SAKARYA
ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

İŞLETİM SİSTEMLERİ ÖDEVİ

Furkan YILDIZ (G201210014)
Mert ÇALIŞKAN(G211210086)
İrfan Eren ÇİFTÇİ(G211210032)
Enes Samet AYDI(G201210078)
Enes Kartancı(G201210021)

Bellek Tahsis Algoritmaları

- ## First Fit

First Fit, bellek tahsis algoritmalarından biridir ve çalışma zamanında bellek yönetimini düzenlemek için kullanılır. Bu algoritma, bellekteki ilk uygun boşluğa veriyi yerleştirme prensibine dayanır. Yani, bellekteki boşluklardan ilki bulunur ve bu boşluğa veri tahsis edilir. Bu algoritma, tahsis edilen bellek bloğunu hızlı bir şekilde bulup yerleştirdiği için basit ve hızlı bir yapıya sahiptir. Ancak, boşlukların optimize edilmesi konusunda etkisiz olabilir. Örneğin, küçük bir boşluğa sığacak bir veri varken, daha büyük bir boşluğa yerleştirme yapabilir ve bu durum bellek kullanımını optimize etmeyebilir. Yine de, basitliği ve düşük işlem maliyeti nedeniyle First Fit, birçok sistemde tercih edilen bir bellek tahsis stratejisi olabilir.

- ## Best Fit

  Best Fit bellek tahsis algoritması, çalışma zamanında bellek yönetimi için kullanılan bir stratejidir. Bu algoritma, bellekteki en küçük uygun bloğa veriyi yerleştirme prensibine dayanır. Yani, mevcut boşluklar arasından en küçük olanına veri tahsis edilir. Bu, bellek kullanımını optimize etme amacını taşır, çünkü küçük boşlukların kullanılması büyük boşlukları bırakmaktan daha az kaynak israfına neden olabilir. Ancak, bu avantajın yanı sıra, Best Fit'in işlemleri sıralama ve uygun boşluğun aranması nedeniyle daha fazla zaman alabilir. Bu nedenle, işlemleri optimize etmek ve bellek kullanımını iyileştirmek arasında bir denge kurma ihtiyacı vardır. Best Fit, genellikle dinamik bellek tahsisini yöneten işletim sistemleri ve programlama dillerinde kullanılır, ancak performans önceliklerine bağlı olarak diğer tahsis stratejileri de tercih edilebilir.

- ## Worst Fit

  Worst Fit bellek tahsis algoritması, bir programın çalışma zamanında bellek yönetimini sağlamak amacıyla kullanılan bir stratejidir. Bu algoritma, bellekte en büyük uygun bloğa veriyi yerleştirir. Yani, veri tahsis edilmeden önce mevcut boşluklar arasında en fazla bellek alanını bırakan boşluğa odaklanır. Worst Fit'in avantajı, büyük veri kümelerinin saklanmasında etkili olabilmesidir. Ancak, bu strateji genellikle bellek kullanımını optimize etme konusunda diğer stratejilere göre daha az etkilidir. Çünkü büyük boşluklar bırakarak, bellek kullanımında verimsizliklere yol açabilir. Worst Fit, genellikle daha az talep gören bir bellek tahsis algoritması olarak kabul edilir ve performans açısından diğer stratejilere göre geride kalabilir.

- **Next Fit**

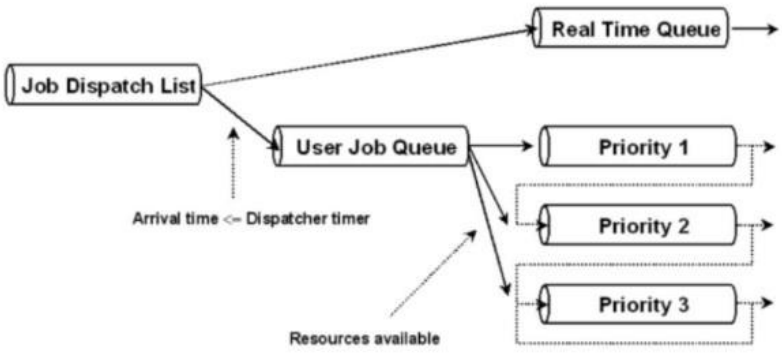
Next Fit, bellek tahsisinde kullanılan bir algoritmadır ve işlemci tarafından işlenen verilerin bellekteki uygun bir boşluğa yerleştirilmesini sağlar. Bu algoritma, veriyi en son yerleştirilen veriden sonraki ilk uygun boşluğa yerleştirir. Yani, bellekteki boşlukları sırasıyla kontrol eder ve ilk uygun boşluğu bulduğunda veriyi oraya yerleştirir. Bu, bellek yönetimini basitleştirir ve hızlı bir şekilde bellek tahsisi yapılmasına olanak tanır. Ancak, bu yaklaşımın bazı dezavantajları vardır. Özellikle büyük bloklar arasında küçük boşluklar bırakabilir, bu da bellek kullanımını optimize etme konusunda diğer algoritmalar kadar etkili olmayabilir. Next Fit, basitliği ve işlemlerin bellek içindeki sıralı olarak yerleştirilmesi avantajlarıyla bilinir.

Kullanılan Bellek Tahsis Algoritması

İlk olarak Best Fit algoritmasını kullanmayı düşündük. Fakat yaptığımız araştırmalar sonucu Best Fit algoritmasının , First Fit algoritmasına göre daha iyi bellek tahsisatı yapmasına rağmen , First Fit algoritmasına göre daha yavaş çalıştığını fark ettik ve performans kaybını minumuma indirmek için projemizde First Fit algoritmasını kullanmayı tercih ettik.

Görevlendirici Yapısı

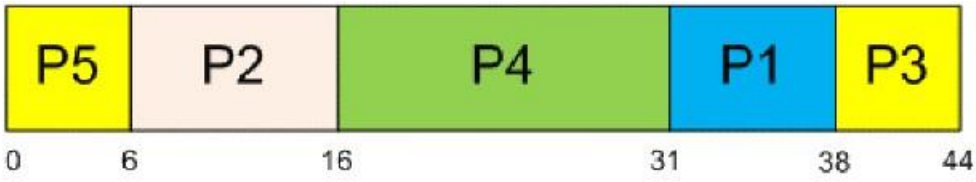
Dispatcher sınıfının run methodunda proccess listesinde ki her bir proccess in Sort Proccess methodu ile kuyruğa alınması sağlanmaktadır. Kuyruğa aldığımız her bir proccess için Load Proccess in to Memory methodu ile yardımcı diğer methodlara erişim sağlıyoruz. Yardımcı methodlardan Kaynak Kısıt methodu bize proccess in çalışması için yeterli kaynak olup olmadığını kontrol ediyor. Bir diğer yardımcı method Fits at Memory bize bellekte yeteri kadar yer olup olmadığına göre proccess i kuyruğa alır. Run methodunun içindeki do , while ve if blokları içinde proccessler arasındaki geçiş sağlanır. Bu geçişe de yardımcı olarak da parametre olarak gönderilen Scheduler sınıfından kalıtım alan çocuk sınıfların addProccess ve getNextProccess methodlarını kullanır.



(Görevlendirici Mantık Akışı)

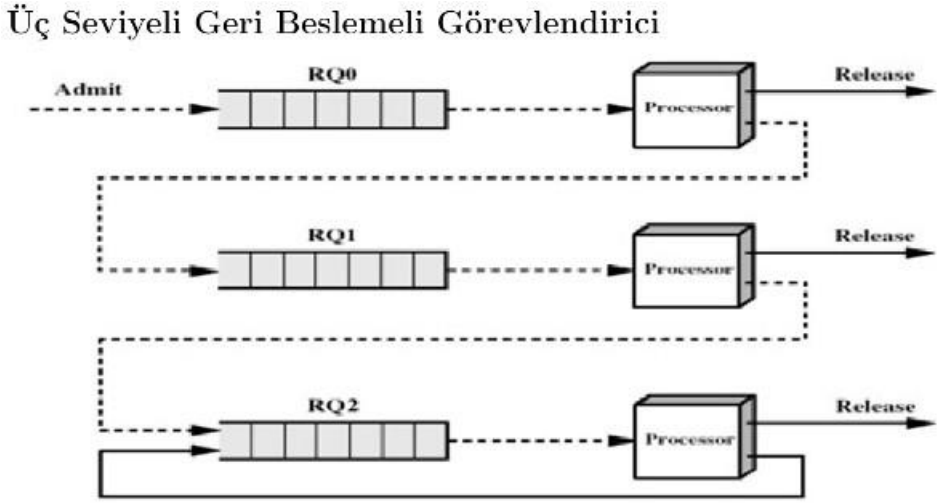
Proje'nin Genel Yapısı

- **Proccess Control Block**: Burada proccess'in durumu , başlama ve bitiş zamanları , pid değeri , toplam çalıştığı zaman değeri tutulmaktadır.
- **Proccess State** : Burada proccess'in bulunacağı olan dört durum(NEW , READY , RUNNİNG , TERMINATED) değerleri tutulmaktadır.
- **Proccess** : Burada proccess control block değerlerinide kullanarak proccess'in varış zamanı , patlama zamanı , bellek kullanımı , kullandığı printer , modem , tarayıcı , cd ve öncelik değerleri tutulmaktadır. Proccess eğer çalışıyorsa durumunun ready durumundan running durumuna , çalışmıyorsa proccess'in bitip bitmediğine göre proccess durumunun terminated ya da ready durumuna getirilmesi işlemleri yapılmaktadır. Proccess'in bekleme zamanı cevap verme zamanı ve bir sonraki çalışma zamanı değerleri hesaplanmaktadır.
- **FCFS** : Burada CPU çizelgeleme algoritmalarından olan FCFS algoritması gerçeklenmiştir. Kuyruktaki ilk proccess in çalışıp kuyruktan atıldıktan sonra yeni proccess e geçme esasına dayanır.

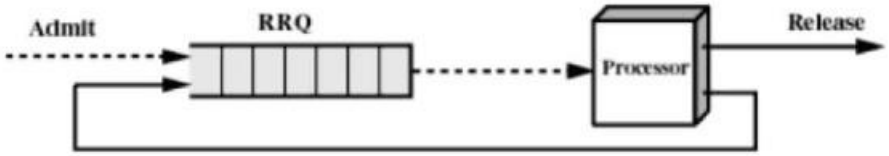


Bknz (FCFS Algoritması Gantt Diyagramı)

- **Üç Seviyeli Geri Besleme Kuyruğu**: Öncelik değeri bir den büyük olan proccesslerin üç seviyeli geri besleme kuyruğuna atılarak , en yüksek öncelikli kuyrukta proccesslerin birer saniye ççalıştırılıp eğer bitmedilerse daha düşük öncelikli kuyruğa geçmeleri sağlanmıştır. Eğer hala bitmeyen proccess var ise o proccess round robin kuyruğuna atılmıştır.



- **Round Robin** : Üç seviyeli kuyrukta bitmeyen proccesslerin round robin algoritmasına göre kuantum zamanı bir saniye olacak şekilde çalıştırılıp proccesslerin tamamlanması sağlanmıştır ve round robin kuyruğundan atılmıştır.



Bknz (Round Robin Yapısı)

- **Hybrid Schedular** : Burada proccesslerin öncelik değerine ve bitip bitmediğine göre gerekli cpu çizelgeleme algoritmalarına aktarılması yapılmıştır.
- **First Fit Algoritması**: Burada bellek tahsis algoritmasından kalıtım alarak içine gönderdiğmiz proccess in first fit algoritması ile bellekteki en uygun boşluğa yerleşmesini sağlıyoruz.
- **Memory Managment Unit** : Burada parametre olarak bir bellek tahsis algoritması isteyen ve bu algoritmanın proccess i belleğe yazma işlemlerini ve proccess sonlandırıldıktan sonra belleğin serbest bırakılmasını sağlıyoruz. Ayrıca sistem kısıtlarını da burada belirliyoruz.
- **Dispatcher**: Burada projede kullanılan tüm sınıfların birleştiği ayrı bir sınıftır. Proccessler ilk olarak buraya gelir. Sort Proccess methodu ile sıraya konulur. Fits at Memory ve Kaynak Kısıt methodları ile proccesslerin sistemde çalışıp çalışamayacağı kontrol edilir. Tick methodu ile clock pulse değeri hesaplanır. Run methodu ile belirlenen algoritmalara göre proccessler sırayla işlenir.

- **Main** : Dosyadan okunan değerleri her bir proccess e ayrı ayrı işlediğimiz ardından gerekli sınıflar new lenerek dispatcher ın run methodunun çalıştığı daha sonra gerekli çıktılarımızın alındığı programın ana sınıfıdır.

Olası İyileştirmeler

İlk olarak gerçek zamanlı prosesleri ele aldığımızda, FCFS algoritmasına göre çalıştıklarından sistemi timeout süresi kadar meşgul edebilirler. Bu durumda periyodik yapılabilecek çağrılar sistemin akışını bozup sistemi kullanılamayacak hale getirebilir. Bu durumu engellemek için yüksek öncelikli prosesleri de proses geri besleme görevlendirici kuyruğuna alıp daha fazla zaman tanımak ve kuyrukta direkt ilk sıraya alınması gibi yöntemler denenebilir. Proses geri besleme görevlendiricisindeki kuantum (q) süresi sabit 1 sn olmamalıdır. Proseslerin kalan zamanlarına ve sistemin çalışma hızına göre dinamik olarak değişen bir değerde olmalıdır. Gerçek işletim sistemlerinde olan Olay (Event) bekleyicileri sisteme dahil edilmelidir. Örneğin, proseslerden biri, bir input bekliyor ise işlemciyi meşgul etmemeli olay gerçekleşene kadar bekletilmelidir. Ayrıca yedekleme gibi aciliyeti olmayan prosesler işlemcinin boş kaldığı zamanlarda devreye alınmalı ve yeni prosesler geldiğinde tekrar uyku moduna geçecek şekilde bir algoritma eklenmelidir.

GİTHUB LİNKİ

https://github.com/ESA19/OS-Project