**İŞLETİM SİSTEMLERİ DERSİ PROJE ÖDEVİ**

**Görevlendirici (Dispatcher) Kabuğu**

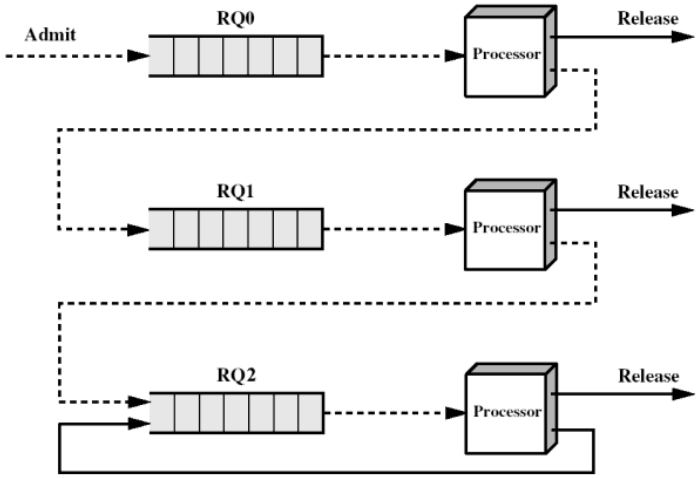
Projeye konu olan sistem sınırlı kullanılabilir kaynakların kısıtlamaları içinde çalışan dört seviyeli öncelikli proses görevlendiricisine sahip bir çoklu programlama sistemidir.

**DÖRT SEVİYELİ ÖNCELİKLİ GÖREVLENDİRİCİ**

Dağıtıcı dört öncelik düzeyinde çalışır:

1. İlk Gelen İlk Çalışır (FCFS) algoritması temelinde hemen çalıştırılması gereken gerçek zamanlı prosesler, daha düşük öncelikle çalışan diğer proseslerden daha yüksek bir önceliğe sahiptir (öncelik değeri 0). Bu prosesler tamamlanıncaya kadar kesilmeden yürütülür.

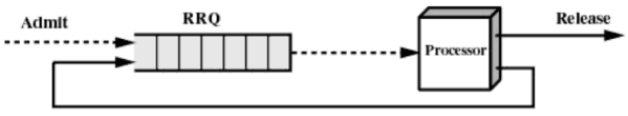
2. Normal kullanıcı prosesleri, üç seviyeli bir geri beslemeli görevlendiricide çalıştırılır. Dağıtıcının temel zamanlama kuantumu (q) 1 saniyedir. Bu aynı zamanda geri besleme sıralayıcısının zaman kuantumunun değeridir.



Şekil 1. Üç Seviyeli Geri Beslemeli Görevlendirici

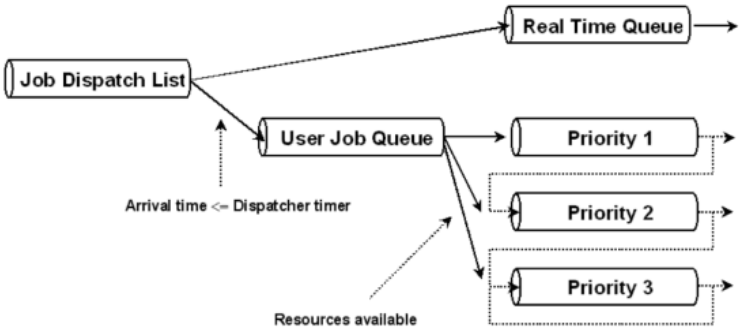
Görevlendirici, proses listesinden (giriş.txt) beslenen iki adet kuyruğa sahiptir: Gerçek Zamanlı ve Kullanıcı Proses kuyrukları. Proses listesi, her zaman adımında sürekli işlenir ve gelen prosesler uygun kuyruğa aktarılır. Kuyruklar işleme alınır; tüm gerçek zamanlı prosesler tamamlanmak üzere çalıştırılır ve o anda çalışmakta olan diğer düşük öncelikli prosesler kesilir.

Düşük öncelikli geri beslemeli görevlendirici yeniden etkinleştirilmeden önce gerçek zamanlı kuyruktaki prosesler bitirilmelidir. Kullanılabilir kaynaklar (bellek ve g/ç aygıtları) içinde çalışabilen kullanıcı kuyruğundaki tüm prosesler, uygun öncelik kuyruğuna aktarılır. Bir geri besleme kuyruğunun normal çalışması, en yüksek öncelik düzeyindeki prosesleri alır, uygun kuantuma göre işler ve daha sonra önceliğini düşürerek bir alt kuyruğa yerleştirir. Ancak her iş öncelik değerine uygun bir kuyruğa yerleştirilir (bknz Şekil 3). Eğer tüm prosesler en alt seviye kuyrukta ise o zaman basit bir çevrimsel sıralı (round robin) algoritma çalıştırılır (Şekil 2).



Şekil 2. Round Robin Görevlendiricisi

Tüm "hazır" yüksek öncelikli prosesler tamamlandığında, geri beslemeli görevlendirici, en yüksek öncelikli ve boş olmayan kuyruğun başındaki prosesin kaldığı yerden devam etmesiyle çalışmasını sürdürür. Bir sonraki zaman adımında, eşit veya daha yüksek önceliğe sahip başka "hazır" prosesler varsa, mevcut proses askıya alınır (veya sonlandırılır ve kaynakları serbest bırakılır).
Mantık akışı Şekil 3'te verilmiştir.



Şekil 3. Görevlendirici Mantık Akışı

**KAYNAK KISITLAMALARI**

Sistem aşağıdaki kaynaklara sahiptir:

- 2 Yazıcı
- 1 Tarayıcı
- 1 modem
- 2 CD Sürücü
- İşlemler için 1024 MB Bellek

Düşük öncelikli prosesler bu kaynakların herhangi birini veya tamamını kullanabilir, ancak Görevlendiriciye proses gönderildiğinde prosesin hangi kaynakları kullanacağı bildirilir (bknz. giriş.txt). Görevlendirici, talep edilen her kaynağın hazır kuyruklarında da geçirilen süre dahil yalnızca talep eden prosese ait olmasını garanti eder. Sadece proses tamamlandığında kaynaklar iade edilecektir.

Gerçek Zamanlı prosesler herhangi bir giriş/çıkış kaynağına (Yazıcı / Tarayıcı / Modem / CD) ihtiyaç duymayacaktır, ancak bellek tahsisine ihtiyaç duyarlar. Gerçek Zamanlı prosesler için bellek gereksinimi her zaman 64 Mbayt veya daha az olacaktır.

**BELLEK TAHSİSİ**

Bellek tahsisi, bir prosesin ömrü boyunca kullandığı bir bellek bloğu olmalıdır. Gerçek Zamanlı proseslerin yürütülmesinin engellenmemesi için yeterli bir yedek bellek bırakılmalıdır - Çalışan bir Gerçek Zamanlı proses için 64 Mbayt, "etkin" kullanıcı işleri arasında paylaşılmak üzere 960 Mbayt.

**PROSESLER**

Sistem üzerindeki prosesler, sevk edilen her iş için yeni bir proses oluşturan Görevlendirici tarafından simüle edilir. Bu proses, herhangi bir öncelikli proses için kullanılabilen genel bir prosestir. Prosesin fonksiyonları aşağıda listelenmiştir:

1. İşlem başladığında proses kimliğini gösteren bir mesaj;
2. İşlemin yürütüldüğü her saniye düzenli bir mesaj; ve
3. İşlem Askıya Alındığında, Devam Edildiğinde veya Sonlandırıldığında bir mesaj.

Görevlendirici tarafından sonlandırılmazsa, proses 20 saniye sonra kendiliğinden sona erecektir. İşlem, her benzersiz proses için rastgele oluşturulmuş bir renk şeması kullanarak yazdırır, böylece proseslerin tek tek "dilimleri" kolayca ayırt edilebilir.

Bir prosesin yaşam döngüsü:

1. Proses, varış zamanı, önceliği, gereken proses süresini (saniye cinsinden), bellek blok boyutunu ve talep edilen diğer kaynakları belirten bir ilk proses listesi aracılığıyla Görevlendirici giriş kuyruklarına gönderilir.
2. Bir proses "geldiğinde" ve gerekli tüm kaynaklar mevcut olduğunda "çalışmaya hazırdır".
3. Bekleyen Gerçek Zamanlı işler, FCFS esasına göre yürütülmek üzere gönderilir.
4. Daha düşük öncelikli bir kullanıcı prosesi için yeterli kaynak ve bellek varsa, proses geri besleme Görevlendirici birimi içindeki uygun öncelik sırasına aktarılır ve kalan kaynak göstergeleri (bellek listesi ve g/ç aygıtları) güncellenir.
5. Bir proses başlatıldığında Görevlendirici proses parametrelerini (Kimliği, öncelik, kalan proses süresi (saniye olarak), bellek konumu ve blok boyutu ve istenen kaynakları görüntüler. )
6. Bir Gerçek Zamanlı prosesin**,** zaman süresi dolana kadar çalışmasına izin verilir.
7. Düşük öncelikli bir Kullanıcı prosesinin, askıya alınmadan veya süresi dolmuşsa sonlandırılmadan önce bir saniye boyunca çalışmasına izin verilir. Askıya alınırsa, öncelik seviyesi düşürülür (mümkünse) ve yukarıdaki Şekil 1 ve 3'te gösterildiği gibi uygun öncelik kuyruğunda yeniden kuyruğa alınır.
8. Gelen prosesler kuyruğunda daha yüksek öncelikli Gerçek Zamanlı prosesler beklemediği sürece, geri besleme kuyruklarındaki en yüksek öncelikli bekleyen proses başlatılır veya yeniden başlatılır.
9. Bir proses sonlandırıldığında, kullandığı kaynaklar, daha sonraki proseslere yeniden tahsis edilmek üzere Görevlendiriciye döndürülür.
10. Proses listesinde, giriş kuyruklarında ve geri besleme kuyruklarında başka proses olmadığında, Görevlendirici sona erer.

## PROSES LİSTESİ

Proses Listesi, işlenecek proseslerin listesidir. Liste, komut satırında belirtilen bir metin dosyasında bulunur. Yani

**>systemd giriş.txt**

Listenin her satırı, aşağıdaki verilerle *"virgül"* ile sınırlandırılmış liste olarak bir prosesi tanımlar:
<varış zamanı>, <öncelik>, <proses zamanı>, <Mbayt>, <#yazıcılar>, <#tarayıcılar>, <#modemler>, <#CD'ler>
Mesela,

**12, 0, 1, 64, 0, 0, 0, 0**
**12, 1, 2, 128, 1, 0, 0, 1**
**13, 3, 6, 128, 1, 0, 1, 2**

şunu belirtir:

1. Proses: 12 zamanında varış, öncelik 0 (Gerçek Zamanlı), 1 saniyelik proses zamanı ve 64 Mbayt bellek gerektirir – başka hiçbir g/ç kaynağı gerekmez.
2. Proses: 12 zamanında varış, öncelik 1 (yüksek öncelikli Kullanıcı prosesi), 2 saniye proses süresi, 128 Mbyte bellek, 1 yazıcı ve 1 CD sürücüsü gerektirir.
3. Proses: 13 zamanında varış, öncelik 3 (en düşük öncelikli Kullanıcı prosesi), 6 saniyelik proses süresi, 128 Mbayt bellek, 1 yazıcı, 1 modem ve 2 CD sürücüsü gerektirir.

İş/proses listesini içeren metin dosyası projede verilecektir.

**PROJE GEREKSİNİMLERİ**

1. Yukarıdaki kriterleri karşılayan bir Görevlendirici tasarlayın.
   Rapor içerisinde:
   a. Hangi bellek tahsis algoritmalarını kullanabileceğinizi açıklayın ve tartışın ve son tasarım seçiminizi gerekçelendirin.
   b. Görevlendirici tarafından bellek ve diğer kaynakları kuyruğa almak, göndermek ve tahsis etmek için kullanılan yapıları tanımlayın ve açıklayın.
   c. Çeşitli modülleri ve ana işlevleri açıklayarak programınızın genel yapısını tanımlayın ve gerekçelendirin ('arayüzler' işlevinin açıklamaları beklenir) .
   d. "Gerçek" işletim sistemleri tarafından kullanılan şemalarla karşılaştırarak, böyle çok düzeyli bir görevlendirme şemasının neden kullanılacağını tartışın. Olası iyileştirmeler önererek, böyle bir şemadaki eksiklikleri ana hatlarıyla belirtin. Tartışmalarınıza bellek ve kaynak ayırma şemalarını dahil edin.

   Resmi tasarım belgesinin derinlemesine tartışmalara, açıklamalara ve argümanlara sahip olması beklenir.
   **Tasarım belgesi herhangi bir kaynak kodu İÇERMEMELİDİR. Pdf olarak sisteme yüklenecek ve tüm grup üyelerinin adlarını içeren bir kapağa sahip olacaktır.**

2. Projeyi JAVA dilini kullanarak kodlayın.

3. Kaynak kodu, meslektaşlarınızın kodu anlamasını ve kolayca değerlendirmesini sağlamak için kapsamlı bir şekilde yorumlanmalı (açıklama satırları - Türkçe) ve uygun şekilde yapılandırılmalıdır. Düzgün yazılmış ve düzenlenmiş kodu yorumlamak çok daha kolaydır ve projenizi değerlendirecek öğretim elemanının zihinsel jimnastik yapmak zorunda kalmadan kodlamanızı anlayabilmesini sağlamak sizin yararınızadır!

4. SABIS'e yüklenecek zip dosya Eclipse derlenmiş projesini ve pdf halindeki raporu içermelidir.

5. Tüm proje notu sınıf notunuzun %15'i değerindedir. Kopya ödevler 0 (sıfır) ile notlandırılacaktır. Ödevler GiHub'a grup temsilcisi tarafından yüklenecektir. Yüklenen proje dosyasının adı grup adı olacaktır. Ders yürütücülerine ve ders yardımcılarına GitHub üzerinde proje dosyalarına erişim yetkisi (sadece okuma) verilecektir. Grubun diğer üyeleri en az bir commit gerçekleştirmelidir. Commit yapmayan üyeler 0 olarak değerlendirilecektir.

**PUANLAMA**

- Bellek ayırma algoritmaları seçiminin tanımı, tartışılması ve gerekçesi - 5 puan
- Görevlendirici tarafından kullanılan yapıların tanımı ve tartışılması-5 puan
- Program yapısının ve bireysel modüllerin tanımı ve gerekçesi-5 puan
- Görevlendiricinin tartışılması (bellek ve kaynak tahsisi dahil), eksiklikler ve olası iyileştirmeler - 5 puan.
- FCFS yüksek öncelikli sıralayıcının çalışması (FCFS.java) - 5 puan
- Kullanıcı Geri Beslemeli sıralayıcının çalışması (GBG.java) - 5 puan
- Kullanıcı Geri Beslemeli sıralayıcının Round Robin modunda çalışması (RR.java) - 5 puan
- Karışık sıralayıcı çalışması - 5 puan
- Kaynak tahsisi - 30 puan
- Bellek yönetimi - 30 puan