T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ
İŞLETİM SİSTEMLERİ

GÖREVLENDİRİCİ(DİSPATCHER) KABUĞU
PROJE RAPORU

Mahir KEZER G201210019
Kerim Emre KURT G181210057
Merve KARTAL b211210557
Veysel Can ŞAHİN b211210382
Mahmut Yiğit AYKOL g201210350

GÖREVLENDİRİCİ KABUĞU PROGRAM YAPISI

Bu program dört seviyeli öncelikli bir proses görevlendiricisi içeren karmaşık bir çoklu programlama sistemini tanımlamaktadır. Bu sistem, gerçek zamanlı prosesler ve normal kullanıcı prosesleri için ayrı kuyruklar kullanarak, her bir prosesi öncelik düzeyine göre işler.

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.

Programın adımları, anahtar kelimeleri ve ne yaptığı,

-Proses Görevlendiricisi:** Sistem, iki temel proses kuyruğuna sahip: gerçek zamanlı prosesler için bir kuyruk ve normal kullanıcı prosesleri için bir kuyruk. Bu kuyruklar, proseslerin varış zamanlarına ve önceliklerine göre düzenlenir

-Gerçek zamanlı prosesler, kesintisiz yürütülür ve en yüksek öncelik seviyesine sahiptir. Bu prosesler, tamamlanana kadar diğer tüm proseslerin işlenmesini durdurur.

-Kullanıcı prosesleri, üç seviyeli geri beslemeli bir görevlendirici tarafından işlenir. Bu prosesler, belirli bir zaman kuantumu (1 saniye) için çalıştırılır ve sonra durdurulur. Eğer bir proses tamamlanmamışsa, bir sonraki öncelik seviyesine düşürülür ve sıra tekrar geldiğinde işlenmeye devam eder.

- En düşük öncelik seviyesindeki tüm prosesler için basit bir Round Robin algoritması kullanılır. Bu, her prosese eşit miktarda işlem süresi tahsis eder.

-Prosesler, varış zamanları, öncelikleri, gereken işlem süreleri, bellek blok boyutları ve talep edilen diğer kaynaklarla tanımlanır. Prosesler, gereken kaynaklar mevcut olduğunda işlenmeye hazır hale gelir.

-Kaynak Yönetimi:** Sistemin belleği 1024 MB'dır ve gerçek zamanlı prosesler için her zaman en az 64 MB bellek ayrılır. Kullanıcı prosesleri, sistem kaynaklarını (yazıcılar, tarayıcı, modem, CD sürücüler) kullanabilir. Her proses, kullanacağı kaynakları başlangıçta belirtir ve bu kaynaklar proses tamamlanana kadar sadece o prosese ayrılır.

-Her proses, kendine özgü rastgele bir renk şeması ile görselleştirilir. Bu, proseslerin izlenmesini ve tanımlanmasını kolaylaştırır.

- Bir proses işlendiğinde, kimliği, öncelik seviyesi, kalan işlem süresi, bellek konumu ve blok boyutu gibi bilgiler gösterilir. Proses sonlandığında, kullanılan kaynaklar bir sonraki proses için yeniden tahsis edilir.

-Eğer proses listesinde, giriş kuyruklarında ve geri besleme kuyruklarında işlenecek başka proses kalmazsa, görevlendirici sona erer.

Ana sınıflar ve içerikleri:

`Process` ve `Dispatcher`. İşte her iki sınıfın bileşenleri ve işlevleri:

-`Process` Sınıfı
Bu sınıf, görevlendiricideki bir işlemi temsil eder ve aşağıdaki özelliklere sahiptir:

1. **id (Kimlik):** Prosesin benzersiz kimliği. Her yeni proses oluşturulduğunda, `Dispatcher.idCounter` kullanılarak arttırılır.
2. **arrivalTime (Varış Zamanı):** Prosesin görevlendiriciye ne zaman vardığını belirtir.
3. **priority (Öncelik Düzeyi):** Prosesin öncelik seviyesini tanımlar.
4. **burstTime (İşlem Süresi):** Prosesin tamamlanması için gereken süreyi ifade eder.
5. **startingBurstTime (Başlangıç İşlem Süresi):** Prosesin ilk oluşturulduğu andaki orijinal işlem süresi.

Bu sınıf, bu değişkenlerin değerlerini alarak ilgili özelliklere atar ve metodlar sayesinde bu özelliklere erişim sağlar, işlem süresini değiştirir.

-`Dispatcher` Sınıfı
Bu sınıf, önceliğe dayalı zamanlamayı kullanan bir görevlendiriciyi temsil eder. Özellikleri ve metodları şunlardır:

1. **Öncelik Düzeyleri:** Dört öncelik düzeyi bulunmaktadır: `REAL_TIME`, `PRIORITY_1`, `PRIORITY_2`, ve `PRIORITY_3`.
2. **run() Metodu:** Öncelik sıralarındaki işlemleri belirli bir sırayla çalıştırır. Bu metot, en yüksek öncelikten en düşüğe doğru işlem sıralarını kontrol eder ve her birini FCFS veya RR algoritmalarına göre çalıştırır.
3. **Round-Robin (RR) Algoritması:** `PRIORITY_3` sırasındaki işlemleri belirlenen bir zaman dilimi (kuantum) için çalıştırır ve sonra durdurur. Eğer işlem tamamlanmadıysa, sıranın sonuna eklenir.
4. **runProcess() ve runProcessForQuantum() Metotları:** Bir işlemi belirli bir süre veya tamamlanana kadar çalıştırır ve işlemin bitip bitmediğini kontrol eder. Tamamlanmazsa, işlem sıraya geri eklenir.

Ek olarak, programda prosesler, her biri için rastgele oluşturulmuş renk şemaları ile görselleştirilmektedir. Bu renkler, `java.util.Random` sınıfı kullanılarak rastgele RGB renk kodları üreterek oluşturulur.

TARTIŞMA VE ÖNERİLER

Dört seviyeli öncelikli proses görevlendiricisi, işletim sistemlerinde kullanılan ve prosesleri dört farklı öncelik seviyesine göre sıralayan bir yöntemdir. Bu sistem, yüksek öncelikli prosesleri hızlıca çalıştırarak genel sistem verimliliğini artırır. Ancak, bu durum bazen düşük

öncelikli proseslerin uzun süre beklemesine ve kaynak dağılımında dengesizliklere yol açabilir.

Öncelikli görevlendiricinin avantajları, özellikle acil ve önemli işlemlerin hızlı bir şekilde tamamlanması açısından önemlidir. Bu, kritik uygulamaların gecikmesiz çalışmasını sağlayarak genel sistem performansını olumlu yönde etkiler. Öte yandan, düşük öncelikli proseslerin gecikmesi, kaynakların adil olmayan bir şekilde dağılımı gibi sorunlara yol açabilir. Bu durum, özellikle çok sayıda düşük öncelikli işlemin biriktiği durumlarda sistem performansını düşürebilir.

Bu eksiklikleri gidermek için, proseslerin önceliklerinin daha dengeli ve dinamik bir şekilde belirlenmesi önemlidir. Sistemin mevcut durumuna ve iş yüküne göre önceliklerin ayarlanması, daha esnek bir kaynak yönetimi sağlayabilir. Ayrıca, kaynak ve bellek ayırma şemalarının etkin kullanımı, sistemin toplam verimliliğini artırabilir. Örneğin, CPU zamanı veya bellek gibi kaynakların dinamik olarak tahsis edilmesi, bir uygulamanın gereksinimlerine göre ayarlanabilir. Bu, özellikle çok sayıda uygulamanın eş zamanlı olarak çalıştığı durumlarda sistemin genel performansını iyileştirebilir.

Bellek yönetiminde ise, daha az kullanılan işlemlerin bellekten çıkarılması ve aktif işlemlere daha fazla bellek ayrılması, sistemde daha fazla iş yükünün yönetilmesine olanak tanır. Bellek ayırma şemalarının uygulanması, özellikle uzun süre çalışan veya yoğun bellek kullanımı gerektiren uygulamalar için önemlidir.

Sonuç olarak, dört seviyeli öncelikli proses görevlendiricisi, işletim sistemlerinin verimliliğini ve yanıt süresini iyileştirme potansiyeline sahiptir. Ancak, bu yaklaşımın etkinliği, kaynakların ve belleğin dengeli yönetilmesine ve dinamik öncelik atamasına bağlıdır. Bu faktörler dikkate alındığında, öncelikli görevlendiricinin avantajları, karşılaşılan dezavantajları aşabilir ve sistem performansını genel olarak iyileştirebilir.

GitHub linki: https://github.com/KerimEmre/IsletimSistemleriGrup23