Prosesleri Yönetimi:

- **RealTimeQueueScheduler:**

RTQS, First Come First Served(FCFS) çizelgeleme algoritmasından faydalanarak sistemdeki en yüksek önceliğe sahip olan gerçek zamanlı sistem proseslerini hiç bir kesme(interrupt) olmadan icra eder.

1. **Sınıf ve Değişkenler:**
    - RealTimeQueueScheduler sınıfı, gerçek zamanlı sistem proseslerinden sorumlu FCFS algoritmasına dayalı bir çizelgeleyiciyi gerçekler.
    - _ram ve _resourses nesneleri bellek ve işlemci sınıflarına ulaşabilmemiz ve gerekli kaynakları tutabilmemizi sağlar.
    - realTimeQueue bağlı listesi sistem proseslerini sıralayan ve FCFS algoritmasının üzerine uygulanacağı yapıyı oluşturur. Burda Queue veri yapısı yerine Linked List veri yapısının kullanılmasının nedeni, realTimeQueue'da sırası gelen proseslerin icra süreleri tamamlanana kadar kuyruktan çıkarmamamızın gerekmesi ve Linked List veri yapısındaki peek() fonksiyonunun bunu sağlaması.
2. **Proses Eklenmesi:**
    - Eklenen her prosese eğer 20 saniyelik proses limitini aşmıyorsa "ready" proses state'i verilir ve kuyruğa eklenir.
3. **Tetiklenme:**
    - Birim zaman aralığında, MFQS tetikleyicinden önce tetiklenir ve kuyruktaki proses varlığını kontrol eder.
    - En yüksek önceliğe sahip prosesleri içerdiğinden dolayı bu kuyruk MFQS yapısındaki kuyrukları kesebilir(interrupt).
    - Kuyrukta herhangi bir proses varsa FCFS algoritmasına göre bu zaman aralığında icra edilir.
4. **Kuyruğun Çalışması:**
    - İlk proses kuyruktan alınır ve çalıştırılır.
    - Belirlenen zaman kuantumu kadar beklenir.
    - Prosesin icrası tamamlanmazsa bir sonraki zaman aralığında da icrasına devam eder.
    - Aksi takdirde, proses kuyruktan çıkarılır ve kaynakları sisteme iade edilir.

Kullanıcı Prosesleri Yönetimi:

- **User Job Queue:**

Dispatcher, varış zamanı gelmiş kullanıcı prosesleri, User Job Queue modülüne aktarmaktadır. Bu modül, kullanıcı işlemlerini yöneten bir kuyruk sınıfını içerir. Aynı zamanda, bu kullanıcı işlemlerini çoklu öncelikli kuyruk tabanlı bir Geri Beslemeli Görevlendirici olan **MultilevelFeedbackQueueScheduler** sınıfına ekler ve yönetir.

1. **Sınıf ve Değişkenler:**
    - UserJobQueue sınıfı, kullanıcı işlemlerini yöneten ana sınıftır.
    - "queue" değişkeni, kullanıcı işlemlerini içeren bir kuyruğu temsil eder.
    - "mfqs" değişkeni, MultilevelFeedbackQueueScheduler sınıfından bir örneği tutar.

2. **Proses Eklenme:**
    - Kuyruğa yeni prosesleri ekleyen bir metoda sahiptir.
3. **Tetiklenme:**
    - Sistem zamanlayıcısı tarafından her 1 saniyede tetiklenir.
    - Kuyruktaki her prosesi kontrol eder.
    - Proses 20 saniye limitine ulaştıysa, proses sonlandırılır.
    - İşlem için gerekli kaynaklar (cihazlar ve bellek) uygunsa, proses çoklu öncelikli kuyruğa eklenir.
    - En sonda da, Geri Beslemeli Görevlendirici tetiklenir.

- **MultilevelFeedbackQueueScheduler:**

  Bu modül, çoklu öncelikli bir kuyruk (Multilevel Feedback Queue) tabanlı bir Geri Beslemeli Görevlendirici tasarımını içerir.

  1. **Sınıf ve Değişkenler:**
     a. MultilevelFeedbackQueueScheduler sınıfı, çoklu öncelikli kuyruk tabanlı bir Geri Beslemeli Görevlendiriciyi temsil eder.
     b. "numberOfLevels" değişkeni, kuyrukların sayısını belirtir (bu projede 3 seviye bulunmaktadır).
     c. "timeQuantums" dizisi, her seviye için zaman kuantumlarını tutar.
     d. "queues" dizisi, her seviyedeki kuyrukları temsil eder.
     e. RRQ değişkeni, en düşük öncelikli kuyruk için Round Robin algoritmasını uygular.
  2. **Proses Eklenme:**
     a. Belirli bir seviyeye yeni prosesleri ekleyen bir metoda sahiptir.
     b. Eklenen işlem "hazır" durumuna getirilir ve ilgili kuyruğa eklenir.
  3. **Tetiklenme:**
     a. Sistem zamanlayıcısı tarafından belirli aralıklarla (1 saniye) tetiklenir.
     b. Her seviyedeki kuyruklar kontrol edilir.
     c. Bir seviyedeki kuyrukta proses varsa, işlemlerin süre sınırlarına bakılır.
     d. İşlem 20 saniyeden fazla sürdüyse, işlem sonlandırılır ve bellek/cihazlar serbest bırakılır.
     e. Geriye kalan işlemler için önceliklere göre çalıştırma işlemi yapılır. Eğer proses en düşük seviyedeki kuyrakta ise, çalıştırma işlemi **Round Robin** sınıfı tarafından yapılır.
     f. Gerçek zamanlı bir işlem varsa, kullanıcı işlemi kesilir.
  4. **Kuyruğun Çalışması:**
     a. İlk proses kuyruktan alınır ve çalıştırılır.
     b. Belirlenen zaman kuantumu kadar beklenir.
     c. İşlem tamamlanmazsa, bir sonraki seviyedeki kuyruğa eklenir.
     d. Aksi takdirde, işlem tamamlanır ve bellek/cihazlar serbest bırakılır.

- **Round Robin:**
  Proses kuyruğunu Round Robin algoritması ile çalıştıran bir modül.

  Kuyruğun Çalışması:

- İlk proses kuyruktan alınır ve çalıştırılır.
- Belirlenen zaman kuantumu kadar beklenir.
- İşlem tamamlanmazsa, kuyruğun sonuna eklenir.
- Aksi takdirde, işlem tamamlanır ve bellek/cihazlar serbest bırakılır.

Proje Github Linki: https://github.com/poqob/SimulationOfAnOperatingSystem