T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

SAKARYA
ÜNİVERSİTESİ

2023-2024 Güz Dönemi

İşletim Sistemleri Dersi Proje Raporu

Grup Üyeleri :

Sude ÇAKMAK - B211210012

Dilay KAL – B201210006

Elif Şevval SAÇLI - B211210068

Esma YILDIZ - B211210081

Merve ŞENTÜRK - B211210087

Git-Hub Linki:

https://github.com/cakmaksude/Grup5_IsletimSistemleri

a) Hangi bellek tahsis algoritmaları kullanılabilir?

- Bellek blokları, talep edilen bellek miktarına uygun olarak sırayla taranır ve ilk uygun blok tahsis edilir. Bu algoritma basit ve hızlıdır ancak bellek alanlarının parçalanmasına neden olabilir. (İlk Uygun)
- Talep edilen bellek miktarına en yakın, ancak onu geçmeyecek şekilde en küçük blok seçilir. Bu şekilde parçalanma miktarı azalır, ancak blok seçimi daha karmaşıktır ve işlem süresi uzayabilir. (En Uygun)
- Talep edilen bellek miktarına en yakın ancak onu geçecek şekilde en büyük blok seçilir. Bu yöntem, büyük blokların küçük blokları daha fazla parçalamasına neden olabilir. (En Kötü)
- Sayfalama algoritmaları, sistem belleği ve disk arasında etkili bir denge sağlamak için kullanılır ve farklı algoritmalar farklı durum ve gereksinimlere uygun olarak tercih edilebilir. Fiziksel bellek 4kb, 8kb gibi 2'nin üssü olan sayılara ayrılır ve offset numaraları ile bu sayfaların başlangıç adresleri kaydedilir.

b) Görevlendirici tarafından bellek ve diğer kaynakları kuyruğa almak, göndermek ve tahsis etmek için kullanılan yapıları tanımlayın ve açıklayın.

- **Bellek kuyrukları:** Bellek kuyrukları, sistemdeki bellek kaynaklarını etkili bir şekilde yönetmek için kullanılır. İşletim sistemi, belleği işlemlere tahsis etmek ve geri almak için bellek kuyruklarını kullanır. İşlemlere ayrılan bellek blokları bu kuyruklarda saklanır ve işletim sistemi bu kuyrukları kullanarak bellek kaynaklarını optimize eder.
- **Bellek Tahsisi:** Bellek tahsisi, işletim sisteminin programlara ve işlemlere bellek alanı sağlama sürecidir. Bu, programların çalışması için gerekli olan bellek bloklarını tahsis etmek ve serbest bırakmak anlamına gelir. İşletim sistemi, bellek tahsisini izleyerek, bellek sızıntılarını önler ve bellek kaynaklarını verimli bir şekilde kullanır.
- **Giriş/Çıkış Yönetimi:** Giriş/Çıkış yönetimi, çeşitli giriş ve çıkış aygıtları arasındaki veri transferini kontrol eder. Bu yönetim, dosya sistemleri, ağ iletişimi ve diğer giriş/çıkış operasyonlarını içerir. İşletim sistemi, çeşitli giriş/çıkış kuyrukları kullanarak bu işlemleri sıralar ve düzenler.
- **CPU Yönetimi (İşlemci Kuyrukları, İşlemci Tahsisi):**
    - **İşlemci Kurukları:** İşlemci kuyrukları, işlemcideki işlemleri sıralamak ve önceliklendirmek için kullanılır. İşletim sistemi, işlemciye erişim taleplerini bu kuyruklar aracılığıyla düzenler.
    - **İşlemci Tahsisi:** İşletim sistemi, işlemciyi farklı işlemler arasında adil bir şekilde paylaştırmak için işlemci tahsisini kullanır. İşlemci tahsisinde çeşitli algoritmalar kullanılabilir, örneğin Round Robin gibi.

c) Projenin genel yapısı

Verilen giriş dosyasındaki işlemleri simüle eder ve bu işlemleri gerçek zamanlı ve kullanıcı kuyrukları aracılığıyla işleyerek sonuçları ekrana yazdırır.

FCFS: Kod içerisinde kullanılan Process sınıfının varlığı varsayılmıştır. Bu sınıf, bir işlemin temel özelliklerini içerir (örneğin, işlem numarası, patlama süresi vb.)

processQueue: İşlemleri temsil eden bir kuyruk (Queue) nesnesi

n: Toplam işlem sayısı

burstTimes: İşlemlerin patlama sürelerini içeren bir dizi.

Kod, her bir işlem için işlem numarası, patlama süresi, bekleme süresi ve iş bitirme süresini ekrana yazdırır.

Round Robin:

RoundRobinScheduler sınıfı, gerçek zamanlı ve kullanıcı proses kuyruklarını yönetir. Ayrıca, çevrimsel sıralama algoritmasını uygular ve prosesleri sırayla çalıştırır.

Round Robin algoritması, iki kuyruk kullanır: biri gerçek zamanlı, diğeri kullanıcı prosesleri içindir. Algoritma, gerçek zamanlı prosesleri öncelikle çalıştırır ve ardından kullanıcı proseslerine geçer. Her proses, belirli bir zaman diliminde çalıştırılır ve sırayla işlem yapar.

Dispatchers: Bu kod parçası bir işlem dağıtıcı (dispatcher) uygulamasını temsil eder. İşlem dağıtıcı, bir bilgisayar sistemine gelen işlemleri yöneten bir bileşen olarak düşünülebilir. İşlemler, sistemdeki kaynaklara ihtiyaç duyarlar (örneğin, modem, yazıcı, bellek, tarayıcı, CD sürücüsü vb.), bu nedenle işlemlerin hangi sırayla çalışacakları ve hangi kaynakları kullanabilecekleri konularında bir düzen sağlar.

Kod, farklı öncelik seviyelerine sahip işlemleri yönetmek için beş farklı kuyruk kullanır. Bu kuyruklar şöyle: realTimeQueue, priority1Queue, priority2Queue, priority3Queue, veuserJobQueue. Her bir kuyruk, o kuyruktaki işlemlerin özelliklerine göre gruplandırılmasına yardımcı olur.

allocateProcessResources fonksiyonu, bir işlemin sistem kaynaklarını kullanmasına izin verir ve işlemin bu kaynakları kullanmasını sağlar.

ProcessZamanAsimi fonksiyonu, sistemde belirli bir süre aşan işlemleri kontrol eder ve bunları sonlandırır.

GBG: Proje içerisinde kullanılan Prosess sınıfı bu sınıf içinde kullanıldı. Süre Kuantumu q belirtilen zaman kuantumu, işlemlerin bir öncelik seviyesinden diğerine geçişini kontrol eden bir parametre olarak kullanılmıştır. Düşük öncelikli geri beslemeli görevlendirici yeniden etkinleştirilmeden önce gerçek zamanlı kuyruktaki prosesler bitirilmelidir. GBG kuyruğunun normal çalışması, en yüksek öncelik düzeyindeki prosesleri alır, uygun kuantuma göre işler ve daha sonra önceliğini düşürerek bir alt kuyruğa yerleştirir.
1-Öncelikli Kuyrukların Oluşturulması

2-Süre Kuantumu

3-İşlem İşlemek

4-GBG Algoritması Çalıştırma

5-Boş Kuyruk Kontrolü

Kaynak Tahsisi: Ödevde 2 yazıcı, 2 CD sürücü, 1 modem ve 1 tarayıcı kaynak olarak simüle edilmiştir. Kaynak tahsisi sınıfında bu kaynakların kullanımda olup olmadığını kontrol eder, eğer boştaysa ilgili prosese atar, proses işlemini bitirince kaynakları iade eder.

Bellek Tahsisi: Kaynak tahsisi yönetimi de bellek tahsisinin içinde yapılır. Fiziksel bellek sayfalama algoritması kullanılarak 4 mb'lık parçalara ayrılır. Rastgele bir offset numarası atanır. Gerçek zamanlı prosesler ve kullanıcı prosesleri için ayrı ayrı işlemler yapılır. Gerçek zamanlı proseslere 64mb, kullanıcı proseslerine 960 mb bellek ayrılmıştır. Gelen proses kaç sayfa istediğini belirtir ve yeterli sayıda sayfa varsa prosese ayrılır. Gerçek zamanlı proseslerin ve kullanıcı proseslerinin sayfaları ayrı tutulur. Proses işini bitirdiğinde kullandığı sayfaları iade eder. Eğer gelen prosesin isteği karşılanamıyorsa hata mesajı yazdırılır.

d) Çok düzeyli bir görevlendirme şeması neden kullanılabilir? Öneriler?

Gerçek Zamanlı Prosesslere Öncelik: Bu şema, gerçek zamanlı proseslere öncelik verir ve bu proseslerin kesintiye uğramadan çalışmasını sağlar. Bu, özellikle zaman duyarlı uygulamalar ve sistemler için önemlidir.

Geri Beslemeli Görevlendirme: Düşük öncelikli kullanıcı prosesleri için üç seviyeli geri beslemeli görevlendirme kullanılması, işlem süreçlerini daha etkili bir şekilde yönetebilir. Öncelik düzeyi ve zaman kuantumu kullanılarak işlemlerin öncelikleri dinamik olarak ayarlanabilir.

Kaynak Kısıtlamaları: Sistem, sınırlı kaynakları (yazıcılar, tarayıcı, modem, CD sürücüler, bellek) yönetmek üzere tasarlanmıştır. Bu, kaynak kullanımını optimize etmeye yönelik bir yaklaşımı yansıtır.

Bellek Tahsisi ve Yönetimi: Proseslerin bellek kullanımı kontrol altında tutulmuş ve gerçek zamanlı proseslerin çalışması engellenmemiştir. Bellek tahsisinin prosesler arasında adil bir şekilde yapıldığı bir sistemdir.

Eksiklikler ve İyileştirme Önerileri:

Performans Ölçümleri ve Optimizasyon: Sistem performansını daha iyi anlamak için ölçümler yapılabilir ve performans optimizasyonları yapılabilir. Örneğin, öncelik düzeylerinin dinamik olarak ayarlanması için daha sofistike bir strateji uygulanabilir.

Güvenlik ve Hata Yönetimi: Şema, güvenlik ve hata yönetimi konularına odaklanmamaktadır. Gerçek işletim sistemlerinde bu unsurlar kritik öneme sahiptir ve bu konulara daha fazla odaklanmak gerekebilir.

Kuyruk Yönetimi İyileştirmeleri: Kuyrukların yönetimi, özellikle yüksek talep durumunda daha etkili bir şekilde optimize edilebilir.

Gelişmiş Geri Besleme Mekanizmaları: Geri besleme mekanizmaları, daha karmaşık algoritmalar veya adaptif yöntemlerle iyileştirilebilir.

Dinamik Kaynak Tahsisi: Sistem, kaynakları statik bir şekilde tahsis etmektedir. Dinamik kaynak tahsisi stratejileri, sistemdeki değişken taleplere daha iyi uyum sağlayabilir.

Bu öneriler, sistemdeki eksiklikleri gidermeye ve performansı artırmaya yöneliktir. Gerçek işletim sistemleri genellikle daha karmaşık ve güvenli olacak şekilde tasarlanır, ancak bu proje, temel bir çoklu programlama sistemi simülasyonu sunmaktadır.