OCAK 08, 2023

SAKARYA
ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

İŞLETİM SİSTEMLERİ PROJE RAPORU

HAZIRLAYANLAR:

Nur Çağlı	B201210350
Ayça Gecü	B201210094
Yiğit Ağar	G201210088
Hikmet Saylam	G191219556

https://github.com/aycagecu/OperatingSystemsProject

Proje Aşamaları:

Tasarım:

Projemize ilk başta görevlendirici nedir, nasıl bir algoritma ile çalışır , FCFS, Çevrimsel Sıralı nedir gibi sorular eşliğinde yerli ve yabancı kaynaklardan bilgi toplayarak başladık. Bilgi toplama işlemi bittikten sonra kullanmamız gereken algoritmayı tartışıp görevlendirme kısmına geçtik.

Görevlendirici (Dispatcher) :

İşletim sistemleri bağlamında, görevlendirici, işletim sisteminin çekirdeğinin,
işlemciyi (veya işlemcileri) yürütülmeyi bekleyen farklı işlemlere veya iş parçacıklarına tahsis etmekten sorumlu olan bir parçasıdır. Görevlendirici, hangi işlemin veya iş parçacığının daha sonra yürütülmesi gerektiğine karar vermek için çeşitli algoritmalar kullanır. Bu algoritmalar, sürecin önemi, yürütülmesi için beklediği süre veya ihtiyaç duyduğu kaynak miktarı gibi farklı faktörlere öncelik verecek şekilde tasarlanabilir.

İlk Gelen İlk Çalışır (FCFS) :

İşlemlerin varış zamanlarına göre programlanan en basit CPU planlama algoritması . FCFS algoritması, CPU'yu ilk isteyen işleme önce CPU'nun tahsis edildiğini belirtir. FIFO kuyruğu kullanılarak uygulanır . Bir işlem hazır kuyruğuna girdiğinde, PCB'si (Prosese ait pid vb.) kuyruğun kuyruğuna bağlanır. CPU boş olduğunda, kuyruğun başındaki işleme tahsis edilir. Çalışan işlem daha sonra sıradan kaldırılır.

Özellikleri:
1)FCFS, önleyici olmayan ve önleyici CPU zamanlama algoritmalarını destekler.
2)Görevler her zaman İlk gelen alır konseptine göre yürütülür.
3)FCFS'nin uygulanması ve kullanımı kolaydır.
4)Bu algoritma performansta pek verimli değildir ve bekleme süresi oldukça fazladır

Program
Proses:
Process classımızdır içinde askıya alındığı zamanı,geliş zamanını,durumlarını,proseslere özgü renkleri ve kalan zamanlarını tutar.İçinde javanın proses classından oluşan bir proses barındırır. Çalışıp durdururlurken o da başlatılır ve durdurulur.

Data Reader:
Bu sınıf bize verilen dosyadaki verileri okuduktan sonra onları bir proses listesine atar.
Bu proses listesi daha sonra fcfs ve userjob kuyrukları içine proses bilgilerini atama yapmamıza yardımcı olacak.

JobDispacth:
Bu sınıfta proses listesinden aldığımız prosesleri istenilen önceliğe göre fcfs veya userjob kuyruklarına atamasını yaparken bu atamayı geliş zamanlarını dikkate alarak yapmaktadır. Aynı zamanda programı çalıştıran bu sınıftır.

Statement

Statement enum, bir işlemin içinde olabileceği durumları temsil eder. Aşağıdaki değerlere sahiptir: New: Bu, sürecin yeni oluşturulduğunu gösterir. Ready: Bu, işlemin yürütülmeye hazır olduğunu gösterir. Running: Bu, işlemin şu anda yürütülmekte olduğunu gösterir. Terminated: Bu, işlemin yürütmeyi sonlandırdığını veya tamamladığını gösterir. TimeOut: Bu, işlemin zaman aşımına uğradığını, yani yürütme için izin verilen süre sınırını aştığını gösterir. Statement enum, bir işlemin mevcut durumunu takip etmek için SpecialProcess sınıfında kullanılır. SpecialProcess sınıfının getStatement () ve setStatement (Statement statement) yöntemleri, sırasıyla bir işlemin Statement almanıza ve güncellemenize olanak tanır.

Multilevel Queue:

RealTimeQueueda eleman yoksa, JobDispatch tarafından bu classa yönlendirilir,burada sırayla önce önceliği 1 olan kuyruğu sonra önceliği 2 olan kuyruğu en son da önceliği 3 olan kuyruğu kontrol eder.

Multilevel Queue'da Son Kuyruğun Round Robin Modunda Çalışması

Eğer queuelar içindeki proseslerin varış zamanı şu anki zamandan küçükse onları round robin modunda çalıştırır,değilse userjob queue da boş demektir ve yeni process gelene kadar bekler.

Karışık Sıralayıcı Çalışması

Görevlendirici çalışmaya başladıpında öncelikle gerçek zamanlı kuyruğu kkontrol edecektir.Eğer burada çalışacak proses bulunuyorsa öncelikle bu çalıştırılır.Burada çalışacak proses yoksa ya da çalışması sonlanmışsa kullanıcı kuyrukları kontrol edilir. Buradaki kuyruklardaki prosesler birer saniye çalıştırılıp alt kuyruga eklenecektir.Daha sonra her saniye bitişinde gerçek zamanlı kuyruk kontrolü yapılır ve eğer bu kuyruga proses eklenmişse programın çalışması buradan devam eder.

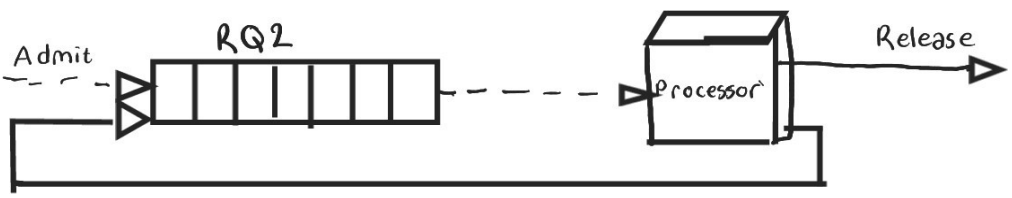
a)

Görevlendiricimiz Proses listesinden (giriş.txt) beslenen iki adet kuyruğa sahiptir bunlar gerçek zamanlı ve kullanıcı proses kuyruklarıdır.

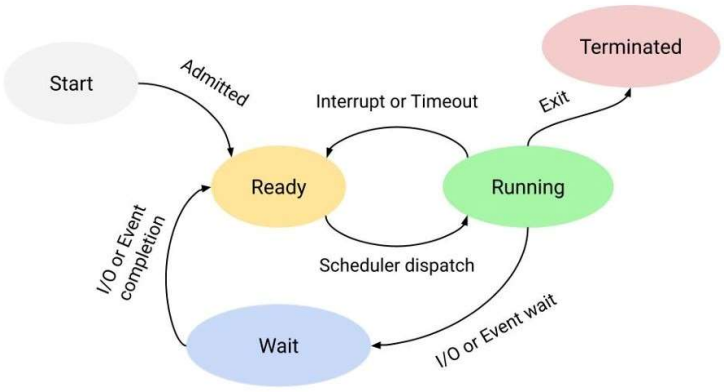
Proses listesi her zaman adımında işlemektedir.

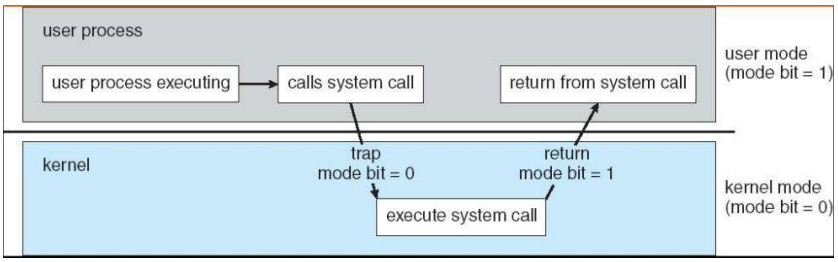
Gelen prosesleri gerçek zamanlı veya kullanıcı prosesleri olarak farklı kuyruklara alırız eğer gerçek zamanlı bir proses ise direk olarak çalışmaya başlar ve bitene kadar çalışmaya devam eder, gerçek zamanlı proses geldiğinde çalışan bir gerçek zamanlı proses varsa onun bitmesini bekler eğer ki çalışan proses kullanıcı prosesi ise kullanıcı prosesi kesilir ve gerçek zamanlı proses çalışmaya başlar ve bu döngü program sonuna kadar devam eder. Yani gerçek zamanlı proses bitene kadar kullanıcı prosesi çalışamaz eğer gerçek zamanlı proses bittiğinde sırada yine gerçek zamanlı proses varsa kullanıcı prosesi beklemeye devam eder. Kullanıcı prosesleri ise geri beslemeli kuyruğunda çalışır Burada ise öncelik sırası sırasıyla 1,2,3 olarak önceliklendirilen proseslerden geliş zamanına da bağlı olarak öncelik sırası yüksek olan 1 sn çalışır ve öncelik sırası Düşürülerek bir alt kuyruğa yerleştirilir.

Eğer tüm prosesler en alt seviye kuyrukta ise bu sefer bu kuyrukta basit çevrimsel sıralı (Round Robin) algoritması çalıştırır.

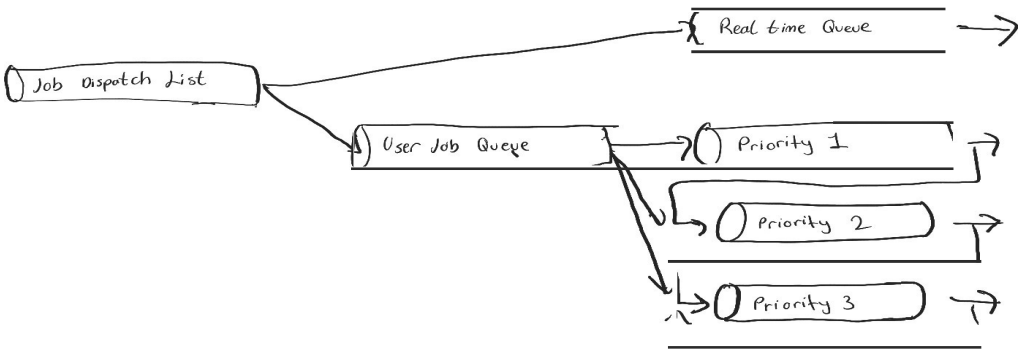


b) **Gerçek işletim sistemleri tarafından kullanılan şemalar**





Projedeki görevlendiricinin çalışma şeması



Tartışma:

Gerçek işletim sistemleri işlemlerini yürütmesi için bir görevlendirici içerir.Bir işlem çalışmaya hazır olduğunda, hazır işlemler kuyruğuna alınır. Görevlendirici, kuyruğun önündeki işlemi seçer ve CPU'yu bu işleme tahsis eder. Bu işlem, yürütmeyi bitirene veya bazı kaynakları beklerken bloke edilene kadar devam eder. İşlem yürütmeyi bitirdiğinde veya bloke edildiğinde, dağıtıcı kuyruktaki bir sonraki işlemi seçer ve CPU'yu buna tahsis eder.Projede kullandığımız JobDispatcher sınıfı, simüle edilmiş bir işletim sisteminde işlemlerin yürütülmesini yönetmekten sorumludur. Ve bunun için yüksek öncelikli gerçek zamanlı kuyruk ve çok seviyeli kullanıcı kuyruğu kullanılır. Her düşük öncelikli kuyruğa bir saniye tahsis edilmesi gerçek işletim sistemleri için kullanışlı bir durum değildir.İşletim sisteminin görevlerini daha düşük parçalara ayırarak çalıştırması daha efektif olacaktır.