SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
İŞLETİM SİSTEMLERİ
PROJE

GRUP ÜYELERİ:

Bartu Dönmez G231210561 – 2A
Kağan Aydoğan G231210375 – 1A
Mehmet Tüysüz G231210017 – 1C
Metehan Köse G231210041 – 2C
Yunus Suntay G231210039 – 2A

Dersi Veren: Ahmet ZENGİN/Abdullah SEVİN

GİTHUB LİNKİ

https://github.com/kaganay/isletimsistemigrup02

1. GİRİŞ

Gerçek zamanlı işletim sistemleri (Real-Time Operating Systems – RTOS), belirli zaman kısıtları altında doğru ve öngörülebilir şekilde çalışması gereken sistemler için geliştirilmiş özel işletim sistemleridir. Bu tür sistemlerde yalnızca üretilen sonuçların doğruluğu yeterli değildir; aynı zamanda bu sonuçların belirlenen zaman aralıkları içinde üretilmesi kritik öneme sahiptir. Zaman kısıtlarının ihlali, özellikle güvenlik kritik uygulamalarda ciddi sistem hatalarına yol açabilmektedir.

FreeRTOS, açık kaynak kodlu yapısı, düşük bellek ayak izi, modüler mimarisi ve çok sayıda mikrodenetleyici platformunu desteklemesi sayesinde günümüzde en yaygın kullanılan RTOS çözümlerinden biridir. Görev (task) tabanlı yapısı, önceliklendirme mekanizması, kesme yönetimi ve deterministik zamanlayıcı davranışı sayesinde hem akademik çalışmalarda hem de endüstriyel projelerde tercih edilmektedir.

Bu projede, FreeRTOS'un görev zamanlama ve önceliklendirme yaklaşımından ilham alınarak, PC üzerinde çalışan bir simülasyon ortamında, çok düzeyli geri beslemeli kuyruk (Multi-Level Feedback Queue – MLFQ) mimarisine sahip bir görev zamanlayıcı tasarlanmış ve C dili kullanılarak gerçekleştirilmiştir. Amaç, gerçek bir RTOS çekirdeğinin temel davranışlarını yazılımsal olarak modellemek ve gözlemlenebilir çıktılar üretmektir.

2. PROJENİN AMACI VE KAPSAMI

Bu çalışmanın temel amacı, gerçek zamanlı işletim sistemlerinde kullanılan görev zamanlama mekanizmalarını teorik ve pratik açıdan incelemek ve FreeRTOS'un bu konudaki yaklaşımını daha iyi kavramaktır. Bu doğrultuda proje aşağıdaki hedefleri kapsamaktadır:

- FreeRTOS'un görev yönetimi ve zamanlama mantığını anlamak
- Çok düzeyli bir scheduler yapısının tasarımını ve çalışma prensibini analiz etmek
- Önceliklendirme, zaman paylaşımlı yürütme (time-slicing), askıya alma (suspend) ve zaman aşımı (timeout) kavramlarını simüle etmek
- Deterministik davranışın ve öncelik tabanlı yürütmenin sistem çıktıları üzerindeki etkisini gözlemlemek

Bu kapsamda, POSIX uyumlu bir PC ortamında çalışan bir simülasyon geliştirilmiştir. Görev bilgileri bir giriş dosyasından okunmuş, görevler öncelik seviyelerine göre farklı kuyruklara yerleştirilmiş ve zamanlayıcı tarafından belirlenen kurallara göre yürütülmüştür. Böylece gerçek donanım gerektirmeden FreeRTOS benzeri bir scheduler davranışı gözlemlenmiştir.

3. YÖNTEM VE SCHEDULER TASARIMI

3.1 Görev (Task) Yönetimi Yaklaşımı

Bu projede her görev, gerçek zamanlı sistemlerde yaygın olarak kullanılan temel niteliklere sahip olacak şekilde modellenmiştir. Her görev aşağıdaki özelliklerle tanımlanmıştır:

- Görev Kimliği (ID)
- İlk öncelik seviyesi
- Çalışma sırasında değişebilen mevcut öncelik seviyesi
- Sisteme giriş (varış) zamanı
- Toplam çalışma süresi (burst time)

- Kalan çalışma süresi
- Görev durumu (hazır, çalışıyor, askıda, tamamlandı, zaman aşımı)

Görevler simülasyon süresi boyunca bu durumlar arasında geçiş yapmaktadır. Her durum değişimi, zaman bilgisiyle birlikte terminal çıktısına yazdırılarak sistemin davranışı ayrıntılı şekilde izlenebilir hale getirilmiştir. Bu yaklaşım, FreeRTOS'un görev durum diyagramına benzer bir yapı sunmaktadır.

3.2 Kuyruk Yapıları ve Kaynak Yönetimi

Scheduler tasarımında, görevlerin önceliklerine göre yönetilebilmesi için dört adet ayrı kuyruk tanımlanmıştır:

- RT Kuyruğu (Öncelik 0)
  Gerçek zamanlı görevler için ayrılmıştır. Bu kuyruktaki görevler en yüksek önceliğe sahiptir ve kesintiye uğramadan, tamamlanana kadar çalıştırılır.
- Q1 Kuyruğu (Öncelik 1)
- Q2 Kuyruğu (Öncelik 2)
- Q3 Kuyruğu (Öncelik 3 ve üzeri – Round Robin)

Bu yapı, çok düzeyli geri beslemeli kuyruk (MLFQ) yaklaşımına dayanmaktadır. Görevler CPU'yu kullandıkça, çalışma sürelerine bağlı olarak daha alt öncelikli kuyruklara taşınır. Bu sayede kısa süreli görevler hızlıca tamamlanırken, uzun süre CPU kullanan görevlerin sistemi tek başına meşgul etmesi engellenir.

Bellek ve kaynak yönetimi, görevlerin bu kuyruklar arasında taşınması ve görev durumlarının güncellenmesi ile sağlanmıştır. Bu yaklaşım, gerçek RTOS'larda kullanılan görev kontrol blokları (TCB) ve kuyruk yapılarının sadeleştirilmiş bir benzeridir.

3.3 Zaman Aşımı (Timeout) Mekanizması

Gerçek zamanlı sistemlerde, uzun süre CPU alamayan görevler sistem kararlılığını ve öngörülebilirliği olumsuz etkileyebilir. Bu nedenle projede bir zaman aşımı mekanizması uygulanmıştır.

Uygulanan kurallar şu şekildedir:

- Bir görev 20 saniye boyunca CPU alamazsa zaman aşımına uğrar
- Zaman aşımına uğrayan görev:
  - Terminal çıktısında kırmızı renkte gösterilir
  - Tekrar çalıştırılmaz
  - Sistemden çıkarılmış kabul edilir

Bu mekanizma, FreeRTOS'ta kullanılan watchdog zamanlayıcıları ve zaman denetimi yaklaşımlarına benzer bir davranış sergilemektedir. Böylece sistemde bekleyen görevlerin kontrolsüz şekilde birikmesi engellenmiştir.

4. UYGULAMA VE PROGRAM YAPISI

4.1 Modüler Yapı

Program, okunabilirliği ve sürdürülebilirliği artırmak amacıyla modüler bir yapıda tasarlanmıştır. Ana modüller şu şekildedir:

- Scheduler Modülü:

Görev seçimi, öncelik kontrolü, zamanlama kararları ve görev durum geçişlerinden sorumludur.

- Task Yönetim Modülü: Görevlerin oluşturulması, giriş dosyasından okunması ve simülasyon sonunda bellekten temizlenmesini sağlar.
- Queue Modülü: Kuyrukların oluşturulması, görevlerin kuyruklara eklenmesi ve çıkarılması işlemlerini yönetir.

Bu modüler yapı, gerçek işletim sistemlerindeki çekirdek (ernel) bileşenlerinin mantıksal ayrımını yansıtmaktadır.

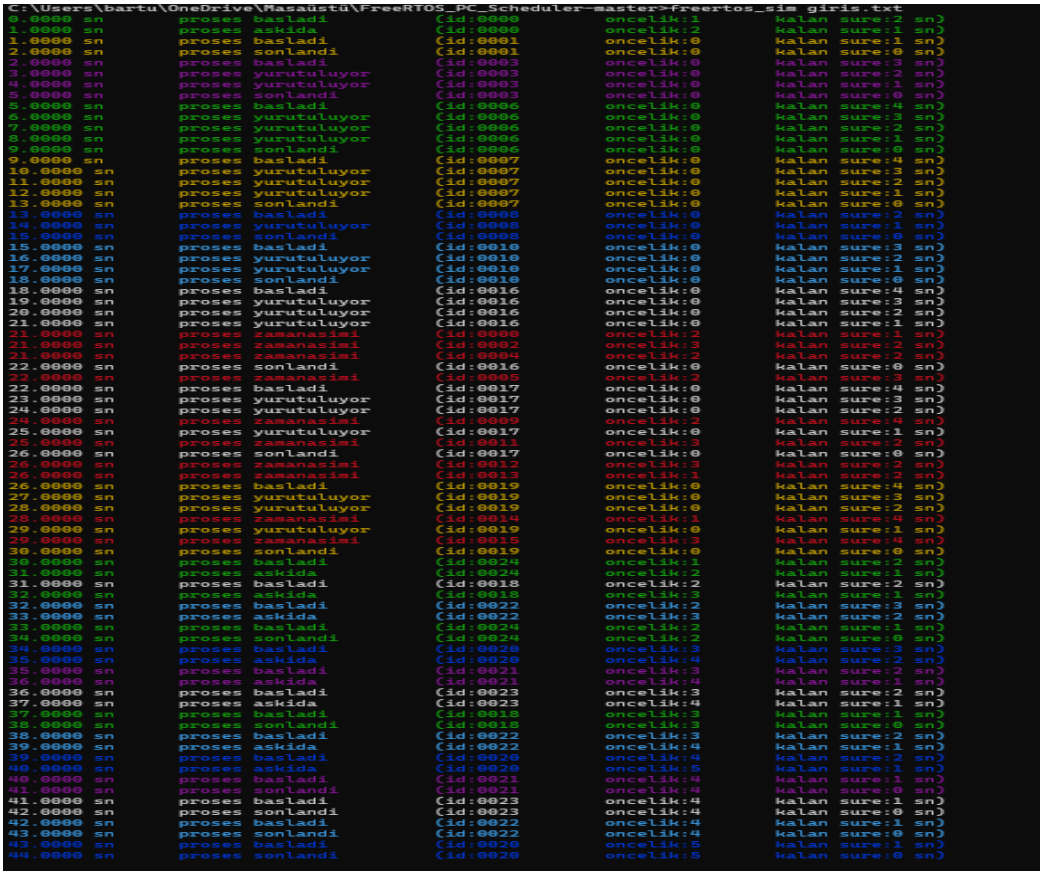
4.2 Arayüzler (Interface) ve Ana İşlevler

Her modül, diğer modüllerle yalnızca tanımlı arayüzler üzerinden iletişim kurmaktadır. Bu yaklaşım:

- Modüller arası bağımlılıkları azaltır
- Hataların izole edilmesini kolaylaştırır
- Gerçek işletim sistemlerindeki kernel–user ayrımına benzer bir yapı sunar

Ana simülasyon döngüsü, zaman adımlarını saniye bazında ilerleterek her adımda şu işlemleri gerçekleştirir:

- Yeni gelen görevleri uygun kuyruklara ekler
- Çalışacak görevi öncelik kurallarına göre seçer
- Zaman aşımı kontrolünü gerçekleştirir
- Görev durum değişikliklerini loglar



Çalışma Çıktısı

5. GERÇEK İŞLETİM SİSTEMLERİYLE KARŞILAŞTIRMA

5.1 Çok Düzeyli Görevlendirmenin Avantajları

Gerçek işletim sistemlerinde (Linux CFS, FreeRTOS, Windows Scheduler vb.) çok düzeyli zamanlayıcılar yaygın olarak kullanılmaktadır. Bunun temel nedenleri şunlardır:

- Adil CPU paylaşımı sağlar
- Öncelik terslenmesi riskini azaltır
- Gerçek zamanlı görevler için deterministik davranış sunar

Bu projede kullanılan MLFQ yaklaşımı, gerçek sistemlerdeki scheduler tasarımlarına kavramsal olarak oldukça yakındır.

5.2 Eksiklikler ve Olası İyileştirmeler

Bu simülasyon bazı basitleştirmeler içermektedir:

- Donanım kesmeleri (interrupt) simüle edilmemiştir
- Bellek ayırma sabit ve sınırlıdır
- Görevler arası senkronizasyon mekanizmaları (mutex, semaphore) bulunmamaktadır

Gelecekte yapılabilecek iyileştirmeler şunlardır:

- Mutex ve semaphore mekanizmalarının eklenmesi
- Dinamik bellek havuzu (heap) yönetiminin simüle edilmesi

6. SONUÇ VE DEĞERLENDİRME

Bu çalışmada, FreeRTOS'un temel görev yönetimi ve zamanlama prensipleri başarıyla modellenmiş ve PC üzerinde çalışan bir simülasyon geliştirilmiştir. Çok düzeyli kuyruk yapısı sayesinde görevlerin CPU kullanımı dengelenmiş, önceliklendirme ve deterministik davranış net bir şekilde gözlemlenmiştir.

Elde edilen sonuçlar, FreeRTOS'un gömülü sistemlerde neden yaygın olarak tercih edildiğini açıkça ortaya koymaktadır. Görev önceliği, zaman aşımı ve deterministik zamanlama gibi kavramlar, gerçek zamanlı sistemlerin güvenilirliğinde kritik rol oynamaktadır.

7. KAYNAKLAR

1. Barry, R. (2023). *FreeRTOS Documentation*. Real Time Engineers Ltd. Erişim adresi: https://www.freertos.org
2. Barry, R. (2016). *Using the FreeRTOS™ Real Time Kernel – A Practical Guide*. Real Time Engineers Ltd.
3. ARM Ltd. (2020). *ARM® Cortex®-M Architecture Reference Manual*. ARM Documentation.
4. FreeRTOS Community. (2024). *FreeRTOS Kernel Source Code Repository*. GitHub. Erişim adresi: https://github.com/FreeRTOS/FreeRTOS-Kernel
5. Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems* (4th ed.). Pearson Education.
6. Stallings, W. (2018). *Operating Systems: Internals and Design Principles* (9th ed.). Pearson.