2025-26 GÜZ YARIYILI İŞLETİM SİSTEMLERİ DERSİ – PROJE ÇALIŞMASI

Konu: FreeRTOS Kullanarak Basit Bir Görev (4 seviyeli öncelikli sıralayıcı) Sıralayıcısı (Scheduler) Simülasyonu

1. Proje Amacı

Bu projenin amacı, FreeRTOS'un görev (task) sıralayıcısının nasıl çalıştığını PC üzerinde POSIX (Linux/Windows) ortamında simüle etmek; görev önceliklerinin, bağlam değişimlerinin ve zaman dilimlerinin (tick) nasıl etkileştiğini göstermektir.

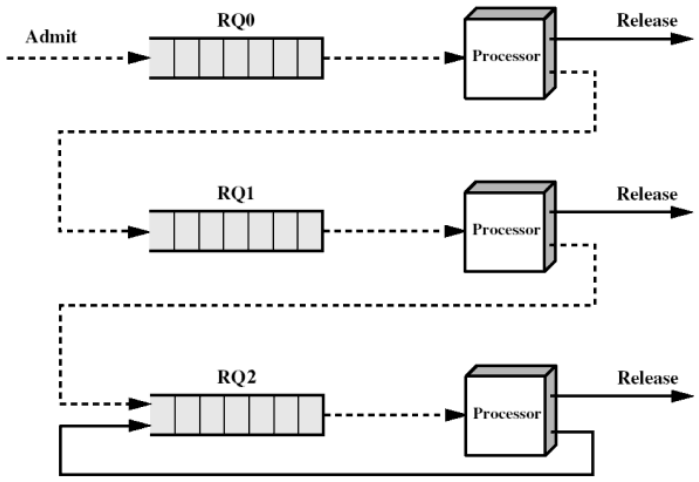
2. Proje Hedefleri

- FreeRTOS çekirdeğini tanımak ve görev zamanlama mantığını öğrenmek
- Kendi görev sıralayıcısını (scheduler) tasarlamak
- Görevlerin öncelik temelli yürütülmesini gözlemlemek
- Gerçek zamanlı görev yönetimini uygulamak

3. DÖRT SEVİYELİ ÖNCELİKLİ GÖREVLENDİRİCİ

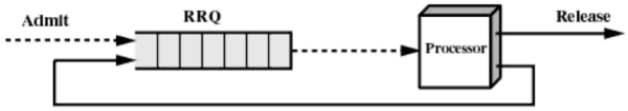
Dağıtıcı dört öncelik düzeyinde çalışır:

1. İlk Gelen İlk Çalışır (FCFS) algoritması temelinde hemen çalıştırılması gereken gerçek zamanlı görevler, daha düşük öncelikle çalışan diğer grevlerden daha yüksek bir önceliğe sahiptir (öncelik değeri 0). Bu görevler tamamlanıncaya kadar kesilmeden yürütülür.
2. Normal kullanıcı görevleri, üç seviyeli bir geri beslemeli görevlendiricide çalıştırılır. Dağıtıcının temel zamanlama kuantumu (q) 1 saniyedir. Bu aynı zamanda geri besleme sıralayıcısının zaman kuantumunun da değeridir.



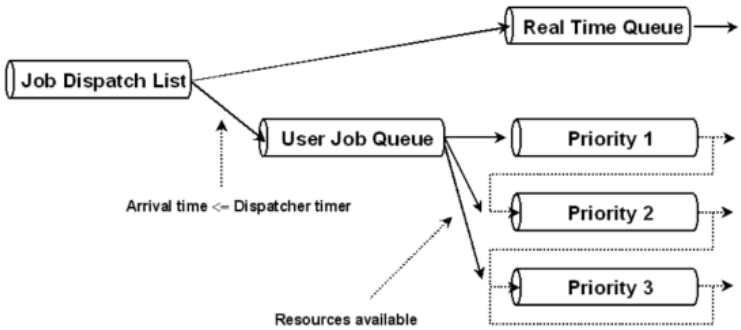
Şekil 1. Üç Seviyeli Geri Beslemeli Görevlendirici

Görevlendirici, görev listesinden (giris.txt) beslenen iki adet kuyruğa sahiptir: Gerçek Zamanlı ve Kullanıcı Görev kuyrukları. Görev listesi, her zaman adımında sürekli işlenir ve gelen görevler uygun kuyruğa aktarılır. Kuyruklar işleme alınır; tüm gerçek zamanlı görevler tamamlanmak üzere çalıştırılır ve o anda çalışmakta olan diğer düşük öncelikli görevler kesilir.
Düşük öncelikli geri beslemeli görevlendirici yeniden etkinleştirilmeden önce gerçek zamanlı kuyruktaki görevler bitirilmelidir. Bir geri besleme kuyruğunun normal çalışması, en yüksek öncelik düzeyindeki görevleri 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 görevler 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 görevler tamamlandığında, geri beslemeli görevlendirici, en yüksek öncelikli ve boş olmayan kuyruğun başındaki görevin kaldığı yerden devam etmesiyle çalışmasını sürdürür. Bir sonraki zaman adımında, eşit veya daha yüksek önceliğe sahip başka "hazır" görevler varsa, mevcut görevler askıya alınır (veya sonlandırılır).
Mantık akışı Şekil 3'te verilmiştir.



Şekil 3. Görevlendirici Mantık Akışı

Görevler

Sistem üzerindeki görevler, sevk edilen her iş için yeni bir görev oluşturan Görevlendirici tarafından simüle edilir. Bu görev, herhangi bir öncelikli görev için kullanılabilen genel bir görevdir. Görevin fonksiyonları aşağıda listelenmiştir:

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

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

Bir görevin yaşam döngüsü:

1. Görev, varış zamanı, önceliği, gereken görev süresini (saniye cinsinden) ilk görev listesi aracılığıyla Görevlendirici giriş kuyruklarına gönderilir.
2. Bir görev "vardığında" çalışmaya hazırdır.
3. Bekleyen Gerçek Zamanlı görevler, FCFS esasına göre yürütülmek üzere gönderilir.
4. Daha düşük öncelikli bir kullanıcı görevi, görev geri besleme Görevlendirici birimi içindeki uygun öncelik sırasına aktarılır.
5. Bir görev başlatıldığında Görevlendirici proses parametrelerini Kimliği, öncelik, kalan görev süresi (saniye olarak) görüntüler.
6. Bir Gerçek Zamanlı görevin, zaman süresi dolana kadar çalışmasına izin verilir.
7. Düşük öncelikli bir Kullanıcı görevinin, 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 görevler kuyruğunda daha yüksek öncelikli Gerçek Zamanlı görevler beklemediği sürece, geri besleme kuyruklarındaki en yüksek öncelikli bekleyen görev alınır veya yeniden başlatılır.
9. Görev listesinde, giriş kuyruklarında ve geri besleme kuyruklarında başka görev olmadığında, Görevlendirici sona erer.

Görev Listesi

Görev Listesi, işlenecek görevlerin listesidir. Liste, komut satırında belirtilen bir metin dosyasında bulunur. Listenin her satırı, aşağıdaki verilerle *"virgül"* ile sınırlandırılmış liste olarak bir görevi tanımlar:
<varış zamanı>, <öncelik>, <görev zamanı>
Mesela,
12, 0, 1
12, 1, 2
13, 3, 6
şunu belirtir:

1. Görev: 12 zamanında varış, öncelik 0 (Gerçek Zamanlı), 1 saniyelik görev zamanı

2. Görev: 12 zamanında varış, öncelik 1 (yüksek öncelikli Kullanıcı görevi), 2 saniye görev süresi,

3. Görev: 13 zamanında varış, öncelik 3 (en düşük öncelikli Kullanıcı görevi), 6 saniyelik görev süresi,

İş/görev listesini içeren metin dosyası projede verilecektir.

4. Gerekli Araçlar ve Ortam

• İşletim Sistemi: Linux, macOS veya WSL/MinGW ile Windows
• Derleyici: gcc
• FreeRTOS kaynak kodu: https://github.com/FreeRTOS/FreeRTOS-Kernel
• POSIX portu: FreeRTOS'un portable/ThirdParty/GCC/Posix klasörü

5. Proje Dosya Yapısı

```
FreeRTOS_PC_Scheduler/
│
├── FreeRTOS/
│   ├── include/
│   ├── portable/ThirdParty/GCC/Posix/
│   └── source/
│
├── src/
│   ├── main.c
│   ├── scheduler.c
│   ├── scheduler.h
│   └── tasks.c
│
├── Makefile
├── giris.txt
└── README.md
```

6. Derleme ve Çalıştırma

Terminal üzerinde aşağıdaki adımlar takip edilmelidir:

1. Proje klasörüne girin: cd FreeRTOS_PC_Scheduler
2. Derleme: make
3. Çalıştırma: ./freertos_sim giris.txt

Beklenen Çıktı:

Terminal veya UART çıktısında aşağıdaki gibi bir sıra gözlemlenir:

```
0.0000 sn  task1 başladı         (id:0000   öncelik:1  kalan süre:2 sn)
1.0000 sn  task1 askıda          (id:0000   öncelik:2  kalan süre:1 sn)
1.0000 sn  task2 başladı         (id:0001   öncelik:0  kalan süre:1 sn)
2.0000 sn  task2 sonlandı        (id:0001   öncelik:0  kalan süre:0 sn)
2.0000 sn  task3 başladı         (id:0003   öncelik:0  kalan süre:3 sn)
3.0000 sn  task3 yürütülüyor     (id:0003   öncelik:0  kalan süre:2 sn)
4.0000 sn  task3 yürütülüyor     (id:0003   öncelik:0  kalan süre:1 sn)
5.0000 sn  task3 sonlandı        (id:0003   öncelik:0  kalan süre:0 sn)
5.0000 sn  task4 başladı         (id:0006   öncelik:0  kalan süre:4 sn)
6.0000 sn  task4 yürütülüyor     (id:0006   öncelik:0  kalan süre:3 sn)
7.0000 sn  task4 yürütülüyor     (id:0006   öncelik:0  kalan süre:2 sn)
8.0000 sn  task4 yürütülüyor     (id:0006   öncelik:0  kalan süre:1 sn)
9.0000 sn  task4 sonlandı        (id:0006   öncelik:0  kalan süre:0 sn)
9.0000 sn  task5 başladı         (id:0007   öncelik:0  kalan süre:4 sn)
10.0000 sn task5 yürütülüyor     (id:0007   öncelik:0  kalan süre:3 sn)
11.0000 sn task5 yürütülüyor     (id:0007   öncelik:0  kalan süre:2 sn)
12.0000 sn task5 yürütülüyor     (id:0007   öncelik:0  kalan süre:1 sn)
13.0000 sn task5 sonlandı        (id:0007   öncelik:0  kalan süre:0 sn)
13.0000 sn task6 başladı         (id:0008   öncelik:0  kalan süre:2 sn)
14.0000 sn task6 yürütülüyor     (id:0008   öncelik:0  kalan süre:1 sn)
15.0000 sn task6 sonlandı        (id:0008   öncelik:0  kalan süre:0 sn)
15.0000 sn task7 başladı         (id:0010   öncelik:0  kalan süre:3 sn)
16.0000 sn task7 yürütülüyor     (id:0010   öncelik:0  kalan süre:2 sn)
17.0000 sn task7 yürütülüyor     (id:0010   öncelik:0  kalan süre:1 sn)
18.0000 sn task7 sonlandı        (id:0010   öncelik:0  kalan süre:0 sn)
18.0000 sn task8 başladı         (id:0016   öncelik:0  kalan süre:4 sn)
19.0000 sn task8 yürütülüyor     (id:0016   öncelik:0  kalan süre:3 sn)
20.0000 sn task8 yürütülüyor     (id:0016   öncelik:0  kalan süre:2 sn)
21.0000 sn task8 yürütülüyor     (id:0016   öncelik:0  kalan süre:1 sn)
21.0000 sn task1 zamanaşımı(id:0000   öncelik:2  kalan süre:1 sn)
22.0000 sn task8 sonlandı        (id:0016   öncelik:0  kalan süre:0 sn)
```

7. Proje Gereksinimleri

1. **Rapor İçeriği Önerisi (**max 5 sayfa)

Giriş: FreeRTOS ve gerçek zamanlı işletim sistemleri kavramı
Amaç: Projenin amacı ve yöntemi, Scheduler yapısının tasarlanması
Yöntem: FreeRTOS görev yönetimi ve kendi scheduler fonksiyonları
Uygulama: Kod açıklamaları ve çalışma çıktısı
Sonuç: Görev önceliği, deterministik davranış ve FreeRTOS'un etkinliği üzerine değerlendirme

- Görevlendirici tarafından bellek ve diğer kaynakları kuyruğa almak, göndermek ve tahsis etmek için kullanılan yapıları tanımlayın ve açıklayın.
- Çeşitli modülleri ve ana işlevleri açıklayarak programınızın genel yapısını tanımlayın ve gerekçelendirin ('arayüzler' işlevinin açıklamaları beklenir).
- "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.

Kaynaklar: FreeRTOS dokümantasyonu, mikrodenetleyici veri sayfaları

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 C 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 **PC üzerinde simülasyon (FreeRTOS+POSIX)** kaynak dosyasını ve pdf halindeki raporu içermelidir.
    - Kaynak kod dosyaları (src klasörü altındaki tüm .c ve .h dosyaları)
    - Çalışan uygulamanın terminal çıktısı (ekran görüntüsü veya metin dosyası)

5. Tüm proje notu sınıf notunuzun %15'i değerindedir. Kopya ödevler 0 (sıfır) ile notlandırılacaktır. **Ödevler GitHub'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.

8. Değerlendirme Kriterleri

- Proje derlenebilir ve çalışır durumda olmalı (%40)
- FreeRTOS görev yapısı doğru kullanılmalı (%30)
- Rapor içeriği açık ve teknik olmalı (%20)
- Kod düzeni, yorumlar ve dokümantasyon (%10)