T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

İŞLETİM SİSTEMLERİ

GÖREVLENDİRİCİ (DISPATCHER) KABUĞU

PROJE RAPORU

B191210038 - Mahir TEKİN
B191210048 - Murat Samet TOKDEMİR
G201210301 - Nagihan YALÇIN
G201210560 - Khadega ALATEYA
B191210084 - Gülay DEMİR

GÖREVLENDİRİCİ KABUĞU PROGRAM YAPISI

Bu programda iki sınıf tanımlanmaktadır. Bunlar **'Process'** ve **'Dispatcher'** sınıflarıdır. 'Process' sınıfı, görevlendiricideki bir işlemi temsil etmektedir. 'Dispatcher' sınıfı ise işlemleri çalıştırmak için önceliğe dayalı zamanlamayı kullanan bir görevlendiriciyi temsil etmektedir.

Geliştirilen görevlendirici **'First Come First Serve (FCFS, İlk Gelen İlk Çalışır)'** algoritması temelinde çalışmaya başlamaktadır. Bu algoritma işletim sistemlerinde kullanılan bir proses görevlendirme algoritmasıdır. Gelen prosesleri sırayla çalıştırır ve proseslerin sırasını proseslerin sisteme girdiği zamana göre belirler. Sisteme ilk gelen proses ilk çalıştırılacak olandır. Sonrasında sisteme giren diğer prosesler sırayla çalıştırılır. FCFS algoritması, proseslerin sırasını belirlerken basit bir yöntem kullanır ve bu sayede oldukça kolay uygulanabilmektedir. Proseslerin çalıştırılma sırası hakkında net bir fikir verir ve bu sayede proseslerin ne zaman çalıştırılacağı konusunda anlaşılır bir yapı oluşur.

'Process' sınıfında yer alan 'id' değişkeni prosesin kimliğini, 'arrivalTime' prosesin görevlendiriciye varış zamanını, 'priority' prosesin öncelik düzeyini, 'burstTime' prosesin yürütülmesini tamamlaması için gereken süreyi, 'startingBurstTime' ise herhangi bir çalışmadan önce yani prosesin ilk oluşturulduğu andaki orijinal işlem süresini tanımlamaktadır. Bu sınıfta 'Dispatcher.idCounter' ile, her yeni proses oluşturulduğunda artırılan bir sayaç kullanarak id değişkenini ayarlanmaktadır. Bu sınıfın yapılandırıcısı (constructor), bu değişkenlerin değerlerini almakta ve bunların değerlerini ilgili özelliklere atamaktadır. Diğer metotlar, bu özelliklerin değerlerine erişimi sağlamakta ve işlem süresini değiştirmektedir.

'Dispatcher' sınıfı işlemleri çalıştırmak için önceliğe dayalı zamanlamayı kullanan bir görevlendiriciyi temsil etmektedir. Zamanlayıcının dört öncelik düzeyi vardır. 'REAL_TIME' en yüksek önceliğe ve 'PRIORITY_3' en düşük önceliğe sahiptir. Bir proses vardığında çalışmaya hazır durumdadır. Bekleyen gerçek zamanlı prosesler FCFS algoritmasına göre yürütülmek üzere gönderilmektedir. 'run()' metodu, öncelik sıralarındaki işlemlerin belirli bir sırada yürütülmesinden sorumludur ve dört öncelik düzeyinin tüm sıraları boşalana dek çalışmaya devam etmektedir. Bu metot önce gerçek zamanlı sırada herhangi bir işlem olup olmadığını kontrol etmektedir. Varsa, o sıradaki bir sonraki işlemi çalıştırmaktadır. 'REAL_TIME' yani gerçek zamanlı, en öncelikli sıra boşsa 'PRIORITY_1' sırasını denetlemekte ve bu sıradaki bir sonraki işlemi çalıştırmaktadır. 'PRIORITY_1' sırası da boşsa,

‘PRIORITY_2’ sırasını denetlemekte ve bu sıradaki bir sonraki işlemi çalıştırmaktadır. Tüm bu sıralar boşsa, ‘PRIORITY_3’ sırasından işlemleri çalıştırmak için ‘**Round-Robin (RR, Çevrimsel Sıralı)’** algoritmasını kullanmaktadır. RR bir işleme algoritmasıdır. Bu algoritma, bir işlem listesi içindeki işlemleri döngü şeklinde işler ve her bir işlem için bir düzenli zaman dilimi ayarlar. Bu zaman dilimi sona erdikten sonra, işlemin tamamlanıp tamamlanmadığına bakılır ve eğer tamamlanmadıysa, işlem listesinin sonuna geri eklenir ve bir sonraki işleme geçilir. Bu şekilde, işlem listesi döngü şeklinde işlenir ve her bir işlem için belirlenen zaman dilimi süresi kadar işlem yapılır. Bu programda RR algoritması, bir sonraki işlemi ‘priority3Queue’ kuyruğundan almayı, sabit bir süre (kuantum olarak adlandırılır) çalıştırmayı ve daha sonra henüz bitmemişse kuyruğa geri koymayı içermektedir. İşlem, ‘PRIORITY_3’ sırasındaki tüm işlemler tamamlanana kadar tekrarlanmaktadır.

‘runProcess()’ metodu, bir işlemi o işlem tamamlanana kadar çalıştırmaktan ve geçerli saat ile önceki işlem değişkenlerini güncellemekten sorumludur. ‘runProcessForQuantum()’ metodu buna benzerdir, ancak işlemi yalnızca belirtilen kuantum zamanı için çalıştırmakta ve çalıştırdıktan sonra işlemin bitip bitmediğini kontrol etmektedir. Tamamlanmazsa, sıraya geri eklenir. Programda prosesler, her benzersiz proses için rastgele oluşturulmuş bir renk şeması kullanılarak yazdırılmaktadır. ‘runProcess()’ ve ‘runProcessForQuantum()’ metotları içinde yer alan renk şemaları bulunmaktadır. Bunun için ‘java.util.Random’ sınıfını kullanılmış ve rastgele RGB renk kodu üretilmiştir.

TARTIŞMA VE ÖNERİLER

Dört seviyeli öncelikli proses görevlendiricisi şeması, işletim sistemlerinde kullanılan bir proses görevlendirme yöntemidir. Bu şema, prosesleri dört farklı öncelik seviyesine göre ayırır ve her seviyedeki proseslerin çalıştırılma sırasını belirler. Öncelik seviyesi yüksek olan prosesler, öncelik seviyesi düşük olan proseslerden önce çalıştırılır.

Dört seviyeli öncelikli görevlendirici şemasının avantajları şunlardır:

- Öncelik seviyesi yüksek olan prosesler çabuk çalıştırılır ve bu sayede sistem daha hızlı bir şekilde çalışır.

- Öncelik seviyesi düşük olan prosesler çalıştırılırken, öncelik seviyesi yüksek olan prosesler için vakit ayırılır ve bu sayede öncelik seviyesi yüksek olan prosesler daha az gecikme ile çalıştırılır.
- Bu şema sayesinde, öncelik seviyesi yüksek olan prosesler için daha fazla kaynak ayırılır ve bu prosesler daha hızlı bir şekilde çalışır.

Ancak avantajlarının yanında aşağıdaki gibi bazı dezavantajları da bulunmaktadır:

- Öncelik seviyesi düşük olan prosesler uzun süreler bekleyebilir ve bu proseslerin çalışması gecikebilir.
- Bu şema, öncelik seviyesi yüksek olan prosesler için daha fazla kaynak ayırır ve bu da diğer proseslerin kaynak kullanımını azaltabilir.
- Bu şema, öncelik seviyesi düşük olan prosesleri çalıştırırken, sistem performansını düşürebilir ve bu proseslerin çalışmasını yavaşlatabilir.

Ortaya çıkan eksiklikleri gidermek için, görevlendiricinin nasıl çalıştığını ve hangi işlemlerin ne zaman çalıştırılması gerektiğini anlamak önemlidir. Öncelikle sistemdeki yazılım hatalarının tespiti ve düzeltilmesi de önem arz eder. Eksiklikleri gidermedeki en önemli nokta ise sistemdeki işlemlerin önceliklerinin doğru bir şekilde belirlenmesi, sistemdeki kaynakların ve belleğin düzgün bir şekilde yönetilmesidir. Bir işletim sisteminde, kaynak ve bellek ayırma şemaları, sistemdeki kaynakların ve belleğin daha etkin bir şekilde yönetilmesini sağlar. Bu sayede, sistem daha verimli bir şekilde çalışır ve daha fazla iş yükünü yönetebilir.

Kaynak ayırma şemaları, sistemdeki kaynakların (örneğin, disk alanı, CPU zamanı veya çıktı cihazları) daha verimli bir şekilde dağıtılmasını sağlar. Örneğin, bir işletim sistemi, CPU zamanını daha önemli işlemler için öncelikli olarak ayırabilir ve daha az önemli işlemler için daha az öncelikli olarak ayırabilir. Bu sayede, önemli işlemler daha hızlı bir şekilde tamamlanır ve sistem daha verimli bir şekilde çalışır. Kaynak ayırma şemaları, özel bir uygulamanın ihtiyaç duyduğu kaynak miktarını azaltarak diğer uygulamalara daha fazla kaynak ayırabilir. Bu durumda, özel bir uygulamanın performansı düşebilir. Özel bir uygulamanın çalışması sırasında diğer uygulamaların çalışmasını yavaşlatabilir. Sistemde çok fazla uygulama olduğu zaman zorluk yaşanabilecek bir durum yaratabilir.

Bellek ayırma şemaları ise, sistemdeki belleğin daha etkin bir şekilde yönetilmesini sağlar. Örneğin, bir işletim sistemi, bellekte çalışan işlemlerin bellekte tutulma sürelerini düzenleyebilir ve daha az kullanılan işlemlerin bellekten çıkarılmasını sağlayabilir. Bu sayede, sistem daha fazla bellek alanına sahip olur ve daha fazla iş yükünü yönetebilir. Bellek ayırma şeması, özel bir uygulamanın ihtiyaç duyduğu bellek miktarını azaltarak diğer uygulamalara daha fazla bellek ayırabilir. Bu durumda, özel bir uygulamanın performansı düşebilir. Bellek ayırma şeması, özel bir uygulamanın çalışması sırasında diğer uygulamaların çalışmasını yavaşlatabilir. Sistemde çok fazla uygulama olduğu zaman zorluk yaşanabilecek bir durum yaratabilir.

Sonuç olarak, kaynak ve bellek ayırma şemaları, bir işletim sisteminde kaynakların ve belleğin daha verimli bir şekilde yönetilmesini sağlar ve bu sayede sistem daha verimli bir şekilde çalışır. Genel resme bakıldığında avantajlarının dezavantajlarından daha önde olduğu kanısına ulaşılır.

Projenin Github Linki: https://github.com/thenghn/1.Grup