SAKARYA
ÜNİVERSİTESİ

FreeRTOS PC Scheduler Odevi

Hazırlayanlar :
| G221210074 | Tunahan Avşar | 1. Öğretim C |
| G221210060 | Mehmet Zahid Görgeç | 2. Öğretim A |
| G221210041 | Anıl Gezertaşar | 1. Öğretim C |
| G221210063 | Talha Kaya | 2. Öğretim A |
| G221210044 | Yunus Emre Sevindik | 1. Öğretim C |

FreeRTOS DÖRT SEVİYELİ ÖNCELİKLİ GÖREVLENDİRİCİ RAPORU

1. GİRİŞ

Gerçek zamanlı işletim sistemleri, görevlerin belirli zaman kısıtlamaları içinde tamamlanmasını garanti eden sistemlerdir. Klasik işletim sistemlerinden farklı olarak, görevlerin ne zaman çalışacağı ve ne kadar sürede tamamlanacağı önceden tahmin edilebilir. FreeRTOS, gömülü sistemler için geliştirilmiş açık kaynaklı bir gerçek zamanlı işletim sistemidir ve küçük bellek ayak izi, hızlı bağlam değişimi ve öncelik tabanlı görevlendirme özellikleri ile yaygın olarak kullanılır.

Bu projede, FreeRTOS çekirdeğini kullanarak dört seviyeli öncelikli bir görevlendirici simülasyonu gerçekleştirdik. Sistemimiz iki ana bileşenden oluşuyor: Priority 0'da çalışan gerçek zamanlı görev kuyruğu (FCFS algoritması ile kesintisiz yürütme) ve Priority 1-2-3'te çalışan üç seviyeli geri beslemeli kullanıcı görev kuyrukları. PC ortamında POSIX portu üzerinden FreeRTOS'un görev yönetim mekanizmalarını kullanarak, gerçek zamanlı görevlendirme mantığını simüle ettik.

2. SİSTEM MİMARİSİ VE TASARIM

Sistemimizde her görev `TaskData` adlı bir yapı ile temsil ediliyor. Bu yapı görevin kimlik numarasını, varış zamanını, öncelik seviyesini, toplam işlem süresini, kalan süreyi ve çeşitli durum bayraklarını saklıyor. Bu sayede bir görevin hangi aşamada olduğunu ve ne kadar süre çalıştığını kolayca takip edebiliyoruz.

Görevlendirici mimarimiz şu şekilde çalışıyor: Priority 0'daki gerçek zamanlı görevler FCFS algoritması ile kesintisiz çalışır ve diğer tüm görevleri kesebilir. Priority 1, 2 ve 3'teki kullanıcı görevleri ise geri beslemeli bir sistemde çalışır. Her görev 1 saniyelik zaman dilimi alır, süre bittiğinde askıya alınır ve önceliği bir seviye düşürülür. Priority 3'e ulaşan görevler bu seviyede kalır ve Round Robin algoritması ile çalışmaya devam eder.

Priority 0: Gerçek Zamanlı Kuyruk (FCFS - Kesintisiz)
↓
Priority 1: Yüksek Öncelikli Kullanıcı Kuyruğu (1 sn quantum)
↓
Priority 2: Orta Öncelikli Kullanıcı Kuyruğu (1 sn quantum)
↓
Priority 3: Düşük Öncelikli Kullanıcı Kuyruğu (Round Robin)

Kuyruk yönetimi için üç temel fonksiyon kullandık: Enqueue görevleri uygun kuyruğa ekliyor, Dequeue kuyruğun başındaki görevi çıkarıyor ve RemoveTaskFromQueue özel durumlar için belirli bir görevi doğrudan kaldırıyor. Bellek yönetiminde statik tahsis kullandık, maksimum 50 görev için önceden ayrılmış bir dizi yapısı var. Bu yaklaşım gerçek zamanlı sistemler için öngörülebilir davranış sağlıyor ama esneklik konusunda sınırlı kalıyor.

3. GÖREV ZAMANLAMA VE YAŞAM DÖNGÜSÜ

Bir görev sistemimizde şu aşamalardan geçiyor: Önce dosyadan okunarak varış zamanında sisteme giriyor, sonra uygun kuyruğa eklenerek hazır duruma geçiyor. Sırası geldiğinde çalışmaya başlıyor, zaman dilimi bittiğinde askıya alınıyor ve önceliği düşürülerek tekrar kuyruğa ekleniyor. Kalan süresi sıfırlanınca veya 20 saniyelik timeout süresi dolunca sonlandırılıyor.

Ana zamanlama algoritması her saniye şu işlemleri yapıyor: Önce o anki zaman adımında sisteme yeni gelen görevler var mı kontrol ediliyor ve varsa uygun kuyruğa ekleniyor. Sonra hangi görevin çalışacağına karar veriliyor - eğer Priority 0'da çalışan bir görev varsa o devam ediyor, yoksa kuyruklar sırayla taranıyor ve ilk boş olmayan kuyruktaki görev seçiliyor. Seçilen görev 1 saniye çalıştırılıyor ve kalan süresi bir azaltılıyor. Eğer görev bittiyse kuyruktan çıkarılıp sonlandırılıyor, bitmediyse ve Priority 0 değilse askıya alınıp önceliği düşürülerek tekrar kuyruğa ekleniyor.

Geri besleme mekanizması şöyle işliyor: Yeni gelen kullanıcı görevleri öncelik değerlerine göre kuyruğa yerleşiyor, her görev 1 saniye çalıştıktan sonra askıya alınıyor ve önceliği bir seviye düşürülüyor. Priority 3'e ulaşan görevler bu seviyede kalıp Round Robin ile çalışmaya devam ediyor. Bu sayede kısa görevler hızlı tamamlanırken, uzun görevler de sistem kaynaklarından adil şekilde yararlanabiliyor.

4. GERÇEK SİSTEMLERLE KARŞILAŞTIRMA VE DEĞERLENDİRME

Linux'un CFS görevlendiricisi sanal çalışma zamanı ve karmaşık veri yapıları kullanırken, bizim sistemimiz sabit kuantum ve basit kuyruklar kullanıyor. Windows'un öncelik sistemi çok daha detaylı ve dinamik öncelik artırma yapabiliyor. Ancak bizim sistemimizin basitliği ve öngörülebilir davranışı, gömülü sistemler için büyük avantaj. Gerçek zamanlı görevlerin kesintisiz çalışması kritik görevlerin zamanında tamamlanmasını garanti ediyor.

Sistemimizin bazı dezavantajları da var. Sürekli yüksek öncelikli görevler gelirse düşük öncelikli görevler uzun süre bekleyebilir, buna açlık (starvation) problemi deniyor. Bunu çözmek için Priority 3'teki görevlere belirli bir süre sonra geçici öncelik artışı uygulanabilir. Ayrıca tüm görevler 1 saniyelik sabit kuantum aldığı için bazı görevler için bu çok kısa bazıları için çok uzun olabiliyor. Statik dizi kullanımı maksimum görev sayısını sınırlıyor, dinamik bellek veya linked list yapıları bu sorunu çözebilir.

Bellek yönetimi konusunda gerçek işletim sistemleri çok daha karmaşık yapılar kullanıyor. Linux'ta buddy allocator ve slab allocator var, Windows heap manager kullanıyor. Biz statik tahsis tercih ettik çünkü gerçek zamanlı sistemlerde dinamik tahsis riskli olabiliyor ve bellek parçalanması problemi yaratıyor. Ancak daha büyük projelerde FreeRTOS'un heap_4 veya heap_5 implementasyonları daha uygun olabilir.

5. SONUÇ

Bu projede dört seviyeli öncelikli bir görevlendiriciyi başarıyla gerçekleştirdik. Priority 0 görevler her zaman kesintisiz çalıştı, kullanıcı görevleri geri besleme mekanizması ile adil şekilde CPU zamanı aldı ve 20 saniyelik timeout mekanizması sonsuz döngüleri engelledi. FreeRTOS'un küçük bellek kullanımı, hızlı bağlam değişimi ve deterministik davranışı sayesinde gerçek zamanlı sistemler için neden bu kadar yaygın kullanıldığını pratik olarak gözlemledik.

Proje sürecinde gerçek zamanlı görevlendirme algoritmalarını derinlemesine öğrendik, çok seviyeli kuyruk yapılarını tasarladık ve öncelik tabanlı sistemlerin avantaj ve dezavantajlarını gözlemledik. Sistem dinamik öncelik ayarlama, I/O görevleri ve semafor/mutex mekanizmaları eklenerek geliştirilebilir. Öncelik tabanlı görevlendirme doğru kullanıldığında hem gerçek zamanlı gereksinimleri karşılar hem de sistem kaynaklarını verimli kullanır.

Github Linki : https://github.com/Ahazzzz/IsletimSistemleriOdevi