Bölüm 17: Soru Cevap

İşletim Sistemleri

Soru

- İşletim sistemi çekirdeğinin amacı nedir?

- A) Kullanıcılara arayüz sağlamak
- B) Donanım kaynaklarını yönetmek
- C) Uygulama yazılımlarını yürütmek
- D) Kullanıcı verilerini depolamak

Cevap

- Cevap: B

- İşletim sistemi çekirdeği, donanım kaynaklarını yönetmekten sorumludur. Bu, bellek, işlemci, disk ve diğer donanım bileşenlerinin etkin ve güvenilir bir şekilde kullanılmasını sağlar. Çekirdek, donanım ve yazılım arasında iletişimi sağlar, giriş/çıkış işlemlerini yönetir, bellek yönetimini gerçekleştirir ve işlemciler arasında zamanlama yapar. Bu nedenle, işletim sistemi çekirdeği, bilgisayar sistemlerinin temel işlevselliğini sağlayan kritik bir bileşendir.

Soru

- Aşağıdakilerden hangisi işletim sisteminin bir fonksiyonu değildir?

- A) Süreç yönetimi
- B) Bellek yönetimi
- C) Ağ yönlendirme
- D) Dosya sistemi yönetimi

Cevap

- Cevap: C

- İşletim sistemi, süreç yönetimi, bellek yönetimi ve dosya sistem yönetimi gibi bir dizi temel görevi yerine getirir. Ancak, ağ yönlendirme işlevi genellikle işletim sistemi tarafından değil, ağ yönlendiricileri veya benzer ağ cihazları tarafından gerçekleştirilir.

Soru

- İşletim sisteminde çizelgeleyicinin (scheduler) temel sorumluluğu nedir?

- A) Bellek kaynaklarını tahsis etmek
- B) Giriş/çıkış işlemlerini yönetmek
- C) CPU'ya atanacak bir sonraki süreci belirlemek
- D) Kullanıcı kimlik doğrulamasını sağlamak

Cevap

- Cevap: C

- Çizelgeleyici (scheduler), işletim sistemindeki temel görevlerden biri olan CPU zamanının etkin bir şekilde kullanılmasını sağlar. Bu görev, bir sonraki çalıştırılacak süreci belirleyerek, işlemciye hangi işin verileceğini belirlemekle ilgilidir. Bu şekilde, sistemdeki farklı süreçler arasında adil bir paylaşım sağlanır ve işlemcideki boş zaman minimumda tutulur.

Soru

- Bir işletim sisteminde aygıt sürücüsünün amacı nedir?

- A) Çevre birimlerini yönetmek
- B) Bellek kaynaklarını tahsis etmek
- C) CPU görevlerini zamanlamak
- D) Yüksek seviye programlama dillerini yorumlamak

Cevap

- Cevap: A

- Aygıt sürücüsü, işletim sistemi ile donanım arasında iletişim sağlayan bir yazılım bileşenidir. Temel amacı, çeşitli donanım aygıtlarını (örneğin, yazıcı, fare, klavye, disk sürücüleri vb.) işletim sistemi ile uyumlu hale getirmek ve bu aygıtların doğru şekilde çalışmasını sağlamaktır.

Soru

- Modern bilgisayar sistemlerinde CPU önbelleğinin temel işlevi nedir?

- A) Uygulama verilerini geçici olarak depolamak
- B) CPU ve ana bellek arasındaki veri erişimini hızlandırmak
- C) Çevre birimlerini yönetmek
- D) Makine komutlarını yürütmek

Cevap

- Cevap: B

- CPU önbelleği, işlemcinin ana bellek (RAM) ile iletişimini hızlandırmak için kullanılan küçük ve yüksek hızlı bellek birimidir. Ana belleğe göre çok daha hızlıdır ve sıkça kullanılan verileri veya komutları depolayarak CPU'nun erişim sürelerini azaltır. Bu, işlemcinin performansını artırır ve sistem genelinde veri erişimini iyileştirir.

Soru

- Bir bilgisayar sisteminde, kesmeleri yönetmek ve donanım işlemlerini koordine etmekten sorumlu olan bileşen hangisidir?

- A) Merkezi İşlem Birimi (CPU)
- B) Rastgele Erişimli Bellek (RAM)
- C) Giriş/Çıkış Denetleyicisi (I/O Controller)
- D) Güç Kaynağı Ünitesi (PSU)

Cevap

- Cevap: C

- Giriş/Çıkış Denetleyicisi (I/O Controller), bilgisayar sisteminde harici cihazlarla iletişimi sağlayan ve kesmeleri yöneten bir bileşendir. Bu denetleyiciler, ana işlemcinin (CPU) yükünü azaltarak ve donanım işlemlerini koordine ederek sistem performansını artırır.

Soru

- Bilgisayarın başlatma işlemi sırasında BIOS (Temel Giriş/Çıkış Sistemi), hangi rolü oynar?

- A) Bellek kaynaklarını tahsis etmek
- B) İşletim sistemi çekirdeğini belleğe yüklemek
- C) Başlangıç sırasında donanım bileşenlerini başlatmak
- D) Dosya sistemlerini yönetmek

Cevap

- Cevap: C

- BIOS (Temel Giriş/Çıkış Sistemi), bilgisayarın başlatma sürecinin ilk adımında donanım bileşenlerini başlatmak ve sistem için temel ayarları yapmakla sorumludur. Bu, işletim sisteminin yüklenmeden önce ana donanım bileşenlerinin doğru şekilde çalışmasını sağlar.

Soru

- Aşağıdakilerden hangisi modern bilgisayar sistemlerinde yaygın olarak kullanılan bir tür kararlı bellek (non-volatile memory) tipi DEĞİLDİR?

- A) Sabit Disk Sürücüsü (HDD)
- B) Katı Hal Sürücüsü (SSD)
- C) Manyetik Bant
- D) Dinamik Rastgele Erişimli Bellek (DRAM)

Cevap

- Cevap: D

- Dinamik Rastgele Erişimli Bellek (DRAM), kararlı olmayan (volatile) bellek türündendir ve güç kesildiğinde verileri korumaz. Diğer seçenekler olan Sabit Disk Sürücüsü (HDD), Katı Hal Sürücüsü (SSD) ve Manyetik Bant ise kararlı bellek türleridir ve veriler güç kaynağı kesilse bile korunabilir.

Soru

- Modern bilgisayar mimarilerinde CPU'yu ana belleğe bağlamak için yaygın olarak hangi veri yol standardı kullanılır?

- A) SATA
- B) PCIe
- C) USB
- D) DDR

Cevap

- Cevap: D

- DDR (Double Data Rate), modern bilgisayar mimarilerinde CPU ile ana bellek arasındaki veri iletişimini sağlamak için yaygın olarak kullanılan bir veri yol standardıdır. SATA, depolama aygıtlarını bağlamak için kullanılan bir arabirim standardıdır. PCIe, genişleme kartları ve diğer cihazları bağlamak için kullanılan bir yüksek hızlı seri veri yoludur. USB ise çeşitli harici cihazları bağlamak için kullanılan bir arayüz standardıdır.

Soru

- Bir işletim sisteminde kabuğun (shell) temel işlevi nedir?

- A) Donanım kaynaklarını yönetmek
- B) Çekirdeğe erişim için kullanıcı arayüzü sağlamak
- C) Çevre birimlerini kontrol etmek
- D) Makine kodu komutlarını yürütmek

Cevap

- Cevap: B

- Kabuk, kullanıcıların işletim sistemiyle etkileşime girmelerini sağlar. Kullanıcıların komutları girip çalıştırmasına, dosyaları yönetmesine ve sistem kaynaklarını kullanmasına izin verir.

Soru

- Bir dizinin içeriğini listelemek için hangi komut kullanılır?

- A) cd
- B) mv
- C) ls
- D) rm

Cevap

- Cevap: C

- "ls" komutu, bir dizinin içeriğini listelemek için kullanılır.

Soru

- GUI'nin (Grafiksel Kullanıcı Arayüzü) temel amacı nedir?

- A) Çekirdek ile etkileşim kurmak
- B) Kullanıcılar için metin tabanlı bir arayüz sağlamak
- C) Kullanıcıların grafiksel öğeler aracılığıyla etkileşimini kolaylaştırmak
- D) Donanım kaynaklarını yönetmek

Cevap

- Cevap: C

- Bir GUI, kullanıcıların bilgisayar sistemleriyle etkileşimde bulunmasını sağlayan grafiksel bir arayüzdür. Grafiksel öğeler, pencereler, menüler, düğmeler vb. aracılığıyla kullanıcılarla etkileşim kurulmasını sağlar. Kullanıcılar bu arayüz sayesinde dosyaları yönetebilir, uygulamaları çalıştırabilir, ayarları yapılandırabilir ve daha birçok işlemi gerçekleştirebilir.

Soru

- Aygıt Yöneticisinin (Device Manager) temel işlevi nedir?

- A) Kullanıcı hesaplarını yönetmek
- B) Ağ trafiğini izlemek
- C) Donanım cihazlarını yapılandırmak ve sorun gidermek
- D) CPU performansını optimize etmek

Cevap

- Cevap: C

- Aygıt Yöneticisi, bir işletim sisteminde donanım cihazlarını yönetmek, yapılandırmak ve sorun gidermekten sorumlu araçtır. Bu araç, bilgisayar donanımını tanımlar, yükler ve yönetir. Donanım sürücülerini güncelleme, eksik veya hatalı donanımı tanıma ve sorunları giderme gibi işlevleri de içerir.

Soru

- Aygıt Yöneticisi bağlamında aygıt sürücülerinin amacı nedir?

- A) Aygıtları fiziksel olarak bilgisayara bağlamak
- B) Donanım bileşenleri için güç ayarlarını yönetmek
- C) Donanım aygıtlarıyla etkileşim için yazılım arayüzleri sağlamak
- D) Cihaz performansını optimize etmek

Cevap

- Cevap: C

- Aygıt sürücüleri, bir işletim sistemindeki Aygıt Yöneticisi tarafından kullanılan yazılım bileşenleridir. Bu sürücüler, donanım aygıtlarıyla iletişim kurmak, tanımak, yönetmek ve kontrol etmek için gereken yazılım arayüzlerini sağlar.

Soru

- Çekirdek (kernel) ile kullanıcı (user) modu arasındaki temel fark nedir?

- A) Çekirdek modu doğrudan donanım kaynaklarına erişime izin verirken, kullanıcı modu böyle bir erişimi kısıtlar.
- B) Kullanıcı modu, çekirdek moduna kıyasla sistem yöneticilerine daha yüksek ayrıcalıklar sağlar.
- C) Çekirdek modu, kullanıcı uygulamalarını sistem işlemlerinin önüne geçirirken, kullanıcı modu sistem işlemlerini önceliklendirir.
- D) Kullanıcı modu sistem belleğini yönetirken, çekirdek modu giriş/çıkış işlemlerini yönetir.

Cevap

- Cevap: A

- Çekirdek (kernel) modu, işletim sisteminin en temel ve en yüksek ayrıcalıklı modudur. Bu mod, doğrudan donanım kaynaklarına erişim sağlar ve işletim sistemi çekirdeği tarafından çalıştırılan kritik işlemleri gerçekleştirir. Kullanıcı (user) modu ise, kullanıcıların uygulamalarının çalıştığı moddur ve sınırlı erişim yetkilerine sahiptir.

Soru

- Doğrudan donanım kaynaklarına erişimi kısıtlayan ve güvenlik amaçları için işlem izolasyonunu sağlayan mod hangisidir?

- A) Çekirdek modu
- B) Kullanıcı modu
- C) Denetleyici modu
- D) Korunan mod

Cevap

- Cevap: B

- Kullanıcı modu (User mode), bir işletim sisteminde kullanıcıların uygulamalarının çalıştığı ve sınırlı erişim yetkilerine sahip olduğu moddur. Bu modda, doğrudan donanım kaynaklarına erişim kısıtlanır ve güvenlik amacıyla süreç izolasyonu sağlanır. Kullanıcı modunda çalışan uygulamalar, işletim sistemi tarafından belirlenen sınırlar içinde çalışır ve birbirlerinden ve işletim sistemi çekirdeğinden izole edilirler. Bu sayede, sistem güvenliği artırılır ve istenmeyen erişim ve zararlı yazılım saldırıları engellenir.

Soru

- Bir işletim sisteminde ayrı çekirdek ve kullanıcı modlarının kullanılmasının temel avantajı nedir?

- A) Aygıt sürücülerini yazma işlemini basitleştirir.
- B) Kritik görevleri kullanıcı uygulamalarından izole ederek sistem istikrarını ve güvenliğini artırır.
- C) Kullanıcı programlarının daha hızlı yürütülmesine olanak tanır.
- D) Bellek yönetiminin karmaşıklığını azaltır.

Cevap

- Cevap: B

- Ayrı çekirdek ve kullanıcı modları, işletim sisteminde kritik görevleri (çekirdek modu) kullanıcı uygulamalarından (kullanıcı modu) ayırarak sistem istikrarını ve güvenliğini artırır. Çekirdek modu, işletim sisteminin en temel ve en yüksek ayrıcalıklı modudur ve doğrudan donanım kaynaklarına erişim sağlar. Bu mod, kritik işlemleri gerçekleştirir ve işletim sistemi çekirdeği tarafından çalıştırılır. Kullanıcı modu ise, kullanıcıların uygulamalarının çalıştığı ve sınırlı erişim yetkilerine sahip olduğu moddur. Kullanıcı modunda çalışan uygulamalar, işletim sistemi tarafından belirlenen sınırlar içinde çalışır ve birbirlerinden ve işletim sistemi çekirdeğinden izole edilirler.

Soru

- Bir mikroişlemcinin iç yapısı ve tasarımını ifade eden terim hangisidir?

- A) Fiziksel Aygıtlar
- B) Mikromimari
- C) Makine Dili
- D) İşletim Sistemi

Cevap

- Cevap: B

- Mikromimari (Microarchitecture), bir mikroişlemcinin iç yapısal özelliklerini ve tasarımını ifade eder. Bu, mikroişlemcinin içindeki işlem birimleri, veri yolları, bellek yönetimi, komut kümesi ve diğer önemli bileşenleri kapsar. Mikromimari, mikroişlemcinin performansını, güç tüketimini, sıcaklık yönetimini ve diğer özelliklerini belirleyen önemli bir faktördür.

Soru

- CPU tarafından doğrudan yürütülebilen ikili komutları içeren dil hangisidir?

- A) Yüksek seviyeli dil
- B) Çevirici dil
- C) Makine dili
- D) Komut dosyası dili

Cevap

- Cevap: C

- Makine dili (Machine language), bilgisayarın anlayabileceği en temel dil olarak bilinir. Bu dil, ikili formatta yani sıfırlar ve birler olarak yazılmış talimatlar içerir. Her bir ikili kod, belirli bir işlemi veya komutu temsil eder. Makine dili, doğrudan işlemcinin yürütebileceği en temel seviyede talimatları içerir.

Soru

- Aşağıdakilerden hangisi mikromimarinin özelliklerinden DEĞİLDİR?

- A) Komut kümesi tasarımı
- B) İşlemci organizasyonu
- C) Önbellek yönetimi
- D) Dosya sistemi yapısı

Cevap

- Cevap: D

- Mikromimarinin temel özellikleri, işlemci içindeki komut seti tasarımı, işlemci organizasyonu ve önbellek yönetimi gibi donanım düzeyindeki özelliklerdir. Dosya sistemi yapısı ise bir işletim sistemi konseptidir ve işlemci mikromimarisinin bir parçası değildir.

Soru

- CPU'ye en yakın ve en küçük kapasiteye sahip olan önbellek hangisidir?

- A) L1 önbelleği
- B) L2 önbelleği
- C) L3 önbelleği
- D) Sanal Bellek

Cevap

- Cevap: A

- L1 önbellek (Level 1 cache), CPU'ye en yakın olan ve genellikle en küçük kapasiteye sahip olan önbellek seviyesidir. L1 önbelleği, CPU çekirdeklerine doğrudan bağlıdır ve işlemcinin hızlı erişim sağlamak için kullandığı küçük bir yüksek hızlı bellek birimidir.

Soru

- Programcıların makine talimatlarını temsil etmek için mnemonik kodlarını kullanmalarına izin veren dil hangisidir?

- A) Yüksek seviyeli dil
- B) Çevirici dil
- C) Makine dili
- D) Komut dosyası dili

Cevap

- Cevap: B

- Çevirici dil (Assembly language), programcıların makine talimatlarını temsil etmek için kullanabilecekleri mnemonik kodları içeren bir düşük seviyeli programlama dilidir. Bu mnemonik kodlar, makine dilindeki talimatları daha anlaşılır hale getirir ve programcıların daha kolay anlamasını sağlar. Çevirici dil, bir derleyici veya çevirici aracılığıyla makine diline çevrilir ve bilgisayar tarafından doğrudan yürütülebilir.

Soru

- İşletim sistemleri bağlamında soyutlama nedir?

- A) Bilgisayar sistemlerinin fiziksel bileşenlerine işaret eder.
- B) Karmaşık detayları gizlemeyi ve kaynakların basitleştirilmiş bir görünümünü sağlar.
- C) CPU performansını optimize etmeye odaklanır.
- D) Donanım ve yazılım arasındaki etkileşimle ilgilidir.

Cevap

- Cevap: B

- Soyutlama, karmaşık detayları gizlemeyi ve kullanıcılara ve uygulamalara kaynakların basitleştirilmiş bir görünümünü sağlamayı içerir. Bu, kullanıcıların ve uygulamaların işletim sistemi kaynaklarına erişimini yönetmeyi ve karmaşık yapıları gizlemeyi içerir. Örneğin, dosya sistemi soyutlaması, kullanıcılara dosyaları ve dizinleri yönetmek için bir arayüz sağlar ve fiziksel depolama cihazlarının karmaşık detaylarını gizler. Bu şekilde, kullanıcılar ve uygulamalar, kaynaklara erişirken karmaşık detaylarla uğraşmak zorunda kalmaz.

Soru

- İşletim sistemlerinde soyutlama, kullanıcılara ve geliştiricilere ne sağlar?

- A) Donanım mimarisine detaylı bakış açısı
- B) Basitleştirilmiş ve tutarlı arayüzler
- C) Düşük seviyeli sistem kaynaklarına erişim
- D) Gerçek zamanlı performans izleme

Cevap

- Cevap: B

- Soyutlama, işletim sistemlerinde kullanıcılara ve geliştiricilere basitleştirilmiş ve tutarlı arayüzler sağlar. Bu, kullanıcıların ve geliştiricilerin karmaşık detaylarla uğraşmadan kaynaklara erişmelerini ve işletim sistemi hizmetlerinden yararlanmalarını sağlar. Örneğin, bir dosya sistemi arayüzü, kullanıcılara dosyaları yönetmek için basit ve tutarlı bir yol sunar, ancak bu, dosyaların fiziksel olarak nasıl depolandığına veya yönetildiğine dair ayrıntıları gizler.

Soru

- Uygulama programlarından fiziksel bellek konumlarını gizleyen soyutlama hangisidir?

- A) CPU çizelgeleme
- B) Süreç yönetimi
- C) Sanal bellek
- D) Dosya sistemi

Cevap

- Cevap: C

- Sanal bellek, uygulama programlarına fiziksel bellek konumlarını gizleyen bir soyutlamadır. Bu, uygulama programlarının bellek üzerinde çalışırken fiziksel bellek adreslerini doğrudan kullanmasını engeller. Bunun yerine, her bir uygulama programı, kendine ayrılmış sanal bellek adreslerini kullanır ve işletim sistemi bu sanal adresleri fiziksel bellek adreslerine eşler. Bu sayede, uygulama programlarının birbirlerini veya işletim sistemini etkilemeden bellek üzerinde çalışması sağlanır.

Soru

- Tek bir CPU üzerinde birden fazla işlemin eş zamanlı olarak çalışmasına olanak tanıyan soyutlama hangisidir?

- A) Dosya sistemi
- B) Süreç yönetimi
- C) Aygıt sürücüleri
- D) Ağ protokolleri

Cevap

- Cevap: B

- Süreç yönetimi, tek bir CPU üzerinde birden fazla sürecin eş zamanlı olarak çalışmasına olanak tanıyan bir soyutlamadır. İşletim sistemi, süreç yönetimi aracılığıyla süreçleri oluşturur, yönetir, zamanında çalıştırır ve sonlandırır. Bu, birden çok işlemin aynı anda çalışmasını ve CPU kaynaklarının etkin bir şekilde kullanılmasını sağlar.

Soru

- İşletim sistemlerinde dosya sistemi soyutlamasının amacı nedir?

- A) Ağ bağlantılarını yönetmek
- B) Verileri depolamak ve erişmek için birleşik bir arayüz sağlamak
- C) CPU kaynaklarını kontrol etmek
- D) Bellek tahsisini optimize etmek

Cevap

- Cevap: B

- Dosya sistemi soyutlaması, işletim sistemi tarafından verileri depolamak ve erişmek için birleşik bir arayüz sağlamak amacıyla kullanılır. Bu soyutlama, kullanıcıların ve uygulamaların dosyaları oluşturmasını, okumasını, yazmasını ve silmesini sağlar. Ayrıca, dosya sistemi, dosyaları organize eder, depolama cihazlarına erişimi yönetir ve dosyaları korur. Bu sayede, kullanıcılar ve uygulamalar, dosyaları işlemek için dosya sistemi aracılığıyla tek bir arayüz kullanabilirler.

Soru

- Aygıtlar ve CPU arasındaki giriş/çıkış işlemlerini yönetmekten sorumlu soyutlama hangisidir?

- A) Süreç yönetimi
- B) CPU çizelgeleme
- C) Aygıt sürücüleri
- D) Sanal bellek

Cevap

- Cevap: C

- Aygıt sürücüleri, işletim sisteminde aygıtlarla (örneğin, yazıcı, klavye, disk sürücüleri) ve CPU arasındaki giriş/çıkış işlemlerini yönetmekten sorumlu olan soyutlamadır. Aygıt sürücüleri, işletim sistemi ve donanım arasında arabirim görevi görür ve kullanıcı programlarına veya işletim sistemine gelen giriş/çıkış taleplerini işler.

Soru

- Kullanıcıların depolama cihazlarında depolanan verileri düzenlemesine ve işlemesine olanak tanıyan soyutlama hangisidir?

- A) CPU çizelgeleme
- B) Sanal bellek
- C) Dosya sistemi
- D) Süreç yönetimi

Cevap

- Cevap: C

- Dosya sistemi, kullanıcıların depolama cihazlarında depolanan verileri düzenlemesine ve işlemesine olanak tanıyan bir soyutlamadır. Dosya sistemi, dosyaları organize eder, depolama cihazlarına erişimi yönetir ve dosyaları korur. Kullanıcılar ve uygulamalar, dosyaları oluşturabilir, okuyabilir, yazabilir, silip yeniden adlandırabilir veya taşıyabilirler.

Soru

- Çevre birimi (peripheral) cihazların fiziksel özelliklerini uygulama yazılımından gizleyen soyutlama hangisidir?

- A) Sanal bellek
- B) CPU çizelgeleme
- C) Aygıt sürücüleri
- D) Dosya sistemi

Cevap

- Cevap: C

- Aygıt sürücüleri, çevre birimi cihazların fiziksel özelliklerini uygulama yazılımından gizleyen bir soyutlamadır. Bu, uygulama yazılımının belirli bir aygıtla iletişim kurarken, cihazın fiziksel yapısı veya bağlantı protokolleri gibi detaylarla uğraşmasını engeller. Aygıt sürücüleri, işletim sistemi ve donanım arasında arabirim görevi görür. Bu sayede, uygulama yazılımı, belirli bir cihaz türüne özgü detaylarla uğraşmak zorunda kalmaz ve farklı cihazlar arasında kolayca geçiş yapabilir.

Soru

- Linux'taki "proc" klasörünün amacı nedir?

- A) Sistem ikili dosyalarını içerir.
- B) Kullanıcı uygulamalarının yapılandırma dosyalarını saklar.
- C) Çalışan süreçler ve sistem kaynakları hakkında bilgi sağlar.
- D) Kullanıcı ana dizinlerine ev sahipliği yapar.

Cevap

- Cevap: C

- "proc" dizini, Linux işletim sisteminde çalışan süreçler ve sistem kaynakları hakkında bilgi sağlar. Bu klasör altında, süreç numaraları, bellek kullanımı, CPU kullanımı, ağ istatistikleri gibi birçok sistem ve süreç bilgisi yer alır. "proc", gerçekte bir dizin değil, çalışan süreçler ve sistem bilgilerini dinamik olarak oluşturan bir sanal dizindir.

Soru

- "proc" klasöründeki "/proc/sys" alt dizini neyi içerir?

- A) Çeşitli sistem hizmetleri için yapılandırma dosyalarını içerir.
- B) Çalışan süreçler hakkında bilgi.
- C) Çekirdek parametreleri ve ayarlar.
- D) Kullanıcı ana dizinlerini içerir.

Cevap

- Cevap: C

- "/proc/sys" alt dizini, Linux işletim sistemi çekirdeğiyle ilgili parametreleri ve ayarları içerir. Bu alt dizin altında, çeşitli sistem ayarlarına ilişkin dosyalar bulunur ve bu dosyalar, sistem davranışını değiştirmek için kullanılabilir. Örneğin, bellek yönetimi, ağ yapılandırması, güvenlik ayarları gibi çeşitli sistem parametreleri bu dizin altında bulunabilir.

Soru

- "proc" klasöründeki "/proc/[pid]" dizinlerinin amacı nedir?

- A) Bağlanmış dosya sistemleri hakkında bilgi içerirler.
- B) Süreç kimlikleri (PIDs) tanımlanan süreçler hakkında bilgi sağlarlar.
- C) Sistem günlüklerini saklarlar.
- D) Sistem genelinde çekirdek parametrelerini içerirler.

Cevap

- Cevap: B

- "/proc/[pid]" dizinleri, süreç kimlikleri (PIDs) tanımlanan süreçler hakkında ayrıntılı bilgi sağlar. Her bir "/proc/[pid]" dizini, o süreçle ilgili çeşitli bilgiler içerir. Örneğin, bellek kullanımı, CPU kullanımı, dosya tanımları, süreç durumu gibi bilgiler bu dizin altında bulunur. Bu dizinler, sistem üzerinde çalışan her bir süreç için ayrı bir dizin oluşturur ve bu dizinler aracılığıyla sistem tarafından süreç hakkında bilgi alınabilir veya etkileşimde bulunulabilir.

Soru

- "proc" klasöründeki hangi dosya, çekirdek sürümü ve derleme parametreleri hakkında bilgi sağlar?

- A) /proc/version
- B) /proc/net
- C) /proc/mounts
- D) /proc/filesystems

Cevap

- Cevap: A

- "/proc/version" dosyası, Linux çekirdeğinin sürüm numarası ve derleme parametreleri gibi bilgileri içerir.

Soru

- Von Neumann mimarisinde, komutlar ve veriler nerede saklanır?

- A) Ayrı bellek birimlerinde
- B) CPU önbelleğinde
- C) Yazmaçlarda
- D) Aynı bellek biriminde

Cevap

- Cevap: D

- Von Neumann mimarisinde, komutlar ve veriler aynı bellek biriminde saklanır. Bu bellek birimi, hem program kodunu (talimatlar) hem de programın çalıştığı verileri içerir. Bu nedenle, program kodu ve veriler aynı bellek biriminde saklanır ve işlemci bu bellek biriminden talimatları alır ve verilere erişir.

Soru

- Von Neumann mimarisinin hangi yönü, performans açısından potansiyel bir darboğaz oluşturur?

- A) Ayrı depolama ve işlem birimleri
- B) Entegre grafik işleme
- C) Paralel işleme yeteneği
- D) Bellek ile CPU arasındaki sınırlı bant genişliği

Cevap

- Cevap: D

- Von Neumann mimarisinde, bellek ve işlemci arasındaki iletişim bellek ile işlemci arasındaki bant genişliği ile sınırlıdır. Bu, işlemcinin bellekten veri okuma ve yazma işlemlerini gerçekleştirirken verimlilik açısından bir darboğaz oluşturabilir. Çünkü işlemci, belleğe erişmek için belirli bir bant genişliğine sahiptir ve bu bant genişliği paylaşılan diğer sistem bileşenleriyle (örneğin, diğer işlemciler, giriş/çıkış aygıtları) de paylaşılabilir.

Soru

- Von Neumann mimarisindeki saklanmış program (stored program) kavramının önemi nedir?

- A) Donanım bileşenleriyle doğrudan etkileşim sağlar.
- B) Komutların veri olarak saklanmasına ve işlenmesine izin verir.
- C) Giriş/çıkış cihazlarına ihtiyacı ortadan kaldırır.
- D) CPU çalışmasını hızlandırır.

Cevap

- Cevap: B

- Saklanmış program kavramı, Von Neumann mimarisindeki önemli bir kavramdır çünkü bu mimaride talimatlar ve veriler aynı bellek biriminde saklanır. Bu, talimatların bellekte veri olarak saklanmasına ve işlenmesine izin verir. Yani, programın kendisi de bellekte veri olarak saklanabilir ve işlenebilir. Bu, programın çalışma zamanında değiştirilebileceği ve programların diğer programlar tarafından manipüle edilebileceği anlamına gelir. Bu da esneklik ve programların dinamik olarak oluşturulması ve değiştirilmesi açısından büyük bir önem taşır.

Soru

- İşletim sistemlerinde bir kesmenin (interrupt) temel amacı nedir?

- a) Mevcut sürecin yürütmesini durdurmak
- b) Bir sürecin CPU kontrolünü isteğe bağlı olarak bırakmasına izin vermek
- c) Asenkron olayları işlemek ve önceliklendirmek
- d) Süreçler arasında iletişimi kolaylaştırmak

Cevap

- Cevap: c)

- Bir işletim sistemi, birçok görevi aynı anda yönetmek zorundadır. Bu görevler arasında kullanıcının klavyeden bir tuşa basması, ağa veri alıp gönderme, diskten veri okuma/yazma gibi işlemler bulunur. Bu tür olaylar "asenkron" olarak adlandırılır, çünkü bunlar işlemciden bağımsız olarak ve önceden belirlenemeyen zamanlarda gerçekleşir. Bir kesme, işletim sisteminin normal akışını keserek bu asenkron olaylara müdahale etmesini ve önceliklerini belirlemesini sağlar. Örneğin, bir klavyeden bir tuşa basıldığında, klavye kesmesi (keyboard interrupt) oluşur ve işletim sistemi bu kesmeyi işler, klavyeden gelen veriyi okur ve uygun işlemi gerçekleştirir. Bu da sistemin daha verimli ve güvenilir çalışmasına olanak tanır.

Soru

- Öncelikli kesmelerin (interrupts) bulunduğu bir önleyici çoklu görevli sistemde işletim sistemi, daha yüksek öncelikli bir kesme aldığında hangi eylemi gerçekleştirir?

- a) Mevcut süreci tamamlar ve sonra kesmeyi işler
- b) Hemen kesme işleyicisinin yürütülmesine geçer
- c) Kesmeyi bir kuyruğa yerleştirir ve mevcut süreçten sonra işler
- d) Mevcut süreç tamamlanana kadar kesmeyi görmezden gelir

Cevap

- Cevap: b)

- Öncelikli bir kesme geldiğinde, işletim sistemi hemen mevcut süreci duraklatır ve kesme işleyicisinin yürütülmesine geçer. Bu, yüksek öncelikli bir olayın hızlı bir şekilde işlenmesini sağlar ve sistemin daha duyarlı olmasını sağlar. Örneğin, bir acil durum alarmı gibi kritik bir olay gerçekleştiğinde, işletim sistemi diğer işlemleri askıya alarak bu olayı derhal işler. Bu, sistem performansını artırır ve kritik işlevlerin zamanında gerçekleştirilmesini sağlar.

Soru

- Aşağıdaki senaryolardan hangisi maskelenemez bir kesme (non-maskable interrupt - NMI) örneğidir?

- a) Bir tuşa basılarak oluşturulan bir klavye kesmesi
- b) Zaman diliminin sona ermesiyle tetiklenen bir zamanlayıcı kesmesi
- c) Sistem tarafından algılanan kritik bir donanım hatası
- d) Bir sistem çağrısı ile çağrılan bir yazılım kesmesi

Cevap

- Cevap: c)

- Non-maskable interrupt (NMI - maskelenemez kesme), sistemdeki kritik durumları işlemek için kullanılır ve genellikle donanım hatası gibi ciddi durumlar için ayrılmıştır. Bu tür bir kesme, diğer normal kesmelerden farklı olarak, işletim sistemi veya kullanıcı tarafından engellenemez veya durdurulamaz. Bu tür hatalar genellikle sistem bütünlüğünü tehlikeye atan veya ciddi hizmet kesintilerine neden olabilecek cihaz veya bileşen arızalarını ifade eder.

Soru

- İşletim sistemleri genellikle kesme işlemlerini yönetmek için hangi mekanizmayı kullanır?

- a) Sorgulama
- b) Vektörlendirme
- c) Round-robin çizelgeleme
- d) Öncelik ters çevirme

Cevap

- Cevap: b)

- İşletim sistemleri, bir kesme oluştuğunda hangi işlem kodunun çalıştırılacağını belirlemek için vektörlendirme mekanizmasını kullanır. Bu, kesmelerin işlenmesinde verimlilik ve doğruluk sağlar. Vektör tablosu genellikle kesmelerin tanımlandığı ve ilgili işlem kodlarının bulunduğu bir tablodur. Bu sayede, bir kesme oluştuğunda işletim sistemi, vektör tablosundaki ilgili girdiyi kullanarak doğrudan ilgili işlem kodunu bulabilir ve çalıştırabilir.

Soru

- Vektörlü kesme işleme şemasında, kesme işleyicisi nasıl belirlenir?

- a) Her bir kesmeye atanan sabit bir öncelikle
- b) Kesme numarası tarafından dizinlenen bir işaretçi tablosu ile
- c) Kesmelerin alındığı sıraya göre
- d) Kesme denetleyici donanımı tarafından

Cevap

- Cevap: b)

- Vektörlü kesme işleme şemasında, bir kesme işleyicisinin belirlenmesi için, kesme numarasına göre dizinlenen bir işaretçi tablosuna başvurulur. Bu tablo, her kesme için bir işaretçiye sahip olup, her bir işaretçi kesmenin işleyicisinin konumunu gösterir. Bir kesme oluştuğunda, işletim sistemi bu tabloya bakarak ilgili işaretçiyi bulur ve bu işaretçinin gösterdiği adrese giderek ilgili kesme işleyicisini başlatır. Bu sayede, her bir kesme için farklı işleyiciler belirlenebilir ve etkin bir şekilde yönetilebilir.

Soru

- Bir kesme (interrupt) ile bir tuzak (trap) arasındaki fark nedir?

- a) Tuzaklar asenkron olaylardır, kesmeler ise senkrondur.
- b) Tuzaklar işletim sistemi, kesmeler aygıt sürücüleri tarafından işlenir.
- c) Kesmeler kontrolü harici bir cihaza, tuzaklar işletim sistemine aktarır.
- d) Kesmeler donanım tarafından, tuzaklar ise yazılım tarafından başlatılır.

Cevap

- Cevap: d)

- Kesmeler, harici bir donanımın işlemciye dikkatini çektiği ve işlemcinin normal akışını geçici olarak keserek kesmeyi işlemesi gerektiği durumlarda donanım tarafından başlatılır. Öte yandan, tuzaklar yazılım tarafından bilerek ve isteyerek oluşturulan özel işaretlerdir. Bir programın çalışması sırasında bir hata durumunda veya belirli bir koşul gerçekleştiğinde, yazılım tuzakları kullanarak işletim sistemine veya belirli bir işlemci işlemine kontrolü aktarabilir. Bu nedenle, kesmeler genellikle donanımın etkileşimiyle ilgili iken, tuzaklar yazılım tarafından başlatılır ve işletim sistemi veya program tarafından kullanılır.

Soru

- Kesme Servis Rutini'nin (ISR) amacı nedir?

- a) Eş zamanlı olarak birden fazla sürecin yürütülmesini yönetmek
- b) Kesmelerin tespit ve onaylanmasını işlemek
- c) Sanal bellek adreslerini fiziksel bellek adreslerine çevirmek
- d) Donanım cihazları arasında iletişim için bir arayüz sağlamak

Cevap

- Cevap: b)

- Kesme Servis Rutini (ISR), bir kesme gerçekleştiğinde işletim sistemi tarafından çağrılan özel bir kod parçasıdır. Bu rutin, kesmenin algılanmasını ve onaylanmasını yönetir. Kesme algılandığında, işletim sistemi, ilgili ISR'yi çağırarak kesmeyi işler ve gereken aksiyonu alır. Örneğin, bir kesme, bir cihazın hazır olduğunu veya bir hata durumunun meydana geldiğini bildirebilir ve ISR, bu durumu ele almak için gereken işlemleri gerçekleştirebilir.

Soru

- Doğrudan Bellek Erişimini (DMA) en iyi tanımlayan ifade hangisidir?

- a) DMA, işletim sistemini dahil etmeden CPU'nun doğrudan belleğe erişmesine izin verir.
- b) DMA, çevre birimlerinin CPU müdahalesi olmadan bellek arasında veri aktarımına izin verir.
- c) DMA, modern işletim sistemlerinde sanal bellek yönetimi için bir mekanizmadır.
- d) DMA, yalnızca işletim sistemi içinde süreçler arası iletişim için kullanılır.

Cevap

- Cevap: b)

- Doğrudan Bellek Erişimi (DMA), çevre birimlerinin (örneğin, disk sürücüsü, ağ kartı) bellek iletişimini yönetmek için kullanılan bir mekanizmadır. DMA, CPU'nun iş yükünü azaltarak, çevre birimlerinin bellekten veri okuması veya belleğe veri yazması gibi işlemleri doğrudan gerçekleştirmesine izin verir. Bu sayede, CPU'nun bu tür transferleri denetlemesi gerekmez, böylece CPU işleme gücünü gereken diğer görevlere odaklayabilir. Bu nedenle, DMA, çevre birimleri ile bellek arasındaki veri transferini optimize etmek için kullanılan önemli bir mekanizmadır.

Soru

- Aşağıdakilerden hangisi DMA kullanmanın faydalarından DEĞİLDİR?

- a) Azaltılmış CPU iş yükü
- b) Artırılmış veri transfer hızı
- c) Bellek erişimi için artırılmış güvenlik
- d) İyileştirilmiş sistem tepkiselliği

Cevap

- Cevap: c)

- DMA kullanmanın avantajları arasında, azaltılmış CPU iş yükü, artırılmış veri transfer hızı ve iyileştirilmiş sistem tepkiselliği bulunmaktadır. Ancak, DMA'nın bellek erişimi için artırılmış güvenlik sağladığı söylenemez. DMA, doğrudan çevre birimlerinin belleğe erişmesine izin verirken, bu durum bazı güvenlik risklerine neden olabilir. Özellikle kötü amaçlı yazılımların, DMA aracılığıyla belleğe doğrudan erişim sağlaması ve hassas verilere zarar vermesi gibi durumlar söz konusu olabilir.

Soru

- DMA yetenekli bir sistemde DMA transferlerini başlatan bileşen hangisidir?

- a) Merkezi İşlem Birimi (CPU)
- b) Çevre birimleri
- c) DMA denetleyici
- d) Bellek Yönetim Birimi (MMU)

Cevap

- Cevap: b)

- DMA transferlerini başlatan bileşen çevre birimleridir. DMA, çevre birimlerinin doğrudan belleğe erişmesine olanak tanır, bu nedenle çevre birimleri DMA transferlerini başlatır ve kontrol eder. DMA denetleyicisi, bu transferleri yönetir ve koordine ederken, çevre birimleri bu transferlerin başlatılmasından sorumludur.

Soru

- Ortalama dönüş süresini minimize etmek için tasarlanmış olan CPU çizelgeleme algoritması hangisidir?

- a) Round Robin
- b) En Kısa İşlem Önce (SJN)
- c) İlk Gelen İlk Hizmet (FCFS)
- d) En Kısa Kalan Süre İlk (SRTF)

Cevap

- Cevap: d)

- En Kısa Kalan Süre İlk (SRTF) CPU çizelgeleme algoritması, mevcut sıradaki süreçler arasından en kısa süreli süreci seçerek işlem sırasını belirler. Bu şekilde, ortalama dönüş süresi minimize edilir, çünkü her zaman en kısa süreli süreç öncelikli olarak işlenir ve daha kısa sürede tamamlanır. Bu, işlemciyi daha verimli kullanmayı ve süreçlerin daha hızlı tamamlanmasını sağlar.

Soru

- Önleyici (preemptive) CPU çizelgelemenin temel amacı nedir?

- a) CPU kullanımını maksimize etmek
- b) İnteraktif görevler için yanıt süresini minimize etmek
- c) İşlemler arasında CPU tahsisinde adil olmayı sağlamak
- d) Bağlam anahtarlama işleminden kaynaklanan iş yükünü minimize etmek

Cevap

- Cevap: b)

- Önleyici CPU çizelgeleme, öncelikle kullanıcı etkileşimli görevler için yanıt süresini minimize etmeyi amaçlar. Bu, kullanıcıların işlemlere hızlı bir şekilde yanıt almasını sağlar ve kullanıcı deneyimini iyileştirir. Önleyici zamanlama, süreçlerin önceliklerine göre kesmelerle (interrupts) kontrol edilir ve işlemcideki süreç sırası sık sık değişebilir.

Soru

- CPU çizelgeleme algoritmalarının performansını ETKİLEMEYEN faktör hangisidir?

- a) Sistemdeki CPU sayısı
- b) CPU-yoğun ve G/Ç-yoğun işlemlerin karışımı
- c) Sistem belleğinin boyutu
- d) Bağlam anahtarlama işleminden kaynaklanan iş yükü

Cevap

- Cevap: c)

- CPU çizelgeleme algoritmalarının performansını etkileyen faktörler arasında, sistemdeki CPU sayısı, CPU-yoğun ve G/Ç-yoğun işlemlerin karışımı ve bağlam anahtarlama işleminden kaynaklanan iş yükü gibi faktörler bulunur. Ancak, sistem belleğinin boyutu, çizelgeleme algoritmalarının performansını doğrudan etkilemez.

Soru

- İlk Gelen İlk Hizmet (FCFS) algoritmasının başlıca dezavantajı nedir?

- a) Yüksek ortalama bekleme süresi
- b) Zayıf CPU kullanımı
- c) Düşük öncelikli süreçlerin belirsiz bir şekilde bloke olması
- d) Önemli görevlere öncelik verememe

Cevap

- Cevap: a)

- İlk Gelen İlk Hizmet (FCFS), gelen süreçleri sırayla işlemek için kullanılan basit bir çizelgeleme algoritmasıdır. Ancak, bu yaklaşımın başlıca dezavantajı, süreçlerin işlemciye geliş sırasına göre işlenmesi nedeniyle yüksek ortalama bekleme süresidir. Öncelikli olmayan işlemler, öncelikli işlemlerin tamamlanmasını beklemek zorunda kalır, bu da ortalama bekleme süresini artırır. Bu durum, sistemdeki genel performansı olumsuz etkileyebilir, özellikle de sistemde birçok süreç olduğunda.

Soru

- CPU görev dağıtıcısının temel amacı nedir?

- a) CPU kaynaklarını süreçlere tahsis etmek
- b) Çevre birimleri tarafından oluşturulan kesmeleri işlemek
- c) Sistem çağrılarının yürütülmesini yönetmek
- d) Süreçler arasında paylaşılan kaynaklara erişimi senkronize etmek

Cevap

- Cevap: a)

- CPU görev dağıtıcısı, işletim sistemi içindeki bir bileşendir ve temel amacı CPU kaynaklarını süreçlere tahsis etmektir. Bir süreç hazır duruma geldiğinde veya bir süreç tamamlandığında, CPU görev dağıtıcısı bu süreçlere CPU kaynaklarını tahsis eder. Bu, işletim sisteminin çoklu işlem yeteneklerini yönetmesine ve işlemleri adil bir şekilde CPU üzerinde çalıştırmasına olanak tanır.

Soru

- İşletim sistemlerinde önbelleğin temel amacı nedir?

- a) Daha hızlı erişim için sık erişilen veri ve komutları saklamak
- b) Bellek yoğun uygulamalar için ek bellek sağlamak
- c) Disk G/Ç işlemleri için bir arabellek görevi görmek
- d) Süreçler arası iletişimi kolaylaştırmak

Cevap

- Cevap: a)

- Önbellek, işletim sistemlerinde sık erişilen veri ve komutları saklar. Önbellek, RAM'in daha hızlı erişilebilir olduğu ve işlemcinin çalışma hızına daha yakın olduğu bir yerdedir. Bu nedenle, sık sık kullanılan veri ve komutlar önbelleğe kopyalanır, böylece işlemci bu verilere daha hızlı erişebilir. Bu, genel sistem performansını artırır ve işlemcideki bekleme sürelerini azaltır.

Soru

- Tipik olarak en küçük kapasiteye ve en hızlı erişim süresine sahip olan önbellek hiyerarşisi seviyesi hangisidir?

- a) L1 önbelleği
- b) L2 önbelleği
- c) L3 önbelleği
- d) Ana bellek

Cevap

- Cevap: a)

- Önbellek hiyerarşisinde, L1 önbelleği en küçük kapasiteye ve en hızlı erişim süresine sahip olan seviyedir. L1 önbelleği, işlemcinin hemen yanında yer alır ve işlemciye çok hızlı bir şekilde erişilir. Ancak, bu önbellek genellikle küçük bir kapasiteye sahiptir, çünkü bu önbellek içine sığdırılabilecek veri miktarı sınırlıdır. Diğer önbellek seviyeleri (L2 ve L3) daha büyük kapasiteye ve daha yüksek erişim sürelerine sahiptir.

Soru

- Çok işlemcili sistemlerde önbellek tutarlılığı ile ilişkili temel zorluk nedir?

- a) Önbellek seviyeleri arasındaki veri tutarlılığını sağlamak
- b) Önbellek erişim gecikmesini en aza indirmek
- c) Önbellek yerine koyma politikalarını optimize etmek
- d) Önbellek boyutu ile hız arasındaki dengeyi sağlamak

Cevap

- Cevap: a)

- Çok işlemcili sistemlerde, her işlemci kendi önbelleğine sahiptir ve bu önbellekler arasındaki veri tutarlılığını sağlamak önemli bir zorluktur. Birden fazla işlemci aynı bellek hücresine erişirse ve bu hücredeki veriyi değiştirirse, diğer işlemcilerin kendi önbelleklerinde bu değişikliği fark etmeleri ve güncellemeleri gerekmektedir. Bu, önbellek tutarlılığı olarak bilinir.

Soru

- İşletim sistemi önyükleme sürecinde önyükleyicinin rolü nedir?

- a) Sistem donanım bileşenlerini başlatmak
- b) İşletim sistemi çekirdeğini belleğe yüklemek
- c) Kullanıcıya özgü ayarları yapılandırmak
- d) Disk bölümlerini yönetmek

Cevap

- Cevap: b)

- Önyükleyicinin temel görevi, işletim sistemi çekirdeğini diskten belleğe yüklemektir. Bilgisayar başlatıldığında, önyükleyici (boot loader), sabit diskten işletim sistemi çekirdeğini okur ve belleğe yükler. Önyükleme işlemi, bilgisayarın donanımını başlatmak, önyükleme aygıtlarını tanımlamak ve işletim sistemi çekirdeğini yüklemek gibi adımları içerir. Ancak, önyükleyici genellikle işletim sistemi çekirdeğini belleğe yüklemekle sınırlıdır ve diğer sistem yapılandırma veya kullanıcı ayarları işlemleri için kullanılmaz. Bu görevler, işletim sistemi tarafından veya başlatma sonrası betikler tarafından gerçekleştirilir.

Soru

- Önyükleme sürecinde Master Boot Record (MBR)'in amacı nedir?

- a) İşletim sistemi çekirdeğini depolamak
- b) Önyükleme yükleyici programını yüklemek
- c) Diski mantıksal birimlere bölmek
- d) Sistem donanım kesmelerini yönetmek

Cevap

- Cevap: b)

- Master Boot Record (MBR), bilgisayarın önyükleme sürecinde, önyükleme yükleyici programını yüklemek için kullanılan bir alandır. MBR, bir sabit diskin ilk sektöründe bulunur ve disk üzerindeki bölümleri ve bunların dosya sistemlerini tanımlar. MBR, bilgisayarın önyükleme işlemi sırasında ilk olarak yüklenen kod parçasıdır. MBR, önyükleme yükleyici programını yükler ve bu program, işletim sistemi çekirdeğini diskten belleğe yükler. Dolayısıyla, MBR'nin temel amacı, önyükleme sürecini başlatmak ve önyükleme yükleyici programını diskten yüklemektir.

Soru

- Önyükleme sürecinde BIOS/UEFI yazılımını başlatmaktan hangi bileşen sorumludur?

- a) Master Boot Record (MBR)
- b) Önyükleme yükleyici
- c) Çekirdek Yükleyici
- d) Temel Giriş/Çıkış Sistemi (BIOS) / Birleşik Genişletilebilir Yazılım Arabirimi (UEFI)

Cevap

- Cevap: d)

- Önyükleme sürecinde, BIOS veya UEFI yazılımını başlatmaktan sorumlu olan bileşen, işletim sistemi çekirdeğini yükleyecek olan bileşenlerden biri olan temel giriş/çıkış sistemi (BIOS) veya birleşik genişletilebilir yazılım arabirimi (UEFI)'dir. Bu yazılım, bilgisayarın donanımını başlatır ve işletim sistemi yükleyici programını çalıştırır. BIOS veya UEFI, bilgisayarın başlangıç noktasıdır ve önyükleme sürecinin ilk adımını oluşturur.

Soru

- Önyükleme sürecinin POST (Power-On Self-Test) aşamasında ne olur?

- a) İşletim sistemi çekirdeği belleğe yüklenir
- b) Sistem donanım bileşenleri başlatılır ve test edilir
- c) Kullanıcıya özgü yapılandırma ayarları uygulanır
- d) Disk bölüm bilgileri Master Boot Record (MBR)'den okunur

Cevap

- Cevap: b)

- Önyükleme sürecinin POST (Power-On Self-Test) aşaması, bilgisayarın donanım bileşenlerinin başlatılması ve test edilmesi için gereklidir. Bu aşamada, sistem belleği, işlemci, sabit disk, grafik kartı ve diğer donanım bileşenleri gibi temel sistem bileşenleri kontrol edilir ve test edilir. POST, bilgisayarın başlangıç noktasıdır ve donanımın düzgün çalışıp çalışmadığını belirlemek için önemlidir.

Soru

- Linux tabanlı işletim sistemlerinin önyükleme sürecinde GRUB (Grand Unified Bootloader)'un rolü nedir?

- a) Sistem donanım bileşenlerini başlatır
- b) İşletim sistemi çekirdeğini belleğe yükler
- c) Disk bölümlendirme ve biçimlendirme işlemlerini yönetir
- d) Farklı işletim sistemi yapılandırmalarını seçmek için bir menü sağlar

Cevap

- Cevap: b)

- GRUB (Grand Unified Bootloader), Linux tabanlı işletim sistemlerinde önyükleme sürecinde işletim sistemi çekirdeğini belleğe yüklemekten sorumlu olan bir önyükleme yükleyici programıdır. GRUB, bilgisayar başlatıldığında önce yüklenir ve ardından kullanıcıya bir menü veya seçenekler sunarak hangi işletim sisteminin yükleneceğini seçmesine izin verir. Seçilen işletim sistemi çekirdeği daha sonra belleğe yüklenir ve işletim sistemi başlatılır.

Soru

- Önyükleme sürecinde çekirdek yükleyicisinin amacı nedir?

- a) Sistem donanım bileşenlerini başlatmak
- b) Kullanıcıya özgü ayarları yapılandırmak
- c) Aygıt sürücülerini ve sistem modüllerini belleğe yüklemek
- d) Disk bölümlendirme ve biçimlendirme işlemlerini yönetmek

Cevap

- Cevap: c)

- Çekirdek yükleyicisi, önyükleme sürecinde, işletim sistemi çekirdeği tarafından kullanılan aygıt sürücülerini ve sistem modüllerini belleğe yüklemekten sorumludur. Bu, işletim sisteminin gerekli donanım sürücülerini yüklemesini sağlar ve işletim sisteminin kullanıma hazır hale gelmesini sağlar. Çekirdek yükleyicisi, işletim sistemi çekirdeğini yüklemeden sürücüleri ve gerekli sistem modüllerini belleğe yükler.

Soru

- İşletim sistemlerinde monolitik çekirdek yapısının bir özelliği nedir?

- a) Çekirdek bileşenlerini tekil kullanıcı alanı süreçlerine ayırır.
- b) Bileşenler arasında iyi tanımlanmış arayüzler sağlayan modüler bir mimari sunar.
- c) Tüm işletim sistemi işlevselliğini tek, büyük bir ikili dosyada birleştirir.
- d) Süreçler arası iletişim için mikroçekirdek prensiplerine dayanır.

Cevap

- Cevap: c)

- Monolitik çekirdek yapısı, işletim sistemi işlevselliğini tek bir büyük ikili dosyada birleştirir. Bu, çekirdeğin temel işlevlerinin, sürücülerin, sistem çağrılarının ve diğer işletim sistemi bileşenlerinin tümünün aynı ikili dosyada bulunduğu anlamına gelir. Bu yaklaşım, çekirdeğin işlevselliğini konsolide eder ve bu nedenle çekirdeğin boyutunu artırır. Monolitik bir çekirdek, modüler bir yapıya sahip olmasa da, birçok işlevi doğrudan çekirdeğin içine entegre eder.

Soru

- Monolitik çekirdek mimarisinde, aygıt sürücüleri genellikle nasıl uygulanır?

- a) Ayrı kullanıcı alanı süreçleri olarak
- b) Dinamik olarak yüklenebilir çekirdek modülleri olarak
- c) Bağımsız uygulamalar olarak
- d) Çekirdek ikili dosyanın bir parçası olarak

Cevap

- Cevap: d)

- Monolitik çekirdek mimarisinde, aygıt sürücüleri genellikle çekirdek ikili dosyanın bir parçası olarak uygulanır. Bu, aygıt sürücülerinin doğrudan çekirdek koduna dahil edildiği ve çekirdeğin bir parçası olduğu anlamına gelir. Aygıt sürücüleri, işletim sisteminin diğer temel bileşenleriyle birlikte tek bir büyük ikili dosyada bulunur ve bu dosya çalıştırıldığında, tüm çekirdek işlevselliği bir arada yürütülür. Bu yaklaşım, aygıt sürücülerinin çekirdek içinde doğrudan erişilebilir ve hızlı bir şekilde çalışmasını sağlar. Ancak, bu aynı zamanda çekirdek boyutunu artırır ve çekirdeğin değiştirilmesini veya güncellenmesini zorlaştırabilir.

Soru

- Monolitik bir çekirdek, kullanıcı alanı uygulamalarından sistem çağrılarını nasıl işler?

- a) Sistem çağrı işleme görevini ayrı kullanıcı alanı süreçlerine devrederek
- b) Çekirdek adres alanı içinde sistem çağrısı işleyicilerini çağırarak
- c) Çekirdek modülleri arasında mesaj iletişimi mekanizması kullanarak
- d) Sistem çağrılarını ayrı bir mikroçekirdek bileşenine yönlendirerek

Cevap

- Cevap: b)

- Monolitik bir çekirdek, kullanıcı alanı uygulamalarının yaptığı sistem çağrılarını doğrudan çekirdek adres alanı içindeki sistem çağrısı işleyicilerini çağırarak işler. Çekirdek adres alanı içindeki sistem çağrısı işleyicileri, kullanıcı uygulamalarının taleplerini alır, gerektiğinde uygun çekirdek işlevlerini çağırır ve sonuçları uygun şekilde işler. Bu yaklaşım, monolitik çekirdeğin hızlı ve verimli çalışmasını sağlar, çünkü sistem çağrıları doğrudan çekirdek içinde gerçekleştirilir ve gereksiz ara işlemlere gerek kalmaz.

Soru

- İşletim sistemlerinde mikroçekirdek yapısının belirleyici bir özelliği nedir?

- a) Tüm işletim sistemi işlevselliğini tek, büyük bir ikili dosyada birleştirir.
- b) Aygıt sürücülerinin çekirdek alanı içinde çekirdek işletim sistemi bileşenleriyle birlikte çalışmasına izin verir.
- c) Çekirdek boyutunu en aza indirerek yalnızca temel işlevleri çekirdek alanında uygulamasıdır.
- d) Verimli süreçler arası iletişim için monolitik bir mimariye dayanır.

Cevap

- Cevap: c)

- Mikroçekirdek yapısının belirleyici bir özelliği, çekirdek boyutunu en aza indirerek yalnızca temel işlevleri çekirdek alanında uygulamasıdır. Mikroçekirdek, işletim sistemi işlevselliğini mümkün olduğunca küçük bir çekirdek içine yerleştirir ve daha fazla işlevselliği kullanıcı alanına veya çekirdek dışı modüllere taşır. Bu, çekirdeğin boyutunu azaltır ve işletim sistemi daha modüler hale gelir, böylece bileşenler daha bağımsız olarak geliştirilebilir ve değiştirilebilir hale gelir. Mikroçekirdek mimarisi, genellikle esneklik, güvenilirlik ve bakım kolaylığı sağlamak için tercih edilir.

Soru

- Mikroçekirdek mimarisinde, aygıt sürücüleri tipik olarak nerede bulunur?

- a) Çekirdek alanında
- b) Kullanıcı alanında ayrı süreçler olarak
- c) Ayrılmış donanım katmanında
- d) Yazılım katmanında

Cevap

- Cevap: b)

- Mikroçekirdek mimarisinde, aygıt sürücüleri genellikle kullanıcı alanında ayrı süreçler olarak bulunur. Bu, aygıt sürücülerinin çekirdek dışında, kullanıcı alanında çalıştığı anlamına gelir. Mikroçekirdek yapısı, çekirdeğin boyutunu en aza indirmeyi ve çekirdek içinde sadece temel işlevleri tutmayı amaçlar. Bu nedenle, aygıt sürücüleri gibi özelleştirilmiş işlevler, kullanıcı alanında ayrı süreçler olarak yürütülür ve çekirdek dışında çalışır.

Soru

- Mikroçekirdek mimarisi, monolitik çekirdeklerle karşılaştırıldığında performans açısından hangi zorlukla karşılaşır?

- a) Artan bağlam anahtarlama maliyetleri
- b) Güvenlik açıklarına karşı hassasiyet
- c) Özelleştirme için azalan esneklik
- d) Çekirdek bileşenleri arasında daha yüksek iletişim gecikmesi

Cevap

- Cevap: a)

- Mikroçekirdek mimarisi, monolitik çekirdeklerle karşılaştırıldığında artan bağlam anahtarlama maliyetleri ile karşılaşır. Mikroçekirdek mimarisinde, işlevler çekirdek dışında, ayrı süreçlerde yürütülür ve bu nedenle süreçler arası iletişim ve geçişler daha sık gerçekleşir. Bu durum, mikroçekirdek mimarisinin performansını etkileyebilir, çünkü bağlam anahtarlama ek işlemci zamanı ve bellek kullanımı gerektirir. Monolitik çekirdekler genellikle işlevlerin doğrudan çekirdek içinde çalıştığı için, bu tür maliyetler genellikle daha düşüktür.

Soru

- Mikroçekirdekte Süreçler Arası İletişim (IPC) mekanizması hangi role sahiptir?

- a) Çekirdek alanı ve kullanıcı alanı süreçleri arasındaki iletişimi yönetir.
- b) Farklı çekirdek modülleri arasındaki iletişimi kolaylaştırır.
- c) Çekirdek ve kullanıcı uygulamaları arasındaki bellek yönetimini koordine eder.
- d) Dosya ve aygıtlar gibi sistem kaynaklarına erişimi kontrol eder.

Cevap

- Cevap: b)

- Mikroçekirdekte Süreçler Arası İletişim (IPC) mekanizması, farklı çekirdek modülleri arasındaki iletişimi kolaylaştırır. Mikroçekirdek mimarisinde, işlevler genellikle ayrı çekirdek modülleri olarak uygulanır ve bu modüller arasındaki iletişim IPC mekanizması aracılığıyla gerçekleşir. Örneğin, bir aygıt sürücüsü çekirdek içinde bir modül olabilir ve bir sistem çağrısı ile iletişim kurarak veya mesaj geçirerek diğer çekirdek modülleriyle etkileşimde bulunabilir. IPC mekanizması, çekirdek modüllerinin işbirliğini sağlar ve farklı işlevler arasında veri ve kontrol akışını yönetir.

Soru

- Mikroçekirdek tabanlı bir işletim sisteminin işlevselliğini genişletmek için genellikle hangi yaklaşım kullanılır?

- a) Çekirdek modüllerinin dinamik olarak bağlanması
- b) Aygıt sürücülerinin çekirdek alanına entegrasyonu
- c) Önyükleme sırasında ek çekirdek bileşenlerinin yüklenmesi
- d) Kritik hizmetlerin çekirdek ikili dosyasının bir parçası olarak çalıştırılması

Cevap

- Cevap: a)

- Mikroçekirdek tabanlı bir işletim sisteminin işlevselliğini genişletmek için genellikle çekirdek modüllerinin dinamik olarak bağlanması kullanılır. Bu yaklaşım, işletim sisteminin çekirdek alanında yer alan temel işlevlerini genişletmek veya özelleştirmek için harici modüllerin dinamik olarak yüklenmesini sağlar. Çekirdek modülleri, çalışma zamanında yüklenebilir ve çekirdek dışındaki bir alanda saklanabilir. Bu, işletim sistemi işlevselliğinin daha esnek bir şekilde genişletilmesini sağlar ve çekirdek boyutunu azaltır.

Soru

- Sistem performansı açısından mikroçekirdek mimarisinin potansiyel bir dezavantajı nedir?

- a) Sistem yönetiminin artan karmaşıklığı
- b) Sistem çağrıları ve IPC için yüksek ek maliyet
- c) Üçüncü taraf aygıt sürücülerini entegre etmede zorluk
- d) Büyük ölçekli sistemler için sınırlı ölçeklenebilirlik

Cevap

- Cevap: b)

- Mikroçekirdek mimarisinin sistem performansı açısından potansiyel bir dezavantajı, sistem çağrıları ve işlem arası iletişim (IPC) için daha yüksek ek maliyettir. Mikroçekirdek mimarisinde, işlevler ayrı çekirdek modülleri olarak uygulanır ve bu modüller arasındaki iletişim genellikle sistem çağrıları veya IPC mekanizması aracılığıyla gerçekleşir. Her sistem çağrısı veya IPC işlemi, çekirdek ve kullanıcı alanı arasında veri kopyalamayı gerektirebilir ve bu, işlemci zamanı ve bellek kullanımı açısından maliyetlidir.

Soru

- Katmanlı işletim sistemi yapısında, donanım soyutlama katmanının (HAL) asıl amacı nedir?

- a) Kullanıcı uygulamalarının işletim sistemiyle etkileşim kurması için bir arayüz sağlamak.
- b) Sistem çağrılarının ve çekirdek düzeyindeki görevlerin yürütülmesini yönetmek.
- c) İşletim sisteminin farklı katmanları arasında iletişimi kolaylaştırmak.
- d) Donanıma özgü ayrıntıları soyutlamak ve üst katmanlara birleşik bir arayüz sağlamak.

Cevap

- Cevap: d)

- Donanım Soyutlama Katmanı (HAL), donanıma özgü ayrıntıları soyutlar ve üst katmanlara (genellikle çekirdek ve kullanıcı arayüzü gibi) birleşik bir arayüz sağlar. Bu katman, işletim sistemi yazılımının donanımdan bağımsız olmasını sağlar. Böylece, işletim sistemi farklı donanım yapılandırmalarıyla çalışabilir ve üst katmanlar, donanımın nasıl çalıştığına dair ayrıntılar hakkında endişelenmeden işletim sistemine erişebilir. HAL, donanım değişikliklerini gizler ve üst katmanlara istikrarlı bir arabirim sunar, bu da işletim sisteminin genişletilmesini ve bakımını kolaylaştırır.

Soru

- İşletim sistemi hiyerarşisinde hangi katman, kullanıcı uygulamaları ile çekirdek arasındaki etkileşimi yönetmekten sorumludur?

- a) Çekirdek katmanı
- b) Donanım soyutlama katmanı
- c) Sistem çağrısı katmanı
- d) Kabuk katmanı

Cevap

- Cevap: c)

- İşletim sistemi hiyerarşisinde, kullanıcı uygulamaları ile çekirdek arasındaki etkileşimi yönetmekten sorumlu olan katman, sistem çağrısı katmanıdır. Bu katman, kullanıcı uygulamalarının çekirdeğe erişimini sağlar ve sistem çağrıları aracılığıyla çeşitli işletim sistemi hizmetlerine erişim sağlar. Kullanıcı uygulamaları, sistem çağrıları yaparak çekirdekten hizmet talep ederler ve çekirdek, bu çağrıları işleyerek gerekli işlemleri gerçekleştirir.

Soru

- Katmanlı işletim sistemi yapısının temel avantajı nedir?

- a) Sistem işlevselliğini özelleştirmek ve genişletmek için artan esneklik
- b) Sistem kaynaklarının ve süreçlerin kolay yönetimi
- c) Katmanlar arasındaki iletişimde azalan ek maliyet nedeniyle artan performans
- d) Sistem bileşenlerinin izolasyonu ve kapsülleme yoluyla artan güvenlik

Cevap

- Cevap: d)

- Katmanlı işletim sistemi yapısının temel avantajı, sistem bileşenlerinin izolasyonu ve kapsülleme yoluyla artan güvenliktir. Katmanlı bir yapı, her katmanın belirli bir işlevsellik seviyesine sahip olduğu ve alt katmanların üst katmanlardan gizlendiği bir yapı sağlar. Bu, her katmanın sadece altındaki katmanlarla iletişim kurmasını ve dışarıdaki katmanlardan izole olmasını sağlar. Böylece, bir katman, diğer katmanlardan etkilenmez ve sistem bütünlüğü korunur. Bu, güvenlik açısından önemlidir çünkü saldırganların sistem içinde hareket etmelerini ve zarar vermelerini önler. Katmanlı yapı ayrıca sistem bileşenlerini daha iyi kapsüller ve karmaşıklığı azaltır, bu da sistem yönetimini ve bakımını kolaylaştırır.

Soru

- Sanallaştırmada, bir hipervizörün temel amacı nedir?

- a) Kullanıcı uygulamalarının sanal makinelerle etkileşim kurması için bir arayüz sağlamak.
- b) Sanal makineler için CPU, bellek ve depolama gibi sistem kaynaklarını yönetmek.
- c) Sanal makinelerin işletim sistemlerini çalıştırması için donanım bileşenlerini taklit etmek.
- d) Tek bir fiziksel ana bilgisayarda birden fazla sanal makine oluşturmak ve yönetmek.

Cevap

- Cevap: d)

- Bir hipervizörün temel amacı, tek bir fiziksel ana bilgisayarda birden fazla sanal makine oluşturmak ve yönetmektir. Hipervizör, donanım kaynaklarını sanal makineler arasında paylaştırır ve her bir sanal makineyi izole bir ortamda çalıştırır. Bu, bir fiziksel sunucu üzerinde birden fazla işletim sistemi çalıştırmak için ideal bir çözümdür ve kaynakları etkin bir şekilde kullanmayı sağlar. Hipervizör, sanal makinelerin oluşturulması, başlatılması, durdurulması ve yönetilmesi gibi görevleri yerine getirir ve sanal makineler arasında kaynak paylaşımını düzenler.

Soru

- Tip 1 hipervizörleri, Tip 2 hipervizörlerinden ayıran nedir?

- a) Tip 1 hipervizörler fiziksel donanım üzerinde doğrudan, Tip 2 hipervizörler bir ana işletim sisteminin üstünde çalışır.
- b) Tip 1 hipervizörler tam sanallaştırmayı, Tip 2 hipervizörler para-sanallaştırmayı destekler.
- c) Tip 1 hipervizörler masaüstü için, Tip 2 hipervizörler sunucu sanallaştırması için tasarlanmıştır.
- d) Tip 1 hipervizörler, doğrudan donanım erişimine sahip oldukları için Tip 2 hipervizörlere göre daha iyi performans sağlarlar.

Cevap

- Cevap: a)

- Tip 1 hipervizörler fiziksel donanımın üzerinde doğrudan çalışırken, Tip 2 hipervizörler bir ana işletim sisteminin üstünde çalışır. Tip 1 hipervizörler, işletim sistemi ile donanım arasında aracısız bir şekilde çalışır ve bu nedenle daha az katman vardır. Bununla birlikte, Tip 2 hipervizörler bir ana işletim sistemi üzerinde çalışır ve bu nedenle ana işletim sistemi ve donanım arasında bir aracı olarak işlev görür. Bu fark, her iki tip hipervizörün çalışma ortamını ve performansını etkiler. Tip 1 hipervizörler, doğrudan donanıma erişim sağladıkları için genellikle daha iyi performans sunarlar, ancak Tip 2 hipervizörlerin daha esnek dağıtım seçenekleri sunar.

Soru

- Gömülü sistemlerde bir bekçi zamanlayıcının (watchdog timer) amacı nedir?

- a) Donanım bileşenleri arasında zamanlama sinyallerini senkronize etmek
- b) Sistem sağlığını izlemek ve yanıt vermeyen durumlarda sistemi sıfırlamak
- c) Kritik sistem görevlerinin yürütme süresini ölçmek
- d) Senkronizasyon amaçları için kesin zaman referansları oluşturmak

Cevap

- Cevap: b)

- Bir bekleyici zamanlayıcının (watchdog timer) gömülü sistemlerdeki amacı, sistemin sağlığını izlemek ve yanıt vermeyen durumlarda sistemde sıfırlama işlemi gerçekleştirmektir. Gömülü sistemler genellikle kritik görevleri yerine getirir ve bu nedenle sistemdeki herhangi bir hata veya kilitlenme, ciddi sonuçlara yol açabilir. Bekleyici zamanlayıcı, sistemdeki belirli bir süre boyunca düzenli olarak bir sinyal göndermezse, sistem bu durumu algılar ve otomatik olarak sıfırlanır. Bu, sistemdeki hatalı durumları ele almanın ve sistem istikrarını korumanın önemli bir yoludur.

Soru

- İşletim sistemlerinde zamanlayıcılar tarafından kullanılan sistem saati hangi bileşen tarafından üretilir?

- a) Gerçek zamanlı saat (RTC)
- b) Merkezi İşlem Birimi (CPU)
- c) Programlanabilir aralık zamanlayıcı (PIT)
- d) Bellek Yönetim Birimi (MMU)

Cevap

- Cevap: a)

- İşletim sistemlerinde zamanlayıcılar tarafından kullanılan sistem saatini üreten bileşen gerçek zamanlı saattir (RTC). Gerçek zamanlı saat, genellikle sistem anakartının bir parçası olarak bulunan bir donanım bileşenidir. Bu saat, gerçek dünya zamanını temsil eder ve işletim sistemi tarafından çeşitli zamanlama işlemleri için kullanılır. Zamanlayıcılar, bu gerçek zamanlı saati kullanarak belirli sürelerde işlemler yaparlar, zamanlanmış görevleri yürütürler ve zamanlayıcı olaylarını işlerler.

Soru

- Modern işletim sistemlerinde sistem zamanlaması için tipik olarak hangi zamanlayıcı çözünürlüğü kullanılır?

- a) Milisaniye
- b) Mikrosaniye
- c) Nanosaniye
- d) Pikosaniye

Cevap

- Cevap: b)

- Modern işletim sistemlerinde, sistem zamanlaması için tipik olarak mikrosaniye (µs) çözünürlüğü kullanılır. Mikrosaniye, işletim sisteminin çeşitli zamanlama işlemleri için yeterli ayrıntı düzeyini sağlar ve genellikle kullanıcıların algısal sınırlarının altındaki zaman aralıklarını ölçmek için yeterlidir.

Soru

- Sistem çağrısının (system call) temel amacı nedir?

- a) CPU ve bellek tahsisi gibi sistem kaynaklarını yönetmek
- b) Aynı sistemde çalışan farklı süreçler arasında iletişimi kolaylaştırmak
- c) Kullanıcı düzeyi uygulamaların çekirdekten hizmet talep etmesi için bir arayüz sağlamak
- d) Sisteme bağlı çevre birimi aygıtlarının oluşturduğu kesmeleri işlemek

Cevap

- Cevap: c)

- Bir sistem çağrısının (system call) temel amacı, kullanıcı düzeyi uygulamaların çekirdekten hizmet talep etmesi için bir arayüz sağlamaktır. Sistem çağrıları, kullanıcı uygulamalarının işletim sistemi çekirdeğinin sağladığı çeşitli hizmetlere erişmesini sağlar. Örneğin, dosya sistemi erişimi, ağ iletişimi, bellek yönetimi gibi işlemler için sistem çağrıları kullanılır. Kullanıcı uygulamaları, sistem çağrılarını çağırarak işletim sistemi hizmetlerine erişebilir ve çeşitli işlemleri gerçekleştirebilir.

Soru

- İşletim sisteminin hangi bileşeni sistem çağrılarını işlemekten sorumludur?

- a) Çekirdek (Kernel)
- b) Kabuk (Shell)
- c) Aygıt sürücüleri (Device drivers)
- d) Kesme işleyicisi (Interrupt handler)

Cevap

- Cevap: a)

- İşletim sisteminde, sistem çağrılarını işlemekten sorumlu olan bileşen çekirdektir (kernel). Çekirdek, işletim sisteminin temel parçasıdır ve çeşitli sistem kaynaklarını yönetir, süreçler arasında zamanlama yapar, bellek yönetimi sağlar ve sistem çağrılarını işler. Sistem çağrıları, kullanıcı uygulamalarının çekirdeğe belirli hizmetler talep etmesini sağlar. Örneğin, dosya okuma/yazma, ağ iletişimi, bellek tahsisi gibi işlemler sistem çağrıları aracılığıyla gerçekleştirilir.

Soru

- Bir işletim sisteminde yeni bir süreç oluşturmak için hangi sistem çağrısı kullanılır?

- a) fork()
- b) exec()
- c) wait()
- d) exit()

Cevap

- Cevap: a)

- Yeni bir süreç oluşturmak için fork() sistem çağrısı kullanılır. fork() çağrısı, mevcut süreci çoğaltarak yeni bir süreç yaratır. Yeni süreç, çağıran süreçle aynı program kodunu ve durumu paylaşır, ancak farklı bir süreç kimliği (PID) ile çalışır. Bu, genellikle bir işlemin bir çocuk süreç oluşturmasını ve ardından çocuk süreçte başka bir programı çalıştırmasını sağlamak için kullanılır. Örneğin, bir ata süreç yeni bir süreç yaratmak için fork() kullanabilir ve sonra exec() sistem çağrısını kullanarak yeni sürecin farklı bir programı çalıştırmasını sağlayabilir.

Soru

- Bir ata sürecin, çocuk sürecin sonlanmasını beklemek için kullandığı sistem çağrısı hangisidir?

- a) fork()
- b) exec()
- c) wait()
- d) exit()

Cevap

- Cevap: c)

- Bir ata sürecin, çocuk sürecin sonlanmasını beklemek için wait() sistem çağrısını kullanır. wait() çağrısı, ata sürecin bir veya daha fazla çocuk sürecin sonlanmasını beklemesini sağlar. Eğer çocuk süreç henüz sonlanmamışsa, ata süreç wait() çağrısı üzerinde bekler ve çocuk süreç sonlandığında devam eder. Eğer çocuk süreç zaten sonlanmışsa, wait() çağrısı hemen geri döner ve ata süreç, çocuğun sonlandırma durumunu (örneğin, çocuğun çıkış durumu) alabilir.

Soru

- Unix benzeri bir işletim sisteminde, bir sürecin yürütmesini sonlandırmak için hangi sistem çağrısı kullanılır?

- a) fork()
- b) exec()
- c) wait()
- d) exit()

Cevap

- Cevap: d)

- Unix benzeri bir işletim sisteminde, bir sürecin yürütmesini sonlandırmak için exit() sistem çağrısı kullanılır. exit() çağrısı, mevcut süreci sonlandırır ve kontrolü çağıran sürecin devamına bırakır. Bu sistem çağrısı genellikle bir süreç tarafından tamamlandığında veya bir hata durumunda çağrılır. İşlem, exit() çağrısını gerçekleştirdikten sonra, sürece ait kaynaklar serbest bırakılır ve süreç sonlandırılır.

Soru

- Dosya yönetiminde open() sistem çağrısının amacı nedir?

- a) Dosya sisteminde yeni bir dosya oluşturmak için
- b) Var olan bir dosyayı açmak ve sonraki işlemler için bir dosya tanımlayıcısı elde etmek için
- c) Açık olan bir dosya tanımlayıcısını kapatmak ve ilişkili kaynakları serbest bırakmak için
- d) Bir dosyadan verileri bellek tamponuna okumak için

Cevap

- Cevap: b)

- Dosya yönetiminde, open() sistem çağrısı, var olan bir dosyayı açmak ve bu dosya üzerinde yapılacak sonraki işlemler için bir dosya tanımlayıcısı (file descriptor) elde etmek için kullanılır. Bu sistem çağrısı dosyayı açarken gerekli izinleri ve dosyanın yolunu belirtir. Dosya tanımlayıcısı, işlem sırasında bu dosyaya erişim sağlamak için kullanılır. open() çağrısı, dosya oluşturma, dosya okuma, yazma veya değiştirme gibi çeşitli dosya işlemlerinin başlatılmasını sağlar.

Soru

- İşletim sistemlerinde bir süreç denetim bloğunun (PCB) temel amacı nedir?

- a) Bir sürecin yürütülebilir kodunu saklamak için
- b) Sistem kaynaklarının süreçlere tahsisini yönetmek için
- c) Bir sürecin durumu ve özellikleri hakkındaki bilgileri korumak için
- d) Birden çok süreç arasında süreç dışı iletişimi sağlamak için

Cevap

- Cevap: c)

- Süreç denetim bloğunun (PCB) temel amacı, bir sürecin durumu ve özellikleri hakkındaki bilgileri saklamaktır. PCB, işletim sistemi tarafından her süreç için oluşturulur. Bu yapı, sürecin durumu (çalışıyor, hazır, beklemede vb.), program sayacı (PC), kaynaklar (bellek, açık dosyalar, CPU zamanı vb.) gibi bilgileri içerir. PCB, işletim sistemi tarafından süreç yönetimi ve çizelgeleme için kullanılır. Süreç durumu veya süreç kaynak talepleri değiştiğinde, PCB güncellenir ve ilgili bilgiler saklanır. Bu sayede işletim sistemi, süreçleri etkin bir şekilde yönetebilir ve kaynakları adil bir şekilde dağıtabilir.

Soru

- Çoklu iş parçacığı (multithreaded) sürecin özelliği nedir?

- a) Farklı CPU çekirdeklerinde aynı anda birden fazla işlem yürütür
- b) Aynı adres alanı ve kaynakları birden fazla iş parçacığı arasında paylaşır
- c) İşletim sistemi zamanlayıcısı tarafından kesintiye uğratılamaz
- d) Her iş parçacığı için ayrı bir süreç denetim bloğuna (PCB) sahiptir

Cevap

- Cevap: b)

- Çoklu iş parçacığı (multithreaded) süreç, aynı adres alanını ve kaynakları birden fazla iş parçacığı arasında paylaşır. Bu, aynı süre. içinde birden fazla iş parçacığının (thread) çalışabileceği anlamına gelir. İş parçacıkları, süreç içindeki farklı görevleri eşzamanlı olarak gerçekleştirmek için kullanılır. Ancak, tüm iş parçacıkları süreç içindeki aynı adres alanını ve kaynakları paylaşır. Bu, iş parçacıklarının veri paylaşımı ve iletişimi için uygun bir ortam sağlar. Örneğin, bir iş parçacığı bir dosyayı okurken, diğer bir iş parçacığı aynı dosyaya yazabilir veya başka bir iş parçacığı tarafından açılan bellek bölgesine erişebilir.

Soru

- Bir olayın tamamlanmasını bekleyen, örneğin G/Ç tamamlanması gibi bir olayı bekleyen bir süreci temsil eden durum hangisidir?

- a) Çalışıyor (Running)
- b) Hazır (Ready)
- c) Engellenmiş (Blocked)
- d) Sonlanmış (Terminated)

Cevap

- Cevap: c)

- Bir süreç, bir olayın tamamlanmasını beklerken (örneğin, bir G/Ç işleminin tamamlanması), engellenmiş (blocked) durumda olur. Bu durumda, süreç kaynak talebi nedeniyle beklemek zorunda kalır ve işletim sistemi diğer süreçleri yürütmeye devam eder. Süreç, beklediği olay gerçekleştiğinde veya belirli bir zaman aşımı gerçekleştiğinde yeniden hazır duruma gelir ve işletim sistemi tarafından tekrar yürütülür.

Soru

- Hazır kuyruğundaki işlemlerin ortalama bekleme süresini en aza indirmeyi amaçlayan hangi zamanlama algoritmasıdır?

- a) Round Robin
- b) Shortest Job Next (SJN)
- c) First Come, First Served (FCFS)
- d) Shortest Remaining Time First (SRTF)

Cevap

- Cevap: d)

- Shortest Remaining Time First (SRTF), kısa kalan işlem süresine sahip olan süreci önceliklendiren bir algoritmadır. SRTF algoritması, hazır kuyruğundaki tüm süreçlerin kalan işlem süresini izler ve en kısa kalan süreye sahip süreci seçer. Bu şekilde, kısa işlemler öncelikli olarak işlenir ve ortalama bekleme süresi en aza indirilir. Diğer çizelgeleme algoritmalarına göre, SRTF genellikle daha düşük bekleme süreleri sağlar, ancak işlem süreleri dinamik olarak değiştiği için işlem sırası sık sık değişebilir.

Soru

- Süreç senkronizasyonu bağlamında, bir semafor amacı nedir?

- a) Paylaşılan kaynaklara karşılıklı dışlama sağlayarak yarış koşullarını önlemek
- b) Süreçler arasında mesaj iletişimi için iletişim kanalı sağlamak
- c) Eşzamanlı bir programda kritik bölgelerin yürütülmesini planlamak
- d) Süreç yürütme sırasında bellek tahsis etmek ve serbest bırakmak

Cevap

- Cevap: a)

- Semafor işaret flaması (semaphore), bir senkronizasyon aracıdır. Paylaşılan kaynaklara karşılıklı dışlama sağlamak için kullanılır. Yarış koşullarını önlemek için, birden çok süreç aynı anda paylaşılan bir kaynağa erişmeye çalışırken, semaforlar kullanılarak kritik bölgelere girişin düzenlenmesi sağlanır. Semaforlar, özellikle kritik bölgelerde yalnızca bir sürecin aynı anda çalışmasına izin vererek, paylaşılan kaynakların güvenli bir şekilde kullanılmasını sağlar. Bu şekilde, semaforlar yarış koşullarını ve veri uyumsuzluğunu önlemeye yardımcı olur.

Soru

- Süreç ve iş parçacığını doğru bir şekilde ayırt eden ifade hangisidir?

- a) Süreç birden çok iş parçacığı içerir, iş parçacığı ise hafif bir işlemdir.
- b) Süreç işletim sistemi tarafından oluşturulur, iş parçacığı ise uygulama tarafından oluşturulur.
- c) Süreç kendi adres alanına sahiptir, iş parçacıkları ise aynı süreç içinde aynı adres alanını paylaşır.
- d) Süreç birden çok komutu aynı anda yürütebilir, iş parçacığı ise komutları ardışık olarak yürütür.

Cevap

- Cevap: c)

- Süreç ve iş parçacığı arasındaki temel fark, bellek yönetimidir. Süreç, işletim sistemi tarafından ayrılan ve izole edilen kendi bellek alanına sahiptir. Bu nedenle, bir süreç kendisine ait veri, kod ve kaynaklara sahiptir, diğer süreçlerden izole edilmiştir. Bu izolasyon, süreçlerin bağımsız çalışmasını sağlar ve bir hata verdiğinde diğer süreçleri etkilemez. İş parçacıkları genellikle aynı süreç içinde aynı görevi yerine getirmek üzere kullanılır ve birbirleriyle iletişim kurmak için paylaşılan bellek alanını kullanırlar. Bu farklar, süreçlerin daha fazla izolasyon ve güvenlik sağlamasına rağmen, iş parçacıklarının daha hafif ve daha hızlı olmasını sağlar. Süreç birden çok iş parçacığını barındırabilir, ancak her iş parçacığı kendi izole edilmiş adres alanına sahip olmayacaktır.

Soru

- Çoklu süreç sistemlerde paylaşılan kaynaklara eş zamanlı erişimle ilgili ana zorluk nedir?

- a) Bağlantı
- b) Açlık
- c) Yarış koşulları
- d) Öncelik ters çevirme

Cevap

- Cevap: c)

- Yarış koşulları (race conditions), çoklu süreç sistemlerde paylaşılan kaynaklara aynı anda erişmeye çalışan süreçler arasında beklenmeyen ve istenmeyen sonuçların ortaya çıkmasına neden olur. Bu durumlar, programın sırası veya zamanlaması ile ilgili olduğunda ortaya çıkar ve genellikle programın doğruluğunu veya güvenilirliğini etkiler.

Soru

- Çoklu süreç sistemlerde her bir sürece eşit payda CPU zamanı sağlamayı hedefleyen hangi çizelgeleme algoritmasıdır?

- a) Round Robin
- b) İlk Gelen İlk Hizmet (FCFS)
- c) En Kısa İş İlk (SJN)
- d) Öncelikli çizelgeleme

Cevap

- Cevap: a)

- Round Robin, çoklu süreç sistemlerde CPU zamanını eşit bir şekilde dağıtmayı hedefler. Her süreç belirli bir zaman dilimi (zaman dilimi veya dilim) için CPU'ya atanır. Zaman dilimi sona erdiğinde, işlemci başka bir sürece geçer ve bu işlemciye bir sonraki zaman dilimi atanır. Bu işlem, süreçler arasında adil bir dağılım sağlar ve her sürece eşit fırsat verir.

Soru

- Eşzamanlı programlamada kritik bölgenin amacı nedir?

- a) Birden çok süreç tarafından aynı anda yürütülebilen bir kod bölümü tanımlamak
- b) Süreçlerin iletişim kurmasını ve faaliyetlerini senkronize etmesini sağlamak
- c) Paylaşılan kaynaklara birden çok sürecin eşzamanlı erişiminden korumak
- d) Süreçlere önceliklerine bağlı olarak CPU zamanı tahsis etmek

Cevap

- Cevap: c)

- Kritik bölge (critical region), eşzamanlı programlamada paylaşılan kaynaklara eş zamanlı erişimi kontrol etmek için kullanılan bir mekanizmadır. Birden çok süreç aynı anda paylaşılan bir kaynağa erişmeye çalıştığında, bu kaynağın tutarlılığını sağlamak önemlidir. Kritik bölüm, yalnızca bir sürecin aynı anda paylaşılan kaynağa erişebileceği şekilde tasarlanır. Bu, kaynağın tutarlılığını korur ve yarış koşullarını önler.

Soru

- Çoklu süreç sistemlerde senkronizasyon için kilit kullanmanın potansiyel bir dezavantajı nedir?

- a) Kilitlenme (deadlock)
- b) Öncelik ters çevirme
- c) Meşgul bekleme
- d) Açlık

Cevap

- Cevap: c)

- Kilitler, çoklu süreç sistemlerde senkronizasyonu sağlamak için sıkça kullanılır. Ancak, bu mekanizmanın dezavantajı, meşgul bekleme (busy waiting) olarak adlandırılan durumdur. Meşgul bekleme, bir işlemci veya iş parçacığının kilit çözülene kadar sürekli olarak kontrol etme veya bekletme durumudur. Bu süreç boyunca, işlemci işlemi gerçekleştiremez ve sadece kilit çözülünceye kadar bekler. Bu durumda, işlemciyi verimli bir şekilde kullanmak yerine, işlemci sürekli olarak kontrol edilir ve kaynaklar boşa harcanır.

Soru

- Çoklu işlemci bağlamında, bir semaforun amacı nedir?

- a) Kritik kod bölgelerini eşzamanlı erişime karşı korumak
- b) Süreçler arasında mesaj iletimi için bir iletişim kanalı sağlamak
- c) Süreç yürütülürken belleği dinamik olarak tahsis ve serbest bırakmak
- d) Paylaşılan kaynaklara erişimi kontrol etmek ve yarış koşullarını önlemek

Cevap

- Cevap: d)

- Semaforlar, çoklu süreç sistemlerde paylaşılan kaynaklara erişimi kontrol etmek ve yarış koşullarını önlemek için kullanılan bir senkronizasyon mekanizmasıdır. Bir semafor, belirli bir paylaşılan kaynağa erişim hakkını kontrol eder. Süreçler bu kaynağa erişmek istediklerinde, semaforu kullanarak erişim izni alırlar. Semaforlar, belirli sayıda izin verilen eşzamanlı erişimi kontrol etmek için kullanılabilir veya bir işlem kaynağı kullanırken diğer süreçlerin beklemesini sağlayabilir. Bu şekilde, semaforlar paylaşılan kaynakların güvenli ve tutarlı bir şekilde kullanılmasını sağlar ve yarış koşullarını önler.

Soru

- Eşzamanlı programlamada monitörün rolü nedir?

- a) Süreçlerin iletişim kurmasını ve faaliyetlerini koordine etmesi için bir mekanizma sağlamak
- b) Paylaşılan kaynakları birden çok süreç tarafından eş zamanlı erişimden korumak
- c) Yalnızca bir sürecin aynı anda kritik kod bölümünü yürütmesine izin vermek
- d) Paylaşılan verileri ve bunları değiştiren süreçleri kapsüllemek ve karşılıklı dışlama sağlamak

Cevap

- Cevap: d)

- Bir monitör, eşzamanlı programlamada paylaşılan verileri ve bu verileri değiştiren işlemleri kapsüllemek için kullanılan yapılardır. Monitörler, veri ve işlemleri bir araya getirerek, bu verilere aynı anda sadece bir sürecin erişmesine ve değiştirmesine izin verirler. Bu, karşılıklı dışlama (mutual exclusion) sağlar ve paylaşılan verilere eş zamanlı erişimden kaynaklanan sorunları önler. Monitörler aynı zamanda koşullu değişkenler gibi senkronizasyon araçlarını da içerebilir, böylece işlemler arasında iletişim ve koordinasyonu sağlarlar.

Soru

- Süreç yönetiminde "Zombie" (Ölü) durumunda olan süreç nedir?

- a) Giriş/Çıkış tamamlanması gibi bir olayın gerçekleşmesini bekleyen bir süreci gösterir.
- b) Sonlandırılmış ancak hala süreç tablosunda tutulan bir süreci temsil eder.
- c) Askıya alınmış ve daha sonra devam ettirilebilecek bir süreci belirtir.
- d) CPU'da etkin olarak komutları yürüten bir süreci belirtir.

Cevap

- Cevap: b)

- "Zombie" (Ölü) durumu, önemli bir durumdur çünkü bir süreç, ata süreç tarafından sonlandırıldıktan sonra bile hala süreç tablosunda tutuluyor olabilir. Bu, süreç tablosundaki kaynağın boşa harcanmasına ve gereksiz kaynak tüketimine neden olabilir. Ölü bir süreç, çocuk süreci sona erdikten sonra ata süreci sona erdirmediğinde veya ata süreç sona erdiğinde ancak çocuk süreç sonlandırma sinyalini almadığında oluşabilir. Ölü bir süreç, sonlandırıldıktan sonra sistem kaynaklarını serbest bırakmak için özel bir temizlik işlemi tarafından kaldırılmalıdır.

Soru

- Bir sürecin "Sonlandırılmış" durumu neyi ifade eder?

- a) Süreç, Giriş/Çıkış işlemlerinin tamamlanması gibi bir olayın gerçekleşmesini bekliyor.
- b) Süreç yürütmesini tamamladı ve tüm ayrılan kaynakları serbest bıraktı.
- c) Süreç, giriş veya çıkış işlemlerinin tamamlanmasını bekliyor.
- d) Süreç, CPU'da komutları etkin bir şekilde yürütüyor.

Cevap

- Cevap: b)

- Bir sürecin "Sonlandırılmış" durumu, süreç yürütmesinin tamamlandığını ve tüm ayrılan kaynakları serbest bıraktığını gösterir. Bu durum, süreç artık sistem kaynaklarına ihtiyaç duymadığı ve sistem tarafından temizlenmeye veya sonlandırılmaya hazır olduğu anlamına gelir. İşletim sistemi, sonlandırılmış bir işlemi süreç tablosundan kaldırarak ve kullanılan diğer kaynakları serbest bırakarak bu durumu işler.

Soru

- Bir sürecin Giriş/Çıkış işlemlerinin tamamlanması gibi bir olayın gerçekleşmesini beklediği durum hangisidir?

- a) Çalışıyor (Running)
- b) Hazır (Ready)
- c) Engellenmiş (Blocked)
- d) Yeni (New)

Cevap

- Cevap: c)

- "Bloklanmış" durum, bir sürecin Giriş/Çıkış (I/O) tamamlanması gibi bir olayın gerçekleşmesini beklediği durumu ifade eder. Bir süreç, bir Giriş/Çıkış işlemi gerçekleştirdiğinde, süreç genellikle işlemcinin dışındaki bir cihazla etkileşime girer ve sonucu bekler. Örneğin, bir dosya okuma işlemi gerçekleştirirken, işlemci dosyanın okunmasını bekler. Bu süreçte, işlem "Engellenmiş" durumuna geçer ve diğer süreçlerle çalışmak için bekler. İşletim sistemi, Giriş/Çıkış işlemi tamamlandığında süreci uyandırır ve yeniden çalıştırır.

Soru

- Süreç yönetiminde "Hazır" durumunun rolü nedir?

- a) Bir sürecin yürütmesini tamamladığını ve tüm kaynakları serbest bıraktığını belirtir.
- b) Bir sürecin Giriş/Çıkış tamamlanması gibi bir olayın gerçekleşmesini beklediğini gösterir.
- c) Bir sürecin giriş veya çıkış işlemlerinin tamamlanmasını beklediğini belirtir.
- d) Bir sürecin hazır ve bir CPU'ya atanmayı beklediğini belirtir.

Cevap

- Cevap: d)

- "Hazır" durumu, bir işlemin CPU'ya atanma ve yürütme için hazır olduğunu belirtir. İşletim sistemi, süreci çalıştırmak için CPU'ya ihtiyaç duyduğunda, hazır durumunda olan süreçleri seçer ve CPU'ya atanmalarını sağlar. Hazır durumundaki bir süreç, CPU zamanı almak için bekler, ancak o anda yürütülmez. İşletim sistemi, bu süreçler arasında zaman paylaşımı yaparak ve işlemciyi verimli bir şekilde kullanarak süreçleri sırayla çalıştırır.

Soru

- Bir sürecin CPU'da komutları etkin bir şekilde yürüttüğünü belirten durum hangisidir?

- a) Hazır
- b) Engellenmiş
- c) Çalışıyor
- d) Sonlandırıldı

Cevap

- Cevap: c)

- "Çalışıyor" durumu, bir sürecin CPU'da aktif olarak talimatları yürüttüğünü ifade eder. Süreç, CPU üzerindeki işlemci zamanını kullanarak talimatları gerçekleştirir. Bu durumda, süreç sistem kaynaklarına erişir ve belirli bir işlemi gerçekleştirmek için CPU'da çalışır. Çalışan bir süreç, işletim sistemi tarafından CPU'dan alınana veya bir kesinti gerçekleşene kadar çalışmaya devam eder.

Soru

- Bağlam anahtarlama (context switch) işleminin temel amacı nedir?

- a) Bir sürecin durumunu kaydetmek ve başka bir süreci yürütme için durumunu geri yüklemek.
- b) Süreçlere öncelik ve çizelgeleme algoritmasına dayalı olarak CPU zamanı tahsis etmek.
- c) Eşzamanlı yürütülen süreçler arasında paylaşılan kaynaklara erişimi senkronize etmek.
- d) Sisteme bağlı çevre birimleri tarafından üretilen kesmeleri işlemek.

Cevap

- Cevap: a)

- Bağlam anahtarlama, işletim sistemi tarafından bir sürecin durumunu kaydetmek ve CPU'da başka bir süreci yürütmek üzere kaydedilen durumunu geri yüklemek için gerçekleştirilir. Bu işlem, çoklu işlem sistemlerde CPU'nun farklı süreçler arasında geçiş yapmasını sağlar. Bir süreç, CPU'da çalışırken, bir kesinti meydana geldiğinde veya işletim sistemi farklı bir sürecin çalışması gerektiğine karar verdiğinde, süreç durumu kaydedilir ve CPU başka bir sürece geçirilir. Bu şekilde, işletim sistemi farklı süreçleri zaman paylaşımı yöntemiyle çalıştırabilir ve CPU'nun etkin bir şekilde kullanılmasını sağlar.

Soru

- İşletim sisteminin hangi bileşeni bağlam anahtarlamayı gerçekleştirmekten sorumludur?

- a) Çekirdek (Kernel)
- b) Kabuk (Shell)
- c) Aygıt sürücüleri (Device drivers)
- d) Çizelgeleyici (Scheduler)

Cevap

- Cevap: a)

- Bağlam anahtarlama, işletim sistemi içerisinde gerçekleşen önemli bir işlemdir. İşletim sistemi çekirdeği (kernel) tarafından yönetilir. Çekirdek, işletim sisteminin merkezi parçasıdır ve temel işlevleri, süreçleri yönetmek, kaynakları (örneğin, CPU, bellek, dosya sistemi) denetlemek ve sistem kaynaklarını yönetmektir.

Soru

- Sık bağlam anahtarlama işlemlerinin bir sonucu nedir?

- a) Geliştirilmiş sistem tepkisi ve işlem kapasitesi
- b) Süreç durumu kaydedilmesi ve geri yüklenmesi nedeniyle artan ek yük
- c) Sistem kaynaklarının yönetiminde azalan karmaşıklık
- d) Artan hata toleransı ve güvenilirlik

Cevap

- Cevap: b)

- Sık bağlam anahtarlama işlemleri, işletim sistemi tarafından yönetilen çoklu görev bir sistemde yaygın bir durumdur. Ancak, sık bağlam değişimlerinin bir sonucu olarak, süreç durumunun sürekli olarak kaydedilmesi ve geri yüklenmesi işlemi artar. Bu, ekstra işlemci zamanı ve sistem kaynakları gerektiren bir işlemdir, bu da işletim sistemi üzerinde ek bir yük oluşturur.

Soru

- İşletim sistemlerinde bir bağlam değişiminin süresini etkileyen faktör hangisidir?

- a) CPU'nun hızı
- b) Kullanılabilir bellek miktarı
- c) Sistemdeki süreç sayısı
- d) Çizelgeleme algoritmasının verimliliği

Cevap

- Cevap: a)

- Bağlam değişimi, bir işlemcinin bir süreci durdurup başka bir sürece geçiş yapmasıdır. Bağlam değişiminin süresi, birçok faktöre bağlıdır ancak en önemlisi CPU'nun hızıdır. Çünkü bağlam değişimi işlemci tarafından gerçekleştirilir ve işlemci hızı, bu işlemin ne kadar hızlı gerçekleşeceğini belirler.

Soru

- Bir bağlam değişimi sırasında bir işlemin durumunun kaydedilmesinin amacı nedir?

- a) Sürecin kesintiye uğradığı noktadan devam etmesine izin vermek.
- b) Süreçlere öncelik ve çizelgeleme algoritmasına dayalı olarak CPU zamanı tahsis etmek.
- c) Eşzamanlı yürütülen süreçler arasında paylaşılan kaynaklara erişimi senkronize etmek.
- d) Sisteme bağlı çevre birimleri tarafından üretilen kesmeleri işlemek.

Cevap

- Cevap: a)

- Bağlam anahtarlama sırasında, bir süreç durumunun kaydedilmesinin temel amacı, süreci kesintiye uğradığı noktadan devam ettirebilmektir. Bu, işlemci bir başka sürece geçtiğinde veya bir kesme (interrupt) meydana geldiğinde, süreci durdurmak ve daha sonra aynı süreci işlemciye geri döndüğünde sürecin kaldığı noktadan devam etmesini sağlamak için gereklidir. Süreç durumunun kaydedilmesi, işlemcinin durdurulan süreci daha sonra tam olarak aynı durumda devam ettirebilmesini sağlar.

Soru

- Önceden kesmeli çoklu görev sistemlerinde, bağlam anahtarlamayı tetikleyen nedir?

- a) Bir süreç CPU'nun kontrolünü gönüllü olarak bırakır.
- b) Bir kesme meydana gelir, örneğin bir zamanlayıcı kesmesi veya Giriş/Çıkış işleminin tamamlanması.
- c) Bir süreç paylaşılan bir kaynağa erişim isteğinde bulunur.
- d) Bir süreç yürütmesini tamamlar ve çıkış yapar.

Cevap

- Cevap: b)

- Önceden kesmeli çoklu görev sistemlerinde, bağlam anahtarlama genellikle bir kesme (interrupt) meydana geldiğinde tetiklenir. Örneğin, bir zamanlayıcı kesmesi (timer interrupt) belirli bir süre dolduğunda veya bir Giriş/Çıkış işlemi tamamlandığında gerçekleşebilir. Bu tür bir kesme, işletim sistemi çekirdeği tarafından algılanır ve işlemci mevcut süreci duraklatır ve kesme işlemi tarafından işaret edilen başka bir sürece geçiş yapar.

Soru

- Tekrarlanamayan yürütme bağlamında, bir sürecin bir olayın gerçekleşmesini beklerken sürekli olarak geciktiği durumu hangi terim tanımlar?

- a) Kitlenme (Deadlock)
- b) Açlık (Starvation)
- c) Boşa çabalama (Thrashing)
- d) Boşa dönme (Spinning)

Cevap

- Cevap: b)

- "Açlık" terimi, bir sürecin bir olayın gerçekleşmesini beklerken sürekli olarak geciktiği durumu tanımlar. Bu durumda, bir süreç bir olayın gerçekleşmesini beklerken, diğer süreçler tarafından sürekli olarak önceliklendirilir veya olay gerçekleşmezse bekleyici durumda kalır. Bu, süreç için istenen kaynakların sürekli olarak diğer süreçler tarafından alınmasından veya kaynakların hiç serbest bırakılmamasından kaynaklanabilir. Sonuç olarak, süreç bekleyici durumda kalır ve belirli bir olayın gerçekleşmesini beklerken hiçbir işlem yapamaz.

Soru

- Tekrarlanamayan yürütme bağlamında, birden fazla sürecin başka bir süreç tarafından tutulan bir kaynağı sonsuza kadar beklediği durumu hangi terim tanımlar?

- a) Açlık (Starvation)
- b) Kilitlenme (Deadlock)
- c) Boşa çabalama (Thrashing)
- d) Boşa dönme (Spinning)

Cevap

- Cevap: b)

- "Kilitlenme" terimi, birden fazla sürecin belirli bir kaynağı elinde tutan diğer bir süreci beklemesi nedeniyle ilerleyememesi durumudur. Bu durumda, her süreç diğerinin elindeki kaynağa erişmek için beklerken, hiçbiri ilerleyemez ve sistemde bir döngü oluşur. Örneğin, Süreç A, bir kaynağı elinde tutarken, Süreç B aynı kaynağı elde etmek için beklerken, Süreç A da başka bir kaynağı elde etmek için Süreç B'nin kaynağını bekleyebilir. Sonuç olarak, her iki süreç de kaynakları serbest bırakmayı bekler ve sistemde ilerleme olmaz.

Soru

- Birden fazla süreç arasında paylaşılan bellek bölgesine doğrudan erişim sağlayan IPC mekanizması hangisidir?

- a) Mesaj iletişimi (Message passing)
- b) Paylaşımlı bellek (Shared memory)
- c) İşaret flaması (Semaphore)
- d) Boru hattı (Pipes)

Cevap

- Cevap: b)

- Paylaşımlı bellek mekanizması, birden fazla süreç arasında veri paylaşımını sağlar. Süreçlerin aynı bellek bölgesine doğrudan erişim sağlamasına izin verir, bu da veri kopyalama ve iletişim maliyetini azaltır. Süreçler, aynı bellek bölgesine yazabilir ve oradan okuyabilirler. Bu, veri aktarımının hızlı ve etkin olmasını sağlar. Ancak, paylaşımlı bellek kullanırken senkronizasyon önemlidir çünkü aynı bellek bölgesine birden fazla süreç aynı anda erişebilir ve bu da istenmeyen sonuçlara yol açabilir.

Soru

- IPC'de senkron mesaj iletiminin özelliği nedir?

- a) Gönderici ve alıcı, iletişim sırasında senkronize olmak zorunda değildir.
- b) Gönderici, alıcının iletiyi aldığını doğrulayana kadar bekler.
- c) İletiler, alıcıya asenkron gönderim için bir tampon belleğe saklanır.
- d) Süreçler arasında koordinasyon gerektirmeden iletişim gerçekleşir.

Cevap

- Cevap: b)

- Senkron mesaj iletiminde, gönderici bir ileti gönderdikten sonra iletiyi alıcının almasını bekler. Bu, gönderici ve alıcı arasında senkronizasyonu sağlar ve iletişimin güvenilirliğini artırır. Gönderici, iletiyi gönderdikten sonra alıcının onu aldığını doğrulamadan devam etmez. Bu, iletişimin güvenilir ve güvenli bir şekilde gerçekleşmesini sağlar, ancak iletişimde gecikmelere neden olabilir.

Soru

- Aynı ağ üzerinde farklı bilgisayarlarda çalışan süreçler arasındaki iletişim için hangi IPC mekanizması kullanılır?

- a) Paylaşımlı bellek (Shared memory)
- b) Mesaj iletişimi (Message passing)
- c) İşaret flaması (Semaphore)
- d) Boru hattı (Pipes)

Cevap

- Cevap: b)

- Aynı ağ üzerinde farklı bilgisayarlarda çalışan süreçler arasında iletişim için genellikle mesaj iletişimi kullanılır. Mesaj iletişimi, bir süreç veya bilgisayar tarafından diğerine mesaj gönderme ve almayı içerir. Mesaj iletişimi, iletişimdeki taraflar arasında bir bağlantı gerektirmeyen güvenilir bir iletişim mekanizmasıdır. Örneğin, ağ üzerindeki bir bilgisayar bir istemci süreci başlatır ve sunucuya bir istek mesajı gönderir. Sunucu, isteği alır, işler ve yanıt gönderir.

Soru

- IPC bağlamında, işaret flaması (semaphore) amacı nedir?

- a) Süreçler arasında paylaşılan belleğe doğrudan erişim
- b) Süreçler arasında veriyi senkronize bir şekilde transfer etme
- c) Eşzamanlı süreçler arasında paylaşılan kaynaklara senkronize erişim
- d) Süreçler arasında tek yönlü bir iletişim kanalı oluşturma

Cevap

- Cevap: c)

- Semaforlar, eşzamanlı çalışan süreçler arasında paylaşılan kaynaklara erişimi senkronize etmek için kullanılır. Birden fazla süreç aynı anda bir kaynağa erişmeye çalışıyorsa, semaforlar bu erişimi düzenleyerek çakışmaları önler ve kaynakların güvenli kullanılmasını sağlar. Örneğin, bir süreç bir dosyaya yazarken, diğer süreçlerin aynı dosyaya aynı anda yazmamasını sağlamak için kullanılabilir.

Soru

- Hangi IPC mekanizması, bir çift süreç arasında tek yönlü iletişim kanalı sağlar?

- a) Paylaşımlı bellek
- b) Mesaj iletişimi
- c) İşaret flaması
- d) Boru hattı (Pipes)

Cevap

- Cevap: d)

- Boru hattı (pipes), bir çift süreç arasında tek yönlü bir iletişim kanalı sağlar. Bu kanal, veri akışının bir yönden diğerine doğru olduğu bir boru hattı gibi düşünülebilir. Bir süreç, bir boruya veri yazabilir ve diğer süreç bu borudan bu veriyi okuyabilir. Borular bir süreç tarafından oluşturulur ve diğer süreçle iletişim kurmak için kullanılır. Örneğin, bir süreçten çıktı almak ve başka bir süreçten girdi almak için sıklıkla kullanılır. Bu nedenle, borular tek yönlü bir iletişim kanalı sağlar ve çift yönlü iletişim için kullanılamaz.

Soru

- Soketler, ağ ortamında süreçler arası iletişimde hangi rolü oynar?

- a) Uzak süreçler arasındaki paylaşılan bellek bölgelerine erişim sağlar.
- b) Aynı bilgisayarda çalışan süreçler arasında mesaj iletişimini kolaylaştırır.
- c) Farklı bilgisayarlarda çalışan süreçler için iletişim kanalları oluştururlar.
- d) Eş zamanlı süreçler arasında paylaşılan kaynaklara erişimi senkronize eder.

Cevap

- Cevap: c)

- Soketler, farklı bilgisayarlarda çalışan süreçler arasında iletişim kurmak için kullanılır. İki farklı bilgisayar arasında bağlantı sağlayarak, veri alışverişi yapılmasını mümkün kılarlar. Soketler genellikle TCP/IP veya UDP/IP protokollerini kullanarak iletişim kurarlar ve çeşitli ağ uygulamalarında (web tarayıcıları, e-posta istemcileri, dosya paylaşımı programları vb.) yaygın olarak kullanılırlar.

Soru

- Çoklu iş parçacığı programlaması bağlamında, bir iş parçacığının paylaşılan kaynağı değiştirdiği durumda diğer iş parçacığının onu okuma veya değiştirme işlemi sürerken yaşanan durumu hangi terim tanımlar?

- a) Kilitlenme (Deadlock)
- b) Açlık (Starvation)
- c) Yarış koşulu (Race condition)
- d) Kritik bölge (Critical region)

Cevap

- Cevap: c)

- Yarış koşulu (Race condition), çoklu iş parçacığı programlamasında karşılaşılan bir durumdur. Bu durumda, bir iş parçacığı paylaşılan kaynağı okurken veya değiştirirken, başka bir iş parçacığı da aynı kaynağa erişmeye çalışır. Eğer iş parçacıkları arasında senkronizasyon sağlanmamışsa ve kaynağın durumu kontrol edilmeden işlemler gerçekleştirilirse, beklenmedik sonuçlar ortaya çıkabilir. Örneğin, bir iş parçacığı bir değişkeni artırırken, diğeri aynı değişkeni okuyabilir veya değiştirebilir. Bu durumda, değişkenin son değeri tutarsız olabilir.

Soru

- Yarış koşullarını azaltmada bir mutex kilidin temel rolü nedir?

- a) İş parçacıkları arasında paylaşılan kaynaklara karşılıklı dışlama sağlamak
- b) Birden fazla iş parçacığının aynı anda paylaşılan kaynaklara erişmesine izin vermek
- c) Çoklu iş parçacıklı bir uygulamada iş parçacıkları arasında iletişimi kolaylaştırmak
- d) İş parçacıklarına önceliklerine göre CPU zamanı tahsis etmek

Cevap

- Cevap: a)

- Mutex kilidi, yarış koşullarını önlemede önemli rol oynar. Paylaşılan bir kaynağa sadece bir iş parçacığının aynı anda erişmesini sağlayarak, diğer iş parçacıklarının beklemesini ve kaynağın güvenli bir şekilde kullanılmasını sağlar. Bu, eşzamanlı çalışan iş parçacıklarının birbirlerinin çalışmasını engeller ve veri bütünlüğünü korur. Mutex kilitleri, kritik bölge adı verilen kod parçasının aynı anda yalnızca bir iş parçacığı tarafından çalıştırılmasını sağlar.

Soru

- Karşılıklı dışlama bağlamında, kritik bölgenin amacı nedir?

- a) Kodu birden çok iş parçacığında eşzamanlı olarak yürütmek
- b) Süreçler arasında paylaşılan kaynaklara erişimi senkronize etmek
- c) Süreçlerin mesaj iletişimi yoluyla iletişim kurmasına izin vermek
- d) Süreçlere önceliklerine göre CPU zamanı tahsis etmek

Cevap

- Cevap: b)

- Kritik bölge, karşılıklı dışlamanın sağlandığı kod parçasıdır. Bu bölgede, yalnızca bir süreç veya iş parçacığı aynı anda çalışabilir. Bu, paylaşılan bir kaynağa erişimi senkronize eder ve birden fazla süreç veya iş parçacığının aynı anda bu kaynağa erişimini önler. Bu nedenle, kritik bölge, paylaşılan kaynaklara güvenli bir şekilde erişim sağlamak için kullanılır ve yarış koşullarını önler.

Soru

- Karşılıklı dışlama bağlamında, spin lock kullanmanın dezavantajı nedir?

- a) Sık sık bağlam anahtarlamaya bağlı yüksek işlem maliyeti
- b) Kilidin serbest bırakılmasını bekleyen iş parçacıkları arasında kilitlenme
- c) CPU kaynaklarının verimsiz kullanımı
- d) Çok sayıda iş parçacığı için sınırlı ölçeklenebilirlik

Cevap

- Cevap: c)

- Spinlock'lar, bir kaynağa erişimi beklerken işlemciyi meşgul eden döngüleri kullanır. Bu, işlemcinin sürekli olarak belirli bir kilit durumunu kontrol etmesine ve kilit serbest bırakılana kadar tekrar etmesine neden olur. Bu süreç, CPU kaynaklarının verimsiz kullanımına yol açar, çünkü işlemci kilit serbest bırakılana kadar başka işleri yapmak yerine sürekli olarak aynı işlemi tekrarlar. Bu, özellikle işlemcilerin yoğun olarak kullanıldığı durumlarda performans sorunlarına neden olabilir.

Soru

- Çoklu iş parçacıklı programlardaki karşılıklı dışlamanın performans etkisini azaltmak için alınabilecek önlem nedir?

- a) Paylaşılan kaynaklara erişen iş parçacığı sayısını artırmak
- b) Kilitlerin süresini azaltmak için kritik bölgeyi optimize etmek
- c) Senkronizasyon için mutex kilitler yerine spinlock'lar kullanmak
- d) Birden çok iş parçacığının kritik bölgeye eşzamanlı erişimine izin vermek

Cevap

- Cevap: b)

- Karşılıklı dışlamanın performans etkisini azaltmanın bir yolu, kritik bölgeyi optimize etmektir. Kritik bölge, bir iş parçacığının paylaşılan kaynağa erişirken diğer iş parçacıklarını engellemek için kullanılan kod parçasıdır. Kritik bölge, kilitlenme süresini en aza indirmek için mümkün olduğunca küçük olmalıdır. Kilitlerin süresini azaltmak için kritik bölgedeki işlemleri optimize etmek, iş parçacıklarının kilitlerin serbest bırakılmasını beklerken harcadığı süreyi azaltır ve dolayısıyla uygulamanın daha hızlı çalışmasını sağlar.

Soru

- İşletim sistemlerinde kritik bölgeyi ne tanımlar?

- a) Sistem işlemleri için ayrılmış bellek bölümüdür.
- b) Paylaşılan kaynaklara erişilen ve eşzamanlı erişimden korunması gereken kod bölümüdür.
- c) Sık erişilen verilerin saklandığı CPU önbelleğinin bir parçasıdır.
- d) Sistem günlükleri ve hata mesajlarını saklamak için kullanılan disk depolama alanının bir parçasıdır.

Cevap

- Cevap: b)

- Kritik bölge, bir işletim sistemi içinde, birden fazla iş parçacığı veya süreç tarafından kullanılan ve paylaşılan kaynaklara erişimi içeren kod parçasıdır. Bu kaynaklar genellikle bellek, dosya sistemi gibi sistem kaynakları veya donanım aygıtları olabilir. Kritik bölgeye girildiğinde, sadece bir iş parçacığı veya süreç tarafından erişilmesine izin verilir ve diğerlerinin erişimi engellenir. Bu, veri bütünlüğünün korunmasını sağlar ve yarış koşullarını önler.

Soru

- Kritik bölgeleri yönetmede semaforun amacını doğru şekilde açıklayan ifade hangisidir?

- a) Yalnızca bir sürecin aynı anda kritik bölgeye girmesini sağlar.
- b) Birden çok sürecin aynı anda paylaşılan kaynağa erişmesine izin verir.
- c) Süreçler arasında mesaj iletişimini kolaylaştırır.
- d) Paylaşılan kaynaklara erişimi düzenleyerek yarış koşullarını önler.

Cevap

- Cevap: d)

- Semaforlar, kritik bölgelere erişimi koordine ve senkronize ederek, birden çok iş parçacığının aynı anda paylaşılan kaynaklara erişmesini önler. Bu, yarış koşullarını ve veri bütünlüğünü korur. Semaforlar genellikle iş parçacıklarının veya süreçlerin kritik bölgeye girmesine izin veren veya engelleyen bir tür senkronizasyon mekanizması olarak kullanılır.

Soru

- Peterson çözümü, işletim sistemlerindeki hangi sorunu çözmeyi amaçlar?

- a) Kilitlenme (Deadlock)
- b) Yarış koşulu
- c) Açlık
- d) Boşa çabalama (Thrashing)

Cevap

- Cevap: b)

- Peterson çözümü, özellikle karşılıklı dışlama ve senkronizasyonun sağlanması gereken çoklu iş parçacıklı veya çoklu işlemcili sistemlerde, yani eş zamanlı programlama ortamlarında yarış koşullarını çözmeyi amaçlar. Yarış koşulları, birden çok iş parçacığının veya sürecin aynı anda paylaşılan bir kaynağa erişmeye çalışması durumunda ortaya çıkan sorunlardır. Peterson çözümü, bu tür yarış koşullarını önlemek için bir algoritma sağlar, böylece paylaşılan kaynaklara güvenli bir şekilde erişim sağlanır ve veri bütünlüğü korunur.

Soru

- Eş zamanlı programlamadaki Peterson çözümünün temel amacı nedir?

- a) Birden çok sürecin aynı anda paylaşılan kaynaklara erişmesini önlemek
- b) Eş zamanlı süreçler arasında paylaşılan kaynaklara karşılıklı dışlama sağlamak
- c) Süreçlerin mesaj geçişi yoluyla iletişim kurmasını sağlamak
- d) Süreçlere önceliklerine göre CPU zamanı tahsis etmek

Cevap

- Cevap: b)

- Peterson çözümü, eş zamanlı programlamadaki temel sorunlardan biri olan yarış koşullarını çözmek için tasarlanmış bir algoritmadır. Bu şekilde, bir iş parçacığının veya sürecin paylaşılan bir kaynağa eriştiği sırada diğer iş parçacıklarının veya süreçlerin bu kaynağa erişmesi engellenir, böylece veri bütünlüğü korunur.

Soru

- Peterson çözümünü doğru bir şekilde tanımlayan ifade hangisidir?

- a) Yalnızca bir sürecin aynı anda kritik bölgeye girebilmesini sağlar.
- b) Birden çok sürecin aynı anda paylaşılan kaynağa erişimine izin verir.
- c) Süreçler arasında mesaj iletişimini kolaylaştırır.
- d) Yarış koşullarını, paylaşılan kaynaklara erişimi düzenleyerek önler.

Cevap

- Cevap: a)

- Peterson çözümü, eş zamanlı programlama ortamında paylaşılan kaynaklara güvenli bir şekilde erişimi sağlamak için kullanılan bir algoritmadır. Bu çözüm, yalnızca bir sürecin aynı anda kritik bölgeye girebileceğini garanti eder. Bu, kritik bölgedeki paylaşılan kaynakların güvenli bir şekilde kullanılmasını sağlar ve yarış koşullarını önler.

Soru

- Peterson çözümü karşılıklı dışlamayı nasıl sağlar?

- a) Kritik bölgede meşgul bekleyerek
- b) Mutex kilitleri yerine spinlock'ları uygulayarak
- c) Süreçler arasında koordinasyon ve senkronizasyon kullanarak
- d) Birden çok sürecin aynı anda kilitleri tutmasına izin vererek

Cevap

- Cevap: c)

- Peterson çözümü, süreçler arasında karşılıklı dışlamayı sağlamak için uygun koordinasyon ve senkronizasyon mekanizmalarını kullanır. Bu, süreçlerin kritik bölgeye girişlerini ve çıkışlarını düzenler ve böylece aynı anda yalnızca bir sürecin kritik bölgeye girmesini sağlar.

Soru

- Peterson çözümünde paylaşılan kaynaklara erişimi koordine etmek için kullanılan veri yapısı nedir?

- a) Semafor
- b) Mutex kilidi
- c) Bariyer
- d) Bayrak dizisi

Cevap

- Cevap: d)

- Peterson çözümünde, süreçler arasında paylaşılan kaynaklara erişimi koordine etmek için bayrak dizisi kullanılır. Bu bayrak dizisi, her süreç için ayrı bir bayrağın tutulduğu bir veri yapısıdır. Her süreç, kritik bölgeye girmek istediğinde önce kendi bayrağını kaldırır ve daha sonra diğer süreçlerin bayraklarını kontrol ederek kritik bölgeye girebilir. Bu bayraklar, süreçlerin kritik bölgeye girişini düzenlemek için kullanılır ve böylece karşılıklı dışlama sağlanır.

Soru

- İşletim sistemleri bağlamında, "uyandırma" işleminin amacı nedir?

- a) Bir süreci uyutarak geçici olarak çalışmasını askıya almak
- b) Bir süreci sonlandırarak ayrılmış kaynaklarını serbest bırakmak
- c) Daha önce uyuyan bir sürecin çalışmasını yeniden başlatmak
- d) Bir sürece önceliğine bağlı olarak CPU zamanı tahsis etmek

Cevap

- Cevap: c)

- "Uyandırma" işlemi, uyuyan bir sürecin çalışmasını yeniden başlatmayı amaçlar. Bir süreç, genellikle bir olayın gerçekleşmesini beklerken uyur duruma geçer. Bu olay gerçekleştiğinde, işletim sistemi bu işlemi uyandırır ve tekrar çalışmaya başlar. Bu durum, örneğin bir G/Ç işlemi tamamlandığında veya bir sinyal alındığında gerçekleşebilir. Bu sayede, işlem sırayla uyku ve uyanıklık durumları arasında geçiş yapabilir ve işletim sistemi kaynakları etkin bir şekilde yönetebilir.

Soru

- İşletim sistemlerinde "wake up" işlemini tetikleyen nedir?

- a) Bir disk G/Ç işleminin tamamlanması
- b) Süreç tarafından ayarlanan bir zamanlayıcının süresinin dolması
- c) Harici aygıtlar tarafından üretilen bir kesme
- d) Başka bir süreç tarafından yapılan sistem çağrısı

Cevap

- Cevap: a)

- İşletim sistemleri, genellikle bir işlem tamamlandığında süreci uyandırmak için çeşitli mekanizmalar kullanır. İşletim sistemi, disk G/Ç işleminin tamamlandığını algıladığında ilgili işlemi uygulayan iş parçacığını (thread) veya süreci uyandırır.

Soru

- İşletim sistemlerinde "wake up sleep" mekanizmasını kullanmanın temel avantajı nedir?

- a) Bağlam anahtarlamanın getirdiği ek yükü azaltır
- b) Süreçlerin mesaj iletişimi yoluyla iletişim kurmasını sağlar
- c) CPU kaynaklarını etkin bir şekilde yöneterek sistem yanıt verilebilirliğini artırır
- d) Eş zamanlı süreçler arasında adil davranarak açlık durumunu önlemeyi sağlar

Cevap

- Cevap: c)

- İşletim sistemlerinde "wake up sleep" mekanizmasının temel avantajı, CPU kaynaklarını verimli bir şekilde yöneterek sistem yanıt verme tepki süresini azaltmasıdır. Bu mekanizma, işletim sisteminin uyku durumundaki süreçleri uyandırmasına ve süreçlere CPU kaynakları sağlamasına olanak tanır. Bu sayede, sistemdeki süreçler daha hızlı yanıt verebilir ve kullanıcı deneyimi iyileştirilebilir.

Soru

- İşletim sistemlerinde, bir borunun (pipe) asıl amacı nedir?

- a) Bellek kaynaklarını süreçlere tahsis etmek
- b) Eş zamanlı süreçler arasında paylaşılan kaynaklara senkronize erişim
- c) Süreçler arasında tek yönlü iletişim kanalı oluşturmak
- d) Süreçlere önceliklerine göre CPU zamanı tahsis etmek

Cevap

- Cevap: c)

- Bir borunun temel amacı, süreçler arasında tek yönlü iletişim kanalı oluşturmaktır. Bu, bir süreç tarafından yazılan verilerin başka bir süreç tarafından okunabileceği anlamına gelir. Borular, genellikle bir sürecin çıktısını diğer bir sürece giriş olarak aktarmak için kullanılır. Örneğin, bir süreç bir komutun çıktısını üretirken, bu çıktı bir boru aracılığıyla başka bir sürece aktarılabilir ve bu süreç bu çıktıyı kullanarak başka bir işlemi gerçekleştirebilir.

Soru

- İşletim sistemlerinde boru (pipe) hakkında hangi ifade doğrudur?

- a) Süreçler arasında çift yönlü iletişime olanak tanır.
- b) Süreçler arasında paylaşılan bellek bölgelerine doğrudan erişim sağlar.
- c) Süreçler arası iletişim için first-in-first-out (FIFO) veri yapısıdır.
- d) Süreçler arasındaki iletişimi mesaj geçişi yoluyla kolaylaştırır.

Cevap

- Cevap: c)

- Bir boru (pipe), işletim sistemlerinde süreçler arasında first-in-first-out (FIFO) mantığına göre çalışan bir veri yapısıdır. Bu, bir süreç tarafından gönderilen verilerin, alıcı tarafından aynı sırayla alınacağı anlamına gelir. Yani, bir süreç bir veri gönderdiğinde, alıcı bu veriyi aldıktan sonra bir sonraki gönderilen veriyi alır.

Soru

- İşletim sistemlerinde borunun (pipe) özelliği nedir?

- a) Süreçler arasında çift yönlü iletişime olanak tanır.
- b) İki süreç arasında noktadan-noktaya bağlantı kurar.
- c) Süreçler arasında paylaşılan bellek bölgelerine doğrudan erişim sağlar.
- d) Birden fazla süreç arasında çoklu yayın iletişimini destekler.

Cevap

- Cevap: b)

- Bir boru (pipe), işletim sistemlerinde iki süreç arasında noktadan-noktaya bağlantı kurar. Yani, bir boru bir süreç tarafından oluşturulur ve diğer süreç tarafından kullanılır. Bu bağlantı, bir süreç tarafından yazılan verilerin diğer süreç tarafından okunmasını sağlar.

Soru

- Bir süreç boş bir borudan okumaya çalışırsa ne olur?

- a) Süreç, boruda veri mevcut olana kadar engellenir.
- b) Boru, boş olduğunu belirten bir hata döndürür.
- c) Süreç, hiçbir verinin mevcut olmadığını belirten null değeri alır.
- d) Süreç, boruda veri mevcut olana kadar uyku durumuna geçer.

Cevap

- Cevap: a)

- Bir süreç borudan okumaya çalıştığında ve boru boşsa, süreç beklemeye alınır. Yani, süreç boruda veri mevcut olana kadar bekler ve o zamana kadar askıya alınır. Bu işlem, borunun bir noktasına veri yazılana kadar devam eder.

Soru

- İşletim sistemi boruları (pipes) uygulamak için hangi mekanizmayı kullanır?

- a) Paylaşımlı bellek
- b) Semafor tabanlı senkronizasyon
- c) Dosya tanımlayıcıları
- d) Sistem çağrıları

Cevap

- Cevap: c)

- İşletim sistemleri, boruları uygulamak için dosya tanımlayıcıları kullanır. Borular, dosya benzeri bir arayüz sağlarlar ve bu nedenle işletim sistemi bunları dosya tanımlayıcıları aracılığıyla yönetir. Bir süreç, bir boruya yazmak veya bir borudan okumak için ilgili dosya tanımlayıcısını kullanır.

Soru

- Yemek yiyen filozoflar probleminde (Dining Philosophers) her filozof neyi temsil eder?

- a) Bir ortak kaynağa erişim için rekabet eden bir süreci temsil eder.
- b) Bir mutex kilidine erişmeye çalışan bir iş parçacığını temsil eder.
- c) Kritik kod bölgesini yürüten CPU çekirdeğini temsil eder.
- d) Bir kesmeye hizmet etmek için bekleyen aygıtı temsil eder.

Cevap

- Cevap: a)

- Yemek yiyen filozoflar problemi, bir grup filozofun bir masada yemek yemesi ve ortak bir kaynak olan çatal setini kullanması durumunu modelleyen klasik bir senkronizasyon sorunudur. Her filozof, bir süreci temsil eder ve ortak kaynağa (çatal setine) erişmek için rekabet eder. Bu problemde, filozofların çatalları nasıl kullanacakları ve kullandıktan sonra diğer filozoflara ne zaman serbest bırakacakları gibi senkronizasyon sorunları bulunmaktadır.

Soru

- Yemek yiyen filozoflar problemi (Dining Philosophers), işletim sistemlerinde hangi ana zorlukları ortaya çıkarır?

- a) Deadlock (Kilitlenme)
- b) Açlık (Starvation)
- c) Yarış koşulu (Race condition)
- d) Bağlam anahtarlama ek yükü (Context switching overhead)

Cevap

- Cevap: a)

- Yemek yiyen filozoflar problemi, kilitlenme durumlarına yol açma potansiyeline sahip bir senkronizasyon problemidir. Deadlock, sistemdeki süreçlerin birbirlerinin tamamlanmasını beklerken kilitlenmesi durumudur. Yemek yiyen filozoflar problemi özelinde, her filozofun sağındaki ve solundaki çatalı kullanması gerekmektedir. Ancak, bu çatal seti paylaşılan bir kaynaktır ve doğru sırayla çatalı kullanmaları gerekmektedir. Eğer her filozof çatalı almak için beklerse ve çatalı serbest bırakmayı bekleyen diğer bir filozofun çatalı alması gerekiyorsa, deadlock durumu ortaya çıkar.

Soru

- Yemek yiyen filozoflar probleminde (Dining Philosophers) her filozof ne kadar kaynağa (çatala) ihtiyaç duyar?

- a) Bir
- b) İki
- c) Üç
- d) Dört

Cevap

- Cevap: b)

- Yemek yiyen filozoflar problemi, her filozofun iki adet kaynağa (çatala) ihtiyaç duyduğu klasik bir senkronizasyon problemidir. Her filozof, yemek yemek için sağında ve solundaki çatalı kullanmalıdır. Dolayısıyla, her filozof iki adet çatala ihtiyaç duyar.

Soru

- Yemek yiyen filozoflar probleminde (Dining Philosophers) bir semafor kullanmanın amacı nedir?

- a) Filozoflar arasında kilitlenmeyi önlemek için
- b) Paylaşılan kaynaklara (çatal) karşılıklı dışlama sağlamak için
- c) Birden fazla filozofun aynı anda çatala erişmesine izin vermek için
- d) Filozoflar arasında yemek masasına erişimi senkronize etmek için

Cevap

- Cevap: b)

- Yemek yiyen filozoflar problemi, filozofların ortak kaynak olan çatal setine güvenli bir şekilde erişmesini sağlamak için senkronizasyon mekanizmalarının kullanılmasını gerektirir. Bu mekanizmalardan biri semaforlardır. Semaforlar, paylaşılan kaynaklara (çatala) karşılıklı dışlama sağlayarak, yani aynı anda sadece bir filozofun bir çatalı almasını sağlayarak, kaynakların doğru bir şekilde kullanılmasını sağlar.

Soru

- Yemek yiyen filozoflar probleminden kilitlenmeyi önlemek için hangi koşul sağlanmalıdır?

- a) En az bir filozof tüm gerekli kaynakları edinebilmelidir.
- b) Her filozof, sol çatalı almadan önce sağ çatalı beklemelidir.
- c) Filozoflar, dairesel bekleme (circular wait) durumunu önlemek için kaynakları sıkı bir sıra ile serbest bırakmalıdır.
- d) Filozoflar senkronizasyon olmadan kaynaklara aynı anda erişmelidir.

Cevap

- Cevap: c)

- Yemek yiyen filozoflar problemi, deadlock oluşumunu engellemek için filozofların çatalı sıkı bir sıra ile alıp bırakmasını gerektirir. Bu, dairesel bekleme durumunu önler. Örneğin, her filozof, sağ çatalı almadan önce sol çatalı serbest bırakmalıdır. Bu sayede, herhangi bir filozof, bir çatalı almak için diğer filozofun çatalını beklemeyecek ve dairesel bir bekleme durumu oluşturmayacaktır.

Soru

- İşletim sistemlerinde çoklu iş parçacığı kullanmanın temel avantajı nedir?

- a) Azaltılmış bellek tüketimi
- b) Geliştirilmiş hata toleransı
- c) Daha iyi yanıt verme ve eş zamanlılık
- d) Basitleştirilmiş programlama modeli

Cevap

- Cevap: c)

- Çoklu iş parçacığı (multithreading), işletim sistemlerinde birçok avantaj sunar, en önemlisi sistem yanıt tepki verimliliğini artırması ve eş zamanlılık sağlamasıdır. Çoklu iş parçacığı sayesinde, bir süreç içindeki birden fazla iş parçacığı aynı anda çalışabilir, böylece sistem daha hızlı ve daha etkili bir şekilde yanıt verebilir. Aynı zamanda, birden fazla iş parçacığı arasındaki işbirliği ve veri paylaşımı daha kolay hale gelir, bu da genel sistem performansını artırır.

Soru

- Çoklu iş parçacığı (multithreading) kavramında, bir iş parçacığı (thread) neyi temsil eder?

- a) Kendi bellek alanına sahip ayrı bir süreci
- b) Aynı bellek alanını paylaşan bir süreç içindeki bir yürütme birimini
- c) Mesaj geçişi yoluyla iletişim kuran bağımsız bir süreci
- d) Süreçler arasında iletişimi kolaylaştıran donanım bileşenini

Cevap

- Cevap: b)

- Bir iş parçacığı (thread), bir süreç içinde yürütülen birimlerden biridir ve aynı süreç içindeki diğer iş parçacıkları ile aynı bellek alanını paylaşır. Bu, iş parçacıklarının birbirleriyle doğrudan iletişim kurmalarını ve aynı bellek alanını kullanarak veri paylaşmalarını sağlar. Bir iş parçacığı, bir süreç içindeki diğer iş parçacıkları ile işbirliği yaparak belirli bir görevi gerçekleştirmek için birlikte çalışır.

Soru

- Çoklu iş parçacığı (multithreading) çoklu süreçten (multiprocessing) nasıl farklıdır?

- a) Çoklu iş parçacığı, farklı CPU'lar üzerinde yürütülen birden fazla süreci içerir.
- b) Çoklu süreç, tek bir süreç içinde birden fazla iş parçacığının yürütülmesine izin verir.
- c) Çoklu iş parçacığı, bir süreç içindeki iş parçacıkları arasında aynı bellek alanını paylaşır.
- d) Çoklu süreç, süreçleri birbirinden izole ederek hata toleransını artırır.

Cevap

- Cevap: c)

- Çoklu iş parçacığı ve çoklu süreç (multiprocessing), paralel yürütme sağlayan iki farklı yaklaşımdır. Çoklu iş parçacığı, bir süreç içinde birden fazla iş parçacığının yürütülmesini sağlar ve bu iş parçacıkları aynı süreç içindeki diğer iş parçacıkları ile aynı bellek alanını paylaşır. Bu, veri paylaşımını ve işbirliğini kolaylaştırır. Öte yandan, çoklu süreç (multiprocessing), farklı işlemciler üzerinde farklı süreçlerin yürütülmesine izin verir. Her süreç kendi ayrı bellek alanına sahiptir ve iş parçacıkları arasında veri paylaşımı doğrudan olmaz.

Soru

- İşletim sistemlerinde kullanıcı düzeyi iş parçacıklarını (user-level threads) çekirdek düzeyi iş parçacıklarından (kernel-level threads) ayıran özellik nedir?

- a) Kullanıcı düzeyi iş parçacıkları doğrudan donanım kaynaklarına erişime sahiptir.
- b) Çekirdek düzeyi iş parçacıkları tamamen işletim sistemi tarafından yönetilir.
- c) Kullanıcı düzeyi iş parçacıkları, CPU'ya yakınlığı nedeniyle daha hızlıdır.
- d) Çekirdek düzeyi iş parçacıkları, kullanıcı uygulamaları tarafından oluşturulabilir ve yönetilebilir.

Cevap

- Cevap: b)

- Kullanıcı düzeyi iş parçacıkları ve çekirdek düzeyi iş parçacıkları arasındaki ana fark, nasıl yönetildikleridir. Çekirdek düzeyi iş parçacıkları, işletim sistemi çekirdeği tarafından doğrudan yönetilir. İşletim sistemi, iş parçacıklarını oluşturur, çizelgeler, duraklatır ve sonlandırırken, kullanıcı sadece iş parçacıklarını yaratır ve sonlandırır. Öte yandan, kullanıcı düzeyi iş parçacıkları kullanıcı tarafından oluşturulur ve yönetilir. İşletim sistemi, kullanıcı düzeyi iş parçacıklarını görmeyebilir ve onları doğrudan yönetmez. Kullanıcı, iş parçacıklarını oluşturur, planlar ve çeşitli iş parçacığı kütüphaneleri aracılığıyla iş parçacıklarının yönetimini sağlar.

Soru

- Kullanıcı düzeyi iş parçacığı kütüphanesi, engelleyici (blocking) sistem çağrılarını nasıl işler?

- a) Çağrı tamamlanana kadar farklı bir kullanıcı düzeyi iş parçacığına geçer.
- b) Çağrıyı çekirdek düzeyi iş parçacığı yöneticisine devreder.
- c) Çağrı tamamlanana kadar tüm kullanıcı düzeyi iş parçacıklarını askıya alır.
- d) Engellenen çağrıyı iptal eder ve yürütmeye devam eder.

Cevap

- Cevap: a)

- Kullanıcı düzeyi iş parçacığı kütüphanesi, engelleyici sistem çağrılarını işlerken genellikle blokeden kaçınmak için çeşitli yöntemler kullanır. Bu yöntemlerden biri, engellenen bir çağrının etkilerini azaltmak için o anda bloke olan iş parçacığını başka bir kullanıcı düzeyi iş parçacığıyla değiştirmektir. Böylece, engellenen çağrının tamamlanmasını beklerken başka bir iş parçacığı yürütülür ve işlemci zamanı daha verimli kullanılır.

Soru

- Kullanıcı düzeyi iş parçacıklarının dezavantajına yol açabilecek bir senaryo nedir?

- a) Ayrıntılı (fine-grained) iş parçacığı kontrolüne ihtiyaç duyulduğunda
- b) Farklı işletim sistemleri arasında taşınabilirlik gerektiğinde
- c) Uygulama yüksek performans ve ölçeklenebilirlik gerektirdiğinde
- d) Uygulama, G/Ç işlemleri için sistem çağrılarına yoğun bir şekilde bağımlı olduğunda

Cevap

- Cevap: d)

- Kullanıcı düzeyi iş parçacıkları, iş parçacıklarının yönetimi ve planlaması konusunda kullanıcıya daha fazla kontrol sağlar, ancak bazı senaryolarda dezavantajlara yol açabilir. Özellikle, uygulamanın G/Ç (giriş/çıkış) işlemleri için sistem çağrılarına yoğun bir şekilde bağımlı olduğu durumlarda, kullanıcı düzeyi iş parçacıkları dezavantajlı olabilir. Sistem çağrıları, işletim sistemi çekirdeği içinde gerçekleşir ve iş parçacıkları bu çağrıların tamamlanmasını beklerken genellikle bloklanır. Kullanıcı düzeyi iş parçacıkları durumunda, eğer bir iş parçacığı bir G/Ç işlemi başlatırsa ve bu işlem bloklanırsa, bu iş parçacığı tarafından kullanılan tüm diğer iş parçacıkları da beklemek zorunda kalabilir. Bu, genellikle uygulamanın yanıt verme süresini etkileyebilir ve performansı azaltabilir.

Soru

- Öncelik temelli çizelgeleme algoritmalarının özellikleri nelerdir?

- a) Tüm süreçlere eşit CPU zamanı ayırarak adil bir dağılım sağlarlar.
- b) Süreçleri varış zamanlarına göre önceliklendirirler.
- c) Her sürece sayısal bir öncelik değeri atarlar.
- d) Çizelgeleme kararları için süreçlerin çalışma süresine bakarlar.

Cevap

- Cevap: c)

- Öncelik temelli çizelgeleme algoritmaları, her sürece öncelik değeri atayarak çalışır. Bu öncelik değerleri, süreçlerin önem derecesini veya önceliğini temsil eder. Öncelik değeri yüksek olan işlemler, öncelik değeri düşük olanlara göre daha önce çalıştırılır.

Soru

- Süreçlerin ortalama dönüş zamanını en aza indirmeyi hedefleyen çizelgeleme algoritması hangisidir?

- a) En kısa kalan süre önce (Shortest remaining time first - SRTF)
- b) Round-robin (RR)
- c) En kısa iş önce (Shortest job next - SJN)
- d) İlk gelen, ilk hizmet (First-come, first-served - FCFS)

Cevap

- Cevap: a)

- En kısa kalan süre önce (SRTF) algoritması, süreçlerin geriye kalan tamamlanma sürelerine göre önceliklendirilir. Bu algoritma, her zaman CPU'ya en kısa sürecek süreci seçer, bu da ortalama dönüş zamanını minimize etmeyi amaçlar.

Soru

- En kısa iş önce (SJN) çizelgeleme kullanmanın dezavantajlarından biri nedir?

- a) Uzun süre çalışan süreçlerin açlık yaşamasına neden olabilir.
- b) Her sürecin yürütme süresinin bilinmesini gerektirir.
- c) Kesme tabanlı çizelgeleme (preemptive scheduling) desteği sağlamaz.
- d) Daha yüksek bağlam anahtarlama üstünlüğüne neden olur.

Cevap

- Cevap: b)

- En kısa iş önce (SJN), her sürecin yürütme süresinin bilinmesine dayanır ve işlemciye en kısa sürecek süreci seçer. Ancak, gerçek dünya senaryolarında, her sürecin yürütme süresini önceden bilmek zor olabilir veya imkansız olabilir. Bu, SJN zamanlamanın pratikte uygulanmasını zorlaştırabilir ve bazı durumlarda yanıltıcı sonuçlara yol açabilir.

Soru

- Basitliği ve adilliğiyle tanınan çizelgeleme algoritması hangisidir?

- a) İlk gelen, ilk hizmet (First-come, first-served - FCFS)
- b) Round-robin (RR)
- c) Çok seviyeli geri besleme kuyruğu (Multilevel feedback queue)
- d) En kısa iş önce (SJN)

Cevap

- Cevap: b)

- Round-robin (RR), basitliği ve adil davranışıyla bilinir. Bu algoritma, süreçlere eşit parçalar halinde zaman paylaşımı yapar ve her bir süreç için sırayla CPU zamanı tahsis eder. Bu nedenle, tüm süreçler eşit fırsatlara sahiptir ve adil bir şekilde işlemci kaynaklarından yararlanır.

Soru

- Hangi çizelgeleme algoritması süreçlerin zaman içinde CPU'yu eşit olarak paylaşmasına izin verir?

- a) Öncelik tabanlı (Priority based)
- b) Round-robin (RR)
- c) En kısa kalan süre önce (Shortest remaining time first - SRTF)
- d) Çok seviyeli geri besleme kuyruğu

Cevap

- Cevap: b)

- Round-robin (RR) algoritması, CPU'yu eşit olarak paylaşmak için tasarlanmıştır. Bu algoritma, süreçlere eşit zaman dilimleri (çoğunlukla kesintisiz) atar ve süreçler bu zaman dilimleri boyunca CPU'yu kullanır. Bu şekilde, her süreç, diğerleri ile eşit şartlarda CPU kaynaklarına erişebilir ve CPU'nun paylaşılması zaman içinde adaletli bir şekilde sağlanır.

Soru

- Kesme tabanlı (preemptive scheduling) ile kesme tabanlı olmayan zamanlama (non-preemptive scheduling) arasındaki fark nedir?

- a) Kesme tabanlı, süreçlerin CPU'yu isteğe bağlı bırakmasına izin verir.
- b) Kesme tabanlı, süreçleri zorla keserek CPU zamanını daha yüksek öncelikli süreçlere tahsis eder.
- c) Kesme tabanlı olmayan, süreçlerin CPU'yu isteğe bağlı olarak bırakana kadar çalışmasına izin verir.
- d) Kesme tabanlı olmayan, CPU zamanını süreç önceliğine göre ayırır.

Cevap

- Cevap: b)

- Kesme tabanlı çizelgeleme, süreçleri zorla keserek CPU zamanını daha yüksek öncelikli süreçlere tahsis eder. Bu, yüksek öncelikli süreçlerin öncelikli olarak çalışmasını sağlar ve düşük öncelikli süreçlerin CPU'yu kullanmasını engeller. Bu nedenle, kesme tabanlı çizelgeleme, CPU zamanını sürekli olarak yüksek öncelikli süreçlere yönlendirerek sistem yanıt süresini artırabilir.

Soru

- En kısa iş önce (SJN) algoritmasının bir sınırlaması nedir?

- a) Süreç açlığına neden olabilir.
- b) CPU kullanma (burst) sürelerinin kesin bilgisini gerektirir.
- c) Kesme tabanlı çizelgeleme için uygun değildir.
- d) Sık sık bağlam anahtarlamadan dolayı yüksek ek iş yüküne neden olur.

Cevap

- Cevap: b)

- En kısa iş önce (SJN), her süreç için tahmini CPU kullanım (burst) sürelerine dayanır ve her zaman en kısa sürecek süreci seçer. Ancak, gerçek CPU kullanım sürelerini tam olarak bilmek zordur ve çoğu durumda imkansızdır. Bu nedenle, SJN gerçek dünya senaryolarında uygulanması zor olabilir ve yanıltıcı sonuçlara yol açabilir.

Soru

- Hangi çizelgeleme algoritması işlem önceliklerini dinamik olarak davranışlarına göre ayarlar?

- a) En kısa kalan süre önce (Shortest remaining time first - SRTF)
- b) Çok seviyeli geri besleme kuyruğu
- c) Öncelik tabanlı
- d) Round-robin (RR)

Cevap

- Cevap: b)

- Çok seviyeli geri besleme kuyruğu çizelgeleme algoritması, süreç önceliklerini dinamik olarak ayarlayarak süreçlerin davranışlarına uyum sağlar. Bu algoritma, süreçlerin önceliklerini, davranışlarına (örneğin, CPU kullanımı, vb.) göre değiştirebilir. Daha öncelikli süreçler daha fazla CPU zamanı alırken, daha az öncelikli süreçler daha az CPU zamanı alır. Bu, sistemdeki iş yükünü daha dengeli bir şekilde dağıtarak performansı artırabilir.

Soru

- Hangi çizelgeleme politikası, etkileşimli süreçler için daha iyi yanıt süresi sağlamayı amaçlar?

- a) İlk gelen, ilk hizmet (First-come, first-served - FCFS)
- b) Öncelik tabanlı
- c) En kısa iş önce (Shortest job next - SJN)
- d) Round-robin (RR)

Cevap

- Cevap: b)

- Öncelik tabanlı çizelgeleme, süreçlere öncelik değerleri atar ve bu öncelik değerlerine göre süreçleri önceliklendirir. Bu, daha yüksek öncelikli süreçlerin daha hızlı bir şekilde CPU'ya erişmesini sağlar. Etkileşimli süreçler genellikle daha yüksek önceliklere sahiptir çünkü kullanıcılarının hızlı yanıt almasını gerektirirler. Bu nedenle, öncelik tabanlı zamanlama politikası etkileşimli süreçler için daha iyi yanıt süresi sağlamayı amaçlar.

Soru

- Hangi çizelgeleme algoritması konvoy etkisine neden olabilir?

- a) Öncelik tabanlı
- b) En kısa kalan süre önce (SRTF)
- c) İlk gelen, ilk hizmet (First-come, first-served - FCFS)
- d) Round-robin (RR)

Cevap

- Cevap: c)

- İlk gelen, ilk hizmet (FCFS) algoritması, süreçleri sırayla işleme alır ve işlem sırası geldiğinde tüm işlemci kaynaklarını kullanır. Bu nedenle, uzun süre çalışan bir süreç diğer süreçlerinin tamamlanmasını beklemek zorunda kalabilir, bu da kısa süreçlerin bekleme sürelerini artırabilir. Bu durum, kısa süreçlerin "konvoy" halinde uzun süreli süreçlerin arkasında beklemesine neden olabilir.

Soru

- Hangi çizelgeleme politikası yaşlanmaya (aging) eğilimlidir?

- a) En kısa iş önce (SJN)
- b) Öncelik tabanlı
- c) Çok seviyeli geri besleme kuyruğu
- d) Round-robin (RR)

Cevap

- Cevap: c)

- Çok seviyeli geri besleme kuyruğu politikası, süreçlerin önceliklerini ve işlem sıralamasını dinamik olarak ayarlar. Bu politika, bir süreç belirli bir süre boyunca düşük bir öncelik seviyesinde kaldıysa, önceliğini artırarak süreci öne çıkarır. Bu şekilde, yaşlı süreçlerin (uzun süre bekleyen) önceliği artar ve öncelikli bir şekilde işlenir, bu da yaşlanma fenomenini önler.

Soru

- Eşzamanlama için spinlock kullanmanın potansiyel bir dezavantajı nedir?

- a) Sık sık bağlam anahtarlamadan kaynaklanan artan ek iş yükü
- b) Öncelik ters çevirme (priority inversion) riski
- c) Daha yüksek kilitlenme olasılığı
- d) Daha yüksek CPU kullanımı

Cevap

- Cevap: d)

- Spinlock'lar, bir kaynağa erişim sağlamak için sürekli olarak kaynağı kontrol eden iş parçacıklarını bekletme stratejisini kullanır. Bu, işlemciyi sürekli meşgul eder ve işlemciyi sürekli olarak çalışır durumda tutar. Bu nedenle, spinlock'lar kullanıldığında CPU'nun daha fazla kullanılması olasıdır.

Soru

- Hangi senkronizasyon ilkesi, iş parçacıklarının kısa bir süreliğine kilidi bırakmasına ve bir koşulun karşılanmasını beklemesine izin verir?

- a) Bariyer (Barrier)
- b) Mutex kilidi (Mutex lock)
- c) Semafor (Semaphore)
- d) Koşul değişkeni (Condition variable)

Cevap

- Cevap: d)

- Koşul değişkeni, bir iş parçacığının kritik bölgeden (kilitlemeyle korunan bir bölge) çıkmasına ve bir koşulun karşılanmasını beklemesine izin verir. İş parçacığı, koşul değişkeni üzerinde sinyal alana kadar bekler ve bu sinyal, diğer bir iş parçacığı tarafından belirli bir koşulun karşılandığını gösterir. Bu sayede iş parçacığı, kilidi bırakabilir ve beklemeye geçebilir, diğer iş parçacıkları kritik bölgeye erişebilir. Koşul değişkeni, iş parçacıklarının birbirleriyle iletişim kurmasını ve işbirliği yapmasını sağlar.

Soru

- Hangi senkronizasyon mekanizması, iş parçacıklarının yürütmesini belirlenen bir noktada senkronize etmelerine izin verir?

- a) Mutex kilidi
- b) Bariyer
- c) Koşul değişkeni
- d) Semafor

Cevap

- Cevap: b)

- Bariyer, bir dizi iş parçacığının belirli bir noktada eşleşmesini sağlar. İş parçacıkları, bariyer noktasına ulaştıklarında bekler ve diğer iş parçacıklarının da aynı noktaya ulaşmasını beklerler. Tüm iş parçacıkları belirli bir noktaya ulaştığında, bariyer kaldırılır ve iş parçacıkları eşzamanlı olarak devam eder.

Soru

- Senkronizasyonda mutex kilidini spinlock'tan ayıran özellik nedir?

- a) Mutex kilitleri bloke etmeyen (non-blocking) yapılara sahiptir, spinlock'lar ise iş parçacıklarının belirsiz süre beklemesine neden olabilir.
- b) Mutex kilitleri öncelik devrini sağlar, spinlock'lar ise sağlamaz.
- c) Mutex kilitleri kullanıcı alanında (user space) uygulanır, spinlock'lar ise çekirdek alanında (kernel space) uygulanır.
- d) Mutex kilitleri, spinlock'larla karşılaştırıldığında kısa kritik bölgeler için daha az verimlidir.

Cevap

- Cevap: a)

- Mutex kilitleri, kilitleme işlemi başarısız olduğunda iş parçacıklarının beklemesine ve uyumasına neden olurken, spinlock'lar iş parçacıklarının belirli bir koşulu beklemesi için sürekli olarak döngüye girerler. Bu nedenle, bir spinlock kullanılırken, bir iş parçacığı diğer bir iş parçacığının kilidini bırakmasını beklerken sürekli işlemci kaynağı kullanır ve bu, işlemciyi etkili bir şekilde meşgul edebilir. Bunun aksine, bir mutex kilit kullanılırken, iş parçacıkları diğer iş parçacıklarının kilidi bırakmasını beklerken uyuyabilir ve işlemci kaynağı daha etkin bir şekilde kullanılır.

Soru

- Senkronizasyonda bir okuyucu-yazar kilidinin asıl amacı nedir?

- a) Birden fazla iş parçacığının aynı anda paylaşılan kaynağı okumasına izin vermek, ancak yalnızca bir iş parçacığının yazmasına izin vermek
- b) Aynı kaynağa erişen birden fazla okuyucu arasında kilitlenmeyi önlemek
- c) Yazıcıların paylaşılan bir kaynağa erişiminde okuyuculara karşı önceliğini sağlamak
- d) Bir paylaşılan kaynağa erişen hem okuyuculara hem de yazıcılara karşılıklı dışlama sağlamak

Cevap

- Cevap: a)

- Okuyucu-yazar kilidi, bir paylaşılan kaynağa aynı anda birden fazla okuma işlemine izin verirken, yalnızca bir yazma işlemine izin veren bir senkronizasyon mekanizmasıdır. Bu, birden çok iş parçacığının kaynağı okumasına izin vererek performansı artırırken, aynı anda yalnızca bir iş parçacığının kaynağı değiştirmesine izin vererek tutarlılığı korur.

Soru

- Sistemde kilitlenmenin (deadlock) oluşması için gerekli koşul hangisidir?

- A) Karşılıklı dışlama (Mutual exclusion)
- B) Tut ve bekle (Hold and wait)
- C) Kesme yok (No preemption)
- D) Döngüsel bekleme (Circular wait)

Cevap

- Cevap: D)

- Kilitlenme, bir sistemde birden fazla sürecin birbirlerinin kaynaklarını kullanabilmeleri için birbirlerini beklemeleri ve böylece hiçbir sürecin ilerlememesi durumudur. Kilitlenme için gerekli olan koşullardan biri de döngüsel beklemedir. Bu, süreçler arasında bir zincirleme etkisi yaratarak hiçbir sürecin ilerlemesine izin vermez.

Soru

- Sistemde kilitlenmeleri (deadlock) önlemek için hangi algoritma "wait-die" şemasını kullanır?

- A) Banker's algoritması (Banker's algorithm)
- B) Bekleme grafiği algoritması (Wait-for graph algorithm)
- C) Kaynak tahsis grafiği algoritması (Resource allocation graph algorithm)
- D) Öncelik devralma protokolü (Priority inheritance protocol)

Cevap

- Cevap: A)

- Wait-die şeması, Banker's algoritmasında kullanılan bir deadlock önleme tekniğidir. Algoritma, kaynak talebini kabul etmenin sistemi güvenli bir durumda bırakıp bırakmayacağını kontrol eder. Talep kabul edildiğinde sistem güvensiz bir duruma girecekse, wait-die şeması uygulanır. Bu şemada:
- Bekleyen Süreç: Talep eden süreç bekleyen bir duruma getirilir.
- Tutan Süreç: Bekleyen sürecin talep ettiği kaynakları tutan bir süreç başka bir kaynak talebinde bulunursa, tutan süreç sonlandırılır.

Soru

- Banker's algoritmasının kilitlenme önleme için kullanılmasının başlıca dezavantajı nedir?

- A) Kaynak yetersizliğine neden olabilir
- B) Merkezi bir otorite gerektirir
- C) Kilitlenmeleri tespit edemez
- D) Hesaplama maliyeti yüksektir

Cevap

- Cevap: D)

- Banker's algoritması, sistemdeki mevcut kaynak miktarını ve her bir sürecin maksimum kaynak talebini göz önünde bulundurarak kaynak tahsisini gerçekleştirir. Ancak bu, sistemdeki her bir sürecin herhangi bir anda talep edebileceği maksimum kaynak miktarını tahmin etmeyi gerektirir. Bu, sistemdeki kaynakların büyük bir kısmının kullanımda olabileceği durumları hesaba katmak anlamına gelir. Bu nedenle, algoritmanın karmaşıklığı ve hesaplama maliyeti oldukça yüksektir. Bu sürekli hesaplama ve güncelleme işlemi, sistemdeki performansı düşürebilir ve kaynakların etkin kullanımını zorlaştırabilir.

Soru

- Aşağıdakilerden hangisi kilitlenmeleri (deadlock) ele almak için kullanılan yöntemlerden değildir?

- A) Kilitlenme önleme (Deadlock prevention)
- B) Kilitlenme tespiti ve kurtarma (Deadlock detection and recovery)
- C) Kilitlenme kaçınma (Deadlock avoidance)
- D) Kilitlenme kabulü (Deadlock acceptance)

Cevap

- Cevap: D)

- Kilitlenme, işletim sistemlerinde istenmeyen bir durumdur ve bu durumlar önlenmeye, tespit edilmeye ve çözülmeye çalışılır. Ancak, "Kilitlenmeyi kabul" bir çözüm yöntemi değildir. Kilitlenme durumlarının kabul edilmesi, kullanıcıya veya sistem yöneticisine kilitlenme durumunun var olduğunu belirten bir uyarı vermekten öteye gitmez.

Soru

- Hangi yaklaşım, sistemin kilitlenme durumuna girmesine izin verir ve ardından ondan kurtulur?

- A) Kilitlenme önleme
- B) Kilitlenme tespiti
- C) Kilitlenme kaçınma
- D) Kilitlenme kurtarma

Cevap

- Cevap: D)

- Kilitlenme tespit edildiğinde, sistem belirli bir algoritma veya strateji kullanarak kilitlenmenin gerçekleştiğini belirler. Daha sonra, kilitlenmeyi çözmek veya sistemdeki kaynakları serbest bırakmak için uygun adımlar atılır. Örneğin, kilitlenen süreçlerin durdurulması, kaynakların serbest bırakılması veya kilitlenmeye neden olan koşulların değiştirilmesi gibi adımlar atılabilir.

Soru

- Kilitlenme önleme durumunda, sistem "tut ve bekle" koşulunu önlemek için ne yapar?

- A) Kaynak tahsisi, dairesel bekleme olasılığını ortadan kaldıracak şekilde yapılır
- B) Gerektiğinde kaynaklar önceden alınır
- C) Kaynakları hemen sağlanamayan süreçler sonlandırılır
- D) Süreçlere istedikleri tüm kaynaklar birden sağlanır

Cevap

- Cevap: A)

- Bu koşulda, bir süreç, ilerlemek için ihtiyaç duyduğu kaynakları beklerken başka bir süreç tarafından tutulan kaynaklara ihtiyaç duyar ve bu da bir döngüsel bekleyişe yol açar. Sistem, bir sürecin ihtiyaç duyduğu tüm kaynakları almadan önce onları tahsis etmeyebilir. Bazı sistemler, bir sürecin aynı anda tutabileceği kaynak sayısını sınırlayabilir.

Soru

- Bir sürecin tüm kaynakları tek seferde değil, aşamalı olarak talep etmesine izin veren, böylece kilitlenme olasılığını azaltan hangi tekniktir?

- A) Kaynak tahsis grafiği algoritması
- B) Banker algoritması
- C) Öncelik mirası protokolü
- D) Aşamalı kaynak tahsisi

Cevap

- Cevap: D)

- Aşamalı kaynak tahsisi, bir sürecin tüm gereksinimlerini tek seferde talep etmek yerine aşamalı olarak talep etmesine izin veren bir tekniktir. Bu yaklaşım, sürecin sadece ihtiyaç duyduğu kaynakları belirli bir sırayla talep etmesine ve kullanmasına olanak tanır. Bu, diğer süreçlerin kaynakları serbest bırakmasını beklerken diğer kaynakları kullanabilmesini sağlar.

Soru

- Banker algoritmasında, güvenlik (safety) algoritmasının amacı nedir?

- A) Kilitlenmeleri tespit etmek
- B) Kaynakları en uygun şekilde tahsis etmek
- C) Kaynak tahsisinin güvenli bir sırası olduğundan emin olmak
- D) Kilitlenmelerden kurtulmak

Cevap

- Cevap: C)

- Güvenlik algoritması, sistemdeki mevcut kaynak durumunu değerlendirerek, herhangi bir kaynak talebini karşılamak için güvenli bir sıra olup olmadığını kontrol eder. Güvenli bir sıra, tüm süreçlerin kaynakları ihtiyaçlarını karşılayabileceği bir sıradır ve kilitlenme durumunun oluşmasını önler. Bu nedenle, güvenlik algoritması, kaynak tahsisinin güvenli ve adil bir şekilde yapılmasını sağlamak için önemlidir.

Soru

- Bir kaynak tahsis grafiğindeki döngü (cycle) neyi temsil eder?

- A) Potansiyel bir kilitlenme durumunu
- B) Kaynak tahsis geçmişini
- C) Süreçlerin önceliğini
- D) Bir kaynağın örnek nesne sayısını

Cevap

- Cevap: A)

- Kaynak tahsis grafiğinde bir döngü varsa, bu döngü bir kaynağın döngü içindeki süreçler arasında dolaşarak birbirlerinin kaynaklarını beklemelerine neden olur. Bu durum, her bir sürecin kaynağı beklerken diğer sürecin elindeki kaynağı serbest bırakmasını beklemesiyle sonuçlanabilir, bu da kilitlenme durumunu oluşturur. Dolayısıyla, kaynak tahsis grafiğindeki döngüler potansiyel kilitlenme durumlarını gösterir ve sistemde kilitlenmenin oluşma riskini belirtir.

Soru

- Kaynak tahsis grafiğindeki döngüleri tespit etmek için hangi teknik kullanılır?

- A) Derinlik-öncelikli arama (Depth-first search)
- B) Genişlik-öncelikli arama (Breadth-first search)
- C) Topolojik sıralama (Topological sorting)
- D) Dijkstra algoritması (Dijkstra's algorithm)

Cevap

- Cevap: A)

- Kaynak tahsis grafiğindeki döngüleri tespit etmek için yaygın olarak kullanılan teknik, derinlik-öncelikli arama (DFS) algoritmasıdır. Bu algoritma, bir çizge üzerinde dolaşırken bir döngü varsa tespit eder. DFS, bir başlangıç noktasından başlayarak derinlemesine bir yol izler ve bu yol boyunca tüm olası yolları keşfeder. Bir döngü bulunduğunda, bu döngü DFS sırasında zaten ziyaret edilen bir düğüme geri dönüş olduğunu gösterir.

Soru

- Kilitlenmeleri önlemek için kullanılmayan kaynak tahsis stratejisi aşağıdakilerden hangisidir?

- A) Kaynak Önceliği (Resource preemption)
- B) İyimser Kaynak Tahsisi (Optimistic resource allocation)
- C) Bekle ve Öldür Şeması (Wait-die scheme)
- D) Vazgeç ve Bekle Şeması (Wound-wait scheme)

Cevap

- Cevap: B)

- Bu yaklaşım, kaynakların sonunda muhtemelen kullanılabilir olacağını varsayar. Süreçler serbestçe kaynak talep eder ve bir talep hemen karşılanamazsa, süreç basitçe bekler veya daha sonra yeniden dener. Bu iyimser yaklaşım, kilitlenmeleri aktif olarak önlemez. Süreçlerin kaynak talep etmeye ve beklemeye devam etmesi, nihayetinde döngüsel bir bekleme durumu ve kilitlenme oluşturması mümkündür.

Soru

- Bir kaynak tahsis grafiğinde, bir süreç düğümünden bir kaynak düğümüne yönlendirilmiş bir kenar neyi temsil eder?

- A) Süreç, kaynağı tutuyor
- B) Süreç, kaynağı talep ediyor
- C) Kaynak, sürece tahsis edildi
- D) Kaynak tahsisi için kullanılabilir

Cevap

- Cevap: B)

- Kenarlar, bir işlem düğümünden bir kaynak düğümüne yönlendirildiğinde, bu işlem düğümünün o kaynağı talep ettiğini gösterir.

Soru

- Kaynak tahsis çizgesi hakkında aşağıdakilerden hangisi doğrudur?

- A) Çizgede döngü olması kilitlenmeyi gösterir (deadlock)
- B) Döngüsüz çizgeler kaynak tahsisi senaryolarını gösteremez
- C) Çizgede sadece süreçler düğüm olarak temsil edilir
- D) Kaynak düğümleri her zaman süreç düğümlerine yönlendirilmiş kenarlara sahiptir

Cevap

- Cevap: A)

- Kaynak tahsis çizgesinde bir döngü, kilitlenme durumunu gösterir. Kilitlenme, bir veya daha fazla sürecin birbirlerinden aldıkları kaynakları beklemesi ve bu kaynakları serbest bırakmaları gerektiği durumu ifade eder. Bir döngü, bir sürecin bir kaynağı tuttuğu ve diğer süreçlerin aynı döngüdeki kaynakları elde etmek için beklediği durumu simgeler. Dolayısıyla, bu döngüyü çözmek mümkün olmadığında, süreçler kilitlenir ve ilerleyemezler.

Soru

- Bir kaynak tahsis çizgesinde, bir kaynak düğümünden bir süreç düğümüne bir yönlü kenar neyi temsil eder?

- A) Kaynak, tahsise hazır
- B) Kaynak, süreç tarafından serbest bırakılıyor
- C) Süreç, kaynağı elinde tutuyor
- D) Kaynak, süreç tarafından talep ediliyor

Cevap

- Cevap: c)

- Kaynak düğümü ilgili kaynağı, süreç düğümü ilgili süreci temsil eder. Bu yönlü kenar, süreç düğümünden kaynak düğümüne doğru olduğundan, süreç tarafından belirtilen kaynağın elinde tutulduğunu gösterir.

Soru

- Kaynak tahsis çizgesinde, bekleme (wait-for) kenarları nasıl temsil edilir?

- A) Süreçlerden diğer süreçlere doğru yönlü kenarlar
- B) Kaynaklardan süreçlere doğru yönlü kenarlar
- C) Süreçlerden kaynaklara doğru yönlü kenarlar
- D) Süreçler arasında yönsüz kenarlar

Cevap

- Cevap: A)

- Bekleme kenarları, bir süreç düğümünden başka bir süreç düğümüne yönlendirilmiş yönlü kenarlardır. Bu, bir süreç tarafından bir kaynağı beklerken diğer bir sürecin o kaynağı elinde tuttuğunu ifade eder.

Soru

- İşletim sistemlerinde kaynak tahsis çizgesinin temel amacı nedir?

- A) Fiziksel belleğin tahsisini görselleştirmek için
- B) CPU kaynaklarının tahsisini temsil etmek için
- C) Potansiyel kilitlenmeleri tespit etmek ve analiz etmek için
- D) G/Ç işlemlerinin zamanlamasını optimize etmek için

Cevap

- Cevap: C)

- Kaynak tahsis çizgesi, işletim sistemlerinde potansiyel kilitlenme durumlarını tespit etmek ve analiz etmek için kullanılır. Süreçlerin kaynakları kullanma ve talep etme etkileşimlerini görselleştirmeye olanak tanır. Bu, kaynakları kimin kullandığı, kimin beklediği ve hangi kaynakların serbest olduğu gibi bilgileri sağlayarak kilitlenme durumlarını belirlemede ve çözümlemede yardımcı olur.

Soru

- Bir bekleme (wait-for) çizge algoritması, kaynak talepleri ve serbest bırakmalarını nasıl ele alır?

- A) Sistem durumunun anlık görüntüsünü alır
- B) Çizgeyi dinamik olarak günceller
- C) Süreçlerden açık bildirim gerektirir
- D) Kaynak taleplerini ve serbest bırakmaları görmezden gelir

Cevap

- Cevap: B)

- Bekleme çizgesi algoritması, kaynak talepleri ve serbest bırakmalarını dinamik olarak günceller. Bu, süreçlerin kaynakları talep ettiğinde veya serbest bıraktığında, çizgede ilgili değişikliklerin yapılmasını içerir. Bu güncellemeler, kilitlenme durumlarını tespit etmek ve çözmek için çizgedeki ilişkileri doğru şekilde yansıtmaya yardımcı olur.

Soru

- Birden fazla türde kaynak için kilitlenme tespit algoritmasında, kaynakların tahsis ve talep durumunu temsil etmek için hangi veri yapısı kullanılır?

- A) Bekleme çizgesi
- B) Kaynak tahsis çizgesi
- C) Öncelik kuyruğu
- D) İkili ağaç

Cevap

- Cevap: B)

- Kaynak tahsis çizgesi, birden fazla türde kaynak için kilitlenme tespit algoritmalarında yaygın kullanılan bir veri yapısıdır. Bu çizge, sistemdeki süreçlerin hangi kaynakları talep ettiğini ve hangi kaynakları elinde tuttuğunu temsil eder. Bekleme çizgesi (Wait-for graph), bir sürecin diğer bir süreci beklediği durumları gösterirken, kaynak tahsis çizgesi (Resource allocation graph), kaynakların tahsis ve talep durumlarını temsil eder.

Soru

- Birden fazla türde kaynak için Banker algoritmasında, "güvenli sıra" terimi neyi ifade eder?

- A) Güvenle sonlandırılabilecek süreç dizisi
- B) Güvenle tahsis edilebilecek kaynak talebi dizisi
- C) Kilitlenmeye neden olmadan güvenle yürütülebilecek süreç dizisi
- D) Kilitlenmeleri önleyebilecek kaynak serbest bırakma dizisi

Cevap

- Cevap: C)

- "Güvenli sıra", Banker algoritması içinde kilitlenme oluşturmadan güvenle işletilebilecek bir süreç dizisini ifade eder. Bu sıra, sistemdeki mevcut kaynak durumuna ve her bir sürecin talep ettiği ve elinde bulunduğu kaynaklara bağlı olarak belirlenir. Banker algoritması, sistemde bir güvenli sıra varsa, bu sıranın işletilmesini güvence altına alır ve kilitlenme olasılığını önler.

Soru

- Banker algoritmasında bir kaynak talebinin kabul edilmesi için hangi koşulların sağlanması gerekmektedir?

- A) Talep karşılandıktan sonra sistem güvenli bir durumda olmalıdır
- B) Talep eden süreç en yüksek önceliğe sahip olmalıdır
- C) Talep eden süreç en uzun süre bekleyen olmalıdır
- D) Talep eden süreç zaten en fazla kaynağa sahip olmalıdır

Cevap

- Cevap: A)

- Banker algoritmasında bir kaynak talebinin kabul edilmesi için, talebin karşılandıktan sonra sistemde güvenli durumun korunması gerekmektedir. Yani, talep edilen kaynağın tahsis edilmesi durumunda kilitlenme olasılığı olmamalıdır. Bu, sistemdeki mevcut kaynak durumuna ve her bir sürecin talep ettiği ve elinde bulunduğu kaynaklara bağlı olarak belirlenir. Eğer talep edilen kaynak karşılandıktan sonra sistem güvenli bir durumdaysa, talep kabul edilir.

Soru

- Banker algoritmasında, "maksimum" matrisinin önemi nedir?

- A) Her sürecin talep edebileceği maksimum kaynakları temsil eder
- B) Sistemdeki toplam mevcut kaynakları temsil eder
- C) Her sürece şu anda tahsis edilen kaynakları temsil eder
- D) Şu anda tahsis edilebilecek kaynakları temsil eder

Cevap

- Cevap: A)

- "Maksimum" matrisi, Banker algoritmasında her bir sürecin talep edebileceği maksimum kaynak miktarlarını temsil eder. Bu matris, sistemin güvenli bir şekilde işlemesi ve kilitlenmelerin önlenmesi için kullanılır. Her bir satır, bir süreci temsil eder ve her bir sütun, belirli bir türdeki bir kaynağı temsil eder. Bir hücredeki değer, ilgili işlemin o kaynağı talep edebileceği maksimum miktarı belirtir.

Soru

- Banker algoritmasının "gereksinim" (need) matrisinin amacı nedir?

- A) Her sürece şu anda tahsis edilen kaynakları temsil eder
- B) Şu anda tahsis edilebilecek kaynakları temsil eder
- C) Her sürecin maksimum kaynakları ile şu anda tahsis edilen kaynaklar arasındaki farkı temsil eder
- D) Sistemdeki toplam mevcut kaynakları temsil eder

Cevap

- Cevap: C)

- "Gereksinim" matrisi, Banker algoritmasında her bir sürecin maksimum kaynak talepleri ile şu anda tahsis edilen kaynaklar arasındaki farkı temsil eder. Bu matris, her bir sürecin ihtiyaç duyduğu kaynak miktarını gösterir. Her bir satır, bir süreci temsil eder ve her bir sütun, belirli bir türdeki bir kaynağı temsil eder. Bir hücredeki değer, ilgili sürecin o kaynağı talep etme miktarıdır.

Soru

- Tek tip kaynaklar için kilitlenme tespit algoritmalarında kaynakların tahsis ve talep durumunu temsil etmek için genellikle hangi veri yapısı kullanılır?

- A) Bekleme (wait-for) çizgesi
- B) Kaynak tahsis çizgesi
- C) Öncelik kuyruğu
- D) İkili ağaç

Cevap

- Cevap: A)

- Tek tip kaynaklar için kilitlenme tespit algoritmalarında, kaynakların tahsis ve talep durumunu temsil etmek için bekleme çizgesi kullanılır. Bir sürecin diğer bir süreci beklediği durumları gösterir ve bu şekilde kilitlenmeleri tespit etmek için kullanılır. Bir süreç bir kaynağı talep ettiğinde ve bu kaynak şu anda başka bir süreç tarafından kullanılıyorsa, bekleme çizgesi bu ilişkiyi temsil eder.

Soru

- Eşzamanlılık kontrolü bağlamında, "iki aşamalı kilitlenme" protokolü neyi amaçlar?

- A) İletişim kilitlenmelerinin önlenmesi
- B) Canlı kilitlenme durumlarının ortadan kaldırılması
- C) Açlık durumunun önlenmesi
- D) Süreçlerin serileştirilebilirliğinin sağlanması

Cevap

- Cevap: D)

- İki aşamalı kilitlenme protokolü, eşzamanlılık kontrolü bağlamında süreçlerin serileştirilebilirliğini sağlamayı amaçlar. Bu protokol, süreçlerin kilit alma ve serbest bırakma aşamalarını iki parçaya böler. İlk aşamada, süreç tüm gerekli kilitleri alır; ikinci aşamada ise, süreç tüm kilitleri bırakır. Bu şekilde, herhangi bir süreç, diğer süreçlerle uygun bir şekilde sıralanarak, eşzamanlılık sırasında doğru sonuçların elde edilmesini sağlar.

Soru

- İki aşamalı kilitlenme protokolünü diğer kontrol mekanizmalarından ayıran özellik nedir?

- A) Süreçlerin tamamlanmadan önce kilitleri serbest bırakmasına izin verir
- B) Tüm kilitlerin aynı anda alınmasını gerektirir
- C) Kilitlerin iki ayrı aşamada alınmasına ve serbest bırakılmasına izin verir
- D) Süreçleri zaman damgalarına göre önceliklendirir

Cevap

- Cevap: C)

- İki aşamalı kilitlenme protokolü, diğer eşzamanlılık kontrol mekanizmalarından farklılaştıran, kilitlerin iki ayrı aşamada alınmasına ve serbest bırakılmasına izin vermesidir. İlk aşamada tüm gerekli kilitler alınır ve süreç başlar, ikinci aşamada ise tüm kilitler serbest bırakılır ve süreç tamamlanır. Bu, süreçlerin kilitlenme ve serbest bırakma işlemlerini net bir şekilde ayırarak, süreçlerin uygun bir şekilde sıralanmasını ve eşzamanlılığı sağlar.

Soru

- Canlı kilidi (livelock) kilitlenmeden (deadlock) ayıran özellik nedir?

- A) Süreçlerin sürekli durumlarını değiştirmelerini ancak ilerleme kaydetmemelerini içerir
- B) Süreçlerin kaynaklar için belirsiz süre beklemesini karakterize eder
- C) Süreçlerin çakışan kaynak talepleri nedeniyle ilerleyememesi durumunda ortaya çıkar
- D) Süreçler arasında döngüsel bir bağımlılık zincirinden kaynaklanır

Cevap

- Cevap: A)

- Livelock (canlı kilitlenme) ile kilitlenme (deadlock) arasındaki temel fark, davranış açısından ortaya çıkar. Livelock durumunda, süreçler sürekli olarak durumlarını değiştirirler ancak ilerleme kaydetmezler. Bu durumda, süreçler genellikle birbirlerine yanıt verirler ve kaynakları serbest bırakmaya çalışırlar, ancak hiçbir ilerleme sağlanmaz.

Soru

- Canlı kilitlenme durumlarını hafifletmek için yaygın bir strateji nedir?

- A) Sistemin zaman aşımı eşiklerini (timeout thresholds) artırmak
- B) Çizelgelemeye (scheduling) rastgelelik (randomness) katmak
- C) Kaynak tahsisi için geri çekilme (backoff) mekanizmaları kullanmak
- D) Etkilenen süreçleri yeniden başlatmak

Cevap

- Cevap: C)

- Canlı kilitlenme durumlarını hafifletmek için yaygın bir strateji, kaynak edinimi için geri çekilme mekanizmalarının kullanılmasıdır. Geri çekilme mekanizmaları, bir sürecin kaynağı alamadığı durumda rastgele bir süre beklemesini ve ardından tekrar denemesini sağlar. Bu, süreçlerin kaynakları elde etme girişimlerini sıraya koymasını ve tekrarlanan kaynak taleplerinin aynı anda gerçekleşmesini önler.

Soru

- Kaynak tahsisi bağlamında, "açlık" (starvation) terimi neyi ifade eder?

- A) Süreçlerin çakışan kaynak talepleri nedeniyle ilerleyememesi
- B) Süreçlerin sürekli durum değiştirmeleri ancak ilerleme kaydetmemeleri
- C) Süreçlerin kaynaklar için belirsiz süre beklemesi
- D) Süreçlerin ihtiyaç duydukları kaynaklara erişiminin engellenmesi

Cevap

- Cevap: C)

- "Açlık" terimi, kaynak tahsisi bağlamında süreçlerin kaynaklar için belirsiz bir süre beklediği durumu ifade eder. Bir süreç, gerekli kaynaklara erişememekten dolayı sürekli olarak bekler ve ilerleyemez. Bu durumda, süreç kaynakları alamadığı için çalışmaya devam edemez ve sürekli olarak beklemek zorunda kalır. Açlık durumu genellikle adil olmayan kaynak tahsis politikaları veya önceliklendirme eksikliği gibi nedenlerden kaynaklanır. Bu durumda, süreçler bir türlü çalışma fırsatı bulamazlar ve sistem kaynakların etkili bir şekilde tahsis edilmesini sağlayamaz.

Soru

- Sistemde açlık durumunun potansiyel sonucu nedir?

- A) Sık sık kilitlenmelerin meydana gelmesi
- B) Süreçlerin aşırı CPU döngüleri tüketmesi
- C) Azalan sistem verimliliği
- D) Canlı kilitlenme durumlarının artması

Cevap

- Cevap: C)

- Açlık durumunun potansiyel bir sonucu, sistem verimliliğinin azalmasıdır. Açlık durumu, süreçlerin bekleyerek kaynaklara erişemediği bir durumdur. Bu durumda, belirli süreçler belirli kaynakları elde edemeyecekleri için süreç kaynakları kullanamaz ve bu da sistem verimliliğini azaltır. Özellikle kritik kaynaklara erişim gerektiren süreçler açlık durumu yaşarsa, sistemde genel bir verimlilik düşüşü yaşanır.

Soru

- Çizelgeleme algoritmalarında açlığı (starvation) ele almak için genellikle hangi mekanizma kullanılır?

- A) Yaşlandırma (Aging)
- B) Kaynakları elinden alma (Preemption)
- C) Round-robin (Round-robin scheduling)
- D) Öncelik tersine çevirme (Priority inversion)

Cevap

- Cevap: A)

- Açlık durumunu ele almak için çizelgeleme algoritmalarında yaygın kullanılan mekanizma, "yaşlandırma" (aging) olarak adlandırılır. Bu mekanizma, süreçlerin uzun süre beklemesi durumunda önceliklerinin artırılmasını sağlar. Bu sayede, uzun süre bekleyen süreçler daha yüksek önceliğe sahip olurlar ve sistem tarafından öncelikli olarak işlenirler. Bu, süreçlerin açlık durumundan kurtulmasını ve sistemde adil bir kaynak dağıtımı sağlanmasını amaçlar.

Soru

- Dağıtık bir sistemde canlı kilitlenme durumuna yol açacak en olası senaryo hangisidir?

- A) İki Sürecin birbirlerinin mesajlarına sürekli yanıt vermesi ve ilerleme kaydetmemesi
- B) Bir sürecin uzaktaki bir sunucudan süresiz yanıt beklemesi
- C) İki sürecin kaynakları elinde tutması ve birbirlerinin elindeki kaynakları beklemesi
- D) Bir sürecin, başka bir süreç tarafından tutulan kilit için süresiz beklemesi

Cevap

- Cevap: A)

- Dağıtık bir sistemde canlı kilitlenme durumuna yol açacak en olası senaryo, iki sürecin birbirlerinin mesajlarına sürekli yanıt vermesi ve ilerleme kaydetmemesidir. Bu durumda, her iki süreç de birbirinin mesajlarına yanıt verirken, süreçlerin ilerlemesi engellenir ve her iki süreç de birbirine karşılık verir. Sonuç olarak, süreçler birbirlerinin mesajlarına yanıt vermeye devam eder ancak herhangi bir ilerleme kaydedilmez.

SON