GİRESUN ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
İŞLETİM SİSTEMLERİ DERSİ BÜTÜNLEME SINAVI

Adı – Soyadı – Numarası:

Soru 1: Süreç ve iş parçacığı kavramlarını açıklayınız. Farklarını yazınız.

Süreç (Process): Süreç, bir işletim sisteminde çalıştırılan bir programın aktif hâlidir. Süreç, kendi bağımsız adres alanına, kaynaklarına (örneğin, bellek, dosya tanıtıcıları) ve yürütme durumuna sahiptir. İşletim sistemi, süreçleri izole bir şekilde yönetir ve süreçler arasında bağlam anahtarlama (context switching) yaparak çoklu görev sağlar.

Özellikler:

- Kendi sanal adres alanına sahiptir.
- Süreç Denetim Bloğu (PCB) ile yönetilir (program sayacı, yazmaçlar, durum).
- Süreçler, kod, veri, yığın (stack) ve öbek (heap) gibi bölümler içerir.
- Örnek: Bir metin editörü veya web tarayıcı bir süreçtir.

İş Parçacığı (Thread): İş parçacığı, bir sürecin içinde çalışan en küçük yürütme birimidir. Aynı sürecin iş parçacıkları, ortak bir adres alanını ve kaynakları paylaşır, ancak her biri kendi program sayacı, yazmaçlar ve yığınına sahiptir. İş parçacıkları, paralel görevlerin yürütülmesini sağlar.

Özellikler:

- Aynı sürecin bellek alanını paylaşır.
- Hafif süreç (lightweight process) olarak bilinir, çünkü bağlam anahtarlama maliyeti düşüktür.
- Örnek: Bir web tarayıcısında sekme işleme ve ağ istekleri ayrı iş parçacıkları olabilir.

Süreçler, izolasyon ve güvenlik sağlar, ancak kaynak tüketimi yüksektir. İş parçacıkları, verimlilik ve paralellik sunar, ancak paylaşımlı bellek nedeniyle senkronizasyon sorunları riski taşır. Modern sistemler, performans ve güvenlik arasında denge kurmak için her ikisini de kullanır.

Soru 2: Çok iş parçacıklı (multi-threaded) uygulamaların avantaj ve dezavantajları nelerdir?

Avantajlar:

- Paralel Yürütme: İş parçacıkları, birden fazla görevi aynı anda yürütebilir, bu da CPU kullanımını optimize eder. Örnek: Bir web sunucusu, her istemci isteğini ayrı bir iş parçacığında işler, böylece yanıt süreleri kısalır.
- Hızlı Bağlam Anahtarlama: İş parçacıkları, aynı adres alanını paylaştığı için süreçler arası geçişe kıyasla daha hızlı bağlam anahtarlaması yapar. Örnek: Bir video düzenleme yazılımında, arayüz ve render işlemleri ayrı iş parçacıklarında çalışır, kullanıcı deneyimi kesintisiz olur.
- Kaynak Paylaşımı: İş parçacıkları, bellek ve dosya gibi kaynakları paylaştığından kaynak tüketimi düşüktür. Örnek: Bir veritabanı uygulamasında, sorgu işleme iş parçacıkları aynı bellek alanını kullanır, bu da belleği verimli kullanır.
- Duyarlılık (Responsiveness): Kullanıcı arayüzü ve arka plan görevleri ayrı iş parçacıklarında çalışarak daha akıcı bir deneyim sağlar. Örnek: Bir tarayıcıda, sekme yükleme iş parçacığı yavaşlasa bile arayüz duyarlı kalır.
- Çok Çekirdekli Sistemlerde Verimlilik: Modern CPU'ların çok çekirdekli yapısı, iş parçacıklarının paralel çalışmasını destekler. Örnek: Bir oyun motorunda fizik hesaplamaları ve grafik render'ı ayrı çekirdeklerde çalışır.

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 14.06.2025

Dezavantajlar:

- Senkronizasyon Karmaşıklığı: Paylaşılan kaynaklara erişim, yarış durumu (race condition) gibi sorunlara yol açabilir, bu da kilitler (locks) veya semaforlar gerektirir. Örnek: Bir bankacılık uygulamasında, iki iş parçacığı aynı hesaba erişirse, yanlış bakiye güncellemesi olabilir.
- Hata Riski: Bir iş parçacığındaki hata (örneğin, null işaretçi), tüm süreci çökertebilir. Örnek: Bir tarayıcıda, bir sekme iş parçacığındaki hata tüm tarayıcıyı etkileyebilir.
- Kaynak Çatışmaları: İş parçacıkları, paylaşılan kaynaklar için yarışabilir, bu da kilitlenme (deadlock) riskini artırır. Örnek: İki iş parçacığı, birbirinin kilitlediği kaynağı beklerse kilitlenme oluşur.
- Hata Ayıklama Zorluğu: Çok iş parçacıklı uygulamalarda hataları tespit etmek ve düzeltmek zordur, çünkü yürütme sırası öngörülemez. Örnek: Bir iş parçacığının yanlış veri yazması, başka bir iş parçacığında tutarsızlığa yol açabilir.
- Performans Ek Yükü: İş parçacığı yönetimi (oluşturma, senkronizasyon) ek hesaplama maliyeti getirir. Örnek: Çok fazla iş parçacığı, bağlam anahtarlama maliyetini artırabilir.

Soru 3: Sistem çağrısı (system call) nedir? Hangi durumlarda kullanılır?

Sistem Çağrısı (System Call): Sistem çağrısı, bir kullanıcı uygulamasının işletim sistemi çekirdeğinden (kernel) hizmet talep ettiği bir arayüzdür. Kullanıcı modunda çalışan uygulamalar, donanıma doğrudan erişemez veya kritik işlemleri gerçekleştiremez; bu nedenle işletim sisteminin sağladığı sistem çağrılarını kullanır.

Nasıl Çalışır?

- Kullanıcı uygulaması, bir sistem çağrısı (örneğin, `read()`, `write()`) yapar.
- Bu çağrı, bir yazılım kesmesi (software interrupt) tetikler ve CPU, kullanıcı modundan çekirdek moduna geçer.
- Çekirdek, istenen işlemi gerçekleştirir (örneğin, diskten veri okur) ve sonucu kullanıcı uygulamasına döndürür.
- CPU, kullanıcı moduna geri döner ve uygulama devam eder.

Hangi Durumlarda Kullanılır?

- Dosya İşlemleri: Dosya okuma, yazma, oluşturma veya silme gibi işlemler. Örnek: `open()`, `read()`, `write()` ile bir metin dosyasından veri okunur.
- Süreç Yönetimi: Süreç oluşturma, sonlandırma veya süreçler arası iletişim. Örnek: `fork()` yeni bir süreç oluşturur, `exec()` bir programı çalıştırır.
- Bellek Yönetimi: Bellek tahsisi veya serbest bırakma. Örnek: `malloc()` için çekirdek, `brk()` veya `mmap()` çağrısı yapar.
- Ağ İşlemleri: Soket oluşturma, veri gönderme veya alma. Örnek: `socket()`, `send()`, `recv()` ile bir istemci-sunucu bağlantısı kurulur.
- Cihaz Yönetimi: Donanımla iletişim (örneğin, klavye, ekran). Örnek: `ioctl()` ile bir cihazın ayarları yapılandırılır.
- Zaman ve Çizelgeleme: Zamanlayıcı ayarları veya süreç önceliği değiştirme. Örnek: `gettimeofday()` sistem saatini alır.

Soru 4: İşlemci çizelgeleme algoritmalarından 3 tanesi açıklayınız.

1. Round Robin (RR): Süreçlere eşit zaman dilimleri (quantum) atanır ve süreçler bir kuyrukta döngüsel olarak çalıştırılır. Bir sürecin zaman dilimi dolduğunda veya kesildiğinde, kuyruğun sonuna eklenir.

Her süreç, sabit bir süre (örneğin, 10 ms) CPU alır. Zamanlayıcı kesmesi, süreci durdurur ve bir sonrakine geçer.

Avantajlar:
- Adil; her süreç sırayla çalışır.
- İyi yanıt süresi, özellikle etkileşimli sistemlerde.

Dezavantajlar:
- Yüksek bağlam anahtarlama maliyeti, eğer quantum çok küçükse.
- Uzun süren süreçler için tamamlanma süresi artabilir.

Örnek: Bir web sunucusunda, her istemci isteği eşit süre alır, böylece hiçbir istemci uzun süre beklemez.

2. Öncelik Tabanlı Çizelgeleme (Priority Scheduling): Süreçlere öncelik değerleri atanır ve en yüksek öncelikli süreç CPU'yu alır. Öncelik, statik (sabit) veya dinamik (değişen) olabilir.

Öncelik sırasına göre bir kuyruk tutulur. Düşük öncelikli süreçler, yüksek öncelikliler bitene kadar bekler.

Avantajlar:
- Kritik süreçler (örneğin, sistem görevleri) önce çalışır.
- Esnek; öncelikler ihtiyaca göre ayarlanabilir.

Dezavantajlar:
- Açlık (Starvation): Düşük öncelikli süreçler süresiz bekleyebilir.
- Öncelik belirleme karmaşık olabilir.

Örnek: Bir hastane yönetim sisteminde, acil hasta kayıtları yüksek öncelik alır.

3. En Kısa İş Önce (Shortest Job First - SJF): En kısa yürütme süresine sahip süreç önce çalıştırılır. Amaç, ortalama bekleme süresini en aza indirmektir.

Süreçlerin tahmini yürütme süreleri bilinir. En kısa süreli süreç seçilir ve tamamlanana kadar çalışır.

Avantajlar:
- Ortalama bekleme süresi düşüktür (teorik olarak optimal).
- Kısa süreçler hızlı tamamlanır.

Dezavantajlar:
- Uzun süreçler için açlık riski.
- Yürütme süresini doğru tahmin etmek zor.

Örnek: Bir yazıcı kuyruğunda, küçük belgeler önce yazdırılır.

Soru 5: Süreçlerin senkronize çalışmasını sağlayan mekanizmalarını açıklayınız?

1. Kilitler (Locks): Bir kaynak, yalnızca bir süreç veya iş parçacığı tarafından kilitlenir ve diğerleri bekler. En basit senkronizasyon aracıdır.

Türleri:

- Mutex (Mutual Exclusion): Tek bir iş parçacığının kaynağı kilitlemesini sağlar.
- Spinlock: Süreç, kilit serbest kalana kadar döngüde bekler (yoğun sistemlerde kullanılır).

2. Semaforlar (Semaphores): Bir sayaçla kaynak erişimini kontrol eder. İkili semafor (0/1) mutex gibi çalışır; sayaçlı semafor, birden fazla erişime izin verir.

İşlemler:
Wait (P): Sayaç azalır, sıfırsa süreç bekler.
Signal (V): Sayaç artar, bekleyen süreç çalışır.

3. Monitörler (Monitors): Kilit ve durum değişkenlerini (condition variables) birleştiren yüksek seviyeli bir yapıdır. Otomatik kilit yönetimi sağlar.

Özellikler:
- Bir monitör, aynı anda yalnızca bir iş parçacığını çalıştırır.
- Wait/Signal: İş parçacıkları belirli koşulları bekler veya bildirir.

4. Koşul Değişkenleri (Condition Variables): İş parçacıklarının belirli bir koşul sağlanana kadar beklemesini sağlar. Genellikle monitörlerle birlikte kullanılır. Örnek: Bir ağ sunucusunda, veri gelene kadar iş parçacığı bekler.

5. Mesaj Geçişi (Message Passing): Süreçler, paylaşılan bellek yerine mesajlarla iletişim kurar, böylece senkronizasyon dolaylı olarak sağlanır. Örnek: Bir dağıtık sistemde, süreçler kuyruklar aracılığıyla mesaj gönderir.

Soru 6: Bellek yönetiminde mantıksal adresi fiziksel adrese dönüştürmek için kullanılan yöntemler nelerdir?

1. Sayfalama (Paging): Bellek, sabit boyutlu sayfalara (örneğin, 4 KB) bölünür. Mantıksal adres, bir sayfa numarası ve sayfa içi ofset (offset) olarak ayrılır. Sayfa tablosu, sanal sayfa numarasını fiziksel çerçeve numarasına eşler. Harici parçalanmayı önler, süreç izolasyonu sağlar. Sayfa tablosu için ek bellek, dahili parçalanma.

2. Bölümleme (Segmentation): Bellek, değişken boyutlu bölümlere (segment) bölünür (örneğin, kod, veri, yığın). Mantıksal adres, bir bölüm numarası ve ofset içerir. Bölüm tablosu, bölümü fiziksel adrese eşler. Esnek, program yapısına uygun. Harici parçalanma, karmaşık yönetim.

Optimizasyonlar:
Çeviri Önbellek Arabelleği (TLB): Sık kullanılan sayfa eşlemelerini önbelleğe alarak çeviriyi hızlandırır.
Çok Seviyeli Sayfa Tabloları: Büyük adres alanlarında (örneğin, 64-bit) bellek kullanımını azaltır.
Ters Sayfa Tabloları: Fiziksel çerçeveleri indeksleyerek sayfa tablosu boyutunu sınırlar.

Soru 7: Sanal bellek (virtual memory) kullanımının avantajları nelerdir?

Süreç İzolasyonu: Her süreç, kendi sanal adres alanında çalışır, böylece diğer süreçlerin verilerine erişemez. Bu, güvenliği ve sistem stabilitesini artırır. Örnek: Bir tarayıcı, bir bankacılık uygulamasının belleğine erişemez.

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 14.06.2025

Büyük Adres Alanı: Süreçler, fiziksel bellekten daha büyük bir adres alanı kullanabilir. Takas alanı, fiziksel belleği genişletir. Örnek: 4 GB RAM'de, bir süreç 8 GB bellek talep edebilir.

Verimli Bellek Kullanımı: Talebe bağlı sayfalama (demand paging), yalnızca ihtiyaç duyulan sayfaları belleğe yükler, böylece RAM verimli kullanılır. Örnek: Bir veritabanı uygulamasında, yalnızca aktif sorguların sayfaları RAM'de tutulur.

Bellek Paylaşımı: Süreçler, ortak kütüphaneleri veya verileri paylaşabilir (örneğin, kopya üzerine yazma - copy-on-write). Örnek: Birden fazla süreç, aynı dinamik kütüphaneyi (örneğin, libc) paylaşır.

Harici Parçalanmanın Önlenmesi: Sayfalama, belleği sabit boyutlu sayfalara böler, böylece dağınık boş alanlar sorunu ortadan kalkar. Örnek: Değişken boyutlu tahsisin aksine, sayfalar herhangi bir sırayla tahsis edilebilir.

Esnek Süreç Yönetimi: Süreç oluşturma (fork) ve bellek tahsisi (malloc) gibi işlemler, sanal bellekle daha hızlı ve esnek hale gelir. Örnek: Fork işleminde, kopya üzerine yazma ile bellek kopyalaması geciktirilir.

Hata Koruması: Geçersiz bellek erişimleri (örneğin, null işaretçi), işletim sistemi tarafından tespit edilir ve segmentasyon hatası üretilir. Örnek: Bir program, tanımlı olmayan bir adrese erişirse çökmesi önlenir.

Soru 8: Sayfa tablosu (page table) yöntemi 64 bit sistemlerde çalışır mı? Açıklayınız.

1. 64-Bit Adres Alanının Boyutu: 64-bit sistemlerde teorik adres alanı 2^{64} bayt kadardır. Ancak, pratikte tüm adres alanı kullanılmaz (örneğin, modern sistemler 48-bit etkili adres kullanır). Buna rağmen, sayfa tablosu boyutları büyük bir sorun oluşturabilir:

Örnek: 4 KB'lik sayfalarla, 48-bit adres alanı (2^{48} bayt) 2^{36} (68 milyar) sayfa gerektirir. Her sayfa girişi 8 bayt ise, tek bir süreç için sayfa tablosu 512 GB olur.
Sorun: Bu büyüklükteki bir tabloyu fiziksel bellekte tutmak imkansızdır.

2. Çok Seviyeli Sayfa Tabloları: 64-bit sistemlerde, sayfa tablosu boyutunu yönetmek için çok seviyeli sayfa tabloları (multi-level page tables) kullanılır:

Nasıl Çalışır:
Sanal adres, birden fazla seviye için indekslere bölünür (örneğin, 4 seviyeli tablo).
Her seviye, bir sonraki seviyenin tablosuna işaret eder; son seviye fiziksel çerçeveyi belirtir.
Örnek: x86-64'te, 48-bit adres 4 seviyeye bölünür (PML4, PDPT, PD, PT).
Avantajlar:
Yalnızca kullanılan adres alanları için tablo girişleri oluşturulur, böylece bellek kullanımı azalır.
Boş adres alanları için tablo tutulmaz.
Örnek: Bir süreç yalnızca 1 GB kullanıyorsa, yalnızca bu alan için sayfa tablosu girişleri oluşturulur.

3. Ters Sayfa Tabloları (Inverted Page Tables): Bazı 64-bit sistemler, sayfa tablosu boyutunu daha da azaltmak için ters sayfa tabloları kullanır:
Fiziksel çerçeveler indekslenir ve her çerçeve, bir sanal adrese işaret eder.
Avantaj: Tablo boyutu, fiziksel bellek miktarına bağlıdır, sanal adres alanına değil.

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 14.06.2025