Ömü

Subject :

Date : 29 / 11 / 2023

* İşletim sistemin bileşenleri:
1) Donanım: (cpu, memory, I/O) temel kaynakları sağlar
2) Uygulama programları: (derleyiciler, web tarayıcıları) bu kaynakların kullanıcı problemlerinde nasıl kullanılacağını belirler.
3) İşletim sistemi.
4) Kullanıcılar.

* Kullanıcı açısından: İşletim sistemi kolay kullanım için tasarlanır. Kaynak kullanım oranını maksimize edecek şekilde tasarlanır.

* Sistem açısından: İşletim sistemi, kaynak (CPU, Memory, ...) kullanımını planlayan programdır, I/O cihazlarını ve kullanıcı programlarını kontrol eder.

* Bilgisayarların temel amacı, kullanıcı programlarını çalıştırması ve kullanıcı problemlerinin kolay ve hızlı bir şekilde çözülmesidir.

* Kernel: İşletim sistemi bilgisayarda sürekli çalışan bir programdır ve kernel olarak adlandırılır.

* Mobil işletim sistemi: Kernel ve middleware'e sahiptir.
* Middleware: Uygulama geliştiricilere ek servisler sağlayan Framework yazılımlarıdır.

* CPU ile cihaz denetleyicileri eş zamanlı çalışırlar,
* hafıza denetleyicisi cihazların hafızaya erişimini yönetir.
* Bootstrap: Bilgisayar çalışmaya başlandığında, başlangıç programı olarak Bootstrap programını kullanır.
* Bootstrap: Rom, EEPROM da saklanır.
* Bootstrap: Bootstrap programlarına firmware adı verilir.
* Firmware: tüm bilgisayar bileşenleri başlatır.
* Bootstrap programı: işletim sistemi kernelin bulunduğu konumu bilmek ve hafızaya yüklemek zorundadır.
* Bazı servisler kernel dışındaki sistem programları (system deamons, system process) tarafından sağlanır.

Subject: Date: 29/01/2023

* DMA (Direct memory access): blok veri ile hafıza arasında doğrudan aktarır.
* komut kümesindeki tüm komutlar bir işlemci tarafından çalıştırmaktadır
* Tek işlemciye sahip olan sistemler tek işlemcili olarak adlandırılır.
** Çok işlemcili sistemler. ("multiprocesser system", "parallel system" "multicore system")
* Çok işlemcili sistemlerin kullanımı. ① sunucu sistemlerinde ② masaüstü ve dizüstü bilgisayarlarda ③ mobil cihazlarında
** Çok işlemcili sistemlerin temel olarak 3 avantajı vardır.
① yüksek throughput: işlemci sayısı arttıkça daha fazla iş yapılır
② Ekonomik ölçeklendirme: Tek işlemcili sistame gore daha ekonomiktir
③ yüksek güvenilirlik: Bir işlemcide oluşan hata sistemin tümünü çalışmaz hale getirmez
** Asimetrik (AMP): her işlemci bir işe atanmıştır.
** Simetrik (SMP): her işlemci tüm işleri yapabilir
* SMP mimarisinde her cpu kendi register'larına ve local cach'e sahiptir. (windows, Mac OS X, linux) desteklemektedir
* UMA: RAM'e erişim süresi tüm işlemcilerde aynı
* NUMA: RAM'e erişim " " " farklı
* Multicore sistemleri daha hızlıdır ve az enerji tüketirler.
* Clustered sistemleri Bağımsız iki veya daha fazla sistemden oluşurlar. LAN (local area network) üzerinde haberleşirler.
* Asymmetric clustering: Bir sistem aktif çalışır değeri bekleme modundadır.
* Clustered sistemlerde. Bir program parçalara bölünerek eş zamanlı çalıştırılabilir (Parelleligation)
* Multiprogramming: işletim sistemleri birden fazla program çalıştırabilir, işini yaparken bekleme olduğunda diğer işleme geçiş yapar.
* Multitasking (time sharing): işletim sistemlerinde CPU işleri arasında çok hızlı geçişler yapar.
* process: hafızaya yüklenen ve çalıştırılmakta olan program. CPU tarafından çalıştırılır
* job scheduling: Hafızaya alınacak olan program.
* CPU scheduling: Hangisi ilk önce çalıştıracak.
* Malicious: iyi tasarlanmış işletim sistemleri, hatalı programların diğerlerini

Subject: Date: 29/11/2023

* İşletim sistemi boot edildikten sonra bir olay (event) gerçekleşmesi için beklemeye başlar.
* Bilgisayar sistemlerinde bir olayın olduğu yazılım veya donanım tarafından interrupt kullanarak bildirilir.
** Donanımlar, CPU'ya bir singal ile interrupt bildirim yapar.
** Yazılımlar, system call işlemlerini çalıştırarak interrupt başlatır.
** CPU bir interrupt aldığında çalışmasını bulunduğu yerde keser ve belirtilen diğer noktaya geçiş yapar.
* interrupt routine adresleri pointer ile gösterilir.
* Komut adresi ve CPU konfigürasyonu stack (yığın) üzerinde saklanmaktadır
* CPU programlara ait komutlar hafızaya yükler
* Bilgisayarlar programları çalıştırmak için ana hafızayı (Ram) kullanır.
* Von Neumann mimarisine sahip sistemlerde, komutları fetch ile çalıştırır.
* Fetch (hafızadan CPU içerisindeki register'a alınması)
* Programlar hafızada saklanmamasının nedenleri:
  1. hafıza sınırlı kapasiteye sahiptir
  2. Enerji kesildiğinde hafızadaki veri kaybolur. (Volatile) (uçucu)
* Tüm programlar (secondary storage) da saklanmaktadır. (magnetic disk, HDD)

1. Register	geçici	• yukarı çıktıkça erişim hızı artar.
2. Cache		• " " bit başına saklama maliyeti artar
3. main memory (RAM)		• " " toplam kapasite azalır.
4. Solid-state disk (SSD)	kalıcı	• " " CPU tarafında kullanılma sıklığı artar.
5. magnetic disk		
6. optical disk		
7. magnetic tapes		

* DRAM ile manyetik disk birlikte kullanılır. Normal işlem sırasında DRAM kullanılır daha sonra manyetik diske aktarma yapılır (dahili bataryayı kullanır)
* İşletim sistemi her cihaz denetleyicisi için cihaz sürücüsüne (device driver) sahiptir.
* I/O işlemini başlatmak için device driver uygun register içeriğini device controller'a aktarır.
* Device controller tarafından device driver'a interrup ile bildirilir.
* Overhead: veri aktarma işlemi gerçekleştirir. İşlemin normal çalışması sırasında ortaya çıkan ek maliyetleri veya kaynak kullanımını ifade eder.

Subject: Date: 29.11.2023

user Process → (user process exculing) → (calls system call) → (return from system call)

trap mode bito

kernal → [execute system call] → return mode bit 1

* Timer her programa geçildiğinde set edilir ve aşağıya doğru sayar
** Timer o değerine ulaşınca kontrol işletim sistemine alınır.
* işletim sistemleri her program için belirlen timer süresini sabit veya değişken alabilmektedir.
* Bir process, CPU süresine, hafızaya, dosyalara, I/O cihazlarına ihtiyaç duyar
* program counter (PC) CPU içerisinde register'dır. Ve sonraki çalıştıracak komutun adresini tutar.
* Bir işletim sistemi process yönetiminde aşağıdaki işlerden sorumludur.
  1) CPU üzerindeki process ve thread'lerin zamanlaması
  2) kullanıcı ve sistem processlerinin oluşturulması ve silinmesi
  3) Process'lerin askıya alınması ve devam ettirilmesi.
  4) process'lerin senkronizasyonu
  5) process'lerin haberleşmesi
* CPU, tüm programlar hafıza üzerinde çalıştırır.
* işletim sistemi hafıza yönetiminde aşağıdaki işlerden sorumludur.
  1) Hafızanın hangi kısmının kullanıldığının izlenmesi
  2) Hangi process'in hafızaya alınacağına karar verilmesi
  3) Boş alanların tahsis edilmesi
* Depolama biriminin özelliklerini soyutlamak için "File" tanımlar.
* işletim sistemi Dosya yönetiminde aşağıdaki işlerden ~~sorum~~ sorumludur.
  Dosya oluşturma ve silme
  Dizin oluşturma ve silme
  Dosya ve Dizin manipülasyon işlemleri
  ~~işletim~~
* Tüm programlar hafızaya yüklenmeden önce disk üzerinde saklanır.
* İşletim sistemi disk yönetiminde aşağıdaki işlerden sorumludur
  1) Boş alan yönetimi
  2) Depolama alanı tahsisi
  4) Disk kullanım zamanlaması.
* Cache Bellekte tutulacak veya atılacak verilerin belirlenmesi için replacement algoritması kullanılır.

Subject: Date: 29./11../2023

** cache belleklerde erişim adrese göre register'larda isime göre yapılır.
** işletim sistemlerinin amaçlarından birisi donanımların özelliklerinden kullanıcıyı soyutlamaktadır.
* I/O sistemi aşağıdaki bileşenlere sahiptir.
① Hafıza yönetim bileşeni (buffering, caching, spooling)
② Device driver arayüzü
③ Donanımlar için driver.
* Bilgisayar sistemindeki kaynaklara kullanıcıların veya process'lerin erişiminin denetlemesine koruma (Protection) denilmektedir.
* protection oluşabilecek hataları arayüzde iken algılar ve sistemin güvenilirliği artırır.
* Güvenlik (security) saldırılara karşı korumayı amaçlar.
* protection ve security sistemdeki tüm kullanıcıların birbirinden ayrıt edilmesini gerektirir. kullanıcı kimliğile (user ID) ve grup ID (admin) ile yapılır.
* Dizinin elemanlarının boyutu sabit linkedlist ise farklı olabilir
* Stack (LIFO)
* işletim sistemleri, iç içe fonksiyon çağırmalarında stack yapısı kullanılır.
* Queue (FIFO)
işletim sistemi yazıcıya iş gönderirken, işlemci tarafından çalıştırmak için bekleyen görevlerin yönetiminde kuyruk yapısını kullanır.
* Ağaç (tree) veriyi hiyerarşik şeklinde göstermek için kullanılır.
* hash fonksiyonu: giriş olarak veri alır, bu veri üzerinde sayısal bir işlem yapar ve bir sayısal değer döndürür. erişim performansı O(1)
* Mobil hesaplama (mobile computing) akıllı telefonlar ve tablet bilgisayarlar ile yapılan işlemleri ifade eder.
* Dağıtık sistem sahip olduğu çok sayıdaki kaynağa kullanıcıların erişimini sağlar.
* Bilgisayar ağları kapsadıkları alana göre LAN < MAN < WAN
Bilgisayarlar ve akıllı telefonların arasındaki oluşturulan ağ "PAN"
* Sunucu (server) sistemi, istemci (client) bilgisayarların isteklerini karşılamak üzere tasarlanır.
* Compute - server: istemciler bir işlemin yapılması için arayüz üzerinden istek gönderirler.
* File - server: istemciler dosya oluşturma, silme veya okuma işlemlerini

Subject: Date: 29/11/2023

* sanallaştırma (virtualization): işletim sistemlerinin uygulamaları başka işletim sistemlerinde çalışmasına izin verir.
* yorumlayıcı (interpreter)
* Derleyici (compiler) → kaynak kodu (source code) makine koduna dönüştürür makina kodu (native code) veya (object code) olarak adandırılır.
* Bulutların tipleri.
1) public cloud: internet üzerinde kullanımına açıktır.
2) private cloud: bir firmanın sahibi olduğu bulottur.
3) Hybrid cloud: private ve public kullanılan bileşenlerine sahiptir.
4) SaaS: Bir veya daha fazla uygulama internet (word, Excel)
5) PaaS: Bir yazılım yığın uygulamalar için internet aracılığıyla kullanıma açıktır (veritabanı sunucu)
6) IaaS: sunucular veya depolama birimleri internet aracılığıyla kullanıma açıktır.
* Gömülü (embedded) sistemler günümüzde araç metotları, üretim robotları mikro dalga fırınlar, uçaklar, gibi çok farklı yerlerde kullanılmaktadır.
* Gömülü sistemler özel amaçlar için geliştirilirler ve sahip oldukları işletim sistemleri sınırlı özelliklere sahiptir.
* Açık kaynak kapalı kaynağa göre daha güvenilirdir.
* işletim sistemi servisleri:
1) kullanıcı arayüzü
2) program çalıştırma
3) I/O işlemleri
4) Dosya sistemi işlemi
5) iletişim: iletişim paylaşımlı hafıza alanlarında mesaj göndererek yapılabilir
6) Hata denetimi:
7) kaynak tahsisi
8) Hesap oluşturma
9) Koruma ve güvenlik
* Arayüz türleri:
1. comman line interface
   * kullanıcı doğrudan komutları yazar
   * öğrenmesi ve kullanımı zordur
2. Graphical user interface (GUI)
   * kullanıcılar farklı bileşenlerle komutları oluşturabilir
   * kullanımı ve öğrenmesi kolaydır.

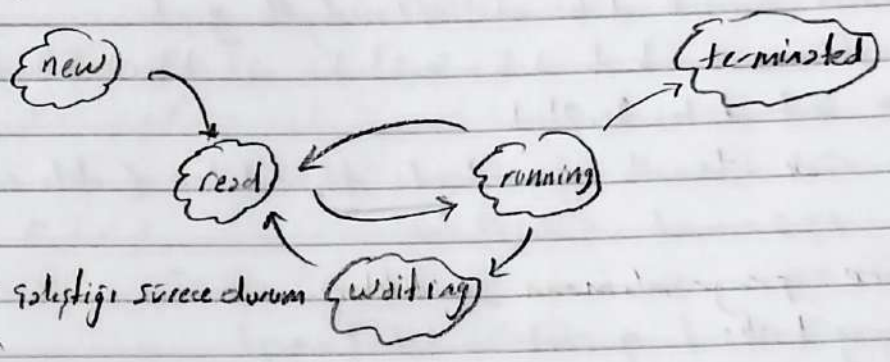
Oms

Subject:

Date: 29/11/2023

* Interpreter shell olarak adlandırılır.
* Sistem çağrı türleri:
  1) Process managment / işlem yönetimi
  2) File management / Dosya yönetimi
  3) Device management / Cihaz yönetimi
  4) Information maintenance / Bilgi Bakımı
  5) Communications / iletişim
  6) Protection / güvenlik
* Bir çok sistem çağrısı kullanıcı programı ile sistem çağrıları arasında veri transferi yapmak için kullanılır.
* Process'ler ~~ta~~ iletişim için iki yöntem vardır.
  1. Massage passing model
  2. Shared memory model
* Sistem programları:
  1. File management / Dosya yönetimi
  2. status information / Durum bilgisi
  3. File modification / Dosya değişikliği
  4. programming language support / programlama dili destekleme
  5. program loading / execute / program yükleme / yürütme
  6. Communications / iletişim
  7. Background services / arkaplanın hizmetleri
* Bir işletim sistemi yüksek seviyeli dillerle yazılması, farklı mikroişlemcilerde çalışabilmesini kolaylaştırır.
* yüksek seviyeli dillerde yazılan işletim sistemleri daha yavaştır ve daha fazla depolama alanına ihtiyaç duyarlar.
* Mikrokernel kısmında, iletişim bileşenleri, hafıza yönetimi ve küçük process'ler kalmıştır.
* Linux kernel, Hafıza yönetimi, Process yönetimi, cihaz sürücüleri ve power management işlemleri için kullanılır.
* Dubbugging: genel olarak sistemdeki hataların bulunması ve giderilmesi işlemlerini ifade eder.
* Debugging, bir kodu yazmaktan çok daha zordur.
* Kernel'in yüklenerek bilgisayarın başlatmasına Booting denilmektedir.
* Bootstrap Rom üzerindedir ve işletim sisteminin kernel kısmını yükler.

* Mobil cihazlarda işletim sisteminin tamamı EPROM üzerinde saklanır.
* Bayağı işletim sistemlerinde Bootstrap Room'da saklanır işletim sistemi ise diskte
* Boot kısmına sahip olan disk, Boot disk veya System disk adlandırılır.
* CPU, aralarında geçiş yaparak tüm process'leri eş zamanlı çalıştırabilir.
** Aynı program birden fazla process ile ilişkili olabilir



* Bir process çalıştığı sürece durum değiştirir
  - new: Process oluşturulmaktadır.
  - Running: Komutlar çalıştırılmaktadır
  - waiting: process bir olayın gerçekleşmesini beklemektedir
  - Ready: Process çalışmak için CPU'ye atanmak üzere bekliyor.
  - Terminated: process çalışmasını sonlandırır

** Modern işletim sistemlerinde bir process ile birden fazla thread çalışmasına izin verilir, Bu özellik Multicore çok faydalıdır ve çok thread eş zamanlı çalıştırılır.
* Bir process'i cpu'da çalışması için process scheduler seçer.
* Cpu'nun bir process'ten başka bir process'e geçmesine context switch denilir.
* Bir process çalışması sırasında birkaç tane başka process'i çalıştırabilir
* Cooperating process'ler shared memory ve message passing modelleri ile veri aktarımı yaparlar.
* Shared memory modeli daha hızlıdır.
* Thread, program counter, bir grup register ve bir stack yapısına sahiptir
* Threadler program kodunu, data kısmını, dosyalar gibi işletim sistemi kaynaklarını ortak kullanır.
* Eğer bir process birden fazla thread'e sahipse birden fazla görevi eşzamanlı yapabilir.

Subject: Date: 29/11/2023

* Thread'lerin sağladığı faydalar:
* Responsiveness: bloklanmış, kilitlenmiş veya uzun süren işlem yürütüyorsa başka bir kısım çalışmasını sürdürür.
* Resource sharing: Thread'ler ait oldukları process'in sahip olduğu hafıza alanını ve diğer kaynakları paylaşabilirler.
* Economy: Thread'ler ait oldukları process'in kaynaklarını paylaştıklarından dolayı context switch daha düşük maliyetle yapılır.
  - Thread oluşturma 30 kat daha hızlıdır ve thread'lerde context switch 5 kat daha hızlıdır
* Scalability: Çok işlemcili mimarilerde thread'ler farklı core'lar üzerinde eş zamanlı çalışabilir.
* Multicore programlamanın zorlukları
  1. Identifying tasks / görevlerin belirlenmesi
  2. Balance / Denge
  3. Data splitting / veri bölme
  4. Data Dependency / veri bağımlılığı
  5. Testing and debugging / Test etme ve hata ayıklama

* user thread'leri kullanıcı uygulamaları tarafından, kernel thread'leri ise işletim sistemi tarafından gerçekleştirilir.
* Multithreaded sistemlerde belirli sayıda thread oluşturulmasına izin vermek için thread pool oluşturulur.
* Cooperating process'ler diğer processleri etkiler veya diğer process'lerden etkilenir.
* Cooperating process'ler paylaşılmış hafıza alanıyla ve dosya sistemleri ile veri paylaşımı yaparlar.
* paylaşılmış veriye eş zamanlı erişim tutarsızlık problemlerine yol açabilir.
* paylaşılmış veri üzerinde işlem yapan process'ler arasında veriye erişimin yönetilmesi gereklidir.
* multicore işlemcilerde çalışan multithread uygulamalardada process senkronizasyonu yapmak zorunludur.
* kritik bölüm problemleri:
  1. Her process kritik bölüme sahip olabilir
  2. Aynı anda iki process kritik bölümüne çalıştırmamalıdır.
  3. her process kritik bölümüne girmek için izin istemektedir.

Subject: Date: 29/11/2023

* kritik bölüm probleminin çözüm aşağıdaki üç gereksinimi sağlamak
① Mutual exclusion (karşıklı dışlama): Bir process'i kritik bölümünü çalıştırılıyorsa
② progress: Hiçbir process kritik bölümünü çalıştırmıyorsa
③ Bounded waiting (sınırlı bekleme): Bir process kritik bölüme giriş izni istedikten sonra ve izin verildikten önceki aralıkta
* sayan semaforların değeri kısıtlı değildir, 0 ve 1 de olabilir
* sayan semaforlar, belirli sayıdaki kaynağa erişimi denetlemek için kullanılır
* process semafor üzerinde wait() işlemi gerçekleştirilir. (sayaç azaltılır)
* process kaynağı serbest bıraktığında ise signal() işlemi gerçekleştirilir (sayaç artırılır)
* semafor değeri = 0 olduğunda tüm kaynaklar kullanılır durumundadır
* Multicore sistemlerde, mutex lock, semafor gibi mekanizmalarda deadlock gibi problemlerin oluşma riski bulunmaktadır
* Thread sayısı arttıkça deadlock problemlerinin ortaya çıkma olasılığı artmaktadır
* CPU scheduling (planlama) multiprogramming çalışan işletim sistemlerinin temelini oluşturur.
* Dispatcher: CPU scheduling işlerini gerçekleştiren bileşen.
* Dispatcher: short-term scheduler tarafından CPU'ya atanacak processi seçer.
* SJF algoritmasındaki en büyük zorluk, sonraki çalışma sürecini tahmin etmektedir
* SJF algoritması long term scheduling için kullanılır. short term scheduling seviyesinde kullanılmaz.
* short-term scheduling'te CPU'da sonraki çalışma süresini bilmek mümkün değildir
* short-term scheduling'te sonraki çalışma süresi tahmin edilmeye çalışır.
* üstel ortalama değeri tahmin edilir: $[T_{n+1} = \alpha t_n + (1-\alpha)T_n]$
$0 < \alpha < 1$ genellikle ($1/2$), T_{n+1} sonraki tahmin edilen süreyi, CPU burst süresi gös.
* öncelikleandırme kriterleri:
① zaman sınırı
② hafıza gereksinimi
③ açılan dosya sayısı
④ I/O burst ve CPU burst oranı
⑤ Process'in önemi
* Multi queue scheduling algoritmasında, process'ler farklı gruplar halinde sınıflandırılır ve arasında geçiş yapabilirler.

Subject:

Date: 29/11/2023

+ Çok işlemci kullanılan sistemlerde yük paylaşımı (load sharing) yapılabilir
Bir process başka bir işlemciye aktarıldığında, eski işlemcideki cache bellek bilgileri aktarılmaz

* Gerçek zamanlı CPU scheduling iki gruba ayrılır.
  - soft real time sistemler: çalışma süresine garanti vermez
  - hard real time sistemler: deadline süresinde çalıştırmayı garanti eder

** Bir process'in durdurularak diğer process'in başlatması için geçen süreye "dispatch latency" denir.

"7. slayt vizeden sonra"

* Multiprogramming ortamlarda çok sayıda process sınırlı kaynağı kullanmak için yarışır.

* Bir process bir kaynağa istek yaptığında, kaynak dolu ise bekleme durumuna geçer.
Bekleme durumundaki process, bazı durumlarda hiç bir zaman durumunu değiştirmez, Buna "deadlock" denir.
Bir process bir kaynağı kullanmadan önce bir istek yapar (request), kullandıktan sonra da serbest bırakır. (release).

** Bir process bir kaynağı aşağıdaki sırasıyla kullanır.
- Request: process kaynağa istek yapar, kaynak kullanılabilir değilse bekler
- Use: process kaynak üzerindeki işlemini gerçekleştirir
- Release: process kaynağı serbest bırakır.

** Aşağıdaki 4 şart aynı anda oluştuğunda deadlock ortaya çıkabilir.

karşılıklı dışlama Mutual exclusion: paylaşımsız kaynak bir process tarafından tutulurken başka bir process bu ~~Hold and wait~~ kaynağa istek yaparsa, ikinci process kaynak boşalıncaya kadar bekler.
No preemption: kaynaklar önceden boşaltılmaz, kullanılan process'in serbest bırakılması gerek
Circular wait: processleri birbirini beklemektedir.
Hold and wait: Bir process bir kaynağı tutarken, başka bir kaynağı da bekle durumundadır.

** Deadlock problemi için 3 farklı yol izlenebilir.
① Deadlock'lardan kaçınmak için protokol kullanılabilir, sistem hiçbir zaman deadlock durumuna düşmez ② sistemin deadlock durumuna düşmesine izin verilir, deadlock algılanır ve çözülür. ③ Deadlock problemi tamamen göz ardı edilir.

Subject : Date : / /

* Bir sistemde hiç bir zaman deadlock olmamasını garanti etmek için deadlock prevention veya deadlock-avoidance yöntemleri kullanılabilir.
* Deadlock prevention, kaynak isteklerini sınırlandırarak deadlock oluşmasını önler.
* Deadlock avoidance, bir kaynağa istek yapan process'in yani sıra, hangi zaman aralığında kullanacağını bilmek ister, kaynak isteği yapan process'in beklenmesine veya kaynağın alınmasına karar verebilir.
* Deadlock-prevention ve Deadlock-avoidance yöntemlerini kullanmazsak deadlock oluşabilir.
* Deadlock algılama ve çözümleme yöntemleri çoğu işletim sisteminde kullanılmaz.
* Bazı sistemlerde, deadlock durumu yoktur. Bunun yerine process'in donmuş durumu vardır (frozen state).
* Deadlock oluşması için 4 durumunda (mutual exclusion, hold and wait no preemption, circular wait) gerçekleşmesi gereklidir.
* ↑ bu şartların birisi engellenirse deadlock önlenmiş olur.
* Mutual exclusion'da en az bir kaynak paylaşılmaz (eş zamanlı erişilmez) durumundadır.
* Read only dosyalar deadlock oluşturmaz.
* Paylaşılmaz kaynaklara birden fazla process tarafından eş zamanlı erişim yapılmaz.
* Bir sistemde hold and wait durumunun olmaması için, bir process bir kaynağa istek yaptığında başka bir kaynağa tutulması gerekir.
* Bir kaynak bir process tarafından tutuluyorsa, kaynak process tarafından serbest bırakılmadan önce boşaltılmaz.
* Circular wait durumunun önlemesi için tüm kaynaklar sıralanır ve bir process kaynak isteğini artan sırada yapılabilir.
* Deadlock önleme algoritmaları kaynak erişimi sınırlanır.
* Eğer bir sistem, kaynakları process'lere belirli bir sırada (safe sequence) maksimum ihtiyaçları kadar alabiliyorsa ve deadlock oluşmuyorsa bu durum safe state olarak adlandırılır.
* Eğer bir sistemde safe sequence varsa sistem safe state durumundadır.
* Bir safe sequence yoksa unsafe durumundadır, deadlock oluşabilir.
* Her kaynaktan bir örnek olan sistemlerde resource-allocation graf değiştirilerek kullanılabilir

Subject : Date : ...../...../.....

* Resource-allocation algoritması, aynı kaynaktan birden fazla olan sistemlerde uygulanmaz.
* Aynı kaynaktan birden fazla olan sistemlerde banker algoritması kullanılabilir
* Bir process bir grup kaynak istediğinde, sistem bu atamanın yapılması halinde safe state'in korunup korunmadığına bakar.
** Banker algoritması aşağıdaki veri yapılarını kullanır
- Available: Bir vektördür, sistemde boştaki tüm kaynakların sayısını tutar
- MAX: n*m matristir, Her process'in maksimum kaynak talebini tutar
- Allocation: n*m matristir. Her process'in kullanmakta olduğu miktarını tutar.
- Needi n*m matristir. Her process'in kalan ihtiyaç miktarını tutar.

* Bir sistem deadlock önleme veya deadlock kaçınma algoritmasına sahip değilse ↓ deadlock oluşabilir.
** Bu tür bir sistem, deadlock probleminin çözüm için aşağıdakileri sağlayabilir
- Sistem deadlock durumunda olup olmadığına karar verecek bir algoritma
- Deadlock durumunun çözülmesi için bir algoritma

* Resource-allocation grafı kullanarak wait-for grafı oluşabilir
** Her kaynaktan birden fazla örnek olması, Bu tür sistemlerde birden fazla veri yapısı kullanılır.

Allocation: n*m, Her process için mevcut atanmış kaynak sayısını tutar
Available: Her kaynak türü için boştaki kaynak sayısını tutar.
Request: n*m matristir, Her process'in mevcut kaynak sayısını tutar.

** process termination yönteminde 2 tür çözüm olabilir:
- Tüm deadlock durumundaki process'ler sonlandırılır.
- Deadlock döngüsü ortadan kalkıncaya kadar her adımda bir deadlock process sonlandırılır.

* Deadlock cycle ortadan kalkıncaya kadar process'ler atanmış kaynaklar alınarak başka process'lere atanır.

Selecting a victim: Kaynak veya process seçiminde maliyet hesaplanabilir
Rollback: Bir process'ten bir kaynağı alınca, normal çalışmasına devam edip etmeyeceğine karar verilir.
Starvation: Eğer bir process sürekli kesiliyorsa görevini hiçbir zaman tamamlayabilir.

Subject: Date: ..../..../....

* Main memory ve general purpose register'lar CPU tarafından adreslenen genel amaçlı kayıt alanlarıdır.
* CPU'nun ihtiyaç duyduğu veri veya komut hafızada değilse, öncelikle hafızaya alınmalıdır.
* CPU içerisindeki register'lara genellikle cycle ile erişilebilmektedir
* Her process kendisine ait ayrı bir hafıza alanına sahiptir.
* Bir process başlangıç adresi (base register) ve boyutu (limit register)
* Bir program disk üzerinde binary dosya olarak bulunur.
* Bir programın çalıştırılabilmesi için hafızaya alınması gereklidir
* Bir process disk üzerinden hafızaya alınmak için kuyruğa alınır (input Queue)
* Processin çalışması tamamlandığında kullandığı hafıza alanı boşaltılır
** komutların veya verilen hafıza adreslerine bağlanması (binding) farklı şekillerde olabilir.

Compile-time: compile aşamasında kodun hafızada yerleşeceği yer bilinirse mutlak code (absolute code) oluşturulabilir.

load-time: compile aşamasında programın yerleşeceği yer bilinmiyorsa, derleyici yeniden yerleştirebilir. (relocatable code) kod oluşturur.

Execution-time: Eğer program çalışması sırasında bir segment'ten başka bir segment'e geçerse, adress binding run time'da yapılır

* CPU tarafından oluşturulan adres mantıksal adres (logical address) adlan.
* Hafıza biriminin gördüğü adres fiziksel adres (physical address) olarak adlan.
* Compile time veya load time adres binding işlemleri mantıksal veya fiziksel adres üretir.
* Execution-time adres binding ise sanal adrestir. (virtual address)
* Runtime'da sanal adresin fiziksel adrese dönüştürmesi donanım bileşeni (Memory-management unit) tarafından yapılır.
* Bir process geçici olarak diske (backing store) aktarılabilir ve tekrar hafızaya alınabilir (swapping)
* Ready Queue: CPU'da çalıştırılmak üzere bekleyen process'leri tutar. CPU scheduler bir process'i çalıştırmaya karar verildiğinde dispatcher'ı çağırır.
* Dispatcher çalışacak process'in hazır kuyruğunda olup olmadığını kontrol eder ve kuyrukta ise çalıştırır, kuyrukta değilse ve hafızada yeterli yer yoksa başka bir process'i hafızadan atar (swap out) ve istenen process'i yükler (swap in)

Subject: Date: ...../...../.....

* iki process'in yer değiştirmesi context-switch işlemini gerektirir ve uzun süre alır
* process'lerin dinamik hafıza gereksinimleri için request_memory() ve release_memory() sistem çağrıları kullanılır.
* Mobil sistemler swapping işlemini desteklemez.
* Mobil cihazlar kalıcı saklama birimi olarak hard disk yerine flash
* bellek kullanır.
* Android işletim sistemi, yeterli hafıza alanı yoksa bir process'i sonlandırır ve durum bilgisi flash belleğe kaydeder.
* bitişik hafıza atama yönteminde hafıza iki parçaya ayrılır.
  - işletim sisteminin yerleştiği kısım
  - kullanıcı process'lerin yerleştiği kısım.
* Hafızada çok sayıda sabit boyutta küçük parçaya ayrılabilir (fixed size partitions) ve her parça bir process'i içerebilir (multiple partition)
* Multiprogramming sistemlerde eş zamanlı çalışan program sayısı partition sayısına bağlıdır.
* * Dynamic storage allocation problemi için 3 farklı çözüm kullanılabilir.
  First fit: yeterli boyuttaki ilk boş alan atanır
  Best fit: yeterli boyutta en küçüğü seçilir.
  Worst fit: yeterli boyuttaki alanların en büyüğü seçilir.
* * first fit yöntemi best fit ve worst fite göre daha kısa sürede atama gerçekleştirmektedir.
* * process'ler hafızaya yüklenirken ve atanırken hafıza alanları süreksiz parçalanır (fragmentation)
  first fit ile yapılan istatistiksel analize göre, N tane kullanılmış blok için N/2 tane boş blok oluşur, Bu durumda hafızanın 1/3 kısmı kullanılmaz
  Buna %50 kuralı denir.
* segmentationi yaklaşımda, her segment bir isim ve uzunluğa sahiptir.
  Bir mantıksal adres, segment adı ile offset değerini belirler.
* * Bir C derleyicisi aşağıdaki için ayrı ayrı segment oluşturabilir:
  - Program kodu, • Global değişkenler
  - Heap (nesneler yerleştirilir)
  - stack (thread'ler kullanır, lokal değişkenler, call/return)
  - Standart C kütüphanesi

Subject : Date : ..../..../....

* paging ile segmentation'da olduğu gibi process'lere bitişik olmayan hafıza adresleri atanabilir.
* paging yönetiminde:
  - Fiziksel hafıza çerçeve adı verilen küçük bloklara bölünür.
  - Mantıksal hafıza ise aynı boyutta page adı verilen bloklara bölünür.
* Bir process çalıştırıcağı zaman kaynak kodu disketten alınarak hafızadaki çerçevelere yerleştirilir.
* CPU tarafında oluşturulan her adres iki parçaya ayrılır: page number (p), page offset (d)
* page number (p) page table içerisindeki indeks değeri için kullanılır.
* Mikroişlemcilerde, küçük boyutta ve hızlı donanımsal önbellek (translation look-aside buffer) ile bu problem çözülür.