SAKARYA
ÜNİVERSİTESİ

İŞLETİM SİSTEMLERİ PROJESİ

ABDULLAH BALABAN	B231210049
ALİ HALİT ÖRNEK	B231210385
ALİ SELMAN ERKAN	B231210103
TALHA FARUK YILDIRIM	B231210583
BELAL ALMASRI	B231210555

GİTHUB LİNKİ: https://github.com/Talhafy/12.grup

FreeRTOS PC Scheduler Proje Raporu

Tarih: 19.12.2025 | Ortam: FreeRTOS POSIX (PC) | Zaman adımı: 1 saniye

1) Giriş: FreeRTOS ve gerçek zamanlı işletim sistemleri

Gerçek zamanlı işletim sistemleri (RTOS), görevlerin belirli zaman kısıtları altında çalışmasını hedefler. Bu tür sistemlerde amaç sadece "hız" değil; özellikle öngörülebilirlik (determinism) ve zamanında tepki vermektir. FreeRTOS, gömülü sistemlerde yaygın kullanılan, öncelik tabanlı zamanlama yapan hafif bir RTOS çekirdeğidir. Bu projede FreeRTOS'un POSIX portu kullanılarak PC üzerinde çalıştırılabilen bir simülasyon ortamı kurulmuş ve ödevde istenen çok düzeyli (multi-level) görevlendirici davranışı uygulanmıştır.

2) Amaç: Projenin amacı ve scheduler tasarımı

Projenin amacı, giris.txt dosyasından gelen görevleri zaman adımı bazında sisteme alıp aşağıdaki kurallara göre yürütmektir:

- Öncelik 0 (RT) görevlerini FCFS mantığıyla seçmek ve bitene kadar kesmeden çalıştırmak.
- Öncelik 1-2-3 (kullanıcı) görevlerini 3 seviyeli geri beslemeli (feedback/MLFQ benzeri) yapıda yürütmek.
- Kuantumu q = 1 saniye olacak şekilde her saniye çalıştırma/askıya alma davranışını gözlemlemek.
- Her saniyede "başladı / yürütülüyor / askıda / sonlandı / zaman aşımı" çıktılarını üretmek.

Scheduler kuyruğu tasarımı (4 hazır kuyruk):

Kuyruk	Öncelik	Görev sınıfı	Seçim kuralı (öncelik sırası)
q_rt	0	Gerçek zamanlı (RT)	Her zaman önce
q1	1	Kullanıcı	q_rt boşsa
q2	2	Kullanıcı	q_rt ve q1 boşsa
q3	3	Kullanıcı	q_rt, q1 ve q2 boşsa

3) Yöntem: FreeRTOS görev yönetimi ve kendi scheduler fonksiyonları

Uygulama iki katmanlıdır: (i) FreeRTOS POSIX altyapısı zaman ilerlemesini ve temel görev yaşam döngüsü fonksiyonlarını sağlar, (ii) asıl zamanlama kararları ve "kimin çalıştığı" bilgisi scheduler.c içinde simüle edilir. Böylece kararlar her saniye tek bir merkezden verilir ve çıktı deterministik şekilde gözlemlenebilir.

Ana arayüzler (scheduler.h):

- scheduler_baslat(): Kuyrukları ve global durumu hazırlar.
- scheduler_gorev_kabul(Task*): Yeni gelen görevi uygun kuyruğa alır.
- scheduler_calistir(t): Aging, preemption, seçim ve yürütmeyi uygular.

- scheduler_bitti_mi(): Simülasyonun tamamlanıp tamamlanmadığını kontrol eder.

Girdi formatı:

```
varis_zamani, oncelik, sure

Örnek:
0,1,2
1,0,1
```

4) Uygulama: Veri yapıları, bellek/kaynak yönetimi ve modüller

Bu bölüm, görevlendiricinin belleği ve diğer kaynakları kuyruklara almak, göndermek ve tahsis etmek için kullandığı yapıları; ayrıca modüllerin genel mimarisini açıklar.

4.1 Görev (proses) veri yapısı

Her görev, kimlik (id), varış zamanı, öncelik (0..3), kalan süre, bekleme süresi ve durum bilgilerini taşır. Kuyruklar için bağlı liste (next pointer) kullanılır. Uygulamada ayrıca sembolik FreeRTOS TaskHandle_t tutulabilir.

4.2 Kuyruk yapısı ve kuyruk işlemleri

Her öncelik seviyesi için ayrı hazır kuyruk bulunur. Kuyruklar temel olarak enqueue/dequeue operasyonları ile yönetilir. Ayrıca her saniye, kuyruktaki görevlerin bekleme_suresi artırılır; belirli eşiği aşan görevler zaman aşımı sayılarak sistemden çıkarılır.

4.3 Bellek ve kaynak tahsisi (tahsis/gönderme/geri verme)

Görevlendirici tarafında kaynak yönetimi üç adımda ele alınır:

- Tahsis: Yeni görev geldiğinde görev yapısı için bellek ayrılır ve ilgili kuyruğa eklenir.
- Gönderme: Scheduler seçim yaptıktan sonra görev “çalışıyor” durumuna alınır; 1 saniyelik dilim çalıştırılır.
- Geri verme: Görev bittiğinde (kalan_sure=0) veya zaman aşımına uğradığında kuyruktan çıkarılır, handle silinir ve bellek serbest bırakılır.

Not: Gerçek RTOS’larda determinism için dinamik bellek yerine çoğu zaman statik bellek tercih edilir. Bu projede simülasyon amacıyla dinamik tahsis kullanılmıştır.

4.4 Çalışma kuralları (özet)

- RT (öncelik 0): Kuyruk doluyken her zaman önce seçilir ve bitene kadar kesilmeden yürütülür (FCFS).
- Kullanıcı görevleri (1-3): Kuantum q=1 saniye. Bitmezse askıya alınır ve mümkünse bir alt seviyeye düşürülerek tekrar kuyruğa eklenir.
- Preemption: RT görevi geldiğinde, çalışmakta olan kullanıcı görevi kesilerek RT görevlerine öncelik verilir.
- Aging/Timeout: Uzun süre bekleyen görevler belirli eşikte zaman aşımı kabul edilip silinir.

5) Sonuç ve tartışma: Neden çok düzeyli şema, eksikler ve iyileştirmeler

Çok düzeyli (MLFQ benzeri) şemalar, gerçek işletim sistemlerinde farklı iş türlerini birlikte yönetmek için kullanılır. Kısa/etkileşimli işler hızlı tepki isterken, uzun CPU işleri sistemi tıkamadan alt seviyelere kaydırılır. RT sınıfı ise gecikmeye tolerans göstermediği için ayrı ve daha güçlü bir öncelik mekanizması gerektirir.

5.1 "Gerçek" işletim sistemleri ile karşılaştırma

- Benzerlik: Öncelik tabanlı seçim, hazır kuyruk mantığı, kuantumla zaman dilimleme (RR) ve MLFQ fikri.
- Fark: Gerçek sistemlerde bağlam değişimi, kesmeler ve zamanlayıcı tick'i donanım tarafından sürülür; burada saniye adımı ile simüle edilir.
- Kaynak yönetimi: Gerçek RTOS'larda bellek parçalanmasını azaltmak için statik tahsis/heap stratejileri önemlidir; simülasyonda daha esnek bir model seçilmiştir.

5.2 Eksikler ve olası iyileştirmeler

- Öncelik sınırı: 3'ün altına düşürme sonrası önceliğin 3'te sabitlenmesi (min(öncelik+1, 3)).
- "20 saniye kuralı": Zaman aşımını sadece bekleme değil, toplam yaşam süresi (ready+running) üzerinden takip etmek.
- Çıktı okunabilirliği: Her göreve renk/etiket atanması ve sabit hizalı format kullanılması.
- Daha gerçekçi yaklaşım: Her proses için ayrı FreeRTOS task açmak yerine tek simülasyon görevi + veri yapıları (daha düşük bellek), veya gerçekten FreeRTOS öncelikleriyle çalışan task'lar (daha gerçekçi).

6) Kaynaklar

1. FreeRTOS Resmî Dokümantasyonu (Kernel, Task yönetimi, POSIX port).
2. Ders/ödev dokümanı: FreeRTOS_PC_Scheduler_Odev.pdf.
3. İşletim Sistemleri ders notları: Zamanlama (FCFS, RR, MLFQ), öncelik ve aging kavramları.