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

DERS ADI : İŞLETİM SİSTEMLERİ

GRUP NUMARASI : 48

KONU : GÖREVLENDİRİCİ KABUĞU

ÜYELER : AYŞENUR YILDIRIM G211210007
ÖMER FARUK KOCAMAN G221210337
MUHAMMED İHSAN KARAYEĞİT G211210023
VEDAT DOĞAN G211210053
HASAN ÇAKIR B211210018

GITHUB : https://github.com/vedatdogann/IS-Proje

Kullanılabilecek Bellek Tahsis Algoritmaları

FCFS (First-Come, First-Served):

FCFS algoritması, işlemcide çalışacak işlemleri sırayla işleme koyan basit bir işlem planlama algoritmasıdır. İlk gelen işlem (ilk sıraya giren) ilk olarak işlenir, sonra sıradaki işlem işlenir ve bu böyle devam eder.

Bu algoritma, işlemcide uzun süreli işlemleri önce işlerken, kısa süreli işlemleri sıraya alır ve bu nedenle "adil" bir algoritma olarak kabul edilmez.

FCFS'nin avantajı, uygulanması ve anlaşılması kolaydır. Ancak, işlem sürelerinin çok farklı olduğu durumlarda kötü performans gösterebilir. Özellikle kısa işlemlerin uzun işlemlerin önünde beklemesi (sıranın sonunda beklemesi) durumu ortaya çıkabilir.

Round Robin(RR):

Round Robin (RR), tüm işlemleri eşit olarak işlemcide paylaştırmayı amaçlayan bir zaman paylaşımı işlem planlama algoritmasıdır. Her işlem belirli bir zaman dilimi veya "zaman dilimi" (time quantum) içinde işlemcide çalıştırılır.

İşlemcideki her işlem, sıradan çıkıp sıradaki işlem sıraya girmesi gereken bir dönem sonunda sıraya tekrar döner. Bu döngü devam eder.

RR'nin avantajı, kısa işlemlerin hızlı bir şekilde işlenmesini sağlaması ve işlemcideki işlemler arasında adil bir paylaşım yapmasıdır. Ancak, uzun süreli işlemler için bekleme süreleri artabilir ve işlemci zamanı zaman dilimine (time quantum) bağlı olarak ayarlanmalıdır.

RR algoritması, çoklu kullanıcı sistemleri ve çoklu işlemcili sistemler gibi çoklu görevli ortamlarda yaygın olarak kullanılır.

First-Fit: İlk uygun boşluğa bellek tahsis eder.

Best-Fit: En küçük uygun boşluğu seçer.

Worst-Fit: En büyük boşluğu seçer.

Buddy System: Belleği sabit boyutlu bloklara bölerek yönetir.

First-Fit algoritmasına basit ve hızlı olduğu için başlangıç için iyi bir seçenektir. Daha karmaşık senaryolar için, Best-Fit veya Buddy System gibi algoritmalar daha verimli olabilir.

Bellek ve Kaynakları Yönetme Yapıları

1. Kuyruklar (Queues)

Main sınıfında iki kuyruk tanımlanmıştır: shortTermQueue ve longTermQueue. Bu kuyruklar, MyProcess nesnelerini saklamak için kullanılır.

Kullanım: Süreçler (MyProcess nesneleri), belirli özelliklerine (örneğin, çalışma süresine) göre bu kuyruklara yerleştirilir. Kısa süreli süreçler shortTermQueue'ya, uzun süreli süreçler ise longTermQueue'ya eklenir.

Amaç: Bu kuyruklar, süreçlerin işlenme sırasını düzenler ve önceliklendirir. Kısa süreli işlemler hızlı bir şekilde tamamlanabilirken, uzun süreli işlemler daha uzun bir bekleme süresine sahip olabilir.

2. MyProcess Sınıfı

Tanım: Her MyProcess nesnesi, bir sürecin temel bilgilerini içerir: processId, cpuUsage, memoryUsage ve state.

Kullanım: Bu sınıf, sürecin CPU ve bellek kullanımını temsil eder ve sürecin durumunu (yeni, çalışan, bekleyen, sonlandırılmış) yönetir.

Amaç: MyProcess, bellek ve CPU kaynaklarının ne kadarının ve hangi süreç tarafından kullanıldığını takip etmek için kullanılır.

3. Resource Sınıfı

Tanım: Resource sınıfı, CPU kapasitesi ve bellek kapasitesi gibi kaynakları temsil eder. Ayrıca, bu kaynağın bir sürece tahsis edilip edilmediğini (isAllocated) takip eder.

Kullanım: Bir sürecin (MyProcess) kaynak kullanımı, Resource nesnesinin kapasitesiyle karşılaştırılır. Eğer kaynak yeterliyse ve daha önce tahsis edilmemişse, sürece tahsis edilir.

Amaç: Resource sınıfı, kaynakların yönetimi ve tahsisatını sağlar. Bu, kaynakların etkili ve verimli kullanımını sağlar ve çakışmaları önler.

4. Node Sınıfı

Tanım: Node sınıfı, ikili ağaç yapısında MyProcess nesnelerini saklar.

Kullanım: Süreçler, addNode metodu kullanılarak ikili ağaca eklenir. Bu, süreçlerin verimli bir şekilde saklanması ve sıralanması için kullanılır.

Amaç: Node sınıfı, süreçlerin hızlı arama ve yönetimi için bir yapı sağlar.

Bu yapılar, bir işletim sistemi benzeri ortamda süreçlerin yönetimi, bellek ve kaynak tahsisatı için birlikte çalışır. Süreçler, özelliklerine göre sıralanır ve uygun kaynaklarla eşleştirilir. Bu modüler ve etkileşimli yapı, süreçlerin verimli yönetilmesini ve kaynakların optimal kullanımını sağlar.

Programın Genel Yapısı

1. Main

İşlevi: Uygulamanın giriş noktası olarak, süreçleri başlatır ve bunları uygun kuyruklara yerleştirir.

Rolü: Main sınıfı, süreçleri oluşturur ve bunların kısa süreli veya uzun süreli olup olmadığına karar vererek shortTermQueue veya longTermQueue içine yerleştirir. Bu süreçler daha sonra işletim sistemi benzeri ortamda yönetilecektir.

2. MyProcess

İşlevi: Bir işletim sistemi sürecini temsil eder. Sürecin kimliği, CPU kullanımı ve bellek kullanımı gibi özellikleri içerir.

Rolü: MyProcess, süreçlerin durumunu (yeni, çalışan, bekleyen, sonlandırılmış) ve kaynak kullanımını takip eder. Bu sınıf, her bir sürecin işletim sistemi içinde nasıl davranacağını ve kaynakları nasıl kullanacağını belirler.

3. Node

İşlevi: İkili ağaç yapısında süreçleri saklamak için kullanılır. Her Node, bir MyProcess nesnesi ve sol/sağ alt dallara bağlantılar içerir.

Rolü: Node sınıfı, süreçlerin verimli bir şekilde saklanması ve sıralanması için ikili ağaç yapısını kullanır. Bu, süreçlerin aranmasını ve yönetilmesini kolaylaştırır.

4. Resource

İşlevi: Bellek ve CPU gibi sistem kaynaklarını temsil eder. Kaynakların bir sürece tahsis edilip edilmediğini takip eder.

Rolü: Resource sınıfı, kaynakların yönetimi ve tahsisatı için sorumludur. Bu, kaynakların verimli kullanımını sağlamak ve çakışmaları önlemek için önemlidir.

5. FileRead

İşlevi: Dosya okuma işlevlerini gerçekleştirir. Verilen dosya adından dosyayı okur ve içeriğini işler.

Rolü: FileRead sınıfı, dosya işleme ve veri girişi için kullanılır. Bu, programın dosya tabanlı verilerle çalışabilmesini sağlar ve genellikle konfigürasyon dosyaları veya kullanıcı girdilerini okumak için kullanılır.

6. Colors

İşlevi: Farklı renk kombinasyonlarını oluşturur. RGB değerlerini kullanarak bir renk listesi üretir.

Rolü: Colors sınıfı, grafiksel uygulamalar veya kullanıcı arayüzü bileşenleri için renk seçimi yapmak üzere tasarlanmıştır. Bu sınıf, renklerle ilgili işlemleri kolaylaştırır ve görsel öğeler için renk paleti oluşturmakta kullanılabilir.

Bu sınıfların her biri, genel programın farklı yönlerini yöneterek, işletim sistemi benzeri bir ortamın simülasyonunda birlikte çalışır. Bu yapı, modüler ve genişletilebilir bir tasarım sağlar.

Çok Düzeyli Görevlendirme Şeması

Kullanım Nedeni

Çok düzeyli görevlendirme şeması, süreçlerin önceliklerine göre yönetilmesini sağlar. Bu, gerçek işletim sistemlerinde de yaygın bir uygulamadır ve sistem kaynaklarının etkili kullanımını sağlar.

Eksiklikler ve İyileştirmeler

Şu anda, kaynak tahsisatı basit ve dinamik değil. Gerçek zamanlı kaynak yönetimi eksik.

Dinamik bellek tahsisatı ve daha gelişmiş kaynak yönetimi algoritmaları eklenmeli. Ayrıca, süreçler arası iletişim ve senkronizasyon mekanizmaları geliştirilebilir.

Bellek ve Kaynak Ayırma Şemaları

Dinamik bellek ayırma ve geri dönüşümlü bellek kullanımı gibi özellikler eklenebilir.

Kaynakların daha verimli kullanımı için önceliklendirme ve zaman dilimi yöntemleri geliştirilebilir.