Bölüm 15: İşletim Sistemi Tasarımı

İşletim Sistemleri

İşletim Sistemi Hedefleri

- Soyutlamalar tanımlar. (*abstractions*)
- İlkel işlemler sağlar. (*primitive*)
- İzolasyon sağlar. (*isolation*)
- Donanımı yönetir. (*hardware*)

İşletim Sistemleri Tasarlamanın Zorlukları

- İşletim sistemleri son derece büyük programlar.
- Eşzamanlılıkla uğraşmak gerekir (*concurrency*).
- Düşmanca davranan kullanıcılarla uğraşmak zorunda.
- Kullanıcı,
  - bilgilerinin ve kaynaklarının bir kısmını,
  - seçilen kullanıcılarla paylaşmak ister.
- Önceki işletim sistemleriyle geriye dönük uyumlu olmalı.
- Uzun süre bakım onarım ihtiyacı.

Arayüz Tasarımı için Yol Gösterici İlkeler

- **Basitlik**,
  - Eklenecek bir şey kalmadığında değil,
  - Çıkarılacak bir şey kalmadığında mükemmelliğe ulaşılır.
- **Bütünlük**,
  - Her şey olabildiğince basit olmalı, ancak daha basit olmamalıdır.
- **Yeterlik**,
  - Bir özellik verimli kullanılamıyorsa, muhtemelen sahip olmaya değmez.

Yürütme Paradigmaları

- (a) Algoritmik kod. (b) Olaya dayalı kod. (*event driven*)

```
main( )
{
        int ... ;

        init( );
        do_something( );
        read(...);
        do_something_else( );
        write(...);
        keep_going( );
        exit(0);
}
```

(a)

```
main( )
{
        mess_t msg;

        init( );
        while (get_message(&msg)) {
                switch (msg.type) {
                        case 1: ... ;
                        case 2: ... ;
                        case 3: ... ;
                }
        }
}
```

(b)

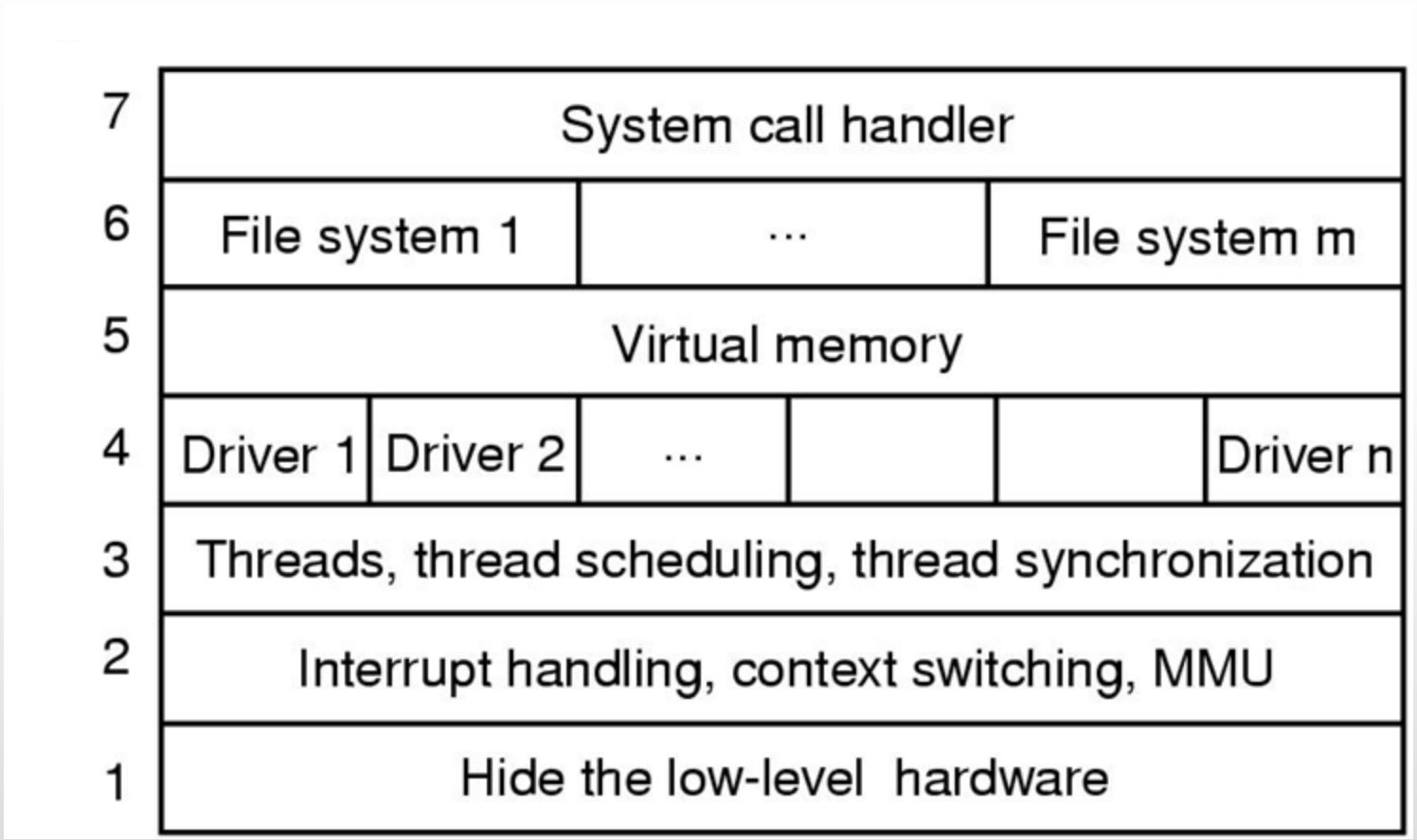
Sistem Yapısı

- Katmanlı sistemler (*layered*).
- Exokernel.
- Mikro çekirdek tabanlı istemci-sunucu sistemleri.
- Genişletilebilir sistemler (*extensible*).
- Çekirdek iş parçacıkları (*kernel threads*)

Katmanlı Sistemler

- İşletim sisteminin farklı işlevleri farklı katmanlara ayrılır.
- Farklı bileşenler arasındaki bağlantıyı azaltarak modülerliği destekler.
- Tipik olarak,
  - Uygulama Katmanı, (*application*)
  - Sistem Çağrısı Katmanı, (*system call*)
  - Kütüphane Katmanı, (*library*)
  - Çekirdek Katmanı, (*kernel*)
  - Donanım Katmanı. (*hardware*)
- İyi tanımlanmış arayüzler ve protokoller aracılığıyla,
  - bir katmandaki değişikliklerin diğer katmanları etkilememesi sağlanır.

Katmanlı Sistemler



Exokernel

- Donanım ve yazılım arasında minimal bir soyutlama katmanı sağlar.
- Ayrıntılı kaynak yönetimi ve tahsisi.
- Çekirdek bileşenlerinin dinamik olarak yüklenmesi ve boşaltılması.
- Tasarım ve uygulamada karmaşıklık.

Mikro Çekirdek Tabanlı İstemci-Sunucu Sistemleri

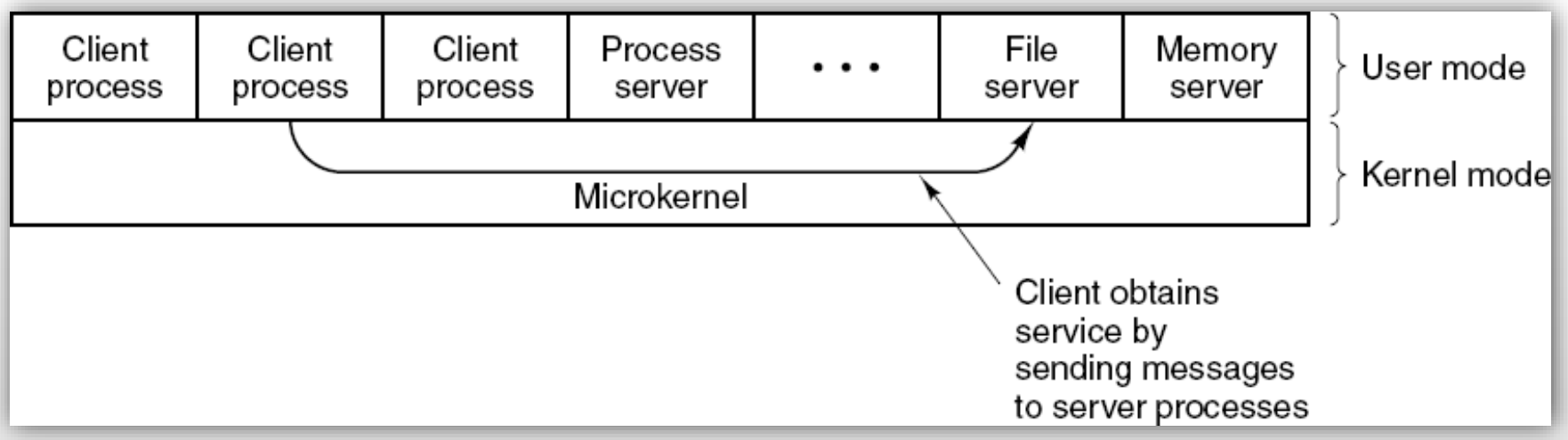
- Çekirdek olabildiğince küçük olmalı.
- Gerekli olmayan tüm hizmetler süreçlere verilir.
- Mikro çekirdek,
  - süreç yönetimi,
  - süreçler arası iletişim,
  - sistem çağrıları gibi temel görevleri yerine getirir.
- Sunucu-istemci mimarisi, modülerlik ve kolay bakım sağlar.
- Sistemin diğer kısımları etkilenmeden sunucu eklenebilir çıkarılabilir.
- Sunucu, dosya sistemi gibi farklı işlevler sağlar.

Mikro Çekirdek Tabanlı İstemci-Sunucu Sistemleri

- İstemci, sunucu tarafından sağlanan hizmetleri kullanır.
- Sistem çağrıları veya mesajlar aracılığıyla sunucu ile iletişim kurulur.
- İstemci ve sunucu arasındaki iletişim güvenlidir,
  - mikro çekirdek tüm süreçler arası iletişimden sorumludur.
- Sunucu, ağdaki farklı düğümlerde çalışabilir,
  - mikro çekirdek tabanlı sistemler yüksek düzeyde ölçeklenebilir.
- Örnek: *Mach, QNX, L4.*

Mikro Çekirdek Tabanlı İstemci-Sunucu Sistemleri

- istemci - microkernel - sunucu.



Genişletilebilir Sistemler

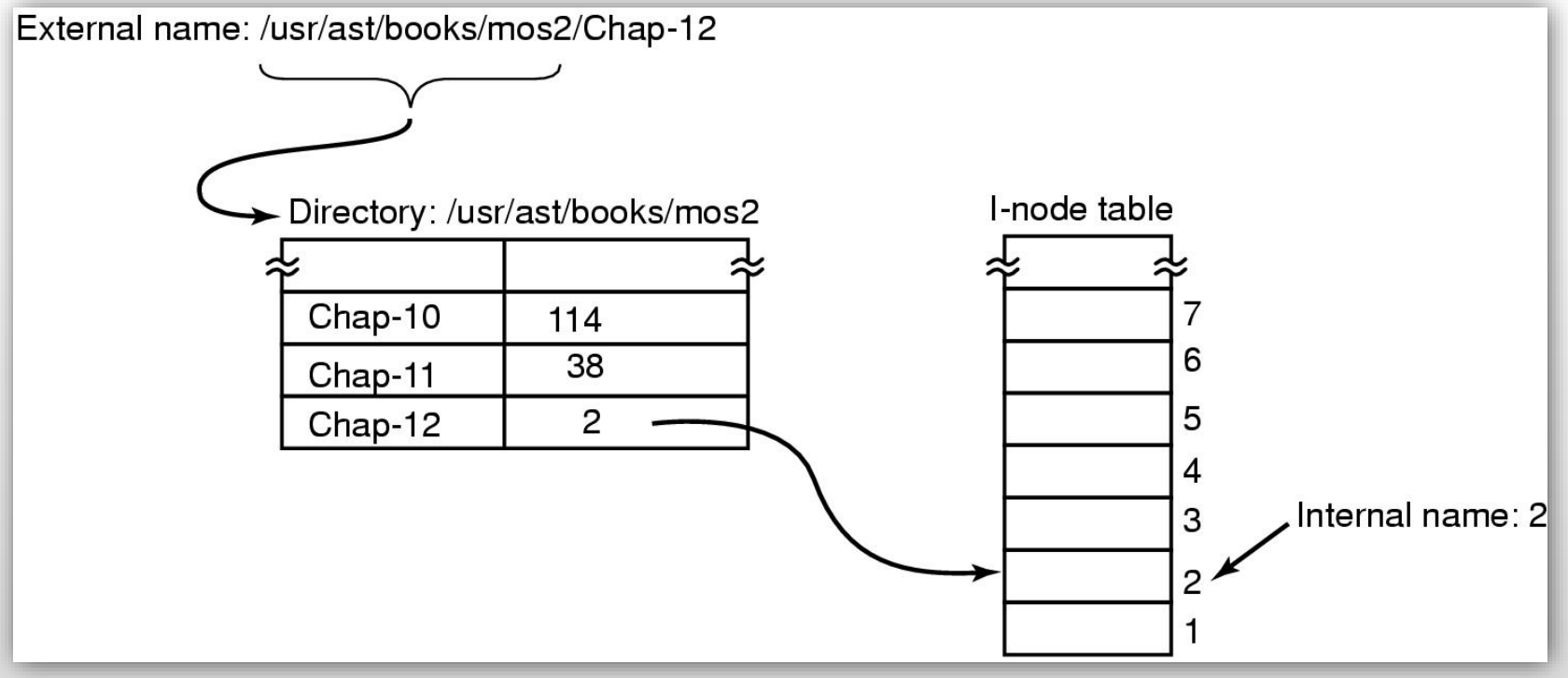
- Çekirdek sistem temel işlevleri sağlar.
- Değişen ihtiyaçlar için sistem genişletilebilir.
- Mevcut sisteme dinamik olarak yeni bileşen eklenmesine izin verir.
- Üçüncü taraf geliştiricilerin, sisteme yeni özellikler eklemesine olanak tanır.
- Yeni bileşenler entegre edilirken uyumluluk ve güvenlik ele alınmalıdır.

Çekirdek İş Parçacıkları

- Çekirdek tarafından yönetilir.
- Çekirdek ile aynı bellek alanını paylaşır.
- Verimli bir çizelgeleme ve yürütme avantajına sahiptir.
- Düşük ek maliyet ve hızlı bağlam anahtarlama.
- Kesme işleme, aygıt G/Ç ve arka plan sistem bakım görevleri.
- Kullanıcı düzeyinde süreçler tarafından, veya
  - sistem çağrıları ile çekirdek çizelgeleyici tarafından,
  - oluşturulabilir, yok edilebilir ve senkronize edilebilirler.

Adlandırma

- Dizinler, harici adları dahili adlara eşlemek için kullanılır.



Bağlama Zamanı

- Erken bağlama (*early binding*)
  - Basit
  - Az esnek
- Geç bağlama (*late binding*)
  - Daha karmaşık
  - Daha esnek

Statik ve Dinamik Yapılar

- Belirli bir *PID* için süreç tablosunu aramak için kullanılan kod.

```
found = 0;
for (p=&proc_table[0]; p<&proc_table[PROC_TABLE_SIZE]; p++) {
  if (p->proc_pid == pid) {
    found = 1;
    break;
  }
}
```

Yararlı Teknikler

- Donanım detaylarını gizleme (*hiding hardware details*)
- Tekrar kullanılabilirlik (*reusability*)
- Yeniden giriş (*reentrancy*)
- Kaba kuvvet (*brute force*)
- Önce hataları kontrol etme (*check for errors first*)

Performans

- İşletim sistemleri neden yavaş?
- Neler optimize edilmeli?
- Uzay-zaman takasları (*swap space-time*)
- Önbelleğe almak (*caching*)
- İpuçları (*hints*)
- Yerelliği kullanmak (*exploiting locality*)
- Genel durumu optimize etme (*optimization*)

Donanımı Gizleme

- CPU türüne bağlı koşullu derleme.

```
#include "config.h"
init() {
#if (CPU == PENTIUM)
/* Pentium initialization here. */
#endif
#if (CPU == ULTRASPARC)
/* UltraSPARC initialization here. */
#endif
```

Donanımı Gizleme

- Sözcük (*word*) uzunluğuna bağlı koşullu derleme.

```
#include "config.h"
#if (WORD_LENGTH == 32)
      typedef int Register;
#endif
#if (WORD_LENGTH == 64)
      typedef long Register;
#endif
Register R0, R1, R2, R3;
```

Uzay Zaman Takasları

- (a) Bir bayttaki bitleri sayma yordamı.

```
#define BYTE_SIZE 8 /* A byte contains 8 bits */
  int bit_count(int byte) { /* Count the bits in a byte. */
  int i, count = 0;
  for (i = 0; i < BYTE_SIZE; i++) /* loop over the bits in a byte */
    if ((byte >> i) & 1) count++; /* if this bit is 1, add to count */
  return(count); /* return sum */
}
```

Uzay Zaman Takasları

- (b) Bitleri saymak için bir makro.

```
/*Macro to add up the bits in a byte and return the sum. */
#define bit_count(b) ((b & 1) + ((b>>1) & 1) + ((b>>2) & 1) +
((b>>3) & 1) + ((b>>4) & 1) + ((b>>5) & 1) + ((b>>6) & 1) +
((b>>7) & 1))
```

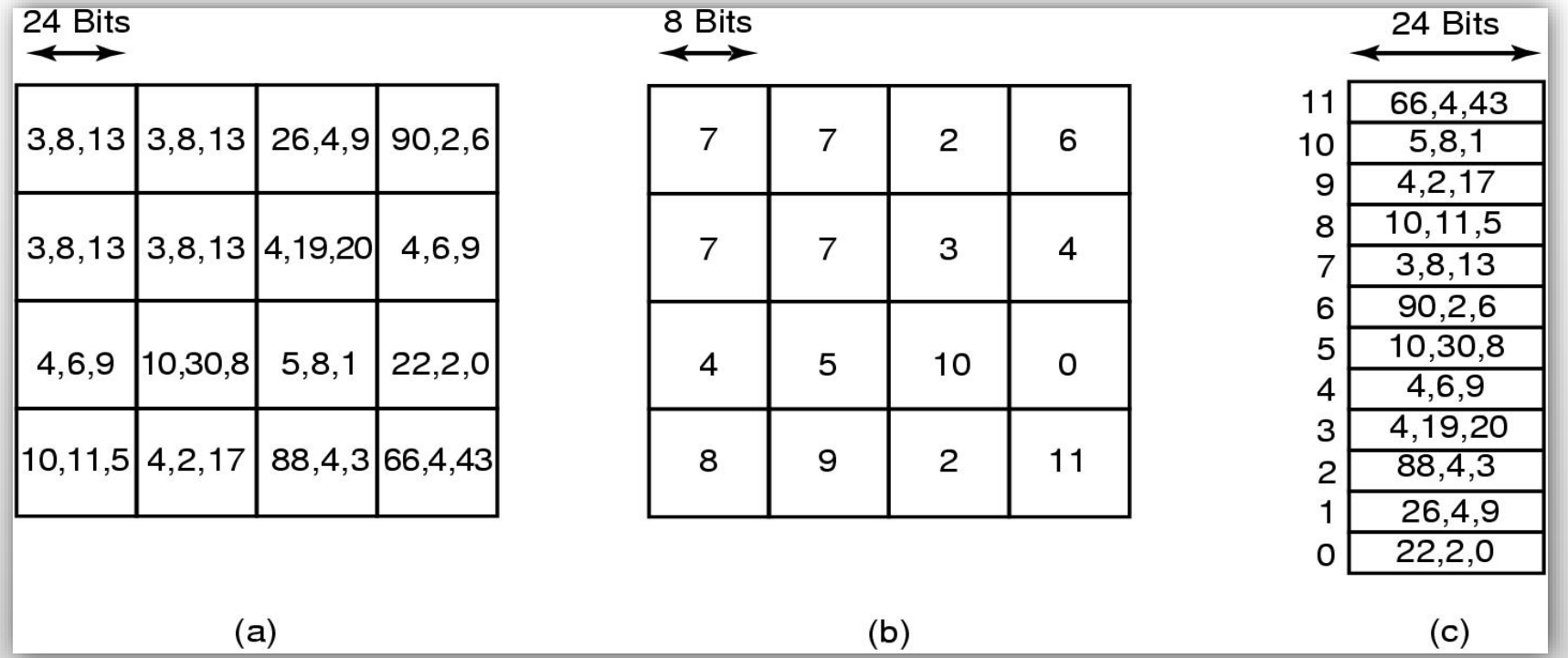
Uzay Zaman Takasları

- (c) Tabloya bakma

```
/*Macro to look up the bit count in a table. */
char bits[256] = {0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3,
3, 4, 1, 2, 2, 3, 2, 3, 3, ...};
#define bit_count(b) (int) bits[b]
```

Uzay Zaman Takasları

(a) 24 *bit* piksellik sıkıştırılmamış. (b) 8 *bit* piksellik GIF ile sıkıştırılmış.
(c) Renk paleti kullanılarak saklanan görüntü.



Önbelleğe Almak

- /usr/ast/mbox'ı aramak için aşağıdaki disk erişimleri gerekir:

- Kök dizin (*root*) için *i-node* okunur (*i-node 1*).
- Kök dizini okunur (*blok 1*).
- /usr için *i-node* okunur (*i-node 6*).
- /usr dizini okunur (*blok 132*).
- /usr/ast için *i-node* okunur (*i-node 26*).
- /usr/ast dizini okunur (*blok 406*).

Önbelleğe Almak

- Yol (*path*) ve yola karşılık gelen i-node numaraları.

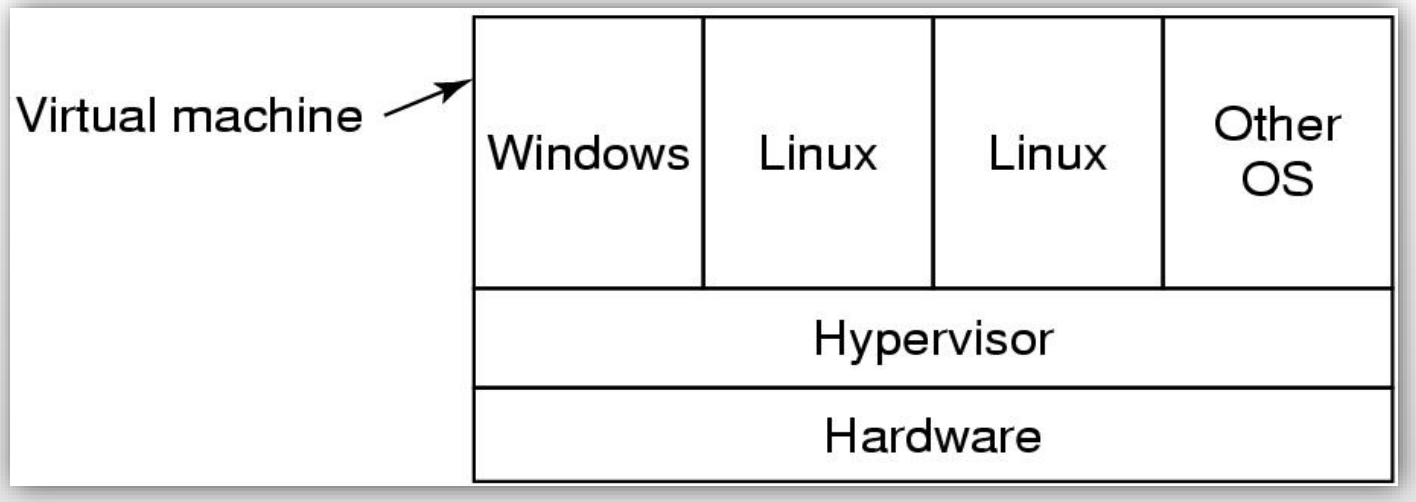
Path	I-node number
/usr	6
/usr/ast	26
/usr/ast/mbox	60
/usr/ast/books	92
/usr/bal	45
/usr/bal/paper.ps	85

İşletim Sistemlerinde Eğilimler

- Sanallaştırma (*virtualization*)
- Çok çekirdekli yongalar (*multicore chips*)
- Geniş adres uzayı (*large address space*)
- Ağ işlemleri (*network operations*)
- Paralel ve dağıtık sistemler (*parallel and distributed systems*)
- Çoklu ortam (*multimedia*)
- Pille çalışan sistemler (*battery powered*)
- Gömülü sistemler (*embedded*)
- Sensör düğümleri (*sensor nodes*)

Sanallaştırma

- Dört sanal makine çalıştıran bir hipervizör (*hypervisor*).



SON