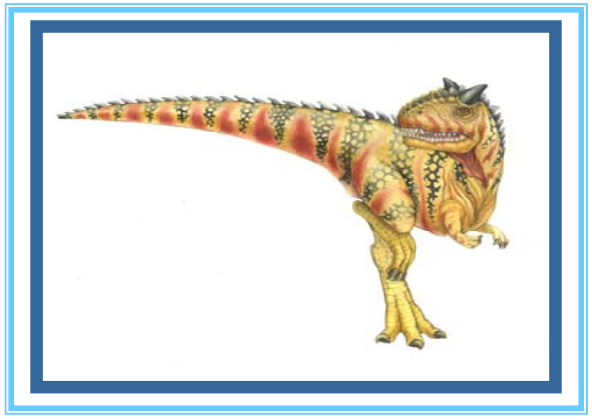
Bölüm 3: Prosesler



Bölüm 3: Prosesler

- Proses Kavramları
- Proses Planlama
- Proses Üzerinde İşlemler
- Prosesler Arası İletişim
- IPC (Prosesler Arası İletişim) Örnekleri
- İstemci-Sunucu Sistemleri Arasındaki İletişim

Bölüm Hedefleri

- Proses mantığını tanıtmak – çalışan bir program
- Proseslerin yapısını anlatmak (Proseslerin zamanlanması, oluşturulması ve sonlandırılması, iletişimi vb.)

- İstemci – sunucu sistemlerin iletişimini tarif etmek

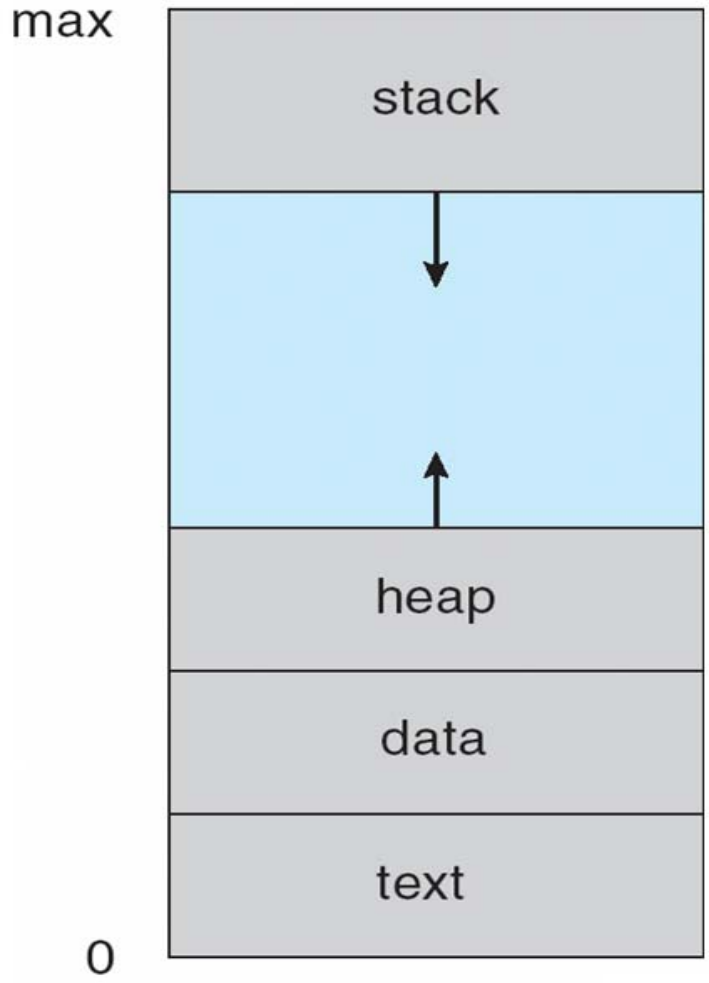
Proses Kavramları

- Bir işletim sistemi çeşitli programlar yürütür:
  - Batch sistemi – görevler (jobs)
  - Zaman paylaşımlı sistemler – kullanıcı programları yada görevleri.
- Ders kitaplarında görev/süreç (job / proses) terimleri neredeyse birbirlerinin yerine kullanılır.
- Proses, yürütülen bir programdır denilebilir ve prosesler sıralı bir biçimde yürütülmelidir.
- Bir proses aşağıdakileri alanları içerir:
  - Program Sayacı
  - Yığın (stack)
  - Veri bölümü (data section)

Proses

- Bölümleri,
  - **program kodu** (text section),
  - **Mevcut işlem**, program sayacı ve diğer kaydediciler tarafından tutulur.
  - **Yığın** (stack) geçici veriyi içerir.
    - Fonksiyon parametreleri, geri dönüş değerleri, yerel değerler
  - **Veri bölümü** (data section) global değişkenleri içerir.
  - **Bellek yığını** (heap) çalışma zamanında ayrılmış bilgileri içerir.
- Program pasif, Proses ise aktiftir
  - Program çalıştırılabilir halde belleğe yüklendiğinde proses halini alır.
- Programın çalıştırılması komut satırından komutun girilmesi, grafik arayüzde program ikonu üzerine tıklanması vb. ile başlar.
- Bir program birden fazla proses içerebilir.
  - Örneğin birden fazla kullanıcının aynı programı çalıştırması.

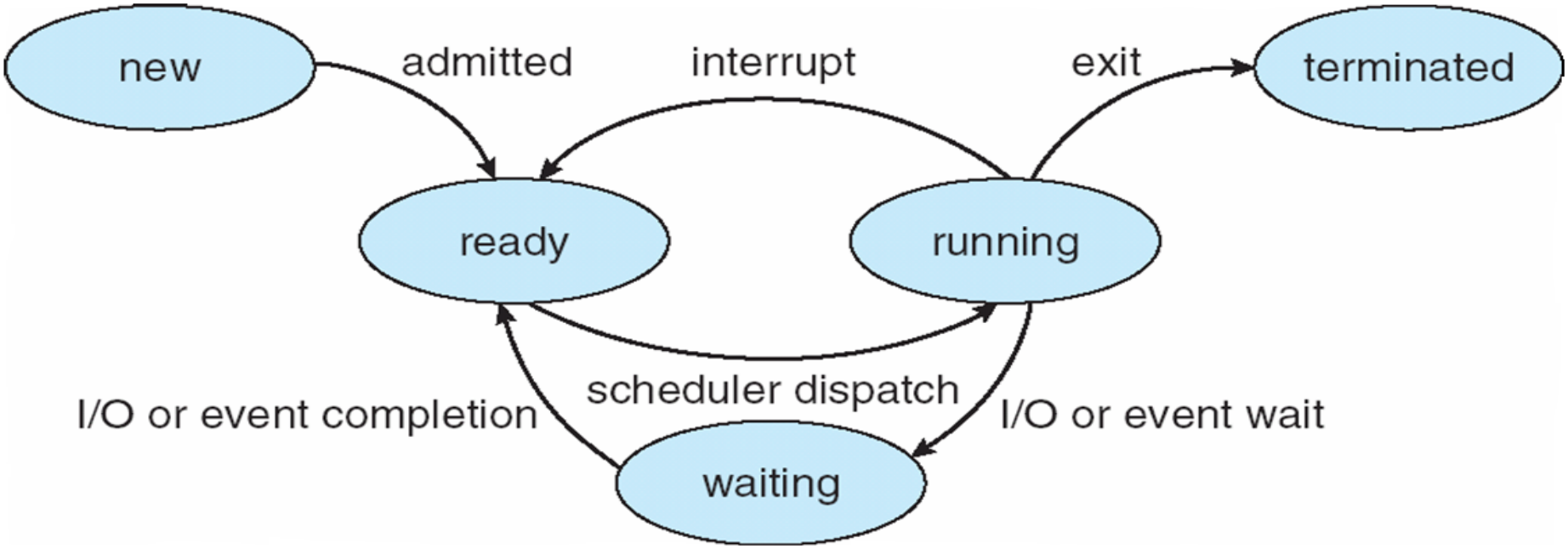
Bellekteki Bir Proses



Proses Durumları

- Çalışma anında proseslerin durumları değişir.
  - **yeni**: Yeni bir proses oluşturuluyor.
  - **çalışıyor**: İşlemler gerçekleştiriliyor.
  - **bekleme**: Proses bazı olayların gerçekleşmesini beklemektedir.
  - **hazır**: Proses, işlemciye aktarılmayı beklemektedir.
  - **sonlandırılmış**: Prosesin yürütülmesi tamamlandığını belirtir.

Proses Durum Diyagramı

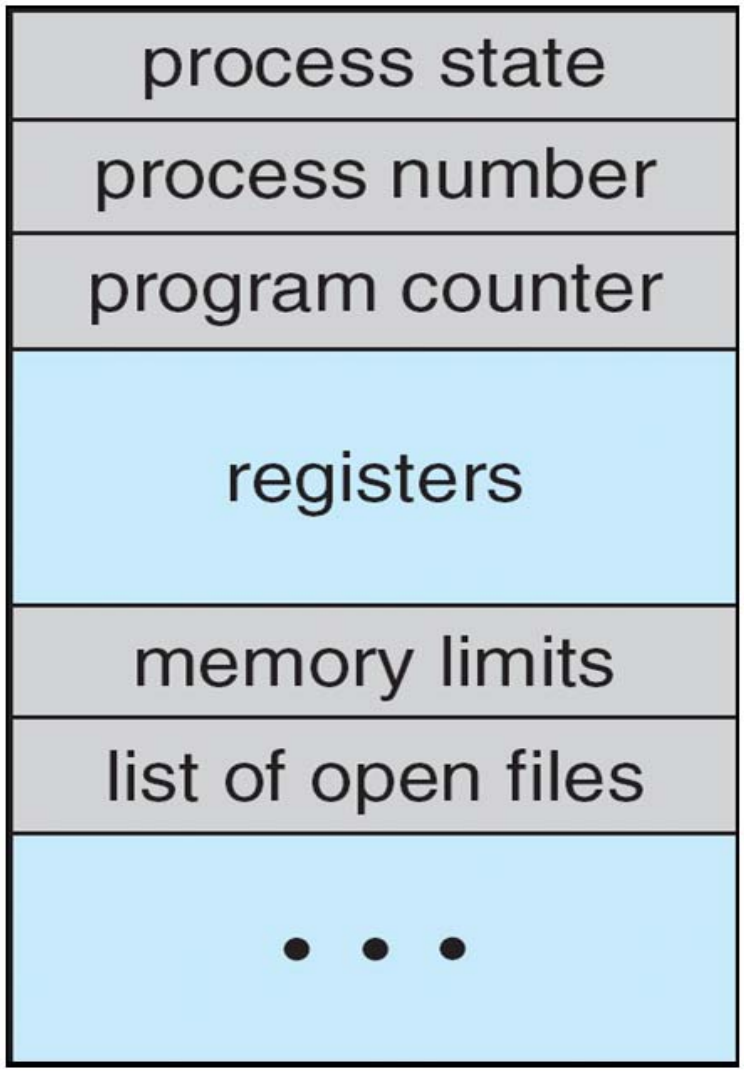


Proses Kontrol Bloğu (PCB)

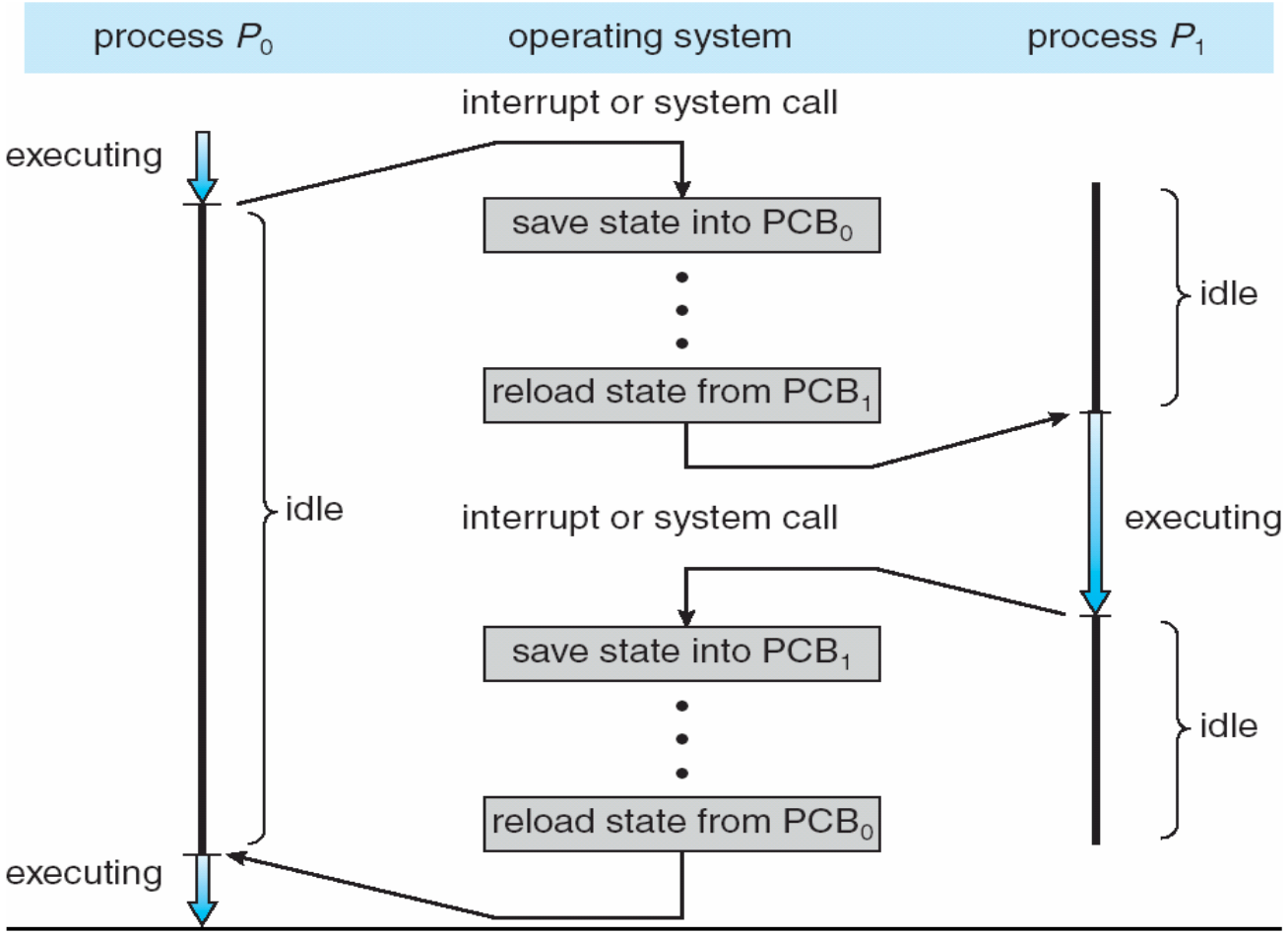
Her proses aşağıdaki bilgileri içerir :

- Proses durumu
- Program Sayacı
- CPU kayıtları
- CPU planlama bilgisi
- Bellek yönetim bilgisi
- Hesaplama bilgisi
- I/O durum bilgisi

Proses Kontrol Bloğu (PCB)



CPU'da Proses Değişimi



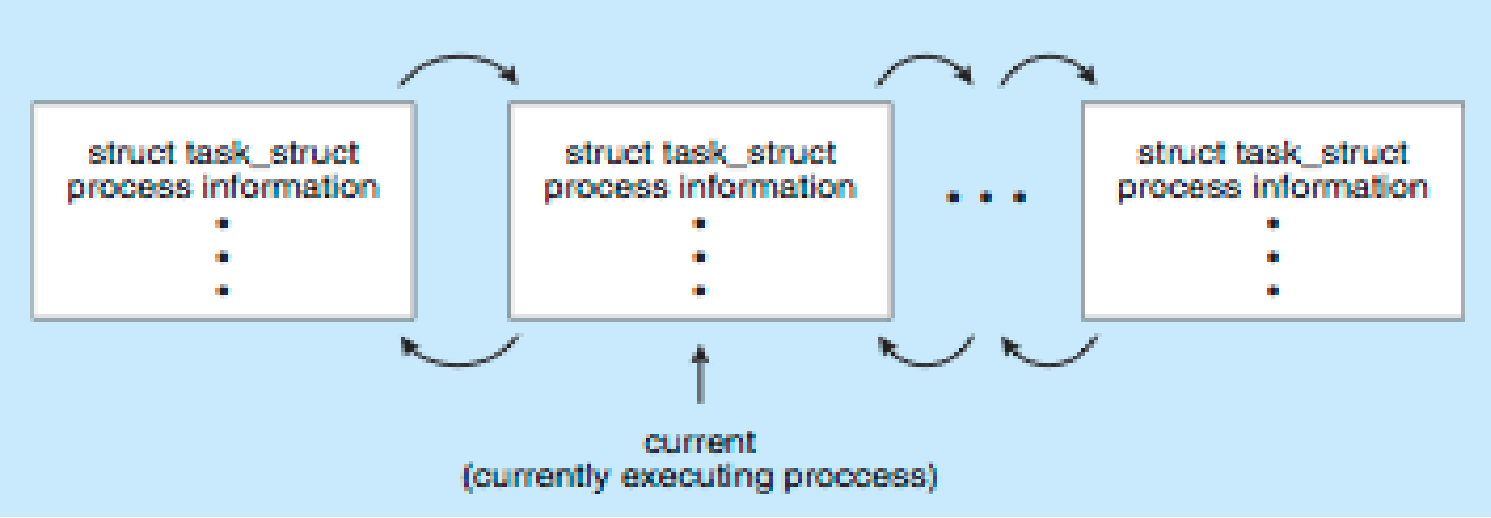
Proses Planlama

- İşlemcinin maksimum kullanımı adına, prosesler arası geçiş ve zaman paylaşımı çok hızlı gerçekleşir.
- **Proses planlayıcı,** işlenecek prosesi işleme hazır prosesler arasından seçer.
- Prosesler aşağıdaki **sıralama kuyruklarında** tutulur.
  - **İş kuyruğu**– sistemdeki tüm prosesler kümesi
  - **Hazır kuyruğu**– ana bellekte bulunan, hazır ve işleme girmeyi bekleyen prosesler kümesi
  - **Aygıt kuyrukları**– I/O aygıtlarından gelecek mesajı bekleyen prosesler kümesi
  - Prosesler kuyruklar arasında geçiş yapabilir.

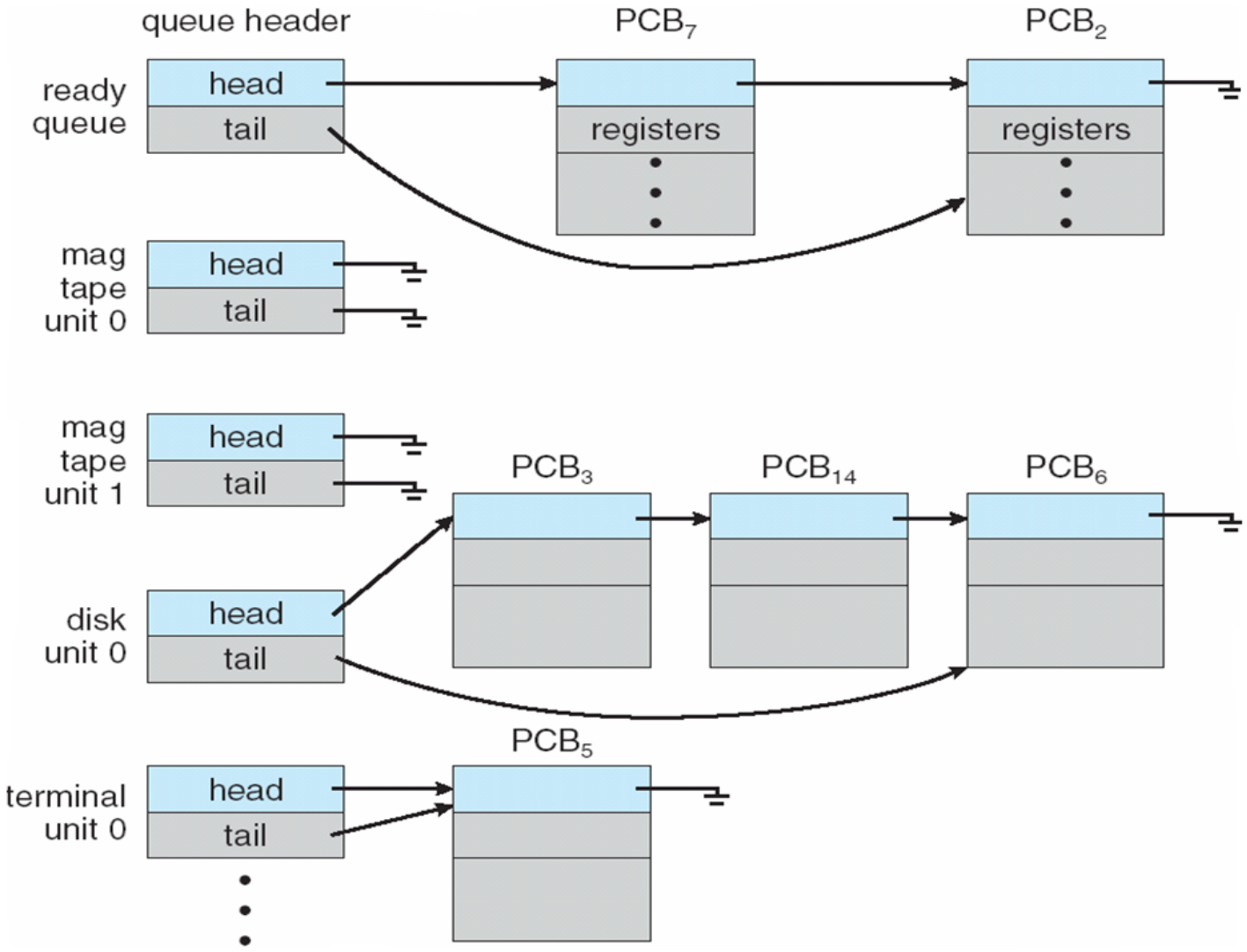
Proseslerin Linux'ta Temsili

- Represented by the C structure `task_struct`

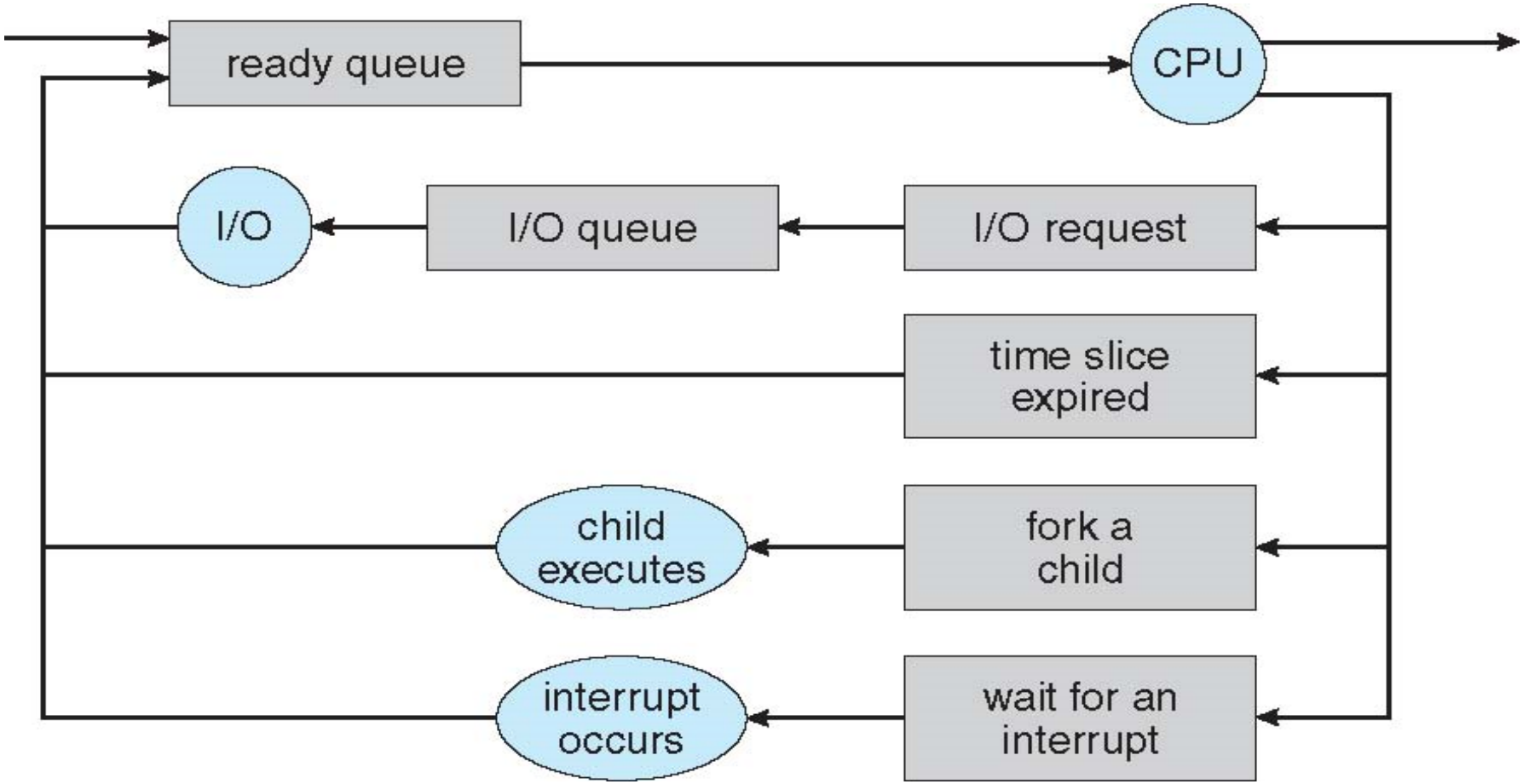
```
pid t pid; /* Proses tanımlayıcı*/
long state; /* Prosesin durumu*/
unsigned int time slice /* planlama bilgisi*/ struct task struct *parent; /* bu
ebeveyn Proses*/ struct list head children; /* bu çocuk Proses*/ struct files struct
*files; /* açık dosyaların listesi*/ struct mm struct *mm; /* address space of this
pro */
```



Hazır Kuyruğu ve Çeşitli I/O Aygıt Kuyrukları



Proses Sıralama Diyagramı



Sıralayıcılar

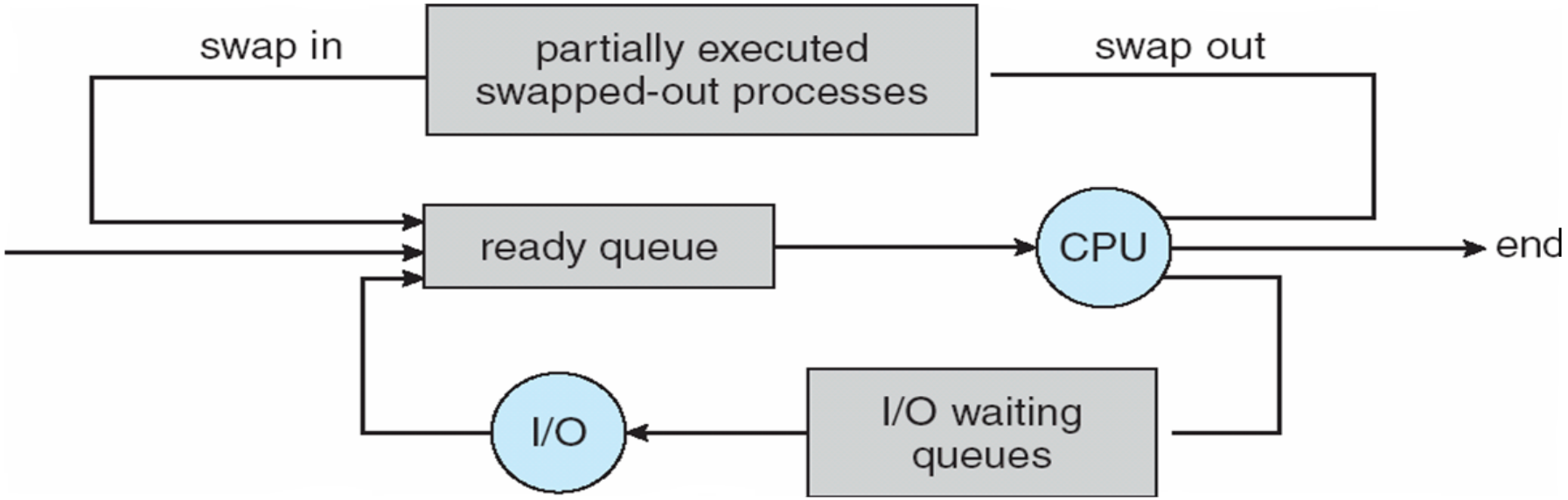
- **Uzun vadeli sıralayıcılar / iş planlayıcı** – Hazır kuyruğundan getirilen prosesler arasından seçim yapar.
- **Kısa vadeli sıralayıcılar / CPU planlayıcı** – Prosesin bir sonraki işlemini ve işlem için ayrılan alanın tahsisini seçer.
  - Bazen sistemde sadece bir sıralayıcı vardır

Sıralayıcılar(Devam)

- Kısa vadeli sıralayıcılar çok sık çağrılırlar.
  - (milisaniyeler düzeyinde) ⇒ (hızlı olmalı)
- Uzun vadeli sıralayıcılar çok seyrek çağrılırlar
  - (saniyeler, dakikalar düzeyinde) ⇒ (yavaş olabilir)
- Uzun vadeli sıralayıcılar, *çoklu programlama* derecesini kontrol eder.

- Prosesler her iki biçimde de tanımlanmış olabilir:
  - **I/O-bağımlı Proses** –giriş/çıkış hesaplaması için daha fazla zaman harcar, çok az CPU sarfiyatı
  - **CPU-bağımlı Proses** – Hesaplamalar için zaman harcar, çok fazla CPU sarfiyatı

Orta vadeli sıralayıcıların eklenmesi



Bağlam Değiştirme (Context Switch)

- CPU diğer prosese geçtiği zaman, sistem mutlaka eski prosesin durumunu kaydetmeli ve yeni prosesin daha önce kaydedilmiş durumunu yüklemeli

- **Bağlam** PCB'dir.
- Aşırı sayıda bağlam değiştirme; sistem geçişler sırasında kullanışlı olmaz
  - Daha karmaşık OS ve PCB -> daha uzun bağlam değişimi

- Donanım desteği zamana bağlıdır.
  - Bazı donanımlar CPU başına birden fazla kaydedici sağlar -> birden fazla bağlam bir kerede yüklenir

Proses Oluşturulması

- Ebeveyn Proses, çocuk prosesleri oluşturur. Bu şekilde ağaç yapısı meydana gelir.
- Genelde prosesler**, bir proses kimlik numarası** (Proses identifier - pid) ile tanımlanır ve yönetilir.

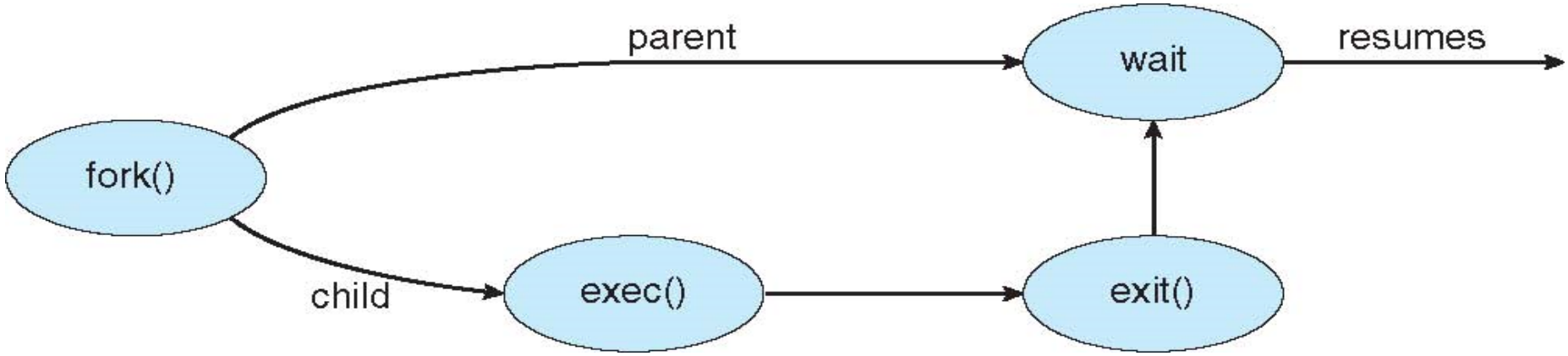
- Kaynak Paylaşımı türleri:
  - Ebeveyn ve çocuk prosesler tüm kaynakları paylaşır.
  - Çocuk prosesler ebeveyn prosesin kaynaklarını kullanır.
  - Ebeveyn ve çocuk hiçbir kaynağı paylaşmaz.

- Uygulama:
  - Ebeveyn ve çocuk proses eşzamanlı çalışır.
  - Ebeveyn proses, çocuk proses sonlanana kadar bekler.

Proses Oluşturulması (Devam)

- Adres alanı
  - Çocuk proses, ebeveyn prosesin alanını kopyalar .
  - Çocuk prosese bir program yüklenmiş olur.
- UNIX örnekleri :
  - **fork** işlevi yeni proses oluşturulması için sisteme çağrıda bulunur.
  - **exec** sistem çağrısı Prosesin bellek alanını yeni bir program ile değiştirmek için bir fork sonrası çağrılır.

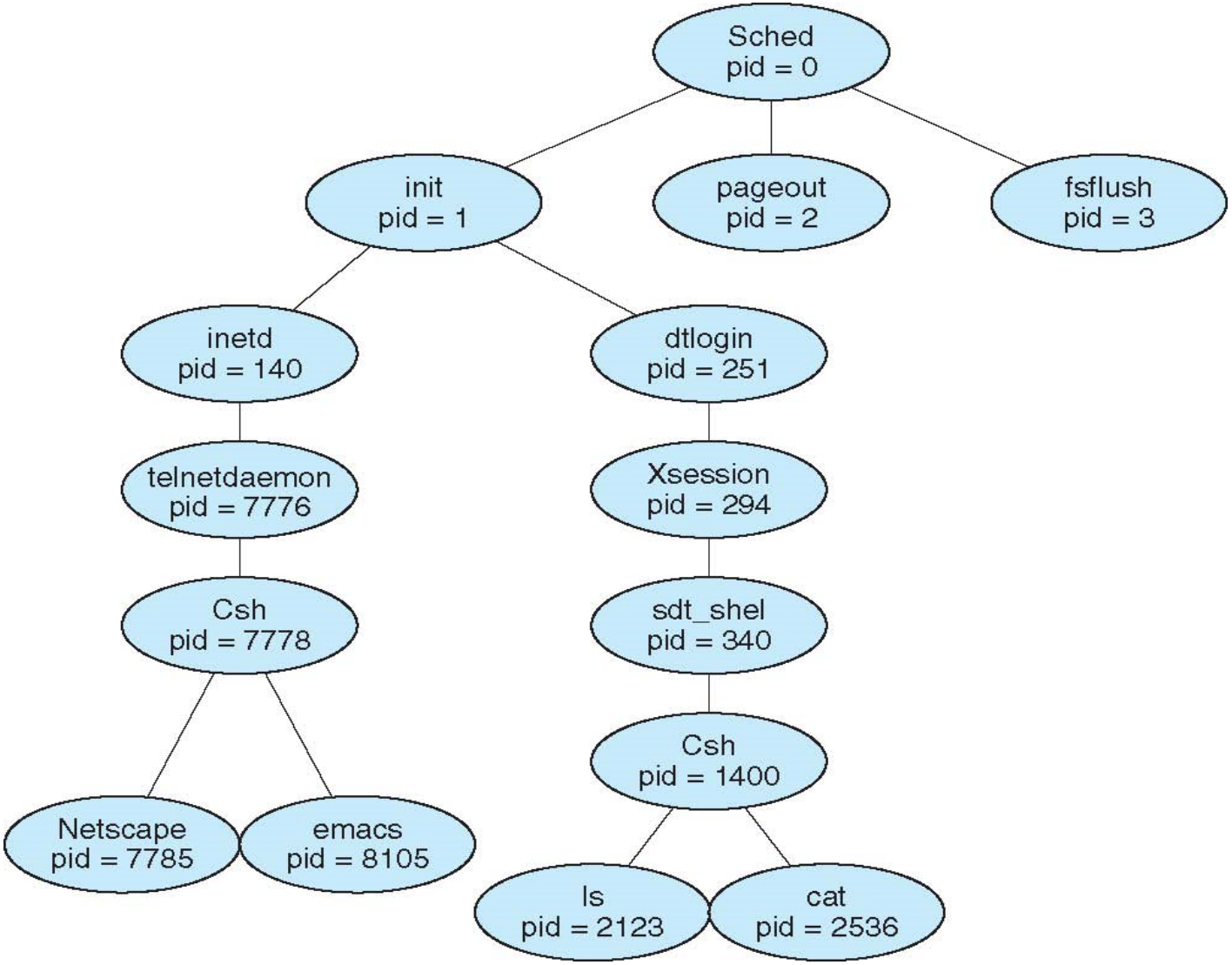
Proses Oluşturulması



Fork İşlemi Yapan C Programı

```
#include <sys/types.h>
#include <studio.h>
#include <unistd.h>
int main()
{
pid_t  pid;
      /* fork another Proses */
      pid = fork();
      if (pid < 0) { /* error occurred */
            fprintf(stderr, "Fork Failed");
            return 1;
      }
      else if (pid == 0) { /* child Proses */
            execlp("/bin/ls", "ls", NULL);
      }
      else { /* parent Proses */
            /* parent will wait for the child */
            wait (NULL);
            printf ("Child Complete");
      }
      return 0;
}
```

Solaris'te Proses Ağaç Yapısı



Proses'lerin Sonlanması

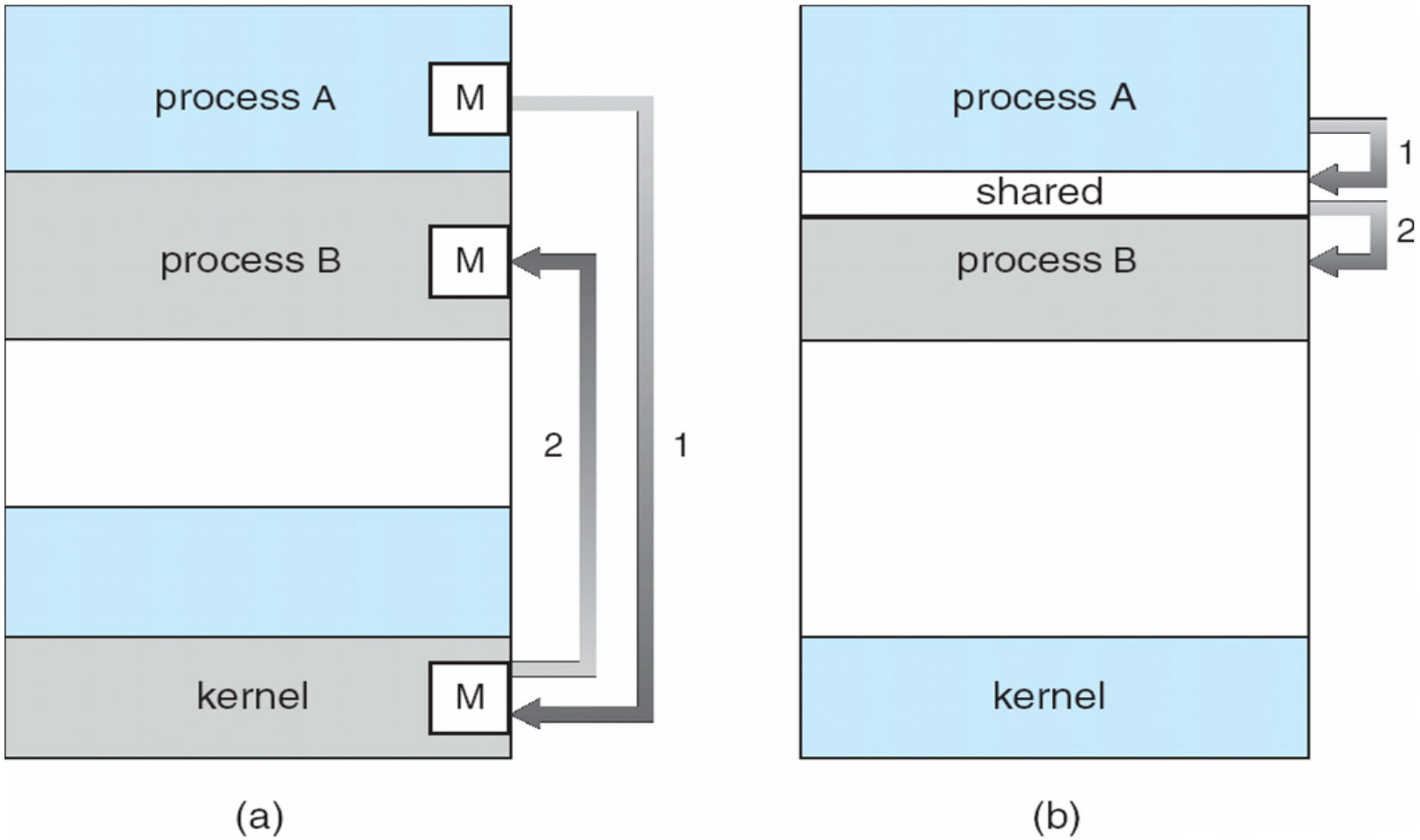
- Proses son kod ifadesinin çalıştırır ve işletim sistemine silinip silinmeyeceğini sorar (çıkış)
  - Çıkış verisi çocuktan ebeveyn prosese (**bekleyerek**)
  - Prosese ayırılan alan işletim sistemi tarafından sonlandırılır.

- Ebeveyn Proses çocuk Prosesin çalışmasını sonlandırabilir (**durdurmak**)
  - Çocuk Proses tahsis edilmiş kaynakların dışına çıkarılır.
  - Artık çocuk Prosese görev tayin etmek gerekmez.
  - Eğer ebeveyn Proses sonlandırılırsa
    - Bazı işletim sistemi ebeveyn proses sonlandırıldıktan sonra çocuk prosesin çalışmasına izin vermez
    - Tüm çocuk Prosesler sonlandırılır.- **basamaklı sonlandırma**

Prosesler Arası İletişim

- Prosesler işletim sistemi içerisinde bağımsız ya da işbirliği içinde çalışabilirler.
- İşbirliği içerisindeki prosesler veri paylaşımı da dahil olmak üzere diğer prosesleri etkileyebilir ya da diğer proseslerden etkilenebilirler.
- Proseslerin işbirliği yapma nedenleri:
  - Bilgi paylaşımı
  - Daha hızlı hesaplama
  - Modülerlik
  - Rahatlık
- İşbirliği içindeki prosesler prosesler arası haberleşmeye (Interproses communication - IPC) ihtiyaç duyarlar.
- 2 temel IPC modeli mevcuttur:
  - Paylaşılmış bellek
  - Mesajlaşma

Haberleşme Modelleri



İşbirliği içerisindeki Prosesler

- **Bağımsız** prosesler, diğer proseslerin çalışmasından etkilenemez ve diğer prosesleri etkileyemezler.
- **İşbirliği yapan** prosesler, diğer proseslerin çalışmasından etkilenebilir ve diğer prosesleri etkileyebilirler.

- Prosesler arası işbirliğinin avantajları :
  - Bilgi paylaşımı
  - Daha hızlı hesaplama
  - Modülerlik
  - Rahatlık

Üretici-Tüketici Problemi

- İşbirliği içindeki proseslere ilişkin bir paradigma: üretici proses tüketici proses tarafından kullanılmak üzere bilgi üretir.
  - *Sınırlandırılmamış tampon*: tampon için limit konulmamıştır
  - *Sınırlandırılmış tampon*: sabit bir tampon boyutu mevcuttur.

Sınırlı-Tamponlu–Paylaşımlı-Bellek Çözümü

- Paylaşılmış veri

```
#define BUFFER_SIZE 10
typedef struct {
   . . .
} item;

item buffer[BUFFER_SIZE];
int in = 0;
int out = 0;
```

- Çözüm doğru, ancak sadece BUFFER_SIZE-1 eleman kullanılabilir.

Sınırlı-Tampon – Üretici

```
while (true) {
   /* data üretilir */
   while (((in = (in + 1) % BUFFER SIZE count)  == out)
    ;   /* çalışmaz – serbest tampon yok */
    buffer[in] = item;
    in = (in + 1) % BUFFER SIZE;
}
```

Sınırlı Tampon – Tüketici

```
while (true) {
     while (in == out)
          ; // çalışmaz -- nothing to consume

     // buffer tarafından veri silinir
     item = buffer[out];
     out = (out + 1) % BUFFER SIZE;
return item;
}
```

Prosesler Arası İletişim - Mesajlaşma

- Proseslerin iletişim mekanizması diğer proseslerin işlemleriyle senkronizedir.

Mesaj sistemi – prosesler birbiri ile, paylaşılan değişkenleri kullanmadan iletişim kurar.

- IPC iki işlemi destekler :
  - **send**(*message*) – gönderilecek mesaj boyutu, sabit ya da değişken olabilir.
  - **receive**(*message*)
- Eğer *P* ve *Q* prosesleri iletişim kurmak istiyorsa, şu işlemleri yapmaları gerekir :
  - Aralarında iletişim bağlantısı var olmalıdır.
  - send /receive yardımı ile mesaj alışverişi gerçekleştirmelidirler.
- İletişim bağlantısı oluşturulması
  - fiziksel (ör., paylaşılmış bellek, donanım veriyolu)
  - mantıksal (ör., mantıksal özellikler)

Doğrudan İletişim

- Proseslerin her biri gönderici ve alıcı olarak isimlendirilmelidir :
  - **send** (*P, message*) – P prosesine mesaj gönder
  - **receive**(*Q, message*) – Q prosesinden mesaj al

- İletişim bağlantısının özellikleri :
  - Bağlantılar otomatik olarak kurulur.
  - Her bir proses çifti arasında tam olarak bir bağlantı vardır.
  - Bir link 2 proses ile ilişkilendirilebilir.
  - Bağlantı tek yönlü olabilir, ancak genellikle iki yönlüdür.

Doğrudan Olmayan İletişim

- Mesajlar port veya posta kutularından alınır veya buralara gönderilir.
  - Her posta kutusu tek bir tanımlayıcıya sahiptir
  - Prosesler paylaşılmış bir posta kutusuna sahipse iletişim kurabilirler.

- İletişim bağlantısı özellikleri şunlardır :
  - Bağlantı, prosesler arası paylaşılmış bir posta kutusu var ise kurulur.
  - Bir bağlantı ikiden fazla proses ile ilişkilendirilebilir.
  - Her bir proses çifti birden fazla bağlantıya sahip olabilir.
  - Bağlantı tek yönlü ya da çift yönlü olabilir.

Doğrudan Olmayan İletişim

- İşlemler
  - Yeni bir posta kutusu oluştur,
  - Posta kutusu aracılığıyla mesaj gönder ve al.
  - posta kutusunu yok et.

- İletişim basitçe şu şekilde gerçekleşir:

  send(*A, message*) – A'nın posta kutusuna bir mesaj gönder

  receive(*A, message*) – A'nın posta kutusundan bir mesaj al.

Doğrudan Olmayan İletişim

- Posta kutusu paylaşımı
  - P_1, P_2, and P_3 Prosesleri A posta kutusunu paylaşıyor.
  - P_1 mesaj gönderiyor; P_2 ve P_3 mesajı alıyor.
  - Mesajı hangisi almıştır?
- Çözüm:
  - Bir bağlantının en fazla iki proses ile ilişkilendirilmesine izin verir.
  - Bir seferde yalnızca bir proses yürütmesine izin verir.
  - Sistemin rastgele bir alıcı seçimine izin verir. Gönderici, alıcının kim olduğunu bildirir.

Senkronizasyon

- Mesaj iletimi engelli ya da engelsiz olabilir.
- **Engelli**, senkron iletim olarak düşünülebilir.
  - **Engelli** gönderim**,** mesaj alınana kadar gönderici engellenir.
  - **Engelli** alım**,** mesaj hazır olana kadar alıcı engellenir.

- **Engelsiz,** asenkron iletim olarak düşünülebilir.
  - **Engelsiz** gönderim, mesaj yollanır ve devam edilir.
  - **Engelsiz** alım, hazır mesaj varsa alır yoksa boş-null değer alır.

Tamponlama

- Bağlantıyla ilişkilendirilmiş mesaj sırası, şu 3 yolla düzenlenir:
  1. Sıfır kapasite – 0 mesaj
     Gönderici, alıcıyı beklemelidir (Buluşma).
  2. Sınırlı kapasite –n adet mesajın sonlu bir uzunluğa sahip olması
     Gönderici, bağlantı dolu ise beklemelidir.
  3. Sınırsız kapasite – sonsuz uzunluk
     Gönderici hiçbir zaman beklemez.

IPC Sistem Örnekleri - POSIX

- POSIX Paylaşılmış bellek
  - Proses öncelikle paylaşılmış bellek alanı oluşturur.

```
segment id = shmget(IPC PRIVATE, size, S IRUSR | S
   IWUSR);
```

  - Proses paylaşılmış belleğe erişmek istemektedir.

```
shared memory = (char *) shmat(id, NULL, 0);
```

  - Şimdi, proses paylaşılan belleğe yazabilir.

```
sprintf(shared memory, "Writing to shared memory");
```

  - İşlem tamamlandığında önceden ayırılan bellek alanı geri alınabilir.

```
shmdt(shared memory);
```

IPC Sistem Örnekleri - Mach

- Mach iletişimi mesaj tabanlıdır.
  - Hatta sistem çağrıları dahi birer mesajdır.
  - Her görev iki posta kutusu oluşturur – Kernel ve Notify
  - Sadece 3 sistem çağrısı mesaj transferine ihtiyaç duyar.

`msg_send(), msg_receive(), msg_rpc()`

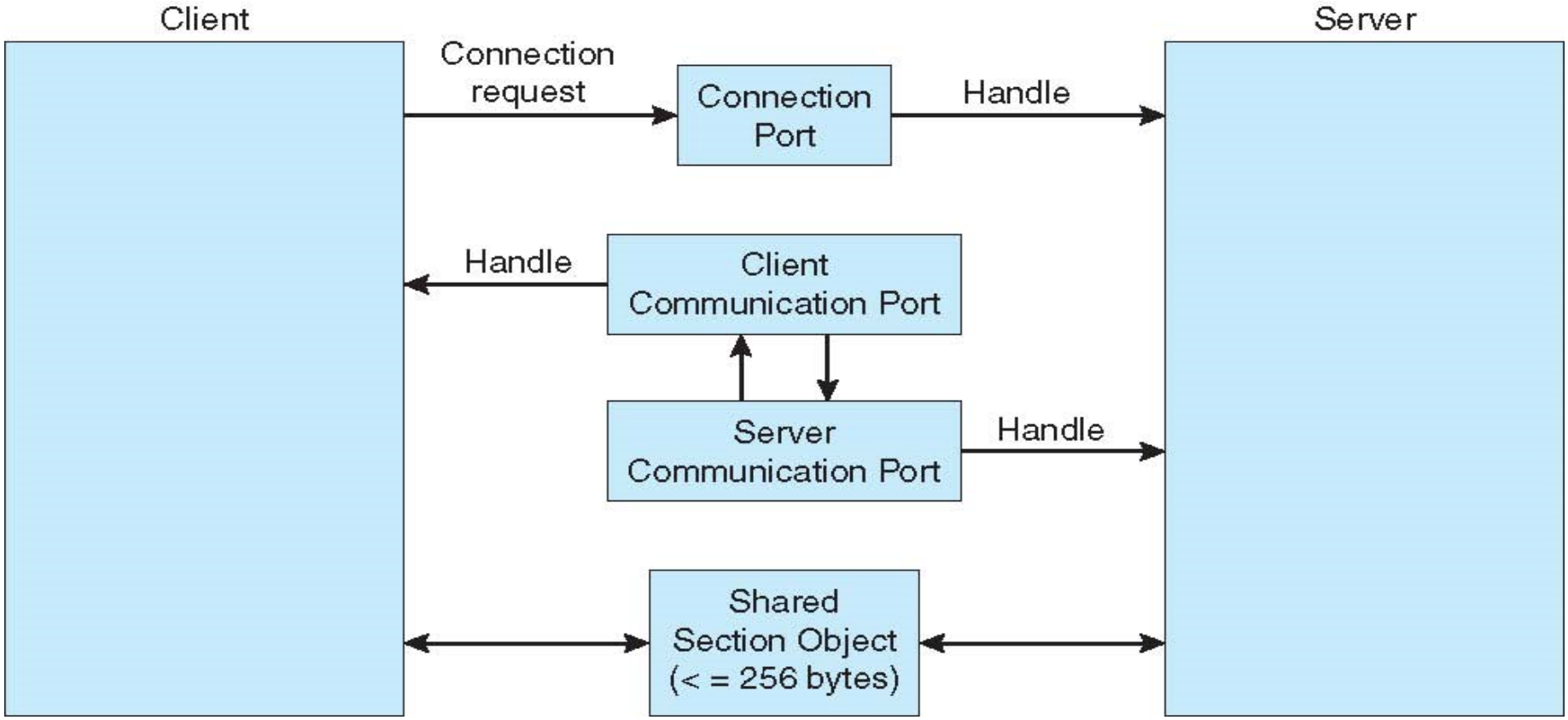
  - Mailboxların oluşturulma sebebi iletişimde mailboxlara ihtiyaç duyulmasıdır.

`port_allocate()`

IPC Sistem Örnekleri – Windows XP

- Mesaj iletimi yerel prosedür çağrı **(local procedure call - LPC)** birimi aracılıyla yönetilir
  - Yalnızca prosesler ve benzeri sistemler arasında çalışır.
  - İletişim kanalları kurmak ve sürdürmek için portları (posta kutuları gibi) kullanır.
  - Haberleşme aşağıdaki gibi çalışır:
    - İstemci altsistemi bir bağlantı nesnesi açar.
    - İstemci bağlantı isteği gönderir.
    - Sunucu iki özel iletişim portu oluşturur ve bunlardan birini istemciye gönderir.
    - İstemci ve sunucu mesajları göndermek, almak ve cevapları dinlemek amacıyla karşılıklı portlar kullanır.

Windows XP'de Yerel İşlem Çağrısı



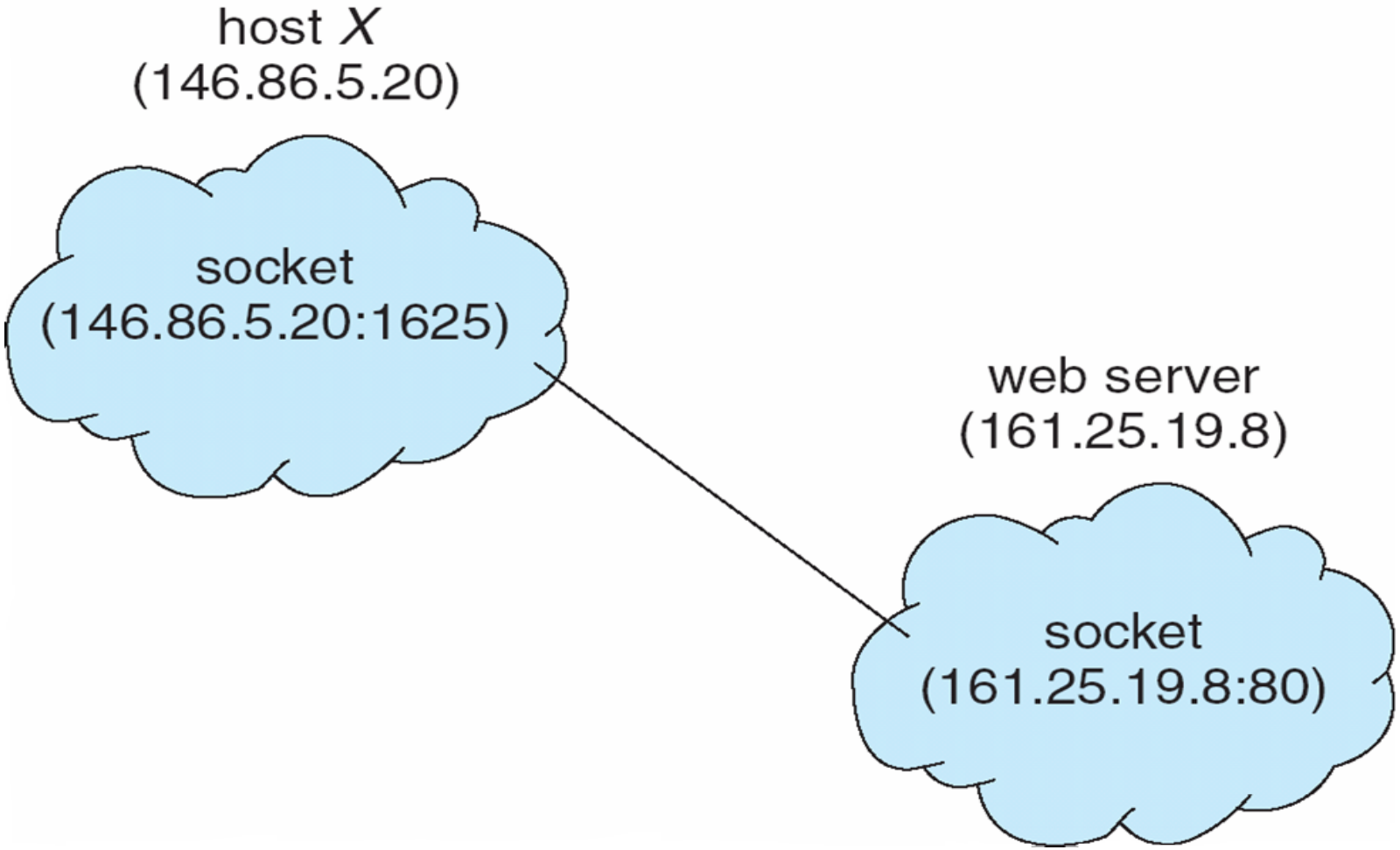
İstemci – Sunucu Sistemlerinde İletişim

- Soketler
- Uzaktan Prosedür Çağrıları
- Tüneller- pipes
- Uzaktan Metot Çağrıları (RMI - Java)

Soketler

- Soket, bir iletişimin bitiş noktası olarak tanımlanabilir.
- IP adresinin ve portun birleşimidir.
- **161.25.19.8:1625** soketi, **1625** portu ve **161.25.19.8** sunucusu demektir.
- İletişim, bir çift soket arasında meydana gelir.

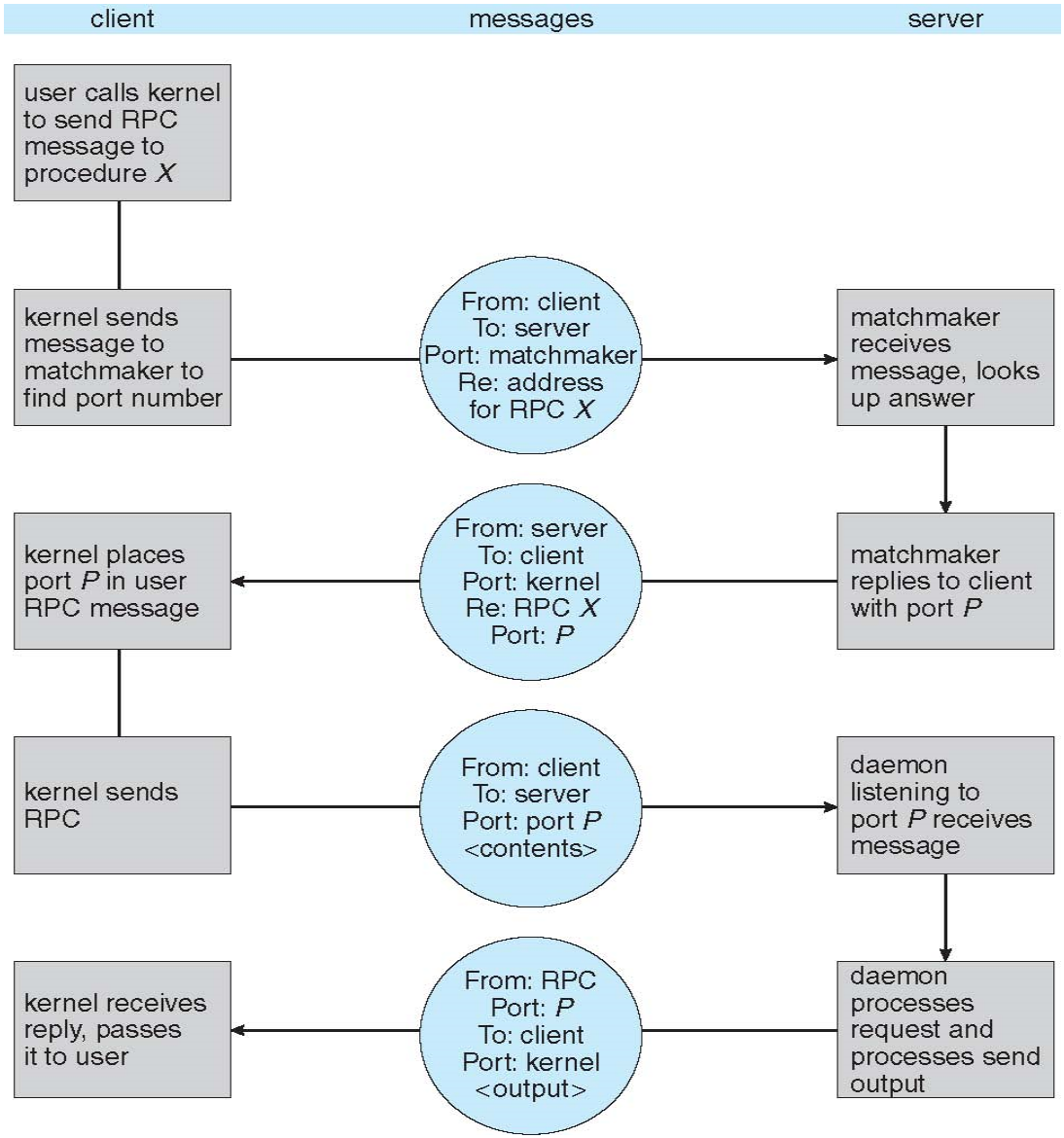
Soket İletişimi



Uzak Yordam Çağrısı

- Uzak prosedür çağrısı (Remote procedure call - RPC), yordam çağrılarını bağlı sistemler üzerindeki işlemlere ayırır.
- **Stub** – sunucu üzerindeki gerçek prosedür için istemci tarafındaki aracı
- İstemci tarafındaki stub, sunucunun yerini belirler ve parametreleri yönlendirir.
- Sunucu tarafındaki stub, mesajı alır, yönlendirilmiş parametreleri açar ve yordamı sunucu üzerinde uygular.

RPC'nin Çalışma Prensibi



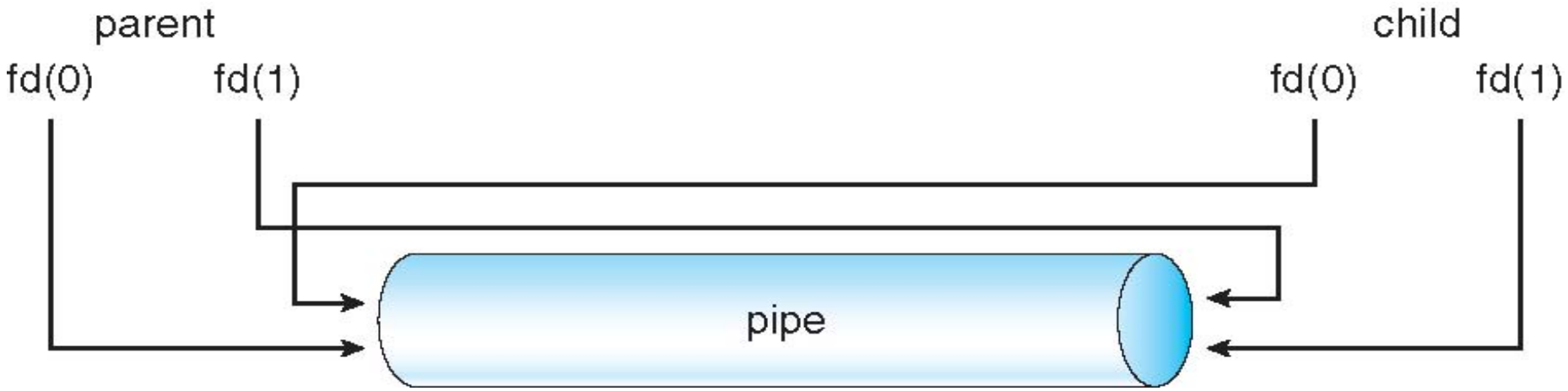
Tüneller- Pipes

- İki proses arasında iletişime izin veren yapıdır.
- **Sorunlar:**
  - İletişim tek yönü mü, çift yönlü müdür?
  - İletişim iki yönlü ise yarı dubleks mi çalışır, yoksa tam dubleks mi çalışır?
  - İletişim halindeki prosesler arasında bir ilişki (ebeveyn-çocuk) olmalı mıdır?
  - Tüneller ağ üzerinden kullanılabilir mi?

Sıradan Tüneller

- **Sıradan tüneller,** standart üretici-tüketici tipi iletişime izin verir.
- Üretici bir uçtan yazar (tünelin yazma ucu)
- Tüketici diğer ucundan okur (tünelin okuma ucu)
- Sıradan tüneller bu nedenle tek yönlü iletişim sağlar.
- Haberleşen prosesler arasında ebeveyn-çocuk ilişkisi gerekir.

Sıradan Tüneller



Adlandırılmış Tüneller

- Adlandırılmış tüneller, sıradan olanlardan daha güçlüdür.
- İletişim çift yönlüdür.
- Haberleşen prosesler arasında ebeveyn-çocuk ilişkisi gerekli değildir.
- Birden fazla proses, kullanabilir.
- UNIX ve Windows işletim sistemlerince desteklenir.

Bölüm 3 - Son