Bölüm 17: Soru Cevap

İşletim Sistemleri

Soru

- Sanal belleğin temel amacı nedir?

- A) Fiziksel belleğin boyutunu artırmak
- B) Bellek paylaşımı için bir mekanizma sağlamak
- C) Süreçlerin fiziksel olarak mevcut olan bellekten daha fazla belleğe erişmesine izin vermek
- D) Bellek erişim süresini hızlandırmak

Cevap

- Cevap: C

- Sanal bellek, fiziksel belleği genişletmek için kullanılan bir mekanizmadır. Bir süreç, ihtiyaç duyduğu verilere fiziksel bellekte erişemiyorsa, işletim sistemi bu verileri sanal bellekten getirir. Bu sayede, süreç daha fazla belleğe erişebilir gibi görünür. İşletim sisteminin fiziksel belleği verimli bir şekilde yönetmesine ve süreçlerin gereksinimlerini karşılamasına olanak tanır.

Soru

- Fiziksel belleği sabit boyutlu bloklara bölen ve her bloğu gerektiğinde bir sürece atayan bellek yönetimi tekniği hangisidir?

- A) Sayfalama (*paging*)
- B) Kesimleme (*segmentation*)
- C) Takas (*swapping*)
- D) Parçalanma (*fragmentation*)

Cevap

- Cevap: A

- Sayfalama, fiziksel belleği sabit boyutlu bloklara böler ve her bloğu bir sürece atar. Bu teknikte, süreçler sabit boyutlu sayfalara bölünür ve her sayfa fiziksel bellek bloklarına atılır. Sayfalama, bellek yönetimi için yaygın olarak kullanılan bir tekniktir ve süreçlere gerektiği kadar bellek sağlamak için esneklik sunar. Bu sayede, bellek kullanımı optimize edilir ve süreçler arasında bellek paylaşımı daha etkin hale gelir.

Soru

- Bitişik (*contiguous*) bellek tahsisi kullanmanın başlıca dezavantajı nedir?

- A) Harici parçalanmaya neden olur
- B) Karmaşık donanım desteği gerektirir
- C) Eşzamanlı olarak çalışabilecek süreç sayısını sınırlar
- D) Sanal belleği desteklemez

Cevap

- Cevap: A

- Bitişik bellek tahsisi, belleği kesintisiz bloklara ayırır ve her bloğu bir sürece atar. Süreçler belleğe yerleştirildiğinde, bellekteki boşluklar parçalanır ve kullanılamaz hale gelir. Bu duruma *harici parçalanma* denir. Harici parçalanma, bellek kullanımını kısıtlar ve bellek verimliliğini azaltır. Ayrıca, bellek bloklarının arasında boşluklar olduğu için, büyük süreçleri belleğe yerleştirmek zorlaşır.

Soru

- Hangi bellek tahsis algoritması, bir isteği karşılamak için yeterli boyutta en küçük bellek bloğunu ayırarak israfı en aza indirmeyi amaçlar?

- A) En İyi Uyan (*best fit*)
- B) İlk Uyan (*first fit*)
- C) En Kötü Uyan (*worst fit*)
- D) Sonraki Uyan (*next fit*)

Cevap

- Cevap: A

- En İyi Uyan (*Best Fit*) bellek tahsis algoritması, bellek isteğini karşılamak için mevcut olan en küçük bellek bloğunu bulur. Bu sayede, boşa harcanan bellek miktarı azalır. Mevcut boş bellek bloklarını tarayarak çalışır ve istek için en küçük uygun bloğu seçer. Bu yöntemde, bellek boşa harcanmadan etkili bir şekilde kullanılır. Ancak, algoritma maliyetlidir.

Soru

- Sanal bellek yönetiminde sayfa tablosunun (page table) amacı nedir?

- A) Sanal adresleri fiziksel adreslere eşlemek
- B) Tüm sayfaların içeriğini ana bellekte saklamak
- C) Sayfaların süreçlere tahsisini yönetmek
- D) Bellek sayfaları için erişim izinlerini kontrol etmek

Cevap

- Cevap: A

- Sayfa tablosu, sanal bellek yönetiminde kullanılan veri yapısıdır. Sanal bellek kullanıldığında, sürecin sanal adresleri fiziksel bellek adreslerine eşlenmelidir. Bu eşleme işlemi sayfa tablosu tarafından gerçekleştirilir. Sayfa tablosu, her bir sanal bellek sayfasının karşılık geldiği fiziksel bellek bloğunu gösterir.

Soru

- Tüm mevcut süreçleri tutmak için fiziksel belleğin yeterli olmadığı durumda hangi mekanizma kullanılır?

- A) Boşa çabalama (*thrashing*)
- B) Sayfa yer değiştirme (*page replacement*)
- C) Takas etme (*swapping*)
- D) Parçalanma (*fragmentation*)

Cevap

- Cevap: C

- Takas etme (*Swapping*), fiziksel belleğin yetersiz olduğu durumlarda kullanılır. Eğer işletim sistemi tüm çalışan süreçleri tutmak için yeterli fiziksel belleğe sahip değilse, kullanılmayan süreçleri veya süreçlerin bazı bölümlerini, sabit disk gibi bir depolama alanına taşıyarak belleği boşaltabilir. Bu sayede, önemli süreçler için yer açılır ve sistem verimliliği artırılır. Takas etme, fiziksel belleğin sınırlarını aşan durumlarda sistemin dengesini korumak için önemlidir.

Soru

- Bellek yönetiminde *Dönüşüm Önbelleği* (*TLB, Translation lookaside buffer*) rolü nedir?

- A) Sık erişilen sayfaların kopyalarını saklamak
- B) Adres dönüşümünü hızlandırmak için sayfa tablosunu önbelleğe almak
- C) Sayfaların süreçlere tahsisini yönetmek
- D) Bellek sayfaları için erişim izinlerini kontrol etmek

Cevap

- Cevap: B

- TLB, sanal bellek adreslerini fiziksel bellek adreslerine çevirirken adres dönüşümünü hızlandırmak için kullanılan bir önbellek türüdür. Sayfa tablosundaki girdilerin bir kopyasını saklar. Sıkça kullanılan çevirilere hızlı bir şekilde erişmek için bu kopyalar kullanılır. Bellek erişim sürelerini azaltır ve sistem performansını artırır.

Soru

- Birden fazla sürecin bellekte bir programın tek bir kopyasını paylaşmasına izin veren bellek yönetimi tekniği hangisidir?

- A) Sayfalama (*paging*)
- B) Kesimleme (*segmentation*)
- C) Talep Üzerine Sayfalama (*demand paging*)
- D) Paylaşımlı Bellek (*shared memory*)

Cevap

- Cevap: D

- Paylaşımlı bellek, farklı süreçler arasında veri paylaşımını kolaylaştıran bir yöntemdir. Bellek kullanımını optimize eder ve kaynak kullanımını azaltır.

Soru

- Sanal bellek sistemlerinde talep üzerine sayfalamanın kullanılmasının dezavantajı nedir?

- A) Daha büyük bir sayfa tablosu gerektirir
- B) Disk G/Ç yükünü artırır
- C) Daha yüksek bellek parçalanmasına neden olur
- D) Çoklu programlamayı desteklemez

Cevap

- Cevap: B

- Talep üzerine sayfalama, bir sürecin çalışma zamanında sadece ihtiyaç duyduğu sayfaların belleğe getirilmesini sağlayarak bellek kullanımını optimize eder. Ancak, bu yöntem disk G/Ç (giriş/çıkış) işlemlerinde artışa neden olur. İhtiyaç duyulan sayfalar fiziksel bellekte bulunamadığında, diskten getirilmesi gerekir.

Soru

- Bellek Yönetim Biriminin (*MMU*) rolü aşağıdakilerden hangisi değildir?

- A) Adres çevirisi
- B) Bellek koruması
- C) Süreç çizelgeleme
- D) Sanal bellek yönetimi

Cevap

- Cevap: C

- Süreç çizelgeleme bellek yönetim biriminin görevlerinden değildir. *MMU*, adres çevirisi, bellek koruması ve sanal bellek yönetimi gibi işlevleri gerçekleştirir. Süreç çizelgeleme, işletim sistemi çekirdeğinin sorumluluğundadır.

Soru

- Aşağıdakilerden hangisi bellek yönetiminde *boşa çabalamanın* (*Thrashing*) bir özelliğidir?

- A) Yüksek disk G/Ç ve düşük CPU kullanımı ile aşırı takas işlemi
- B) Minimal iş yükü ile bellek kaynaklarının etkin kullanımı
- C) İkincil depodan ana belleğe hızlı veri transferi
- D) Parçalanmayı en aza indirerek süreçlere bellek tahsisi

Cevap

- Cevap: A

- *Thrashing*, aşırı takas işlemlerine neden olur. İşletim sistemi yararlı işler yapmak yerine bellek ve disk arasında sürekli veri alışverişi yapar. Sonuç olarak, yüksek disk G/Ç (giriş/çıkış) işlemleri gerçekleşirken, işlemci düşük bir kullanım oranına sahip olur. *Thrashing* bellek yetersizliğiyle ilgilidir ve işletim sistemi yetersiz bellek alanına rağmen birçok süreci aynı anda sürdürmeye çalışır.

Soru

- Talep üzerine sayfalama (*demand paging*) sanal bellek sistemlerinde, hangi sayfa yer değiştirme (*page replacement*) algoritması, en uzun süre erişilmeyecek olan sayfayı değiştirmeyi amaçlar?

- A) En Eski Kullanılan (*LRU*)
- B) İlk Giren İlk Çıkar (*FIFO*)
- C) Saat (*Clock*)
- D) Optimum (*Optimal*)

Cevap

- Cevap: D

- Optimal sayfa yer değiştirme algoritması, bir sayfanın gelecekte belirli bir süre boyunca erişilmeyeceğini tahmin ederek, en uygun sayfayı değiştirmeyi amaçlar. Amacı, gelecekte kullanılmayacak olan sayfaları tahmin ederek, belleği en verimli şekilde kullanmaktır. Pratikte gerçekleştirilmesi zordur çünkü gelecekteki istekleri önceden tahmin etmek imkansız.

Soru

- Bellek yönetiminde Çalışma Kümesi (*Working set*) modelinin amacı nedir?

- A) Süreçlerde bellek sızıntılarını önlemek
- B) Süreçlerin gelecekteki bellek ihtiyaçlarını tahmin etmek
- C) Thrashing'i önlemek için gereken minimum sayfa sayısını belirlemek
- D) Bir sürecin etkin çalışması için gereken bellek miktarını ölçmek

Cevap

- Cevap: D

- Çalışma Kümesi modeli, bir sürecin etkin bir şekilde çalışabilmesi için gereken bellek miktarını ölçer. Bir sürecin çalışması sırasında bellekte hangi sayfaların kullanıldığını ve süreç için ne kadar belleğe ihtiyaç duyulduğunu belirler.

Soru

- Aşağıdakilerden hangisi sayfa tablosunun bir öğesi değildir?

- A) Geçerli/Geçersiz Biti
- B) Kirli Biti
- C) Sayfa Tablosu Taban Yazmacı
- D) Çerçeve Numarası

Cevap

- Cevap: C

- Sayfa tablosu girdisinin bileşenleri, bir sayfanın fiziksel bellekteki konumu hakkında bilgi sağlar. Bu bileşenler arasında Geçerli/Geçersiz biti (*Valid/Invalid Bit*), Kirli Biti (*Dirty Bit*) ve Çerçeve Numarası (*Frame Number*) bulunur. Ancak; Sayfa tablosu taban yazmacı (*Page Table Base Register*), sayfa tablosunun yerini gösterir. Sayfa tablosunun bellekteki başlangıç adresini tutar ve işletim sistemi tarafından kullanılır.

Soru

- Bellek tahsisinde *Buddy* sisteminin amacı nedir?

- A) Bitişik boş bellek bloklarını birleştirerek parçalanmayı yönetmek
- B) Tüm süreçleri bellek ve disk arasında takas ederek talep üzerine sayfalamayı uygulamak
- C) Süreç büyüklüğüne bağlı, bellek kaynaklarını dinamik tahsis etmek
- D) Sık erişilen sayfaları belleğe önden yükleyerek sayfa hatalarını önlemek

Cevap

- Cevap: A

- *Buddy* sistemi, bitişik boş bellek bloklarını birleştirerek parçalanmayı yönetir. Bu sistemde, bellek blokları *eş* (*buddy*) olarak belirli boyutlarda gruplara ayrılır. Bir süreç için istenen bellek miktarı belirlendiğinde, uygun boyuttaki bir blok tahsis edilir. Eğer istenen boyutta bir blok yoksa, daha büyük bir blok bölünerek veya daha küçük bloklar birleştirilerek ihtiyacı karşılayacak bir blok oluşturulur. Bu şekilde, parçalanma azaltılır ve bellek kullanımı optimize edilir.

Soru

- Hangi bellek tahsis algoritması belleği değişken boyutlu bölümlere ayırır ve her bölümü sürecin bellek gereksinimlerine göre atar?

- A) Kesimleme (*segmentation*)
- B) Sayfalama (*paging*)
- C) En iyi uyan (*best fit*)
- D) Sonraki uyan (*next fit*)

Cevap

- Cevap: A

- Kesimleme, belleği değişken boyutlu bölümlere ayırır ve her bir bölümü sürecin bellek gereksinimlerine göre tahsis eder. Her süreç, ihtiyaç duyduğu bellek miktarına uygun bir bölüm alır. Bellek kullanımını optimize etmek için esneklik sağlar ve farklı süreçlerin farklı bellek ihtiyaçlarına yanıt verir.

Soru

- İşletim sistemlerinde bellek korumasının amacı nedir?

- A) Sistem kaynaklarına izinsiz erişimi önlemek
- B) Belleği birden çok süreç arasında verimli bir şekilde tahsis etmek
- C) Bellek tahsis işlemi sırasında parçalanmayı en aza indirmek
- D) Disk G/Ç işlemlerinin performansını artırmak

Cevap

- Cevap: A

- Bellek koruması, işletim sisteminin sistem kaynaklarına (bellek gibi) yetkisiz erişimi önler. Herhangi bir sürecin diğer süreçlerin bellek alanlarına müdahalesini önleyerek güvenliği sağlar. Bellek koruması, yetkilendirilmiş erişim ve izinsiz erişim arasında bir ayırım yaparak, sistemdeki verilerin bütünlüğünü ve gizliliğini korur.

Soru

- Aşağıdakilerden hangisi *Kesimleme* kullanmanın faydalarından değildir?

- A) Birden çok süreç arasında kod kesimlerinin paylaşımına olanak tanır
- B) Belleği sabit boyutlu bloklara bölerek bellek tahsisini basitleştirir
- C) Dinamik yükleme ve kütüphane paylaşımını destekler
- D) Farklı ayrıcalıklar atayarak bellek koruması sağlar

Cevap

- Cevap: B

- Belleği sabit boyutlu bloklara bölmek, Kesimleme (*Segmentation*) tekniğine özgü değildir. Kesimleme tekniği belleği mantıksal olarak ilgili parçalara böler ve her parçayı bir sürece atar. Bu sayede, birden çok süreç aynı bellek alanını paylaşabilir, dinamik yükleme ve kütüphane paylaşımını destekler, ve farklı ayrıcalıklar (*privileges*) atayarak bellek koruması sağlar.

Soru

- Çok seviyeli sayfa tablosu düzeninde, birden fazla sayfa tablosu kullanmanın amacı nedir?

- A) Sayfa tablosunun boyutunu azaltmak bellek gereksinimini en aza indirmek
- B) Sık erişilen sayfaları önbelleğe alarak adres çevirisi performansını artırmak
- C) Hiyerarşik bir yapı ile daha büyük sanal adres alanlarını desteklemek
- D) Bellek sıkıştırma ve birleştirme (*defragmentation*) işlemlerini mümkün kılmak

Cevap

- Cevap: C

- Çok seviyeli sayfa tablosu düzeni, daha büyük sanal adres alanlarını destekler. Sanal adres alanını hiyerarşik bir yapıda düzenler ve daha büyük adres alanlarını yönetmeyi kolaylaştırır. Bellek yönetimini optimize eder.

Soru

- Diğer bellek tahsis tekniklerine göre *Buddy* sistemin avantajı nedir?

- A) Sabit boyutlu bloklar halinde bellek tahsis ederek parçalanmayı en aza indirir
- B) Etkin bellek sıkıştırma ve birleştirme (*defragmentation*) işlemleri sağlar
- C) Sık erişilen sayfaları önbelleğe alarak belleğe hızlı erişim sağlar
- D) Bellek tahsisi ve serbest bırakma işlemlerini basitleştirir

Cevap

- Cevap: D

- *Buddy* sistem, bellek tahsisini yönetmek için kullanılan bir yöntemdir. Bellek blokları *eş* (*buddy*) olarak adlandırılan ve boyutları birbirine yakın bloklar halinde gruplandırılır. Bir süreç bellek tahsis etmek istediğinde, sistem bu süreç için uygun boyutta bir blok bulur veya uygun boyutta bir bloğu böler. Bellek serbest bırakma işlemi ise, artık kullanılmayan bir bloğun işaretlenmesi ve gerektiğinde bu bloğun birleştirilmesiyle gerçekleştirilir. Bellek tahsisi ve serbest bırakma işlemlerini basit ve etkili bir şekilde yönetir.

Soru

- Çok seviyeli sayfa tablosu düzeninde, *TLB* kullanım amacı nedir?

- A) Daha hızlı erişim için sıkça erişilen sayfaların kopyalarını saklamak
- B) Sayfa tablosunun boyutunu azaltmak ve bellek gereksinimini en aza indirmek
- C) Ana bellek ve ikincil depo arasında sayfa değişimi yaparak talep üzerine sayfalama desteği sağlamak
- D) Sayfalara farklı erişim izinleri atayarak bellek koruması sağlamak

Cevap

- Cevap: A

- *TLB* kullanmanın temel amacı, sık erişilen sayfaların kopyalarını saklayarak daha hızlı veri transferi sağlamaktır. TLB, sanal bellek sistemlerinde adres dönüşümünü hızlandırmak için kullanılan bir önbellek türüdür. Bu sayede, sanal bellek adresi fiziksel bellek adresine hızlı bir şekilde çevirilebilir. Eğer TLB'de gerekli bilgi bulunursa, süreç fiziksel bellek adresini TLB'den alır ve doğrudan bu adrese erişir, böylece sayfa tablosuna erişim süresi azalır ve sistem performansı artar.

Soru

- Aşağıdakilerden hangisi bellek tahsisinde *dış parçalanmaya* neden olur?

- A) Ardışık bellek tahsis düzeninde bloklarının tahsisi ve serbest bırakılması
- B) Talep üzerine sayfalamada bellek sayfalarının süreçlere tahsisi
- C) Kesimleme tabanlı bellek yönetiminde bellek kesimlerinin birden fazla süreç arasında paylaşılması
- D) Buddy stratejisinde değişken boyutlu bellek bölümlerinin süreçlere tahsisi

Cevap

- Cevap: A

- Ardışık bellek tahsis düzeninde, bellek blokları ardışık olarak tahsis edilir ve süreçler bellek blokları arasında yer değiştirir. Bellek blokları serbest bırakıldığında, bu bloklar arasında boşluklar oluşur. Bu boşluklar, büyük bir süreç için yeterli bellek bloğu bulunamaması durumunda parçalanmaya neden olabilir. Serbest bırakılan bellek blokları arasında kalan küçük boş alanlar, dış parçalanmaya yol açabilir.

Soru

- Bellek yönetiminde *Takas* (*Swapping*) uygulamanın asıl amacı nedir?

- A) Sayfa tablosu bakımının iş yükünü azaltmak
- B) Süreçlerin birbirleriyle bellek kesimlerini paylaşmasını sağlamak
- C) Bellek alanını serbest bırakmak için ana bellek ve ikincil depo arasında tüm süreci transfer etmek
- D) Sıkça erişilen sayfaları ayrı bir önbelleğe alarak bellek erişim süresini iyileştirmek

Cevap

- Cevap: C

- *Takas* (*Swapping*), tüm sürecin ana bellek ve ikincil depo (genellikle disk) arasında aktarılmasıyla bellek alanını serbest bırakmak için kullanılır. Ana bellek sınırlıdır ve aynı anda birçok süreç çalıştırıldığında, bellek yetersizliği yaşanır. Bu durumda, aktif olmayan süreçlerin bellekten çıkarılması ve ikincil depoya (disk) taşınması gerekir. Bu işlem, bellekte mevcut süreçlerin ihtiyaçlarını karşılamak için yer açar.

Soru

- Aşağıdaki bellek tahsis algoritmalarından hangisi, bir isteği karşılamak için en büyük bellek bloğunu tahsis ederek boşa harcanan belleği en aza indirmeyi amaçlar?

- A) En iyi uyan (*best fit*)
- B) İlk uyan (*first fit*)
- C) En kötü uyan (*worst fit*)
- D) Sonraki uyan (*next fit*)

Cevap

- Cevap: C

- En kötü uyan (*Worst Fit*) bellek tahsis algoritması, bir süreç için yeterince büyük olan en büyük bellek bloğunu tahsis ederek boşa harcanan belleği en aza indirmeyi amaçlar. Mevcut boş bellek blokları arasından en büyük olanını seçer ve bu bloğu sürecin bellek ihtiyacını karşılamak için kullanır. Bu şekilde, küçük boşluklar bırakılmaz ve bellek alanı daha etkin bir şekilde kullanılır. Algoritmanın dezavantajı, küçük boşluklar yerine büyük boşluklar bırakmasıdır, bu da gelecekteki tahsisler için uygun boşlukların azalmasına ve parçalanmanın artmasına neden olabilir.

Soru

- Talep üzerine sayfalamada, hangi algoritma basitliği ve düşük iş yükü ile bilinir, ancak *Belady* anomalisinden etkilenir?

- A) En Eski Kullanılan (*LRU*)
- B) İlk Giren İlk Çıkar (*FIFO*)
- C) Optimum (*Optimal*)
- D) Saat (*Clock*)

Cevap

- Cevap: B

- İlk Giren İlk Çıkar (*FIFO*) sayfa yer değiştirme algoritması, basitliği ve düşük iş yükü ile bilinir ancak Belady Anomalisinden etkilenebilir. Bellekte en uzun süredir bulunan sayfanın yerine yeni bir sayfa getirir. Basit bir kuyruk yapısı kullanılarak gerçekleştirilir; yeni gelen sayfalar kuyruğun sonuna eklenir ve yer değiştirme gerektiğinde kuyruğun başındaki sayfa bellekten çıkarılır. *Belady Anomalisi*, daha fazla sayıda bellek çerçevesi (*page frame*) kullanılmasının sistem performansını beklenmedik şekilde kötüleştirebileceğini belirtir.

Soru

- Bellek yeniden yerleştirme (*relocation*) işleminin temel amacı nedir?

- A) Fiziksel belleğin boyutunu artırmak
- B) Süreçlerin fiziksel adreslerinden bağımsız olarak bellek konumlarına erişmesine izin vermek
- C) Bellek erişim süresini hızlandırmak
- D) Bellek parçalanmasını yönetmek

Cevap

- Cevap: B

- Bellek yeniden yerleştirme, süreçlerin fiziksel adreslerinden bağımsız olarak bellek konumlarına erişmesini sağlar. Bir süreç, çalıştırıldığında fiziksel bellek içinde belirli bir konuma yerleştirilir. Ancak, fiziksel bellek, sınırlı bir kaynaktır ve süreçler arasında paylaşılmalıdır. Ayrıca, bir süreç için ayrılan fiziksel bellek miktarı, sürecin ihtiyaçlarını karşılayabilecek kadar büyük olmayabilir. Süreçleri fiziksel bellekte daha uygun bir konuma taşımak ve bu süreçlere sanal bellek alanları tahsis etmek için bellek yeniden yerleştirme işlemi kullanılır. Bu şekilde, süreçler farklı zamanlarda fiziksel bellekte farklı konumlara yerleştirilebilir.

Soru

- Aşağıdakilerden hangisi mantıksal adresleri fiziksel adreslere çevirmek için Bellek Yönetimi birimi (*MMU*) kullanımını gerektirir?

- A) Ardışık Bellek Tahsisi
- B) Sayfalama
- C) Kesimleme
- D) Parçalanma

Cevap

- Cevap: B

- Sayfalama tekniğinde, bellek fiziksel olarak bölümlere ayrılmaz; bunun yerine, bellek sanal sayfalara bölünür. *MMU*, işlemci tarafından oluşturulan mantıksal adresleri, sayfa tablolarını kullanarak fiziksel adreslere çevirir. Bu işlem, sürecin sanal bellek alanında yer aldığını varsayar ve herhangi bir sayfanın herhangi bir zamanda ana bellekte herhangi bir konuma taşınabileceği anlamına gelir.

Soru

- Talep üzerine sayfalamada, Bellek Yönetimi birimi (*MMU*) ne işe yarar?

- A) Bellek sayfalarının süreçlere tahsisini yönetir
- B) CPU tarafından oluşturulan mantıksal adresleri ana bellekteki fiziksel adreslere çevirir
- C) Bellek sayfaları için erişim izinlerini kontrol eder
- D) Daha hızlı veri transferi için sıkça erişilen sayfaları önbelleğe alır

Cevap

- Cevap: B

- MMU, sanal bellek sistemlerinde mantıksal adreslerin fiziksel bellek adreslerine çevrilmesinden sorumludur. İşlemci, programların yürütülmesi sırasında mantıksal adresler üretir. Mantıksal adresler, sayfa tabloları kullanılarak fiziksel bellek adreslerine çevrilir. MMU, bu adres dönüşüm işlemini gerçekleştirir ve işlemcinin erişmek istediği fiziksel bellek adresini belirler.

Soru

- Yer değiştirme yazmaçlarının (*relocation register*) görevi nedir?

- A) Süreçlerin çeviri yapmadan bellek konumlarına erişmesine izin verir
- B) Belleği sabit boyutta bloklara ayırarak bellek parçalanmasını önler
- C) Bellekte dinamik yükleme ve kütüphanelerin paylaşımını kolaylaştırır
- D) Süreçlerin fiziksel bellek konumlarından bağımsız yürütülmesini sağlar

Cevap

- Cevap: D

- Yer değiştirme yazmaçları, süreçlerin bellek içindeki konumlarını izlemek ve gerekirse yeniden yerleştirmek için kullanılan özel yazmaçlardır. Süreçlerin fiziksel bellek içindeki konumlarından bağımsız olarak yürütülmesini sağlar. Bu sayede, süreçlerin bellek içindeki konumları değişse bile, süreçler çalışmaya devam edebilir.

Soru

- Mantıksal adresleri taban ve limit yazmaçlarını kullanarak fiziksel adreslere çeviren hangi bellek yönetimi tekniğidir?

- A) Sayfalama
- B) Kesimleme
- C) Takas
- D) Ardışık Bellek Tahsisi

Cevap

- Cevap: D

- Ardışık Bellek Tahsisi, CPU tarafından oluşturulan mantıksal adresleri taban ve limit yazmaçlarını kullanarak fiziksel adreslere çevirir. Belleği ardışık bloklara böler ve her süreç için bir başlangıç adresi ve bir limit belirler. CPU tarafından oluşturulan mantıksal adresler, süreç için atanmış olan başlangıç adresi ve limit kullanılarak doğrudan fiziksel bellek adreslerine çevrilir. Süreçlerin fiziksel bellek içinde belirli ve sınırlı bir alanda olmasını sağlar. Basit ve hızlı bir bellek yönetimi tekniğidir. Bellek parçalanmasını önlemez ve bellek kullanımını optimize etmek için dinamik olarak büyüyüp küçülemez.

Soru

- Bellek yer değiştirmede taban ve limit yazmaçlarının kullanılma amacı nedir?

- A) Bellek sayfalarının süreçlere tahsisini yönetir
- B) Belleği sabit boyutta bloklara ayırarak bellek parçalanmasını önler
- C) Mantıksal adresleri fiziksel adreslere çevirerek bellek korumasını sağlar
- D) Daha hızlı transfer için sık erişilen sayfaları önbelleğe alır

Cevap

- Cevap: C

- Taban ve limit yazmaçlarının temel amacı, mantıksal adresleri fiziksel adreslere çevirmek ve bellek koruması sağlamaktır. Mantıksal adresler, işlemcinin ürettiği adreslerdir, ve doğrudan fiziksel belleğe işaret etmezler. İşlemcinin ürettiği mantıksal adresler, gerçek bellek içindeki fiziksel adreslere çevrilmelidir. Bu dönüşüm işlemi, taban ve limit yazmaçları kullanılarak yapılır. Limit yazmacı, sürecin bellek aralığının boyutunu belirtir. Süreç sınırlarını aşamaz, istemeden de olsa başka süreçlerin bellek bölgelerine müdahale edemez.

Soru

- Aşağıdakilerden hangisi bellek parçalanmasına yol açar?

- A) Ardışık bellek tahsisinde bellek bloklarının tahsisi ve geri alınması
- B) Talep üzerine sayfalamada süreçlere bellek sayfalarının tahsisi
- C) Kesimleme bellek yönetiminde bellek bölümlerinin birden fazla süreç arasında paylaşılması
- D) Buddy tahsis stratejisinde süreçlere değişken boyutlarda bellek bölümlerinin tahsisi

Cevap

- Cevap: A

- Ardışık bellek tahsisinde bellek bloklarının tahsisi ve serbest bırakılması, bellek parçalanmasına neden olur. Bellek blokları birbiri ardına sıralanır, bir blok serbest bırakıldığında, kullanılamaz halde küçük parçalar kalabilir. Bellek parçalanması, birçok küçük parçanın kullanılamaz hale gelmesi ve belleğin etkin kullanılamaması durumunu ifade eder.

Soru

- Bellek yönetiminde dinamik yer değiştirmenin (*dynamic relocation*) temel amacı nedir?

- A) Fiziksel belleğin boyutunu artırmak
- B) Süreçlerin fiziksel adreslerinden bağımsız olarak bellek konumlarına erişmesine izin vermek
- C) Bellek erişim süresini hızlandırmak
- D) Bellek parçalanmasını yönetmek

Cevap

- Cevap: B

- Dinamik yer değiştirme, bir sürecin fiziksel bellek adresini değiştirme işlemidir. Süreci farklı bir fiziksel bellek adresine taşımak için yapılır. Bu sayede, sürecin fiziksel bellek içindeki konumu değişse bile, süreç doğru bir şekilde çalışmaya devam eder. Süreçlerin bellek içindeki konumlarının dinamik olarak değiştirilmesi, işletim sisteminin bellek kaynaklarını daha verimli bir şekilde yönetmesine olanak tanır.

Soru

- Aşağıdaki tekniklerinden hangisi bellek adreslerini çalışma zamanında ayarlamak için dinamik yer değiştirme gerektirir?

- A) Sayfalama
- B) Kesimleme
- C) Ardışık Bellek Tahsisi
- D) Parçalanma

Cevap

- Cevap: B

- *Kesimleme* tekniği, belleği mantıksal olarak farklı bölümlere böler ve her bir bölümü bir sürece atar. Her süreç, farklı mantıksal adres aralıklarını kullanır. İşletim sistemi mantıksal adresleri, fiziksel bellek adreslerine dönüştürmek ve sürecin bellek içindeki konumunu belirlemek için dinamik yer değiştirme kullanır.

Soru

- Dinamik yer değiştirme ile bellek adreslerinin çalışma zamanında ayarlanmasının temel faydası nedir?

- A) Adres çevirisinin getirdiği ek yükü azaltır
- B) Bellek tahsisini optimize ederek bellek parçalanmasını en aza indirir
- C) Süreçlerin daha güvenli bir ortamda yürütülmesini sağlar
- D) Süreçlerin herhangi bir değişikliğe gerek kalmadan mevcut boş bir bellek konumuna yüklenmesini sağlar

Cevap

- Cevap: D

- Dinamik yer değiştirme, işletim sistemi tarafından süreç yürütülürken bellek adreslerinin dinamik olarak ayarlanmasını sağlar. Süreçler mevcut boş bir bellek konumuna değişiklik yapmadan yüklenebilir.

Soru

- İşletim sistemlerinde dinamik yüklemenin (*dynamic loading*) amacı nedir?

- A) Kaynak kodunu makine koduna derlemek
- B) Yürütülebilir kodu yalnızca ihtiyaç duyulduğunda belleğe yüklemek
- C) Derleme zamanında harici kütüphaneleri bağlamak
- D) İşletim sistemi başlangıcında süreçler için bellek tahsis etmek

Cevap

- Cevap: B

- Program çalıştırıldığında, işletim sistemi öncelikle yürütülebilir kodu diskten belleğe yükler. Özellikle büyük programlar için bu durumda gereksiz bellek kullanımı olabilir. Dinamik yükleme, yürütülebilir kodun yalnızca ihtiyaç duyulduğunda belleğe yüklenmesini sağlar. Dinamik yükleme aynı zamanda programların yürütme süresini de azaltabilir. Örneğin, bir programın tüm bileşenlerini başlangıçta yüklemek yerine, yalnızca kullanıcı bir işlevi çağırdığında ilgili kodu yüklemek daha hızlı yanıt süresi sağlar.

Soru

- Aşağıdakilerden hangisi dinamik bağlamayı (*dynamic linking*) tanımlar?

- A) Derleme zamanında harici kütüphaneleri bağlamak
- B) Program başlatıldığında kütüphaneleri belleğe yüklemek
- C) Program çalışma zamanında kütüphaneleri programa bağlamak
- D) Dinamik veri yapıları için bellek ayırmak

Cevap

- Cevap: C

- Bir program, belirli işlevleri gerçekleştirmek için harici kütüphanelere ihtiyaç duyabilir. Dinamik bağlama, harici kütüphaneleri çalışma zamanında programa bağlama işlemidir. Programın sadece gerektiği zamanlarda gerekli kütüphaneleri yüklemesini ve bellek kullanımını optimize etmesini sağlar. Programın daha küçük boyutta olmasını sağlar çünkü program ihtiyaç duyulan kütüphaneleri çalışma zamanında yükler.

Soru

- Statik bağlamaya (*static linking*) kıyasla dinamik yükleme (*dynamic loading*) ve bağlamanın (*dynamic linking*) temel faydası nedir?

- A) Bellek kullanımının azalması
- B) Programı daha hızlı yürütme
- C) Kaynak kullanımında daha büyük esneklik ve verimlilik
- D) Programın taşınabilirliğinin artması

Cevap

- Cevap: C

- Dinamik yükleme ve bağlama, bir programın harici kütüphaneleri çalışma zamanında yüklemesine ve programa bağlamasına olanak tanır. Kaynakların daha etkin kullanılmasını, programın güncel kütüphane sürümleri ile çalışmasını sağlar ve esnek bir geliştirme süreci sunar.

Soru

- Aşağıdakilerden hangisi dinamik bağlamaya kıyasla statik bağlamanın bir avantajıdır?

- A) Daha küçük yürütülebilir dosya boyutu
- B) Daha az bellek gereksinimi
- C) Daha iyi çalışma zamanı performansı
- D) Kütüphanelerin daha kolay bakımı ve güncellenmesi

Cevap

- Cevap: A

- Statik bağlama, bir programın derlenmiş sürümüne tüm harici kütüphane kodunu dahil ederek yürütülebilir dosyanın tek bir dosya içinde birleştirilmesini sağlar. Programın harici kütüphanelere olan bağımlılığını ortadan kaldırır. Bir programın herhangi bir ortamda çalışmasını sağlar, çünkü tüm gerekli kodlar yürütülebilir dosyada mevcuttur.

Soru

- Statik bağlamanın dinamik bağlamaya kıyasla başlıca dezavantajı nedir?

- A) Artan bellek kullanımı
- B) Daha uzun program başlatma süresi
- C) Harici kütüphanelere bağımlılık
- D) Kaynak tahsisinde sınırlı esneklik

Cevap

- Cevap: A

- Statik bağlama, tüm gerekli kütüphane kodunu yürütülebilir dosyaya dahil ettiği için bellek kullanımını artırabilir. Özellikle birçok programın aynı kütüphane kodunu paylaştığı durumlarda bu etki daha belirgin olabilir. Statik bağlama, herhangi bir kütüphane güncellemesinin tüm programların yeniden derlenmesini ve dağıtılmasını gerektirir.

Soru

- Bellek yönetiminde boş alan yönetiminin amacı nedir?

- A) Bellek kaynaklarını birden çok süreç arasında dağıtmak
- B) Bellekteki süreçlerin yerleşimini yönetmek ve parçalanmayı en aza indirmek
- C) Süreçlere tahsis edilmek üzere kullanılabilir bellek bloklarını takip etmek
- D) Mantıksal adresleri fiziksel adreslere çevirmek

Cevap

- Cevap: C

- Boş alan yönetimi, bellek yönetiminde kullanılabilir bellek bloklarını takip eder. Bellek tahsis işlemlerinde kullanılacak uygun bellek bloklarını bulmayı ve süreçlere tahsis etmeyi sağlar. Örneğin, işletim sistemi boş alanları bir liste benzeri veri yapısı aracılığıyla izler ve bellek tahsisi gerektiğinde uygun blokları bulur, süreçlere tahsis eder.

Soru

- Boş alan yönetiminde kullanılan en yaygın veri yapısı hangisidir?

- A) Bağlı liste
- B) Hash tablosu
- C) İkili ağaç
- D) Yığın

Cevap

- Cevap: A

- Boş alan yönetiminde, kullanılabilir bellek bloklarını izlemek için bağlı listeler kullanılır. Bağlı liste, her biri bir bellek bloğunu temsil eden düğümlerden oluşur. Her düğüm, belirli bir bellek bloğunun başlangıç adresini, boyutunu ve bağlantıları içerir. Bağlı listeler, bellek bloklarının eklenmesi, kaldırılması ve güncellenmesi işlemlerini etkin bir şekilde yönetir. Örneğin, bir süreç bellek talep ettiğinde, işletim sistemi bağlı listeden uygun bir bellek bloğu bulur ve tahsis eder. Bellek bloğu tahsis edildikten sonra bağlı listeden kaldırılır.

Soru

- Boş alan yönetiminde, iç parçalanma nedir?

- A) Tahsis edilmiş bloklar arasındaki kullanılmayan bellek
- B) Bir bellek bloğunun başındaki kullanılmayan bellek
- C) Tahsis edilmiş bir bellek bloğu içindeki kullanılmayan bellek
- D) Bir bellek bloğunun sonundaki kullanılmayan bellek

Cevap

- Cevap: C

- İç parçalanma, bir bellek bloğunun, sürecin ihtiyacından daha büyük olduğunda veya süreç boyutu tam blok boyutuna eşit olmadığında oluşur. Bu durumda, işletim sistemi, süreç için daha büyük bir bellek bloğu tahsis eder, ancak süreç bloğun tamamını kullanmaz. Blok içindeki bu kullanılmayan alan, iç parçalanma olarak adlandırılır. İç parçalanmayı azaltmak için, bellek tahsisi yapılırken işletim sistemi, süreç boyutuna mümkün olduğunca yakın bir blok tahsis etmeye çalışır.

Soru

- Hangi bellek tahsis algoritması, istenen boyuta uygun olan ilk kullanılabilir bellek bloğunu seçer?

- A) İlk uyan (*first fit*)
- B) Sonraki uyan (*next fit*)
- C) En iyi uyan (*best fit*)
- D) En kötü uyan (*worst fit*)

Cevap

- Cevap: A

- İlk Uyan (*First Fit*) bellek tahsis algoritması, bellek bloklarını tararken, istenen boyuta uygun olan ilk boş bellek bloğunu seçer ve süreç için bu bloğu tahsis eder. Bellek bloklarını tarama sırasında, işletim sistemi, ilk uygun bloğu bulduğunda arama işlemi sona erer ve süreç için bu bloğu tahsis eder. İşletim sistemi bellek tahsisini hızlı bir şekilde gerçekleştirir. İlk uygun bloğu seçmek daha sonraki tahsislerde daha kötü eşleşmelere neden olabilir, iç parçalanmaya yol açabilir.

Soru

- İlk uyan (*first fit*) bellek tahsis algoritmasının temel dezavantajı nedir?

- A) Aşırı dış parçalanmaya neden olur
- B) Her tahsis sonrası boş bellek listesini sıralamayı gerektirir
- C) Tahsis edilen bloklar arasında küçük kullanılamaz boşluklar bırakabilir
- D) Diğer algoritmalara kıyasla daha yüksek hesaplama maliyetine sahiptir

Cevap

- Cevap: C

- İlk uyan bellek tahsis algoritması, uygun boyuttaki ilk boş bellek bloğunu seçer ve süreç için bu bloğu tahsis eder. Ancak, bu yaklaşım bazen küçük, kullanılamaz boşlukların bırakılmasına neden olabilir. Bu boşluklar, işletim sistemi tarafından yeniden kullanılamaz. Bu durum, iç parçalanma olarak adlandırılır ve bellek kullanımını etkili bir şekilde azaltır.

Soru

- Bellek yönetiminde boş alan yönetimi algoritmalarının temel amacı nedir?

- A) Bellek tahsis ve serbest bırakma için gereken zamanı en aza indirmek
- B) Bellek kullanımını maksimize etmek ve parçalanmayı en aza indirmek
- C) Süreçlerin yetkisiz bellek konumlarına erişmesini engellemek
- D) Hızlı erişim için bellek bloklarını ardışık bir şekilde tahsis etmek

Cevap

- Cevap: B

- Boş alan yönetimi algoritmalarının ana hedefi, bellek kullanımını optimize etmek ve parçalanmayı en aza indirmektir. Bellek parçalanması, bellek bloklarının bölünmesi ve verimli bir şekilde kullanılamayan küçük boşluklar bırakılması durumunda oluşur.

Soru

- Hangi bellek tahsis algoritması, tahsis sonrası en küçük boş alan bırakan bellek bloğunu seçer?

- A) İlk Uyan
- B) Sonraki Uyan
- C) En İyi Uyan
- D) En Kötü Uyan

Cevap

- Cevap: C

- En iyi uyan bellek tahsis algoritması, tahsis edilen boyuttan sonra en küçük boş alanın kaldığı bellek bloğunu seçer. Kullanılabilir boş bellek blokları arasından süreç boyutuna en yakın olanı seçer. Bellek bloklarının taranması ve boyutlarına göre sıralanması gibi işlemleri gerektirir, diğer algoritmalara kıyasla daha fazla hesaplama maliyetine neden olabilir.

Soru

- Boş alan yönetiminde, dış parçalanma nedir?

- A) Tahsis edilmiş bloklar arasında kullanılmayan alan
- B) Bellek bloğunun başlangıcında kullanılmayan alan
- C) Tahsis edilen bir bellek bloğunda kullanılmayan alan
- D) Bellek bloğunun sonunda kullanılmayan alan

Cevap

- Cevap: A

- Dış parçalanma, tahsis edilmiş bellek blokları arasında kullanılmayan boş bellek blokları oluştuğunda ortaya çıkar. Bellek blokları arasında boşluklar oluşur ve bu boşluklar birleştirilerek daha büyük bir bellek bloğu elde edilemez. Toplam bellek kullanımını azaltır. Dış parçalanma, bellek tahsisi sırasında kullanılmayan küçük boşlukların birikmesiyle oluşabilir.

Soru

- Talebi karşılamak için yeterli toplam bellek alanı mevcut olsa da belleğin ardışık olmaması durumunda hangi tür parçalanma oluşur?

- A) İç parçalanma
- B) Dış parçalanma
- C) Statik parçalanma
- D) Dinamik parçalanma

Cevap

- Cevap: B

- Dış parçalanma, toplamda yeterli bellek alanı olduğunda ancak bellek blokları arasında kullanılmayan boşluklar olduğunda ortaya çıkar. Bu durumda, bellek blokları dağınık olduğu için, talep yerine getirilemez çünkü boş bellek blokları ardışık değildir.

Soru

- Bellek yönetiminde sayfa tablosunun temel işlevi nedir?

- A) Mantıksal adreslere karşılık gelen fiziksel adresleri saklamak
- B) Bellek sayfalarının süreçlere tahsisini yönetmek
- C) Bellek konumlarına yetkisiz erişimi engellemek
- D) Kütüphanelerin dinamik yüklenmesi ve bağlanmasını kolaylaştırmak

Cevap

- Cevap: A

- Sayfa tablosu, sanal bellek sistemlerinde bellek yönetimi için kullanılan bir veri yapısıdır. Mantıksal adreslerle fiziksel bellek adresleri arasındaki eşlemeyi saklar. Sanal bellek sistemlerinde, her süreç kendi sanal adres uzayını kullanır. Bu sanal adresler, gerçek fiziksel bellek adreslerine eşlenmelidir, bu da sayfa tablosunun rolüdür. Sayfa tablosu, işlemci tarafından üretilen mantıksal adresleri gerçek fiziksel bellek adreslerine çevirmek için kullanılır. Ayrıca bellek erişim haklarını kontrol etmek ve bellek korumasını uygulamak için de kullanılabilir.

Soru

- Sayfalama ile ilgili aşağıdaki ifadelerden hangisi doğrudur?

- A) Dış parçalanmayı ortadan kaldırır.
- B) Tüm süreçlerin aynı boyutta olması gerekir.
- C) İç parçalanmaya neden olabilir.
- D) Bellek tahsisi için dinamik yer değiştirme işlemine dayanır.

Cevap

- Cevap: C

- Sayfalama, bir bellek yönetimi tekniği olup, sanal bellek sistemlerinde sıkça kullanılır. Fiziksel bellek blokları sabit boyutlu parçalara bölünür ve süreçler sanal bellek sayfalarına bölünür. Her bir süreç, fiziksel bellekteki uygun sayfalara eşlenir. Sayfalama, iç parçalanmaya neden olabilir. İç parçalanma, bir süreç tarafından kullanılamayan küçük boşluklara neden olur. Dış parçalanmayı tamamen ortadan kaldırmaz; ancak, iç parçalanmayı azaltabilir. Sayfa boyutu seçimi, iç parçalanmayı en aza indirmek için önemlidir.

Soru

- Sayfa tablosu girdisinin amacı nedir?

- A) Sayfanın mantıksal adresini saklamak
- B) Sayfa çerçevesinin fiziksel adresini saklamak
- C) Sayfa boyutunu saklamak
- D) Sayfa için erişim izinlerini saklamak

Cevap

- Cevap: B

- Sayfa tablosu, her bir sayfanın fiziksel bellek içindeki konumunu belirlemek için kullanılır. Mantıksal adreslerle fiziksel bellek adresleri arasındaki eşlemeyi sağlar. Her bir sayfa tablosu girdisi, bir mantıksal sayfayı (veya sayfa numarasını) fiziksel bellekteki bir çerçeveyle eşler.

Soru

- Sayfalamanın ardışık bellek tahsisine kıyasla temel avantajı nedir?

- A) Daha az iç parçalanma
- B) Daha hızlı bellek erişim süresi
- C) Daha iyi bellek koruması
- D) Bellek tahsisinde daha büyük esneklik

Cevap

- Cevap: D

- Sayfalama (*Paging*), ardışık bellek tahsisine (*contiguous memory allocation*) kıyasla daha fazla esneklik sağlar. Ardışık bellek tahsisinde, bellek süreçler arasında ardışık bloklara bölünür. Bu, süreçlerin bellek içinde sıkışıp kalmasına neden olur. Sayfalamada ise, bellek blokları sabit boyutlu parçalara bölünür (*sayfalar*) ve süreçler bu sayfalar arasında serbestçe yer değiştirebilir. Süreçlere bellekte daha serbestçe yer bulma imkanı sağlar. Bir süreç, sadece gerektiğinde ihtiyacı olan sayfaları belleğe yükleyebilir ve gereksiz sayfaları bellekten çıkarabilir. Ayrıca, süreçler bellekte rastgele yerleştirilebilir.

Soru

- *TLB* hakkında aşağıdaki ifadelerden hangisi doğrudur?

- A) Mantıksal adreslere eşlenmiş fiziksel adresleri içerir.
- B) Ana belleğin bir parçasıdır.
- C) Sayfa tablosuna erişildikten sonra erişilir.
- D) Kesimlemeyi yönetmek için kullanılır.

Cevap

- Cevap: A

- TLB, mantıksal adreslerin fiziksel bellek adreslerine dönüştürülmesinde kullanılan hızlı erişim sağlayan önbellek türüdür. Mantıksal adresler, işlemci tarafından üretilir, doğrudan fiziksel bellek adreslerine karşılık gelmezler. İşlemci, bir sürecin belleğe erişim taleplerini işlemek için mantıksal adresleri kullanır ve bu adreslere karşılık gelen fiziksel adreslerini belirlemek için TLB'yi kullanır. TLB, mantıksal adresleri doğrudan fiziksel adreslere çevirerek işlemcinin bellek erişimini hızlandırır. Sayfa tablosuna (*page table*) her erişim fiziksel belleğe doğrudan erişim gerektirdiğinden, TLB daha hızlı bir işlemi mümkün kılar.

Soru

- *TLB*, bellek erişim performansını nasıl artırır?

- A) Sayfa tablosunun boyutunu azaltarak
- B) Daha hızlı çeviri için sık erişilen sayfa tablosu girdilerini saklayarak
- C) Bellek kesimlerini verimli bir şekilde yöneterek
- D) Ana bellek ile ikincil depo arasında bellek sayfalarını değiştirerek

Cevap

- Cevap: B

- TLB, mantıksal adreslerin fiziksel bellek adreslerine dönüştürülmesini hızlandırır. Sık erişilen sayfa tablosu girdilerini saklar. İşlemci, bir programın mantıksal adresine erişmek istediğinde, bu adresi fiziksel bellek adresine çevirmek için sayfa tablosuna başvurur. Bu işlem zaman alıcı olabilir, sayfa tablosu büyük olabilir, tamamen taranması gerekebilir. Bu durumu hafifletmek için sık erişilen sayfa tablosu girdileri, TLB içinde saklanır.

Soru

- *TLB* içinde istenen adres çevirisi bulunamazsa ne olur?

- A) CPU, çeviriyi ana bellekteki sayfa tablosundan alır.
- B) CPU, bir sayfa hatası oluşturur.
- C) CPU, belirli bir gecikmeden sonra tekrar TLB önbelleğini arar.
- D) CPU, çeviriyi başka bir CPU çekirdeğinin TLB önbelleğinden alır.

Cevap

- Cevap: A

- Her adres çevirisi TLB önbelleğinde bulunmayabilir. Bu durumda, CPU, gerekli çeviriyi ana bellekte bulunan sayfa tablosundan alır. Bu işlem, bellek erişim süresini biraz artırır, ancak sistem performansını ciddi şekilde etkilemez.

Soru

- Aşağıdakilerden hangisi sayfa hatasını tetikler?

- a. Bellekte bulunan bir sayfaya erişme
- b. Bellekte bulunmayan bir sayfaya erişme
- c. Bellekte kilitli olan bir sayfaya erişme
- d. Salt okunur olarak işaretlenmiş bir sayfaya erişme

Cevap

- Cevap: B

- Bellekte bulunmayan bir sayfaya erişmeye çalışmak, sayfa hatasına neden olur. İşletim sistemi, talep edilen sayfanın diskten belleğe yüklenmesi gerektiğini anlar. Sayfa hatası işlemi, istenen sayfanın belleğe yüklenmesiyle tamamlanır ve ardından süreç yeniden başlatılır. Sayfa hataları, sanal bellek sistemlerinde sıkça görülür.

Soru

- Bir sayfa hatası meydana geldiğinde işletim sistemi ne yapar?

- a. Süreci hemen sonlandırır
- b. Tüm süreci diske taşır
- c. İstenen sayfa için yeni bir sayfa çerçevesi ayırır
- d. İstenen sayfayı ikincil depodan alır

Cevap

- Cevap: D

- Sayfa hatası, erişilmek istenen sayfanın bellekte bulunmadığını belirtir. İşletim sistemi, sayfa hatası oluştuğunda bu hatayı ele alır ve istenen sayfayı diskten belleğe yüklemek için gerekli işlemleri gerçekleştirir. İkincil depo (genellikle sabit disk) üzerinde bulunan istenen sayfa, diskten belleğe kopyalanır. Ardından, süreç yeniden başlatılır ve bu kez istenen veri bellekte bulunur.

Soru

- Aşağıdakilerden hangisi yüksek sayfa hata oranının sonuçlarından değildir?

- a. Sistem performansının azalması
- b. Disk G/Ç'sinin artması
- c. CPU kullanımının artması
- d. Bellek kullanımının artması

Cevap

- Cevap: C

- Yüksek sayfa hata oranı, sistem performansının azalmasına, artan disk G/Ç işlemlerine ve artan bellek kullanımına neden olur. Ancak, sayfa hata oranının artması, CPU kullanımının doğrudan artmasına neden olmaz. Bunun yerine, CPU'nun iş yükünü artırabilecek ek disk G/Ç işlemleri ve bellek erişimiyle ilgili işlemler olabilir. Sayfa hata oranı genellikle sistem yükünü artırır, ancak bu, CPU kullanımının doğrudan bir sonucu değildir.

Soru

- Çok seviyeli sayfa tablosunda, en dıştaki sayfa tablosunun amacı nedir?

- a. Sanal sayfaları disk adreslerine eşlemek
- b. Sayfa tablosunun boyutunu azaltmak
- c. Sayfalar için erişim izinlerini yönetmek
- d. Sanal sayfaları bellekte fiziksel sayfalara eşlemek

Cevap

- Cevap: D

- Çok seviyeli sayfa tablosunda, en dıştaki sayfa tablosu, sanal adres uzayındaki sanal sayfaları fiziksel bellekteki fiziksel sayfalara eşler. En dıştaki sayfa tablosu, sanal adres uzayının belirli bölümlerini daha içteki sayfa tablolarına yönlendirir, böylece belirli bir sanal sayfa numarası, fiziksel bellekteki doğru sayfa numarasını bulmak için kullanılabilir.

Soru

- *FIFO* (*First In First Out*) sayfa yer değiştirme algoritmasının dezavantajı hangisidir?

- a. Karmaşık hesaplama gerektirir
- b. *Belady* anomalisine neden olur
- c. Her zaman en son kullanılan sayfayı seçer
- d. Sayfa tablosunda sık sık güncellemeler gerektirir

Cevap

- Cevap: B

- *FIFO* sayfa yer değiştirme algoritması, en eski giren sayfayı önce çıkartır. *Belady* Anomalisi durumunda daha fazla fiziksel belleğin kullanılmasına rağmen, beklenmedik şekilde daha fazla sayfa hatası alınabilir. Daha iyi bir performans elde etmek için kullanılan fiziksel bellek miktarının artmasına rağmen, daha fazla sayfa hatası alınabilir.

Soru

- Bir sistem aynı sayfa için birden fazla sayfa hatasını nasıl önler?

- a. Bellekte sayfayı kilitleyerek
- b. Baştan sona yazma (*write-through*) ön bellek politikası kullanarak
- c. TLB kullanarak
- d. Sayfayı kirli olarak işaretleyerek

Cevap

- Cevap: A

- Bir sayfa için birden fazla sayfa hatası alınmasını önlemek için sistem, sayfayı bellekte kilitler. Sayfanın bellekte tutulmasını ve erişiminin diğer süreçler tarafından değiştirilmesini engeller. Eğer sayfa bellekte kilidini kaybederse ve değiştirilirse, bu durum sayfa hatasına neden olabilir ve bu da gereksiz bellek ve giriş/çıkış işlemlerine yol açabilir.

Soru

- Ters çevrilmiş (*inverted*) sayfa tabloları kullanmanın temel avantajı nedir?

- a. Daha az bellek gereksinimi
- b. Sayfa tablosunda daha hızlı arama
- c. Daha yüksek önbellek tutarlılığı
- d. Sanal bellek parçalanması için destek

Cevap

- Cevap: A

- Ters çevrilmiş sayfa tablolarının avantajı, daha az bellek gereksinimidir. Geleneksel bir sayfa tablosunda, her süreç için bir sayfa tablosu bulunur ve bu tablolar büyük olabilir. Ancak ters çevrilmiş sayfa tablolarında, her bir sayfa girdisi genellikle bir süreç ve sayfa numarası çiftini eşler, böylece her süreç için tek bir girdiye ihtiyaç duyulur.

Soru

- En düşük sayfa hata oranını garanti eden algoritma hangisidir?

- a. İlk Giren İlk Çıkar (*FIFO*)
- b. Optimum (*Optimal*)
- c. En Eski Kullanılan (LRU)
- d. En Az Kullanılan (LFU)

Cevap

- Cevap: B

- Optimal sayfa yer değiştirme, verilen bir iş için en düşük sayfa hata oranını garanti eder. Bu algoritma, gelecekte hangi sayfaların kullanılacağını bilir. En az sayıda sayfa hatası oluşacak şekilde sayfaları yer değiştirir. Ancak, pratikte uygulanması zordur, gelecekte hangi sayfaların kullanılacağını bilmek mümkün değildir. FIFO, en eski giren sayfayı yer değiştirir ve Belady Anomalisi durumunda performansı kötüleşebilir. LRU, en son kullanılan sayfayı yer değiştirir, LFU, en az kullanılan sayfayı yer değiştirir, ancak bazı durumlarda performansları düşük olabilir.

Soru

- Hangi sayfa yer değiştirme algoritması *thrashing* ile karşılaşır?

- a. İlk Giren İlk Çıkar (*FIFO*)
- b. En Eski Kullanılan (*LRU*)
- c. Optimal
- d. Çalışma Kümesi (*Working Set*)

Cevap

- Cevap: A

- *Thrashing*, bilgisayarın işlem hızının düştüğü ve işlemci, disk ve diğer kaynaklar arasında sürekli bir rekabetin olduğu durumdur. İşlemci, yetersiz fiziksel bellek nedeniyle, sürekli olarak sayfa değişimleri yapmak zorunda kalır. FIFO algoritması, çalışma kümelerini takip etmediğinden veya yetersiz fiziksel bellek varken kullanıldığında, en eski giren sayfayı çıkarttığı için, sayfalar sıklıkla tekrar yüklenebilir.

Soru

- Hangi sayfa yer değiştirme algoritması sayfa erişimlerinin yakınlığını dikkate alır?

- a. İlk Giren İlk Çıkar (*FIFO*)
- b. Optimal
- c. En Eski Kullanılan (*LRU*)
- d. Saat (*Clock*)

Cevap

- Cevap: C

- LRU, sayfa erişimlerinin yakınlığını dikkate alır. En son kullanılan sayfanın yerine en uzun süredir erişilmemiş olan sayfanın değiştirilmesini sağlar. Bir sayfaya erişildiğinde, o sayfanın en son kullanıldığı zamana kadar geçen süre ölçülür ve kullanılan sayfa olarak işaretlenir. LRU algoritması, en son kullanılan sayfaların bellekte kalma olasılığını artırırken, nadiren kullanılan sayfaların bellekten çıkarılma olasılığını artırır. Bu, sayfa hatalarını azaltır.

Soru

- Saat sayfa yer değiştirme algoritmasında, el işaretçisi (*hand pointer*) neyi işaret eder?

- a. Değiştirilecek sonraki sayfayı
- b. En son erişilen sayfayı
- c. Geçerli zamanı
- d. Toplam sayfa hata sayısını

Cevap

- Cevap: A

- Saat (*Clock*) sayfa yer değiştirme algoritması, dairesel olarak düzenlenmiş bir sayfa çerçevesi dizisi kullanır. *Clock hand*, dizide bir noktayı temsil eder ve sayfa çerçevelerini sırayla dolaşır. Algoritma, el işaretçisinin işaret ettiği sayfa çerçevesine bakar ve eğer kullanımdaysa, bir sonraki sayfa çerçevesine geçer. Eğer sayfa çerçevesi kullanılmıyorsa, bu sayfa çerçevesi değiştirilmek üzere seçilir. Yani, el işaretçisi, bir sonraki değiştirilecek sayfa çerçevesini gösterir.

Soru

- Hangi algoritma, sık kullanılan sayfaların bellekte daha uzun süre kalmasına neden olan *aging* problemine duyarlıdır?

- a. İlk Giren İlk Çıkar (*FIFO*)
- b. En Eski Kullanılan (*LRU*)
- c. Optimal
- d. En Az Kullanılan (*LFU*)

Cevap

- Cevap: D

- LFU sayfa yer değiştirme algoritması, sayfaların erişimlerinin sıklığına göre sıralandığı ve en az sıklıkla kullanılan sayfanın bellekten çıkarıldığı bir algoritmadır. Yaşlanma (*aging*) problemine duyarlıdır. Yaşlanma problemi, bir sayfanın erişim sıklığı zamanla değiştiğinde, LFU algoritmasının sıkça kullanılan sayfaları doğru bir şekilde tespit etmekte zorlanmasına neden olur. Özellikle, bir sayfanın erişim sıklığı arttıkça, LFU bu sayfayı hala nadiren kullanılan sayfa olarak görebilir ve bellekten çıkartmayabilir.

Soru

- Basitliği nedeniyle gerçek zamanlı sistemlerde kullanılan algoritma hangisidir?

- a. *LRU* (En Eski Kullanılan)
- b. *FIFO* (İlk Giren İlk Çıkar)
- c. Optimal
- d. *LFU* (En Az Kullanılan)

Cevap

- Cevap: B

- FIFO algoritması, basitliği nedeniyle genellikle gerçek zamanlı sistemlerde tercih edilir. Uygulanması oldukça basittir çünkü sadece bir kuyruk yapısı kullanır ve yeni bir sayfa belleğe eklendiğinde, en eski giren sayfanın yerini değiştirir. Gerçek zamanlı sistemler, basit ve öngörülebilir algoritmaları tercih ederler.

Soru

- Aşağıdakilerden hangisi, sayfa erişimlerinin (*referans*) geçmişine bakarak dinamik olarak değişen bir politika uygular?

- a. Optimal
- b. Çalışma Kümesi (*Working Set*)
- c. *LRU* (En Eski Kullanılan)
- d. *LFU* (En Az Kullanılan)

Cevap

- Cevap: B

- Çalışma Kümesi (*Working Set*), sayfa referanslarının geçmişine dayalı olarak sayfa yer değiştirme politikasını dinamik olarak ayarlar. İşlemcinin çalışma kümesini (belirli bir zaman diliminde kullanılan sayfaların kümesi) izler ve bu kümedeki sayfaları bellekte tutar. İşlemcinin çalışma kümesi, zamanla değişebilir, bu nedenle çalışma kümesi algoritması, değişikliklere uyum sağlamak için sayfa yer değiştirme politikasını ayarlar.

Soru

- Saat sayfa yer değiştirme algoritmasında, işaretçi nasıl hareket eder?

- a. Saat yönünde
- b. Saat yönünün tersine
- c. Sayfa referans bitine bağlı olarak
- d. Rastgele

Cevap

- Cevap: A

- Saat sayfa yer değiştirme algoritmasında, işaretçi (*clock hand*) saat yönünde hareket eder. Sayfa çerçevelerini dairesel bir yapıda düzenler ve işaretçi de bu dairesel yapıda dolaşır. Her sayfa erişiminde, işaretçi bir sonraki sayfa çerçevesine ilerler. Bu işlem, dairesel yapıda bir tur tamamlanana kadar devam eder. Eğer bir sayfa çerçevesi kullanılmıyorsa, işaretçi o sayfa çerçevesine gelince, sayfa yer değiştirme işlemi yapılır.

Soru

- Saat sayfa yer değiştirme algoritmasında, bir sayfa hatası oluştuğunda ve mevcut sayfanın referans biti ayarlı ise ne yapılır?

- a. Sayfa değiştirilir
- b. Referans biti temizlenir
- c. El işaretçisi bir sonraki sayfaya hareket eder
- d. Sayfa bellekte tutulur

Cevap

- Cevap: D

- *Saat* sayfa yer değiştirme algoritmasında, sayfa hatası oluştuğunda ve işaretçinin gösterdiği sayfanın referans biti ayarlı ise, sayfa bellekte tutulur. Sayfanın referans biti sıfırlanır, işaretçi bir sonraki sayfayı geçer. Sayfanın bellekte kalıp kalmayacağını belirlemek için referans biti kullanılır. Referans biti, sayfanın son erişimini gösterir. Eğer sayfanın referans biti ayarlı ise, sayfa son bir döngü süresi içinde kullanılmıştır ve dolayısıyla sayfa bellekte kalır.

Soru

- Hangi sayfa yer değiştirme algoritması, en uzun süredir kullanılmayan sayfayı atma prensibine dayanır?

- a. FIFO (İlk Giren İlk Çıkar)
- b. Optimal
- c. LRU (En Eski Kullanılan)
- d. Saat (*Clock*)

Cevap

- Cevap: C

- LRU algoritması, en uzun süredir kullanılmayan sayfayı atma prensibine dayanır. Bir sayfanın son erişiminin üzerinden ne kadar fazla zaman geçmişse, bir sonraki yer değiştirme adayı olur. LRU algoritması, sayfa yer değiştirme stratejileri arasında popülerdir, genellikle etkilidir ve özellikle sayfa erişim örüntülerinin dinamik olduğu durumlarda iyi performans gösterir. Ancak, uygulama maliyeti yüksektir, her bir sayfanın son erişimini takip eder.

Soru

- *LFU* (En Az Kullanılan) sayfa yer değiştirme algoritmasında, sayfa erişim sıklığı nasıl belirlenir?

- a. Sayfa hata sayısını sayarak
- b. Sayfa erişim zamanlarını takip etmek için bir zamanlayıcı kullanarak
- c. Her sayfa için bir sayaç tutarak
- d. Sayfa referans geçmişini analiz ederek

Cevap

- Cevap: C

- *LFU* (En Az Kullanılan) sayfa yer değiştirme algoritması, sayfa erişim sıklığını belirlemek için her sayfa için bir sayaç tutar. Başlangıçta sayaç değeri 1 olarak atanır. Sayfaya her erişimde, ilgili sayfa için sayaç bir artırılır. Sayfa erişim sıklığına dayalı olarak sayfa yer değiştirme kararları verilirken, LFU algoritması, her bir sayfanın erişim sıklığını doğrudan göz önünde bulundurur.

Soru

- *Yaşlanma* (*aging*) sorununa yatkın algoritma hangisidir? (eski sayfaların istenenden daha uzun süre bellekte tutulma olasılığı)

- a. *FIFO* (İlk Giren İlk Çıkar)
- b. Optimal
- c. *LRU* (En Eski Kullanılan)
- d. *LFU* (En Az Kullanılan)

Cevap

- Cevap: D

- *LFU* (En Az Kullanılan) sayfa yer değiştirme algoritması, yaşlanma problemine yatkındır. *Yaşlanma* problemi, bir sayfanın erişim sıklığının zamanla değiştiği durumlarda ortaya çıkar ve LFU'nun bu değişen davranışa uyum sağlayamaması sonucu oluşur. LFU algoritması, en az kullanılan sayfaları bellekten çıkarmaya eğilimlidir; ancak, bir sayfanın sıklığı zamanla azalırsa, LFU bu sayfayı hala sık kullanılan bir sayfa olarak algılayabilir ve bellekten çıkarmayabilir. Sonuç olarak, eski sayfalar beklenenden daha uzun süre bellekte tutulabilir.

Soru

- Saat sayfa yer değiştirme algoritmasının temel dezavantajı nedir?

- a. Sayfa erişim örüntülerini dikkate almaz
- b. Sayfa referans bitlerinin sık sık güncellenmesini gerektirir
- c. *Belady* anomalisiyle karşılaşır
- d. Sık erişilen sayfaları gereksiz yere bellekte tutabilir

Cevap

- Cevap: A

- Saat sayfa yer değiştirme algoritması, sayfa erişim örüntülerini dikkate almaz. Bir sayfaya ne sıklıkta erişildiğini veya en son ne zaman erişildiğini göz önünde bulundurmaz. Dairesel liste yapısı kullanır ve bir referans bitine dayalı olarak sayfa yer değiştirme kararları verir. Ancak, bu kararlar, sayfanın gerçek kullanım sıklığına veya erişim zamanına dayanmaz.

Soru

- FAT32 dosya sistemi tarafından desteklenen maksimum dosya boyutu nedir?

- A) 2 GB
- B) 4 GB
- C) 16 GB
- D) 32 GB

Cevap

- Cevap: B

- FAT32 dosya sistemi, bir dosyanın maksimum boyutunu 4 GB ile sınırlar. Tek bir dosyanın boyutu 4 gigabaytı geçemez. FAT32 dosya sistemi, dosya boyutu sınırlamaları açısından FAT16'dan daha esnek olsa da, modern dosya sistemleri olan NTFS (Windows) veya exFAT (genel olarak taşınabilir aygıtlarda kullanılır) gibi diğer dosya sistemleri daha büyük dosya boyutlarına izin verir.

Soru

- Aşağıdakilerden hangisi ext4 dosya sisteminin özelliğidir?

- A) 4 TB'a kadar dosya boyutunu destekler
- B) Maksimum birim (*volume*) boyutu 16 TB'dir
- C) Dosya adları 256 karakterle sınırlıdır
- D) Günlükleme (*journaling*) desteği vardır

Cevap

- Cevap: D

- Ext4 dosya sistemi, günlükleme (*journaling*) desteği ile bilinir. Günlükleme, bir dosya sisteminde yapılan değişikliklerin bir günlük dosyasına kaydedilmesi işlemidir. Sistemde beklenmedik bir güç kesintisi veya çökme meydana geldiğinde dosya sistemi bütünlüğünü korumaya yardımcı olur. Ext4 dosya sistemi, dosya adlarını 255 karakterle sınırlar.

Soru

- Unix dosya sistemine göre, "chmod 755 dosya.txt" komutu ne yapar?

- A) Sahibine okuma, yazma ve yürütme izinleri, grup ve diğerlerine okuma ve yürütme izinleri verir
- B) Sahip, grup ve diğerlerine okuma ve yürütme izinleri verir
- C) Sahibine tüm izinleri verir, grup ve diğerlerine okuma izni verir
- D) Dosyadan tüm izinleri kaldırır

Cevap

- Cevap: A

- Bu komut dosya.txt adlı dosyaya izinler atar. *chmod* komutu, dosya veya dizinlerin izinlerini değiştirmek için kullanılırken, *755* verilecek izinleri belirtir.
- Dosya sahibi (*owner*): 7 = 4 (okuma) + 2 (yazma) + 1 (yürütme).
- Grup (*group*): 5 = 4 (okuma) + 1 (yürütme).
- Diğerleri (*others*): 5 = 4 (okuma) + 1 (yürütme).

Soru

- *MacOS* hangi dosya sistemini kullanılır?

- A) NTFS
- B) FAT32
- C) APFS
- D) exFAT

Cevap

- Cevap: C

- APFS, macOS ve diğer Apple cihazlarında (iOS, iPadOS, tvOS, watchOS) varsayılan dosya sistemidir. APFS, HFS+ (*Hierarchical File System Plus*) dosya sistemine kıyasla birçok avantaj sunar. Özellikle, performans, veri bütünlüğü, dosya şifreleme ve depolama alanı yönetimi gibi alanlarda iyileştirmeler getirir.

Soru

- Dosya sisteminde günlükleme (*journaling*) özelliğinin temel amacı nedir?

- A) Dosyaları daha verimli depolayabilmek için sıkıştırmak
- B) Dosya sistemi meta verilerinin yedeğini sağlamak
- C) Çökme veya güç kesintisi durumunda kurtarma için dosya sistemi değişikliklerinin bir günlüğünü tutmak
- D) Güvenlik amacıyla dosyaları şifrelemek

Cevap

- Cevap: C

- Dosya sisteminde günlükleme (*journaling*), değişikliklerin günlüğünü tutar. Eğer çökme veya güç kesintisi gibi beklenmedik bir durum olursa, dosya sistemi bu günlük dosyasını kullanarak son yapılan değişiklikleri geri alabilir ve dosya sisteminin bütünlüğünü koruyabilir. Bu, veri kaybını önlemeye yardımcı olur ve dosya sisteminin kurtarılabilirliğini artırır.

Soru

- Aşağıdakilerden hangisi dosya sistemlerinde kullanılan bir dosya tahsis yöntemi değildir?

- A) Ardışık tahsis (*contiguous*)
- B) Bağlı tahsis (*linked*)
- C) İndeksli tahsis (*indexed*)
- D) Bölümlenmiş tahsis (*partitioned*)

Cevap

- Cevap: D

- Bölümlenmiş tahsis (*Partitioned allocation*) dosya sistemlerinde kullanılan bir tahsis yöntemi türü değildir. Ardışık tahsis (*Contiguous allocation*), dosyaların ardışık bir şekilde yerleştirilmesini sağlar. Bağlı tahsis (*Linked allocation*), dosyaların bağlı listeler şeklinde yerleştirilmesini sağlar. İndeksli tahsis (*Indexed allocation*), dosyaların indeks tabloları aracılığıyla yerleştirilmesini sağlar.

Soru

- Aşağıdakilerden hangisi *FAT* dosya sisteminin bir kısıtlamasıdır?

- A) Dosya parçalanması için sınırlı destek
- B) Her dizinde sınırlı sayıda dosya
- C) Uzun dosya isimlerini desteklemez
- D) Dosya izinlerini desteklemez

Cevap

- Cevap: B

- Her dizinde sınırlı sayıda dosya bulunabilir. FAT dosya sistemi, FAT12, FAT16 ve FAT32 olmak üzere farklı sürümlerde mevcuttur ve her bir sürüm, dosya ve dizinler için belirli bir maksimum sayıda girdiye izin verir. Özellikle eski FAT dosya sistemleri, her dizin için sınırlı sayıda dosya ve alt dizin girdisi destekler. Büyük ve karmaşık dosya yapılarına sahip sistemlerde sınırlamalara neden olur.

Soru

- Unix dosya sisteminde, *ls -l* komutu neyi görüntüler?

- A) Dosya adlarını
- B) Dosyalar hakkında izinler, sahiplik, boyut ve değiştirilme zamanı dahil detaylı bilgiler
- C) Gizli dosyalar dahil tüm dosyaların listesini
- D) Disk kullanımının özetini

Cevap

- Cevap: B

- *ls -l* komutu, dosyalar hakkında ayrıntılı bilgileri görüntüler. Dosya adlarının yanı sıra her dosya için izinler, sahiplik, boyut ve değiştirilme zamanı gibi detaylı bilgileri gösterir. Kullanıcıya bir dizindeki dosyalar hakkında daha kapsamlı bilgi sunar ve bu bilgiler dosyaları tanımlamak, erişmek veya yönetmek için önemlidir.

Soru

- Dağıtık dosya sistemi kullanmanın temel avantajı nedir?

- A) Merkezi depolama nedeniyle artan performans
- B) Dosyaların şifrelenmesi yoluyla artan güvenlik
- C) Artan güvenilirlik ve hata toleransı
- D) Kullanıcılar için basitleştirilmiş dosya yönetimi

Cevap

- Cevap: C

- Dağıtık dosya sistemi kullanmanın temel avantajı, artan güvenilirlik ve hata toleransıdır. Bu sistemler, verileri birden fazla sunucu veya depolama cihazı arasında dağıtarak, herhangi bir sunucu veya depolama cihazının arızalanması durumunda bile veriye erişimi sürdürebilirler. Sistemlerin daha yüksek çalışma sürekliliği sağlamasına ve veri kaybını en aza indirmesine olanak tanır.

Soru

- Hangi dosya sistemi özelliği, ardışık depolama alanlarını ayırarak parçalanmayı azaltmaya yardımcı olur?

- A) Birleştirme (*Defragmentation*)
- B) Günlükleme (*Journaling*)
- C) Blok alt tahsis (*Block suballocation*)
- D) Ön tahsis (*Preallocation*)

Cevap

- Cevap: D

- Ön tahsis, dosya sistemi tarafından ardışık depolama alanlarının belirli bir dosya için önceden ayırılması işlemidir. Dosya büyüdüğünde veya değiştiğinde, dosyanın parçalanmasını önler ve ardışık depolama alanları kullanmasını sağlar. Bu sayede, performans artar ve dosya sistemi üzerindeki disk parçalanması azalır.

Soru

- *Kopya Üzerine Yazma* (*Copy on Write (COW)*) tekniğinin dezavantajı nedir?

- A) Artan disk alanı kullanımı
- B) Yavaş dosya erişim hızı
- C) Büyük dosyalara sınırlı destek
- D) Günlükleme ile uyumsuzluk

Cevap

- Cevap: A

- *Copy-on-Write* tekniği, bir dosyaya yazma işlemi yapılmadan önce kopyalanarak, orijinal dosyanın değişmeden kalmasını sağlar. Özellikle bir dosyanın birden çok kullanıcısı veya süreci olduğunda yararlıdır, çünkü her süreç orijinal dosyayı değiştirmez, değişikliklerini kendi kopyasında yapar. Ancak, COW tekniğinin bir dezavantajı artan disk alanı kullanımıdır. Her yazma işlemi sırasında yeni bir kopya oluşturulduğu için, diskte fazladan yer tutabilir. Özellikle büyük dosyalar veya sık sık değiştirilen dosyalar için, zamanla bu ekstra kopyalar önemli miktarda disk alanını işgal edebilir.

Soru

- Hangi dosya sistemi, ayrıntılı izinler için Erişim Kontrol Listelerini (*ACL*) destekler?

- A) ext4
- B) HFS+
- C) ZFS
- D) NTFS

Cevap

- Cevap: D

- *NTFS* (*New Technology File System*), ayrıntılı izinler sağlamak için Erişim Kontrol Listeleri (ACL) tutar. ACL, her dosya veya dizin için özelleştirilmiş izinler atamaya olanak tanır.

Soru

- Unix benzeri dosya sistemlerinde *inode*'un amacı nedir?

- A) Dosya verilerini saklamak
- B) Dosya izinlerini yönetmek
- C) Dosya meta verilerini saklamak
- D) Dosya erişim zamanını izlemek

Cevap

- Cevap: C

- *inode,* dosya sistemlerinde dosyaların meta verilerini saklar. Her dosya veya dizin için bir tür kimlik numarası işlevi görür ve dosyanın fiziksel konumu, boyutu, erişim izinleri, sahibi, değiştirme ve erişim zamanları gibi meta verileri içerir. Bu nedenle, inode'lar dosya sistemi tarafından dosya ile ilgili temel bilgilerin depolanmasını ve yönetilmesini sağlar. Dosyanın kendisi değil, dosyanın özellikleri hakkında bilgi içerirler.

Soru

- Aşağıdakilerden hangisi *Btrfs* dosya sisteminin özelliği değildir?

- A) Kopya üzerine yazma (*copy-on-write*)
- B) Anlık görüntü desteği (*snapshot*)
- C) Şeffaf sıkıştırma (*transparent*)
- D) Kaplam tabanlı tahsis (*extent based*)

Cevap

- Cevap: D

- *Btrfs* (*B-tree file system*), modern bir dosya sistemidir ve birçok gelişmiş özelliği içerir. Şeffaf sıkıştırma, dosyaların otomatik olarak sıkıştırılmasını sağlar, böylece disk alanı daha etkin bir şekilde kullanılabilir. Kopya üzerine yazma (*Copy-on-Write*) ve anlık görüntü desteği Btrfs dosya sisteminin temel özelliklerindendir. Bu özellikler, dosyaların güvenliği ve yönetimini sağlamak için önemlidir.

Soru

- Dosya sistemleri bağlamında *RAID* kısaltması neyi ifade eder?

- A) Bağımsız Disklerin Yedeklenmiş Dizisi (*Redundant Array of Independent Disks*)
- B) Hızlı Erişim ve Sızma Tespiti (*Rapid Access and Intrusion Detection*)
- C) Bellekteki Verilerin Güvenilir Tahsisi (*Reliable Allocation of In-memory Data*)
- D) Yazmadan Sonra Okuma Atomik Disk (*Read-after-Write Atomic Disk*)

Cevap

- Cevap: A

- *RAID* (*Redundant Array of Independent Disks*), disklerin paralel olarak çalıştığı ve veri depolama ve korumanın geliştirilmiş bir yöntemidir. Bu terim, *Redundant* (Yedekli), *Array* (Dizisi), *Independent* (Bağımsız) ve *Disks* (Diskler) kelimelerinin baş harflerinden oluşur. RAID, bir dizi farklı disk sürücüsünün bir araya getirilmesiyle oluşturulan bir depolama teknolojisidir. Bu teknoloji, veriye yedekleme, yüksek performans, artan veri erişilebilirliği veya her üçünü de bir arada sağlamak için kullanılabilir.

Soru

- Flash tabanlı depolama aygıtları, USB sürücüler ve SD kartlar için tasarlanmış olan dosya sistemi hangisidir?

- A) JFS
- B) F2FS
- C) ReiserFS
- D) XFS

Cevap

- Cevap: B

- *F2FS* (*Flash-Friendly File System*), flash tabanlı depolama aygıtları için tasarlanmıştır. Geleneksel dosya sistemleri, sabit disk sürücüleri gibi dönme tabanlı depolama ortamlarına odaklanmışken, F2FS özellikle flash bellek aygıtlarının özelliklerine uygun tasarlanmıştır. F2FS, flash bellek aygıtlarının performansını artırmak, ömrünü uzatmak ve dosya sistemi üzerindeki yazma işlemlerini daha etkin hale getirmek için optimizasyonlar içerir.

Soru

- Unix dosya sisteminde *chown* komutunun amacı nedir?

- A) Dosya sahipliğini değiştirmek
- B) Dosya izinlerini değiştirmek
- C) Dosya konumunu değiştirmek
- D) Dosya içeriğini değiştirmek

Cevap

- Cevap: A

- *chown* (*change owner*) komutu, dosyanın sahipliğini değiştirmek için kullanılır. Dosya sahibi, dosyanın tam izinlerine ve diğer kontrol düzeylerine (örneğin, grup izinleri) erişime sahiptir. Bir dosyanın sahibini başka bir kullanıcıya atamak için kullanılabilir.

Soru

- Dosya sistemlerinde *seyrek dosya* (*sparse*) özelliğinin amacı nedir?

- A) Yalnızca boş olmayan (*non-empty*) bloklar için depolama alanı ayırmak
- B) Depolama gereksinimlerini azaltmak için dosyaları sıkıştırmak
- C) Güvenlik için dosya içeriğini şifrelemek
- D) Gelecekteki dosya büyümesi için ekstra alan ayırmak

Cevap

- Cevap: A

- *Seyrek dosya* (*sparse file*) özelliği, dosya sistemlerinde yalnızca boş olmayan bloklar için depolama alanı tahsis etmenin bir yoludur. Dosyanın belirli bir boyutta olmasına rağmen, gerçek depolama alanının dosyanın içeriğindeki boşluklara göre dinamik olarak değişmesine olanak tanır. Seyrek dosyalar, genellikle büyük dosyaların saklanması veya veri yedeklemesi gibi durumlarda kullanışlıdır. Örneğin, bir seyrek dosya 1 TB boyutunda olabilir, ancak gerçekte sadece birkaç megabayt veri içerebilir.

Soru

- Dosya sistemleri bağlamında, *inode* terimi ne anlama gelmektedir?

- A) İndeks Düğümü (*index node*)
- B) İç Düğüm (*internal node*)
- C) Bilgi Düğümü (*information node*)
- D) İndeksli Düğüm (*indexed node*)

Cevap

- Cevap: A

- *I-node* (*Index Node*), dosya sistemlerinde dosyaların ve dizinlerin meta verilerini saklar. Her dosya veya dizin için bir tür kimlik numarası işlevi görür ve dosyanın fiziksel konumu, boyutu, erişim izinleri, sahibi, değiştirme ve erişim zamanları gibi meta verileri içerir.

Soru

- Aşağıdakilerden hangisi bir dosyanın birden fazla dosya adıyla erişilmesine izin verir?

- A) Katı bağlantılar (*hard links*)
- B) Yumuşak bağlantılar (*soft links*)
- C) Sembolik bağlantılar (*symbolic links*)
- D) Bağlantı noktaları (*junction points*)

Cevap

- Cevap: A

- *Katı bağlantılar* (*Hard links*), bir dosyanın birden fazla dosya adıyla (yol adı) erişilmesine izin verir. Bir dosyanın katı bağlantıları, dosya sistemine ekstra bir dosya adı ekleyerek oluşturulur. Bu dosya adları, orijinal dosya adıyla aynı dosya içeriğine ve aynı disk konumuna işaret eder. Bir dosyanın birden fazla yerde kullanılmasına izin verir ve dosyanın silinebilmesi için tüm bağlantıların kaldırılması gerekir. Aynı dosyanın farklı adlar altında farklı bağlantıları olduğundan, bir adı değiştirme veya silme işlemi, diğer adlarla ilişkili dosyaları etkilemez.

Soru

- İşletim sistemlerinde *mount* komutunun amacı nedir?

- A) Aygıt sürücülerini yüklemek
- B) Disk bölümlerini başlatmak
- C) Bir dosya sistemini dosya hiyerarşisinde bir dizine bağlamak
- D) Sanal bellek alanı oluşturmak

Cevap

- Cevap: C

- *Mount* komutu, bir dosya sistemini belirli bir dizine bağlamak için kullanılır. Belirli bir disk bölümündeki dosyaların, işletim sistemi tarafından belirli bir dizin altında erişilebilir hale getirilmesini sağlar. Bir dosya sistemiyle belirli bir dizini ilişkilendirir. Örneğin, bir USB flash sürücüsü Linux sisteme takıldığında, bu sürücünün içeriğine erişmek için *mount* komutu kullanmak gerekir. *mount* komutu, USB sürücüsündeki dosya sistemini, önceden belirlenmiş bir dizine (örneğin, "/mnt/usb") bağlayarak, bu dizindeki dosyalara erişim sağlar.

Soru

- Hangi dosya sistemi, dosya sisteminin zamanında (*point-in-time*) bir kopyasını oluşturmak için *snapshot* özelliğini destekler?

- A) XFS
- B) Btrfs
- C) UFS
- D) JFS

Cevap

- Cevap: B

- *Snapshot* kavramı, bir dosya sisteminin belirli bir anında tüm dosyalarının kopyasını oluşturmayı sağlar. Dosya sisteminin anlık bir görüntüsü oluşturulur ve bu görüntü, dosyaların o zamandaki durumunu korur. Dosyaların ilerideki değişikliklerinden etkilenmeyen bir kopya sağlar. *Btrfs* (*B-Tree File System*), snapshot özelliğini destekler. Btrfs, Linux işletim sistemlerinde yaygın olarak kullanılır. Snapshot daha sonra o zamana geri dönebilmek veya yedekleme amacıyla kullanılabilir.

Soru

- Geleneksel dosya sistemlerine kıyasla Günlüklü Yapılandırılmış Dosya Sistemi'nin (*LFS*) başlıca avantajı nedir?

- A) Daha hızlı dosya erişim hızı
- B) Veri bozulma riskinin azalması
- C) Geliştirilmiş disk alanı kullanımı
- D) Büyük dosyalar için daha iyi destek

Cevap

- Cevap: B

- Günlüklü Yapılandırılmış Dosya Sistemi (*Log Structured File System*), geleneksel dosya sistemlerine kıyasla veri güvenliğini artırır. Veri bütünlüğünü korumak ve veri kaybını en aza indirmek için günlükleme yöntemlerine dayanır. Geleneksel dosya sistemlerinde, bir dosya üzerinde yapılan değişiklikler doğrudan dosya üzerine yazılırken veri bütünlüğü tehlikeye atılır. Örneğin, bir kesinti veya sistem çökmesi sırasında verinin kaybolma, bozulma riski vardır. Ancak LFS'de, değişiklikler önce bir günlük dosyasına (*log*) yazılır ve sonra ana depolama alanına taşınır.

Soru

- Unix benzeri işletim sistemlerinde *truncate* komutunun amacı nedir?

- A) Yeni bir dosya oluşturmak
- B) Bir dosyayı kaldırmak
- C) Bir dosyanın boyutunu daraltmak veya genişletmek
- D) Dosya izinlerini değiştirmek

Cevap

- Cevap: C

- *Truncate* komutu, bir dosyanın boyutunu değiştirmek için kullanılır. Bir dosyanın boyutunu belirli bir boyuta düşürmek veya genişletmek için kullanılır. Dosya boyutu genişletildiğinde, eklenen bölümler sıfırlanır. Bir dosyanın içeriği silinmeden boyutunu daraltmak için kullanılabilir.

Soru

- Veri yedekleme ve hata toleransı sağlayan dosya sistem özelliği hangisidir?

- A) Günlükleme
- B) RAID
- C) ACL
- D) Anlık Görüntüler

Cevap

- Cevap: B

- *RAID* (*Redundant Array of Independent Disks*), dosya sistemlerinde veri yedekleme ve hataya tolerans sağlar. RAID, birden fazla disk sürücüsünü bir araya getirerek veri parçalarını depolar ve bu parçaları farklı disk sürücülerine dağıtarak veri artıklığı oluşturur. RAID'de kullanılan bir yöntem, parite bilgisi kullanarak veri artıklığı sağlamaktır. Parite, RAID dizisi içindeki veri blokları arasında bir tür kontrol verisidir. Diğer veri bloklarından elde edilir ve veri bozulması durumunda orijinal veriyi geri yüklemek için kullanılır.

Soru

- *ZFS* dosya sisteminin *Kopya üzerine yazma* (*Copy on Write*) mekanizmasının başlıca avantajı nedir?

- A) Okuma performansının artması
- B) Yazma genişlemesinin (*amplification*) azalması
- C) Düşük depolama maliyeti
- D) Veri bütünlüğünün artması

Cevap

- Cevap: D

- *ZFS* (*Zettabyte File System*), veri bütünlüğünü sağlamak ve veri kaybını önlemek için güçlü bir dosya sistemidir. Bu özelliğin temelinde Yazma üzerine kopyala (*COW*) mekanizması bulunur. Bir dosyanın üzerine yazıldığında, öncelikle ilgili blokların kopyaları alınır ve yeni veri bu kopyalar üzerine yazılır. Eski veri blokları değiştirilmez. Bu durum, veri bütünlüğünü artırır çünkü orijinal veriye dokunulmadan yeni veri yazılabilir. Bu mekanizma aynı zamanda veri güvenliği ve geri dönüşü sağlar. Yazma genişlemesinin azalması genellikle SSD'lerde kullanılan bir terimdir.

Soru

- Unix benzeri bir dosya sisteminde *inode tablosunun* başlıca amacı nedir?

- A) Dosya sistemindeki her dosya hakkında meta verileri depolamak
- B) Kurtarma için dosya sistemi değişikliklerinin günlüğünü tutmak
- C) Dosya isimlerini bunların karşılık gelen disk adreslerine eşlemek
- D) Dosya depolama için disk alanı tahsis etmek ve yönetmek

Cevap

- Cevap: A

- *Inode* tablosu, her dosya ve dizin için ayrıntılı meta verilerin (*metadata*) depolandığı bir veri yapısıdır. Bu meta veriler dosyanın adını, boyutunu, sahibini, izinlerini, oluşturma ve değiştirme tarihlerini ve dosyanın fiziksel konumunu içerir. Her dosya ve dizin için bir *inode* girdisi bulunur ve bu girdi o dosya veya dizinin tüm özelliklerini tanımlar. Dosya sistemine eklenen her dosya veya dizin için bir *inode* oluşturulur ve inode tablosuna eklenir.

Soru

- Dosya sistemlerinde *süper bloğun* (*superblock*) amacı nedir?

- A) Dosya sistemi hakkında boyut ve durum gibi meta verileri depolamak
- B) Tahsis edilebilecek boş disk bloklarının listesini tutmak
- C) Günlükleme amaçları için dosyalarda yapılan değişiklikleri izlemek
- D) Küçük dosyaların içeriğini doğrudan süper blok içinde depolamak

Cevap

- Cevap: A

- Dosya sistemi süper bloğu, dosya sisteminin tamamıyla ilgili meta verileri depolayan bir veri yapısıdır. Bu meta veriler dosya sisteminin boyutunu, kullanılan ve kullanılmayan alanları, dosya sisteminin durumunu (örneğin, bağlı veya bağlı olmayan) ve dosya sisteminin diğer özelliklerini içerir. Dosya sisteminin başlangıcında bulunur ve genellikle dosya sistemi tarafından kullanılan sabit bir konumda bulunur. Dosya sistemi başlatıldığında veya bağlandığında, işletim sistemi bu süper bloğa erişir ve dosya sistemi hakkındaki önemli bilgileri okur.

Soru

- Microsoft tarafından sunulan ve büyük miktarda veri ve akan medya (*streaming*) dosyalarını depolamak için optimize edilen dosya sistemi hangisidir?

- A) NTFS
- B) FAT32
- C) exFAT
- D) ReFS

Cevap

- Cevap: D

- *ReFS* (*Resilient File System*), Microsoft'un sunucu ve depolama çözümleri için geliştirdiği dosya sistemidir. Büyük miktarda veri ve akan medya dosyalarını depolamak için optimize edilmiştir. Geleneksel dosya sistemlerine göre daha sağlam ve dayanıklıdır. Veri bütünlüğünü korumak için özellikler içerir ve büyük veri depolama gereksinimlerini karşılamak için tasarlanmıştır. ReFS, yüksek performans ve güvenilirlik sağlamak için tasarlanmıştır, bu nedenle büyük veri merkezlerinde, bulut depolama sistemlerinde ve medya yayıncılığı gibi alanlarda tercih edilir.

Soru

- Hangi Linux dosya sistemi anlık görüntüleme (*snapshotting*), sıkıştırma ve *RAID* işlevselliğini yerleşik olarak destekler?

- A) ext2
- B) ext4
- C) Btrfs
- D) XFS

Cevap

- Cevap: C

- *Btrfs* (*B-tree File System*), Linux için geliştirilmiş bir dosya sistemidir ve *snapshotting*, sıkıştırma ve yerleşik RAID işlevselliği gibi özellikleri destekler. Dosya sistemlerinin güncel ihtiyaçlarına uyum sağlamak amacıyla tasarlanmıştır. Anlık görüntüleme özelliği, dosya sistemini belirli bir zamandaki *anlık görüntüsünü* oluşturarak dosyaların durumunu koruma ve geri dönüş yapma imkanı sağlar. Sıkıştırma, depolama alanını daha verimli bir şekilde kullanmak ve diskteki veri miktarını azaltmak için kullanılır. Yerleşik RAID işlevselliği, disk arızalarına karşı yüksek düzeyde veri koruması sağlar.

Soru

- *ReiserFS* dosya sisteminin geleneksel Linux dosya sistemleri (örneğin ext4) karşısındaki başlıca avantajı nedir?

- A) Büyük dosyalar için geliştirilmiş destek
- B) Parçalanmaya karşı direnç
- C) Büyük depolama dizileri için artırılmış ölçeklenebilirlik
- D) Rastgele G/Ç işlemleri için daha yüksek performans

Cevap

- Cevap: D

- *ReiserFS*, yüksek performanslı ve rastgele G/Ç işlemleri için optimize edilmiştir. Geleneksel Linux dosya sistemlerine göre en önemli avantajı, rastgele G/Ç işlemleri için daha yüksek performans sağlamasıdır. Dosya sistemi verilerini depolamak için daha verimli bir B-ağacı (*B-tree*) kullanır. Bu yapının bir sonucu olarak, büyük dosyaları işlemek ve rastgele G/Ç işlemlerini daha hızlı gerçekleştirmek için uygun hale gelir.

Soru

- Hangi Linux dosya sisteminde veri ve meta verilerin veri bütünlüğünü sağlamak için yerleşik *sağlama toplamı* (*checksum*) bulunur?

- A) ext4
- B) XFS
- C) Btrfs
- D) JFS

Cevap

- Cevap: C

- *Btrfs* (*B-tree File System*), Linux için geliştirilmiş modern bir dosya sistemidir ve veri bütünlüğünü sağlamak için yerleşik sağlama toplamı (*checksums*) kullanır. Sağlama toplamı, dosya sistemi içindeki verilerin ve meta verilerin doğruluğunu kontrol etmek için kullanılan özet değerlerdir. Btrfs, her veri ve meta veri bloğu için bir değer hesaplar ve bu değerlerin depolama alanına yazılmasını sağlar. Bu sayede, dosya sistemi herhangi bir veri bütünlüğü hatası algıladığında, bu hatayı otomatik olarak düzeltebilir veya etkilenen veriyi doğru bir kopyadan geri yükleyebilir.

Soru

- Linux'ta, hızlı günlükleme (*journaling*) ve kurtarma yetenekleri nedeniyle genellikle kök (/) bölümleri için hangi dosya sistemi kullanılır?

- A) ext3
- B) XFS
- C) Btrfs
- D) ReiserFS

Cevap

- Cevap: A

- ext3, özellikle kök (/) bölümleri için tercih edilir. Hızlı günlükleme (*journaling*) ve kurtarma yetenekleri sağlar. Günlükleme, dosya sistemi üzerindeki değişikliklerin bir günlük dosyasına kaydedilmesini içerir. Beklenmedik sistem kapanmaları veya çökme durumlarında dosya sistemi bütünlüğü korunur ve veri kaybı önlenir. Kök (/) bölümü, işletim sisteminin temel dosya sistemi ve sistem dosyalarının bulunduğu bölümdür. Bu nedenle, kök bölümü için güvenilir ve hızlı bir dosya sistemi tercih edilir.

Soru

- *HFS+* dosya sisteminin *FAT32* dosya sistemine kıyasla avantajı nedir?

- A) Büyük dosya boyutları için daha iyi destek
- B) Windows sistemleriyle geliştirilmiş uyumluluk
- C) Veri depolama için artırılmış güvenilirlik
- D) Günlükleme için yerel destek

Cevap

- Cevap: D

- *HFS+* dosya sisteminin FAT32'ye göre avantajı, günlükleme (*journaling*) için yerel destek sağlamasıdır. Dosya sisteminin bütünlüğünü korur ve veri kaybını azaltır. Bazı durumlarda, günlükleme işlemi dosya sistemi performansını azaltabilir, ancak genel olarak, küçük bir etkidir ve veri bütünlüğünün sağlanması göz önüne alındığında avantajları ağır basar. FAT32 dosya sistemi, günlükleme özelliğini desteklemez.

Soru

- *Extended File Allocation Table* (*exFAT*) dosya sisteminin temel amacı nedir?

- A) Dosya sıkıştırması için artırılmış destek
- B) Windows ve büyük depolama cihazları ile geliştirilmiş uyumluluk
- C) Veri güvenliği için yerleşik şifreleme desteği
- D) Gelişmiş veri yinelenme teknikleri

Cevap

- Cevap: B

- *Extended File Allocation Table* (*exFAT*) dosya sisteminin amacı, Windows ve büyük depolama aygıtları ile uyumluluğunun artırılmasıdır. exFAT, Windows işletim sistemine daha iyi uyumluluk sağlamak için geliştirilmiştir. FAT32 gibi, exFAT de dosya ve depolama aygıtları arasında dosya transferini kolaylaştırır. exFAT, Windows yanı sıra diğer işletim sistemleriyle de uyumludur. FAT32, tek bir dosya için 4 GB'lık boyut sınırına sahiptir. Bu boyut kısıtı, büyük medya dosyaları için uygun değildir. exFAT bu sınırları genişleterek, dosyanın daha büyük olmasına izin verir.

Soru

- Saydam sıkıştırma uzantısının (*Transparent Compression Extension*) temel özelliği nedir?

- A) Sıkıştırılmış dosyaların otomatik şifrelenmesi
- B) Dosya verilerinin gerçek zamanlı sıkıştırılması
- C) Sıkıştırılmış dosyaların anlık olarak açılmasını destekleme
- D) Sıkıştırılmış dosya yedeklemesi için bulut hizmetleriyle entegrasyon

Cevap

- Cevap: B

- Saydam Sıkıştırma Uzantısı (*TCE*) dosya sistemlerinin gerçek zamanlı sıkıştırma özelliğine sahip olmasıdır. Dosyaları saklarken veya aktarırken otomatik olarak sıkıştırır. Gerçek zamanlı sıkıştırma, dosya sistemlerinin performansını etkileyebilir. TCE, dosyaları sıkıştırırken veri bütünlüğünü korur. Bu, sıkıştırılmış dosyaların açılması veya kullanılması sırasında veri kaybını önler.

Soru

- Günlük yapılı (*log structured*) dosya sistemi, depolama optimizasyonu açısından ne sunar?

- A) Disk bloklarının otomatik olarak birleştirilmesi
- B) Depolama tahsisinin sürekli izlenmesi ve ayarlanması
- C) Dosya değişikliklerinin ardışık günlüklenmesi ile yazma performansının artırılması
- D) Disk alanı kullanımını azaltmak için dosya verilerinin saydam olarak sıkıştırılması

Cevap

- Cevap: C

- *LSFS*, dosya sistemi üzerinde yapılan değişiklikleri ardışık olarak günlükler, yani değişiklikler sıralı olarak kaydedilir. Diske yazma işlemlerini optimize eder ve disk erişimini iyileştirir. LSFS, silinen dosyaların yerini geri kazanmak için disk boşaltma işlemlerini yönetir. Diskte boş alanın etkin bir şekilde kullanılmasını sağlar, performansı artırır. Dosyaların diske ardışık olarak yazılmasını sağladığından, parçalanmayı azaltır, okuma işlemlerini hızlandırır.

Soru

- Dağıtık dosya sisteminin (*Distributed File System*) birincil özelliği nedir?

- A) Veri bütünlüğünün artırılması için merkezi depolama yönetimi
- B) Depolama düğümleri arasında dosyaların saydam olarak kopyalanması
- C) Dosya düzeyinde şifreleme ve erişim kontrolü için destek
- D) Ağlar üzerinde dosya değişikliklerinin gerçek zamanlı senkronizasyonu

Cevap

- Cevap: B

- *DFS*, dosyaların birden çok depolama düğümü arasında saydam bir şekilde kopyalanmasını sağlar. Dosyaları farklı düğümler arasında otomatik olarak kopyalar ve kullanıcılar için bu işlemi saydam (iç karmaşıklığını kullanıcıya hissettirmeden) hale getirir. Dosyalar tek bir merkezi depolama noktasına yüklenmek zorunda değildir. Bir depolama düğümünde arıza olursa, dosyalara diğer düğümlerden erişilebilir. Bu sayede, veri kaybını önler ve hizmet kesintilerini azaltır. Aynı anda birden çok kullanıcı tarafından erişilen dosyalar için yükü dengeleyebilir. Yeni depolama düğümlerinin kolayca eklenmesine olanak tanır.

Soru

- Windows'ta *exe* uzantısı ile ilişkilendirilen dosya türü nedir?

- A) Metin dosyası
- B) Yürütülebilir dosya
- C) Görüntü dosyası
- D) Ses dosyası

Cevap

- Cevap: B

- *exe* uzantısı yürütülebilir dosyalarla ilişkilendirilir. Bilgisayar tarafından doğrudan çalıştırılabilir ikili (*binary*) dosyalardır. *exe* dosyası çift tıklandığında, ilgili program başlatılır. exe dosyaları, zararlı veya kötü amaçlı yazılım tarafından da kullanılabilir. Bu nedenle, exe dosyalarını açarken veya çalıştırırken dikkatli olmak gerekir, güvenilir kaynaklardan onaylanmış dosyaları çalıştırmak önemlidir.

Soru

- Unix tabanlı sistemlerde *sıkıştırılmış arşiv* dosyasını gösteren dosya uzantısı hangisidir?

- A) .zip
- B) .tar.gz
- C) .iso
- D) .rar

Cevap

- Cevap: B

- *tar.gz* uzantısı, sıkıştırılmış arşiv dosyasını temsil eder. İki farklı sıkıştırma işlemi olan *tar* ve *gzip* tarafından oluşturulan bir arşiv dosyasını belirtir. *tar* komutu, dosyaları bir araya getirir ve tek bir dosya veya dizin halinde arşiv oluşturur. Bu arşiv dosyası daha sonra *gzip* programıyla sıkıştırılır, bu da *tar.gz* uzantılı sıkıştırılmış arşiv dosyasını oluşturur.

Soru

- *mp3* uzantısıyla tanımlanan dosya türü nedir?

- A) Görüntü dosyası
- B) Video dosyası
- C) Ses dosyası
- D) Belge dosyası

Cevap

- Cevap: C

- *mp3* uzantılı dosyalar, ses içeriğini depolar. Ses verilerini sıkıştırarak depolayan bir formatı temsil eder. *MP3* formatı, ses verilerini sıkıştırırken kaliteyi koruyarak dosya boyutunu küçültür. Bu, daha küçük dosya boyutlarıyla yüksek kalitede ses kaydı sağlar ve internet üzerinden ses içeriğinin paylaşılmasını ve aktarılmasını kolaylaştırır.

Soru

- *pdf* uzantısı hangi dosya türünü temsil eder?

- A) Görüntü dosyası
- B) Ses dosyası
- C) Belge dosyası
- D) Video dosyası

Cevap

- Cevap: C

- *pdf* uzantılı dosyalar, Taşınabilir belge biçimini (*Portable Document Format*) kullanarak belgeleri depolar. Belgelerin farklı bilgisayarlar ve işletim sistemleri üzerinde tutarlı bir şekilde görüntülenmesini sağlar. *PDF* dosyaları metin, görüntü, grafik, tablo ve diğer içerikleri bir araya getirebilir. Broşürler, raporlar, kitaplar, sunumlar ve diğer birçok belge türü için kullanılır. Farklı cihazlarda ve platformlarda açılabilir ve görüntülenebilir. Belgenin orijinal biçimini korur.

Soru

- Excel'de elektronik tablo dosyaları için hangi dosya uzantısı kullanılır?

- A) .xls
- B) .docx
- C) .ppt
- D) .txt

Cevap

- Cevap: A

- *xls* uzantısı, Microsoft Excel'in eski sürümlerinde kullanılan standart elektronik tablo dosya biçimidir. Bu dosyalar, tabloları, grafikleri, formülleri ve diğer verileri içerebilir. Microsoft Excel 2007'den itibaren, *xlsx* uzantısı xml tabanlı Office Open XML (*OOXML*) biçimine geçiş yapmıştır. Daha iyi sıkıştırma, ve veri bütünlüğü sağlar.

Soru

- *avi* uzantısı hangi dosya türünü belirtir?

- A) Ses dosyası
- B) Video dosyası
- C) Görüntü dosyası
- D) Arşiv dosyası

Cevap

- Cevap: B

- *AVI* (*Audio Video Interleave*), ses ve görüntü içeriğini aynı dosya içinde birleştirir. Hem ses hem görüntü içerir. Genellikle filmler, video klipler, animasyonlar ve diğer video içeriklerini depolamak için kullanılır. AVI formatı, ses ve video kalitesini korurken dosya boyutunu yönetmek için farklı sıkıştırma yöntemlerini destekler.

Soru

- Linux'ta kabuk betik (*shell script*) dosyasının uzantısı hangisidir?

- A) .sh
- B) .exe
- C) .bat
- D) .cmd

Cevap

- Cevap: A

- Kabuk betikleri, kabuk (*shell*) tarafından yürütülen komut dizilerini içeren metin dosyalarıdır. Sistem yönetimi, otomasyon ve diğer görevler için kullanılır. *sh* uzantısı, dosyanın kabuk betiği olduğunu belirtir. Uygun izinler verildiğinde doğrudan yürütülebilir hale gelir. Çalıştırıldığında betiğin içindeki komutlar gerçekleştirilir. *sh* uzantılı dosyalar, farklı kabuklarda (örneğin, Bash, Zsh, KornShell) çalışabilir.

Soru

- *jpg* uzantısı hangi dosya türünü temsil eder?

- A) Ses dosyası
- B) Video dosyası
- C) Görüntü dosyası
- D) Belge dosyası

Cevap

- Cevap: C

- *jpg* uzantılı dosyalar, dijital görüntüleri depolar. Bu dosyalar, fotoğraflar, çizimler, grafikler ve diğer görsel içerikleri saklamak için kullanılır. *JPEG* (*Joint Photographic Experts Group*) olarak bilinen format, sıkıştırılmış bir görüntü biçimidir. Bu sıkıştırma, görüntünün dosya boyutunu küçültürken, görsel kaliteyi korumaya çalışır. Farklı renk derinliklerini ve kalite seviyelerini destekler. *JPEG* formatı, geniş bir kullanıcı tabanı tarafından desteklenir ve birçok cihaz ve platformda kullanılır.

Soru

- Engellemeli G/Ç (*Blocking I/O*) hakkında hangisi doğrudur?

- A) Birden fazla G/Ç işleminin aynı anda işlenmesine izin verir.
- B) G/Ç işlemi tamamlanana kadar çağıran işlemi bekletir.
- C) CPU, G/Ç beklerken kesintiye uğramaz.
- D) Engelleme olmayan G/Ç olarak da bilinir.

Cevap

- Cevap: B

- Engellemeli G/Ç (*Blocking I/O*), işlemcinin bir G/Ç işlemi tamamlanana kadar beklemesine neden olur. Örneğin, dosya okuma veya yazma işlemi sırasında, işlemci dosya işlemini tamamlamadan diğer işlemlere devam edemez. G/Ç işlemi sıralı olarak gerçekleşir. Bir G/Ç işlemi başlatıldığında, diğer işlemler sırayla bekler. Tek iş parçacıklı (*single-threaded*) veya tek işlemcili sistemlerde verimlilik kaybına neden olur.

Soru

- Aşağıdakilerden hangisi Doğrudan bellek erişiminin (*DMA*) özelliklerinden değildir?

- A) Veri transferi sırasında CPU üzerindeki yükü azaltır.
- B) Her veri transferi için CPU müdahalesini gerektirir.
- C) Çevre birimlerinin, CPU'yu dahil etmeden doğrudan belleğe/veya bellekten veri transfer etmesine izin verir.
- D) CPU'ya G/Ç görevlerini aktararak genel sistem performansını artırır.

Cevap

- Cevap: B

- *DMA*, veri transferi işlemlerinde CPU'nun rolünü azaltır. Veri transferi işlemleri, geleneksel olarak işlemci tarafından yürütülür. Ancak *DMA* kullanılarak, bu işlemler CPU müdahalesi olmadan gerçekleştirilebilir. Bunun yerine, *DMA denetleyicisi*, veri transferini doğrudan yürütür ve CPU'ya sadece gerekli olduğunda bilgi verir. DMA, çevre birimlerinin (örneğin, disk sürücüleri, ağ kartları) doğrudan belleğe veya bellekten veri transfer etmesine izin verir.

Soru

- Aşağıdakilerden hangisi *asenkron G/Ç*'nin bir özelliğidir?

- A) G/Ç işlemi tamamlanana kadar çağıran süreci bloke eder.
- B) İşlemin tamamlandığının kontrolü için çağıran süreç tarafından sorgulama gerektirir.
- C) Senkron G/Ç olarak da bilinir.
- D) G/Ç işlemlerinin hemen tamamlanmasını garanti eder.

Cevap

- Cevap: B

- Asenkron G/Ç, bir işlem sırasında G/Ç işlemlerinin arka planda tamamlanmasına olanak tanır. G/Ç işlemi başlatıldıktan sonra işlemci beklemek zorunda kalmaz. İşlemi başlatan süreç, G/Ç işleminin tamamlandığını kontrol etmek için sorgulama yapar. Süreçlerin G/Ç işlemlerini başlatırken diğer süreçlerle aynı anda devam etmelerine olanak tanır. Sistemdeki kaynakların daha etkili bir şekilde kullanılmasını sağlar.

Soru

- İşletim sistemi içinde aygıt sürücüsünün amacı nedir?

- A) Aygıtları yönetmek için grafiksel kullanıcı arayüzü sağlamak.
- B) Uygulama taleplerini donanım aygıtları tarafından anlaşılabilir komutlara çevirmek.
- C) G/Ç işlemleri için sistem kaynaklarını tahsis etmek.
- D) G/Ç yoğun işlemler için CPU çizelgelemesini optimize etmek.

Cevap

- Cevap: B

- Aygıt sürücüsü, uygulamaların aygıtlarla iletişim kurmak için kullandığı yüksek düzeyli komutları, donanım aygıtları tarafından anlaşılabilir düşük düzeyli komutlara dönüştürür. Uygulamaların donanım aygıtlarına erişmesini ve onlarla etkileşime girmesini sağlar. Donanım aygıtlarının doğru çalışması için donanım kaynaklarını yönetir. Donanım aygıtlarının performansını artırmak için optimize edilmiş sürücü kodu sağlar. Donanım aygıtlarında oluşabilecek hataları algılar ve işletim sistemi veya kullanıcıya bu hataları bildirir. İşletim sistemi ve aygıtlar arasında uyumluluğu sağlar.

Soru

- Unix benzeri işletim sistemlerinde *select* sistem çağrısının amacı nedir?

- A) Kullanıcıdan girdi almak.
- B) Bir veya daha fazla G/Ç işleminin tamamlanmasını beklemek.
- C) Bir komutu arka planda çalıştırmak.
- D) Yeni bir süreç oluşturmak.

Cevap

- Cevap: B

- *select* sistem çağrısı, birden fazla dosya tanımlayıcısı ve bir zaman aşımı değerini alarak, dosyalar üzerindeki G/Ç işlemlerinin tamamlanmasını bekler. Tek bir işlemde birden fazla dosyadan gelen verileri izlemeyi ve işlemeyi mümkün kılar. Birden fazla G/Ç işlemi arasında geçiş yapılmasını ve verimliliğin artırılmasını sağlar. Bir sunucu, birden fazla istemciden gelen bağlantıları kabul etmek için *select* çağrısını kullanabilir. Bir zaman aşımı değeri belirterek, belirli süre içinde G/Ç işlemlerinin tamamlanmasını bekler. Sistemdeki kaynakların verimli kullanılmasını sağlar.

Soru

- Dosya G/Ç işleminde *seek* işleminin görevi nedir?

- A) Dosya içindeki mevcut konumu alır.
- B) Dosya işaretçisini dosya içinde belirtilen bir konuma taşır.
- C) Dosya önbelleğinin içeriğini diske boşaltır.
- D) G/Ç işlemi tamamlandıktan sonra dosyayı kapatır.

Cevap

- Cevap: B

- Her dosya, dosyanın içindeki belirli bir konumu izlemek için dosya işaretçisine sahiptir. *seek* işlemi, dosya işaretçisini belirli bir konuma taşır. Dosya içinde gezinmeyi ve istenen verilere erişmeyi sağlar. Dosya işaretçisinin taşınması, dosya içinde belirli bir konuma gidilmesini sağlar, dosya içinde arama yapmak veya belirli bir veri bloğuna erişmek için gereklidir. Örneğin, bir dosyada belirli bir yerden okuma veya yazma için dosya işaretçisinin konumlandırılması gerekir.

Soru

- Hangi G/Ç çizelgeleme algoritması, G/Ç isteklerinin tamamlanmasını beklemek için harcanan toplam zamanı en aza indirmeyi amaçlar?

- A) İlk Gelen, İlk Hizmet (*FCFS*)
- B) En Kısa Arama Zamanı Önce (*SSTF*)
- C) Round Robin
- D) İlk Giren, İlk Çıkan (*FIFO*)

Cevap

- Cevap: B

- *SSTF* algoritması, mevcut başlık konumuna en yakın veri bloğuna erişmek isteyen G/Ç isteğini seçer. Başlık konumuna hareket süresini azaltarak G/Ç isteklerinin tamamlanması için geçen toplam zamanı azaltır. *SSTF*, mevcut disk başlık konumunu optimize ederken, işlemci kullanımını da maksimize etmeyi amaçlar. Adil bir G/Ç hizmeti dağılımı ve tüm G/Ç isteklerinin hızlı şekilde karşılanmasını sağlar.

Soru

- İşletim sisteminde tampon önbelleğinin (*buffer cache*) amacı nedir?

- A) İkincil depolama aygıtlarından sık sık erişilen verileri saklamak.
- B) Sıkça yürütülen süreçler için talimatları önbelleğe almak.
- C) Aygıt sürücüleri için bellek tahsisini yönetmek.
- D) G/Ç yoğun süreçler için CPU çizelgelemesini optimize etmek.

Cevap

- Cevap: A

- Tampon bellek (*Buffer cache*), sık kullanılan veri bloklarını bellekte (*RAM*) tutarak disk erişim sürelerini ve G/Ç işlemlerini optimize etmeyi amaçlar. Bu sayede veriye erişim hızlanır ve performans artar. İşletim sisteminin disk üzerindeki veri bloklarını okuyup yazarken kullandığı tampon bellek alanıdır. İşletim sistemleri, kullanılmayan veya daha az kullanılan verileri tampon bellekten çıkarıp yeni veriler için yer açarak dinamik yönetim sergiler. Yazma işlemleri öncelikle tampona yapılır ve daha sonra toplu olarak diske yazılır. Veri tutarlılığı ve güvenliği sağlar.

Soru

- Aşağıdakilerden hangisi geçerli bir G/Ç aygıtı kategorisi değildir?

- A) Blok tabanlı aygıtlar
- B) Karakter tabanlı aygıtlar
- C) Ağ aygıtları
- D) Sıralı aygıtlar

Cevap

- Cevap: D

- Blok tabanlı aygıtlar, veri depolama ve erişimini bloklar halinde yapar. Örneğin, sabit disk ve SSD blok aygıtlarına örnektir. Karakter tabanlı aygıtlar, karakterlerin veya baytların seri bir şekilde okunup yazılmasını sağlar. Örneğin, klavye ve fare gibi giriş aygıtları ve yazıcı gibi çıkış aygıtları karakter tabanlı aygıtlara örnektir. Ağ aygıtları, bilgisayarlar arasında veri iletişimini sağlar. Örneğin, ağ arabirimi kartları (*NIC*), yönlendirici (*router*), anahtarlayıcı (*switch*) ve modem ağ aygıtlarına örnektir. Ağ üzerinde iletişim kurmak için veri paketlerini kullanırlar.

Soru

- G/Ç işlemlerinde aygıt denetleyicisinin temel amacı nedir?

- A) CPU'dan gelen komutları yorumlamak ve aygıt üzerinde çalıştırmak.
- B) CPU ile aygıt arasındaki fiziksel bağlantıyı yönetmek.
- C) CPU ve aygıt arasında veri dönüşümü yapmak.
- D) Aygıt ile etkileşimde bulunmak için kullanıcı arayüzü sağlamak.

Cevap

- Cevap: A

- Aygıt denetleyicisi, CPU tarafından gönderilen komutları alır ve bu komutları aygıtta gerçekleştirir. Örneğin, dosya okuma veya yazma gibi işlemler, aygıt denetleyicisi tarafından yönetilir. Aygıt denetleyicisi, ilgili aygıtın durumunu ve yapılandırmasını yönetir. Aygıtın etkin bir şekilde kullanılmasını sağlar ve uygun işlem için hazır durumda olmasını sağlar. CPU ve aygıt arasında veri aktarımını yönetir. Aygıtta oluşabilecek hataları algılar, işler, raporlar ve hataların giderilmesini sağlar.

Soru

- *Scatter-gather* G/Ç'nin başlıca avantajı nedir?

- A) Verilerin aynı anda birden fazla aygıt arasında aktarılmasına izin verir.
- B) Büyük veri transferlerini yönetme iş yükünü, ayrı önbellekler kullanarak azaltır.
- C) Aygıt sürücülerine gerek duymaz.
- D) CPU ve G/Ç aygıtları arasında gerçek zamanlı senkronizasyon sağlar.

Cevap

- Cevap: B

- Dağıtma ve toplama (Scatter-gather I/O) yöntemi, veri transfer işlemlerini optimize etmek için kullanılan bir tekniktir. Büyük verilerin parçalara bölünerek farklı bellek alanlarına dağıtılması (*scatter*) ve daha sonra bu parçaların bir bütün halinde toplanmasını (*gather*) sağlar. Özellikle yüksek performanslı veri transferleri gerektiren sistemlerde kullanılır.

Soru

- G/Ç aygıtlarının CPU'yu ele geçirmesini önlemek için hangi teknik kullanılır?

- A) Kesme tabanlı G/Ç (*Interrupt*)
- B) Sorgulama (*Polling*)
- C) Doğrudan Bellek Erişimi (*DMA*)
- D) Döner Kilit (*Spinlock*)

Cevap

- Cevap: A

- Kesme tabanlı G/Ç'de, bir G/Ç aygıtı işlemi tamamladığında CPU'ya bir kesme sinyali gönderir. CPU, kesmeyi alır almaz, mevcut süreci askıya alır ve G/Ç aygıtının beklediği işlemleri gerçekleştirir. Çoklu görev ortamlarında, birden fazla süreç aynı anda çalışır. Kesme tabanlı G/Ç, bu süreçlerin birbirini engellemeden çalışmasını, CPU'nun zamanını etkin bir şekilde kullanmasını sağlar. G/Ç aygıtlarıyla etkileşimi kolaylaştırır. Çünkü aygıtlar, işlemlerini tamamladıklarında CPU'ya bir kesme gönderirler ve CPU, bu kesintiyi hemen işleyebilir.

Soru

- Halka arabelleği (*ring buffer*) G/Ç performansını nasıl artırır?

- A) Donanım yazmaçlarına doğrudan erişerek aygıt sürücülerine gereksinimini azaltır.
- B) Sürekli ve döngüsel veri depolamasına izin vererek çekişme (*contention*) ve ek yükü (*overhead*) azaltır.
- C) Sürekli G/Ç aygıtlarını sorgular, kesme gereksinimini ortadan kaldırır.
- D) Birden fazla G/Ç aygıtı arasında gerçek zamanlı senkronizasyon sağlar.

Cevap

- Cevap: B

- Halka arabelleği, veriyi sürekli ve döngüsel bir şekilde saklar. Veri depolama alanının sonuna ulaşıldığında, yeni veri eski verinin üzerine yazılır. Veri depolama alanının verimli kullanılmasını sağlar, boşa harcanmasını önler. Çekişme ve ek maliyeti en aza indirir. Halka arabelleği, veriyi sürekli olarak saklar ve süreçler arasında veri aktarımını sağlar. Veri aktarımında çakışma olasılığını azaltır, G/Ç işlemlerinin daha düzenli gerçekleştirilmesini sağlar. Veri erişimi sırasında duraksama olmaz. Veri kaybını önler. Veri, halka arabelleğine yazıldığında, en eski veriler otomatik olarak silinir ve yerine yeni veriler yazılır.

Soru

- Bellek eşlemeli (*memory-mapped I/O*) ile G/Ç eşlemeli (*I/O-mapped*) adresleme bakımından nasıl farklılık gösterir?

- A) Bellek eşlemeli, bellek ve G/Ç aygıtları için ayrı adres alanlarını kullanırken, G/Ç eşlemeli birleşik bir adres alanını paylaşır.
- B) Bellek eşlemeli, aygıtın yazmaçlarını doğrudan CPU'nun adres alanına eşlerken, G/Ç eşlemeli aygıtlara erişmek için özel komutlar kullanır.
- C) Bellek eşlemeli, özel donanım desteği gerektirirken, G/Ç eşlemeli yalnızca yazılım öykünümüne (*emulation*) dayanır.
- D) Bellek eşlemeli, belirli CPU mimarilerine özelken, G/Ç eşlemeli evrensel olarak desteklenir.

Cevap

- Cevap: B

- Bellek eşlemeli G/Ç'de, aygıtın yazmaçları doğrudan CPU'nun adres alanına eşlenir. Aygıta erişmek için normal bellek erişim komutları kullanılabilir. Aygıta veri yazmak veya okumak için özel komutlara gerek yoktur. G/Ç aygıtı yazmaçları, normal bellek adresleri gibi işlenir.
- G/Ç eşlemeli G/Ç'de ise, aygıtlara erişmek için ayrı bir adres alanı kullanılır. CPU, aygıta erişmek için özel komutlar kullanır. Normal bellek erişim komutları ile aygıta erişilemez.

Soru

- Aşağıdakilerden hangisi vektörlü kesme-tabanlı G/Ç'nin bir özelliğidir?

- A) CPU'nun düzenli olarak aygıtları sorgulamasını gerektirir.
- B) Birden çok aygıtın tek bir kesme isteği hattını paylaşmasına izin verir.
- C) Aygıtlar arası doğrudan veri transferi için DMA kullanır.
- D) Gerçek zamanlı işletim sistemlerinde yaygın olarak kullanılır.

Cevap

- Cevap: B

- Vektörlü kesme-tabanlı G/Ç, birden çok aygıtın tek bir kesme isteği hattını paylaşmasına izin verir. Çok sayıda aygıtın aynı anda kesme talebinde bulunması durumunda kullanışlıdır çünkü her aygıt için ayrı bir kesme hattı gerektirmeden, tek bir hat üzerinden birden çok aygıtın kesme isteklerini işlemeyi sağlar. Yoğun G/Ç iş yüklerini etkin bir şekilde yönetmek için kullanılır. Sistem tasarımını basitleştirir ve donanım kaynaklarını daha verimli kullanılmasını sağlar.
- *Vektörlü kesme*, farklı çevre birimlerinden kaynaklanan kesme istekleri farklı adreslerdeki hizmet yordamlarına yönlendirilir.

Soru

- Aşağıdakilerden hangisi talepleri mevcut disk kafa konumuna yakınlığına göre önceliklendirir?

- A) SCAN
- B) C-LOOK
- C) LOOK
- D) SSTF

Cevap

- Cevap: D

- *SSTF* (*Shortest Seek Time First*), En kısa arama süresi gerektiren yani en yakın talep önce işlenir. Mevcut disk kafa konumuna göre yakın olan veri taleplerini önceliklendirir. Disk hareketlerini minimize eder. Ortalama erişim süresini kısadır ve disk performansını artırır. Algoritmanın dezavantajı ise, talepler uzak bölgelerde sıkışıp kalabilir.

Soru

- Disk G/Ç işlemlerinde *arama zamanı* (*seek time*) ne anlama gelir?

- A) Disk ve bellek arasında veri aktarımı için harcanan zaman.
- B) Disk kolunun istenen izin üzerine konumlanması için harcanan zaman.
- C) CPU'nun bir disk G/Ç isteğini işlemesi için harcanan zaman.
- D) Diskin gerekli dönme hızına ulaşmak için harcanan zaman.

Cevap

- Cevap: B

- *Arama zamanı*, disk G/Ç işleminde, disk kolu veya okuyucunun istenen iz üzerine gelmesi için geçen zamandır. Disk kafasının bir izden diğerine hareket etmesi ve istenen veriye erişmek için gerekli konuma gelmesi anlamına gelir. Daha kısa arama süreleri, daha hızlı disk erişimi anlamına gelir, daha iyi bir performans sağlar. Disk çizelgeleme algoritmaları, arama zamanını minimize etmek için optimize edilir, böylece disk kafası mümkün olduğunca az yer değiştirir ve veri taleplerini hızlı bir şekilde işler.

Soru

- Hangi G/Ç tekniği, CPU ve G/Ç aygıtı arasında sorgulama veya kesmelere gerek duymadan veri transferine izin verir?

- A) Doğrudan Bellek Erişimi (*DMA*)
- B) Programlı G/Ç
- C) Kesme-tabanlı G/Ç
- D) Bellek Eşlemeli G/Ç

Cevap

- Cevap: A

- Doğrudan Bellek Erişimi (*DMA*), CPU'nun doğrudan müdahale etmeden, G/Ç aygıtı ve bellek arasında veri transferi yapmasını sağlayan bir G/Ç tekniğidir. Sürekli sorgulama veya kesmelere gerek olmadan, veri transferini daha verimli bir şekilde gerçekleştirir. DMA, genellikle blok transferler yapar. Yani, tek bir komutla birden çok veri parçası (*blok*) transfer edilebilir. DMA, yüksek bant genişliği gerektiren uygulamalarda kullanılır.

Soru

- Aşağıdakilerden hangisi *spooling*'i doğru tanımlar?

- A) *Simultaneous Peripheral Operations Online*'nin kısaltmasıdır.
- B) Birden çok G/Ç işleminin aynı anda işlenmesine izin verir.
- C) İşleme hızını dengelemek için verilerin geçici bir kuyruğa depolanması.
- D) Toplu işlem sistemlerde yazıcı çıktısını yönetmek için kullanılır.

Cevap

- Cevap: A, C

- *Spooling* (*Simultaneous Peripheral Operations Online*), veri giriş/çıkış işlemlerini optimize etmek için kullanılır. Verileri geçici bir kuyruğa (*spool*) depolayarak işleme hızını dengeler. Özellikle, yavaş bir aygıtın hızlı bir şekilde işlenebilmesini sağlar. Örneğin, bir yazıcının yavaş işleme hızı, spooling sayesinde yazdırma işlemlerinin hızlı bir şekilde sırayla gerçekleştirilmesini sağlar. Bir aygıtın meşgul olması durumunda diğer işlemlerin sıraya alınmasını ve beklemesini sağlar. Genellikle yazıcı çıktısını yönetmek için kullanılır.

Soru

- Hangi disk çizelgeleme algoritması, talepleri aldığı sıraya göre hizmet verir?

- A) SCAN
- B) C-SCAN
- C) FCFS (İlk Gelen, İlk Hizmet)
- D) SSTF (En Kısa Arama Süresi Önce)

Cevap

- Cevap: C

- *FCFS* (*First-Come, First-Served*), algoritması, talepleri aldığı sıraya göre hizmet verir. İlk gelen talep, ilk hizmet edilen olur. Talepleri aldığı sıraya göre işlediği için, tüm talepleri eşit şekilde ele alır. Talepler arasında adil bir dağılım sağlar. Taleplerin gelme sırası, disk üzerindeki veri dağılımına göre optimize edilmemişse, verimli olmayabilir.

Soru

- G/Ç işlemleri bağlamında, *pipelining* ne anlama gelir?

- A) Birden fazla G/Ç isteğini tek bir veri akışına birleştirme süreci.
- B) G/Ç işleminin birden fazla aşamasının eşzamanlı olarak yürütülmesi.
- C) Verinin boru hattı ile giriş/çıkış aygıtları arasında yönlendirilmesi
- D) G/Ç görevlerinin paralel yürütülmesi için küçük alt görevlere bölünmesi.

Cevap

- Cevap: B

- *Pipelining*, bir G/Ç işleminin farklı aşamalarının aynı anda yürütülmesini sağlar. Örneğin, bir veri transferi işlemi, veri okuma, işleme ve yazma aşamalarından oluşabilir. *Pipelining* sayesinde, bu aşamalar eşzamanlı olarak yürütülür ve böylece işlem tamamlanma süresi azalır. G/Ç işlemlerinin paralel olarak işlenmesini sağlar.

Soru

- Programlanmış G/Ç'yi, kesme-tabanlı G/Ç'den ayıran nedir?

- A) Programlanmış G/Ç, CPU'nun sürekli olarak G/Ç aygıtını sorgulamasını gerektirir, kesme-tabanlı G/Ç ise CPU'nun G/Ç işlemi tamamlanana kadar diğer görevleri gerçekleştirmesine izin verir.
- B) Programlanmış G/Ç, veri transferi için DMA kullanırken, kesme-tabanlı G/Ç doğrudan CPU müdahalesine dayanır.
- C) Programlanmış G/Ç, kesme-tabanlı G/Ç'ye göre hızlı ancak verimsizdir.
- D) Programlanmış G/Ç, yalnızca düşük hızlı G/Ç aygıtları için uygunken, kesme-tabanlı G/Ç yüksek hızlı aygıtlar için kullanılır.

Cevap

- Cevap: A

- Programlanmış G/Ç (*Programmed I/O*): CPU sürekli G/Ç aygıtını sorgular ve G/Ç işlemi tamamlanana kadar başka görev gerçekleştiremez. CPU, G/Ç işlemi için aygıta komut gönderir ve ardından aygıtın hazır olduğunu kontrol etmek için tekrar tekrar sorgular.
- Kesme-tabanlı G/Ç (*Interrupt-driven I/O*): G/Ç aygıtı işini tamamladığını kesmeler ile işlemciye bildirir. CPU, G/Ç işlemi tamamlanana kadar diğer görevleri gerçekleştirebilir. Kesme-tabanlı G/Ç, CPU'nun boşa harcanan zamanını azaltır ve sistem verimliliğini artırır.

Soru

- Hangi G/Ç tekniği, birden fazla G/Ç işleminin aynı anda başlatılmasına ve bağımsız olarak tamamlanmasına izin verir?

- A) Dağıtık G/Ç (*Scatter-gather I/O*)
- B) Asenkron G/Ç (*Asynchronous I/O*)
- C) Senkron G/Ç (*Synchronous I/O*)
- D) Tamponlu G/Ç (*Buffered I/O*)

Cevap

- Cevap: B

- Asenkron G/Ç, birden fazla G/Ç işleminin aynı anda başlatılmasına izin verir. Farklı G/Ç işlemlerinin aynı anda çalışmasını sağlar. Her bir G/Ç işleminin bağımsız olarak tamamlanmasına izin verir. Bir işlem tamamlandığında, diğer işlemler etkilenmez ve kendi süreçlerine devam eder.

Soru

- Disk çizelgeleme algoritmalarından hangisi, disk kafasının mevcut konumuna yakın taleplere öncelik verir, ardından iz takip numaralarının artış yönünde ilerler ve son izine ulaştıktan sonra yönünü tersine çevirir?

- A) LOOK
- B) C-SCAN
- C) SCAN
- D) SSTF

Cevap

- Cevap: C

- *SCAN* algoritması, disk üzerindeki veri taleplerini belirli bir yönde tarar ve taleplere hizmet verir. Mevcut konuma yakın olan talepler öncelikli olarak işlenir. SCAN, iz takip numaralarının artış yönünde ilerler. Disk kafasının mevcut konumundan daha uzak taleplere öncelik verildiğinde bile, mevcut konumdan sonraki taleplere hizmet edilmesini sağlar.

Soru

- Eşzamanlı G/Ç'yi, eşzamanlı olmayan G/Ç'den ayıran nedir?

- A) Eşzamanlı G/Ç, CPU'nun G/Ç tamamlanana kadar diğer görevleri gerçekleştirmesine izin verirken, eşzamanlı olmayan G/Ç, CPU'yu G/Ç işlemi tamamlanana kadar bekletir.
- B) Eşzamanlı G/Ç, G/Ç işlemlerinin hemen tamamlanmasını garanti ederken, eşzamanlı olmayan G/Ç'nin tamamlanma durumunu belirlemek için açıkça sorgulanmasını gerektirir.
- C) Eşzamanlı G/Ç, CPU ve G/Ç aygıtı arasında doğrudan iletişim içerirken, eşzamanlı olmayan G/Ç iletişim için kesmelerden yararlanır.
- D) Eşzamanlı G/Ç, veriyi doğrudan bellekten G/Ç aygıtlarına aktarırken, eşzamanlı olmayan G/Ç veriyi aktarmadan önce bellekte tamponlar.

Cevap

- Cevap: A

- Eşzamanlı G/Ç (*Synchronous I/O*): CPU, bir G/Ç işlemi tamamlanana kadar diğer görevleri gerçekleştirebilir. CPU, G/Ç işlemi tamamlanana kadar beklemek zorunda değildir.
- Eşzamanlı Olmayan G/Ç (*Asynchronous I/O*): CPU, G/Ç işlemi tamamlanana kadar diğer görevleri gerçekleştiremez. CPU, G/Ç işlemi tamamlanana kadar beklemek zorundadır.

Soru

- Aşağıdakilerden hangisi *tampon bellekli G/Ç*'nin bir özelliğidir?

- A) Veriyi bellekten doğrudan G/Ç aygıtlarına aktarır.
- B) Her veri aktarım işlemi için CPU müdahalesi gerekir.
- C) G/Ç aktarımları sırasında veri geçici olarak arabellekte saklanır.
- D) Doğrudan G/Ç tekniklerine göre yavaştır.

Cevap

- Cevap: C

- Tampon bellekli G/Ç, veriyi geçici olarak saklamak için bir tampon kullanır. Örneğin, bir dosyanın diskten okunması sırasında, okunan veri önce bir tampona yazılır ve ardından işlenir. Verinin doğrudan bellekten G/Ç aygıtlarına aktarılmasını sağlar. Tampon, G/Ç aygıtları arasındaki hız farklarını dengelemek ve veri işleme sürecini iyileştirmek için kullanılır.

SON