



Bölüm 15: İşletim Sistemi Tasarımı

İşletim Sistemleri



İşletim Sistemi Hedefleri

- Soyutlamaları tanımlayın. (abstractions)
- İlkel işlemler sağlayın. (primitive)
- İzolasyonu sağlayın. (isolation)
- Donanımı yönetin.



İşletim Sistemleri Tasarlamanın Zorlukları

- İşletim sistemleri son derece büyük programlar haline geldi
- Eşzamanlılıkla uğraşmak gerekir (concurrency)
- Potansiyel olarak düşmanca davranan kullanıcılarla uğraşmak zorunda
- Birçok kullanıcı, bilgilerinin ve kaynaklarının bir kısmını seçilen diğer kullanıcılarla paylaşmak ister.



İşletim Sistemleri Tasarlamanın Zorlukları

- İşletim sistemleri uzun süre yaşar
- Tasarımcılar hatırı sayılır bir genellik sağlamalı
- Sistemler genellikle taşınabilir olacak şekilde tasarlanmalı
- Önceki bazı işletim sistemleriyle geriye dönük uyumlu olmalı



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 veya sistem çağrısı verimli bir şekilde uygulanamıyorsa, muhtemelen buna 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'ler
- Mikro Çekirdek Tabanlı İstemci-Sunucu Sistemleri
- Genişletilebilir Sistemler (extensible)
- Çekirdek İş Parçacıkları (kernel threads)



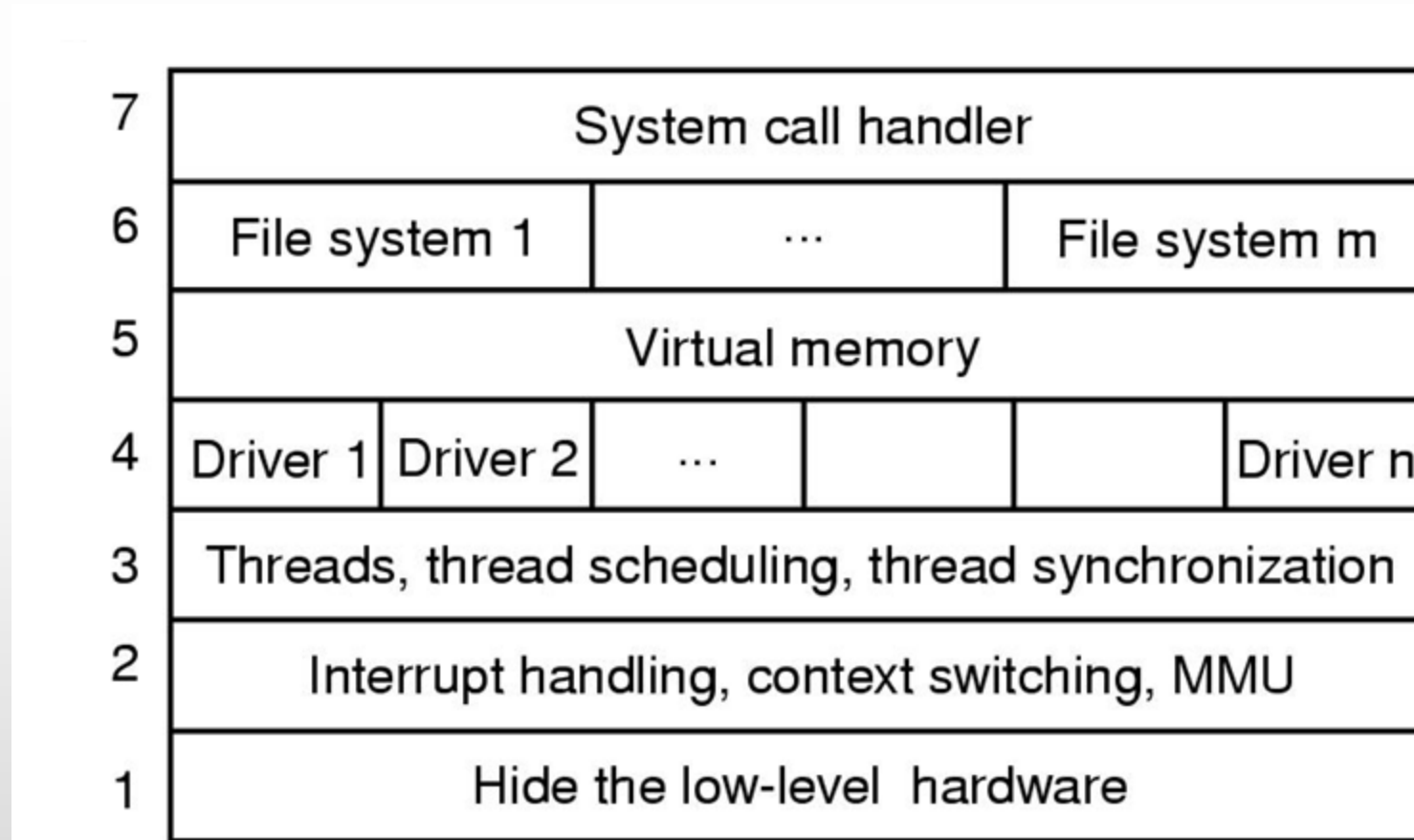
Katmanlı Sistemler

- Katmanlı Sistem mimarisi, bir işletim sisteminin farklı işlevlerinin katmanlara ayrıldığı bir tasarım yaklaşımıdır.
- Farklı bileşenler arasındaki bağlantıyı azaltarak modülerliği destekler
- Tipik olarak, işletim sistemleri Uygulama Katmanı, Sistem Çağrısı Katmanı, Kitaplık Katmanı, Çekirdek Katmanı, Donanım Katmanı gibi birkaç katmana ayrılır.
- İyi tanımlanmış arayüzler ve protokoller aracılığıyla birbirleriyle iletişim kurarak, bir katmandaki değişikliklerin diğer katmanları etkilememesini sağlar.
- Örnek olarak Unix ve Windows NT işletim sistemi verilebilir.



Katmanlı Sistemler

■ .





Exokernel

- Donanım ve yazılım arasında minimal bir soyutlama katmanı sağlayan bir işletim sistemi türüdür.
- Ayrıntılı kaynak yönetimi ve tahsisi
- Çekirdek bileşenlerinin dinamik olarak yüklenmesi ve boşaltılması
- Gömülü sistemler, yüksek performanslı bilgi işlem ve sanallaştırma ve bulut bilişim alanlarında kullanılır
- Tasarım ve uygulamada karmaşıklık,
- Özel bilgi ve uzmanlık gerektirir

Mikro Çekirdek Tabanlı İstemci-Sunucu Sistemleri



- Çekirdek işletim sistemini olabildiğince küçük tutma ve gerekli olmayan tüm hizmetleri ayrı süreçlere boşaltma ilkesini izler.
- Mikro çekirdek, yalnızca süreç yönetimi, süreçler arası iletişim ve sistem çağrıları gibi temel görevleri yerine getirir.
- Sunucu-istemci mimarisi, modülerlik ve kolay bakım sağlar.
- Sunucular, sistemin diğer kısımlarını etkilemeden eklenebilir veya kaldırılabilir.
- Sunucular, dosya sistemleri, aygıt sürücüler ve iletişim protokolleri gibi farklı işlevler sağlayabilir.



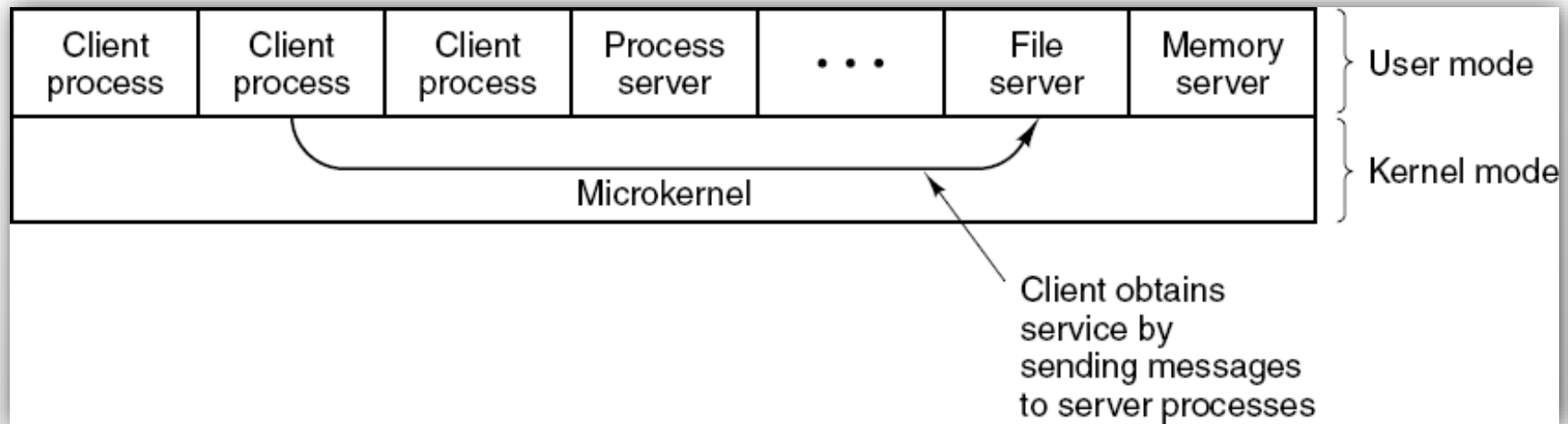
Mikro Çekirdek Tabanlı İstemci-Sunucu Sistemleri

- İstemciler, sunucular tarafından sağlanan hizmetleri kullanır.
- Sistem çağrıları veya mesajlar aracılığıyla sunucularla iletişim kurarlar.
- İstemciler ve sunucular arasındaki iletişim son derece güvenlidir, çünkü mikro çekirdek tüm işlemler arası iletişimin yürütülmesinden sorumludur.
- Sunucular bir ağdaki farklı düğümlerde çalışabildiğinden, mikro çekirdek tabanlı sistemler yüksek düzeyde ölçeklenebilir olabilir.
- Örnekleri arasında Mach, QNX ve L4 bulunur.



Mikro Çekirdek Tabanlı İstemci-Sunucu Sistemleri

■ .





Extensible Sistemler

- Geniřletilebilir sistemler, yeni bileřenlerin mevcut sisteme dinamik olarak eklenmesine izin verir.
- Çekirdek sistem temel iřlevleri saęlar ve iřlevsellięini geniřletmek için ek bileřenler takılabilir.
- Kullanıcıların deęiřen ihtiyaçlarını karřılamak için sistemin özelleřtirilmesine ve geniřletilmesine izin verir.
- Üçüncü taraf geliřtiricilerin sistemin iřlevsellięini geliřtirebilecek yeni özellikler eklemesine olanak tanır.
- Ancak sisteme yeni bileřenler entegre edilirken uyumluluk ve güvenlik konuları ele alınmalıdır.



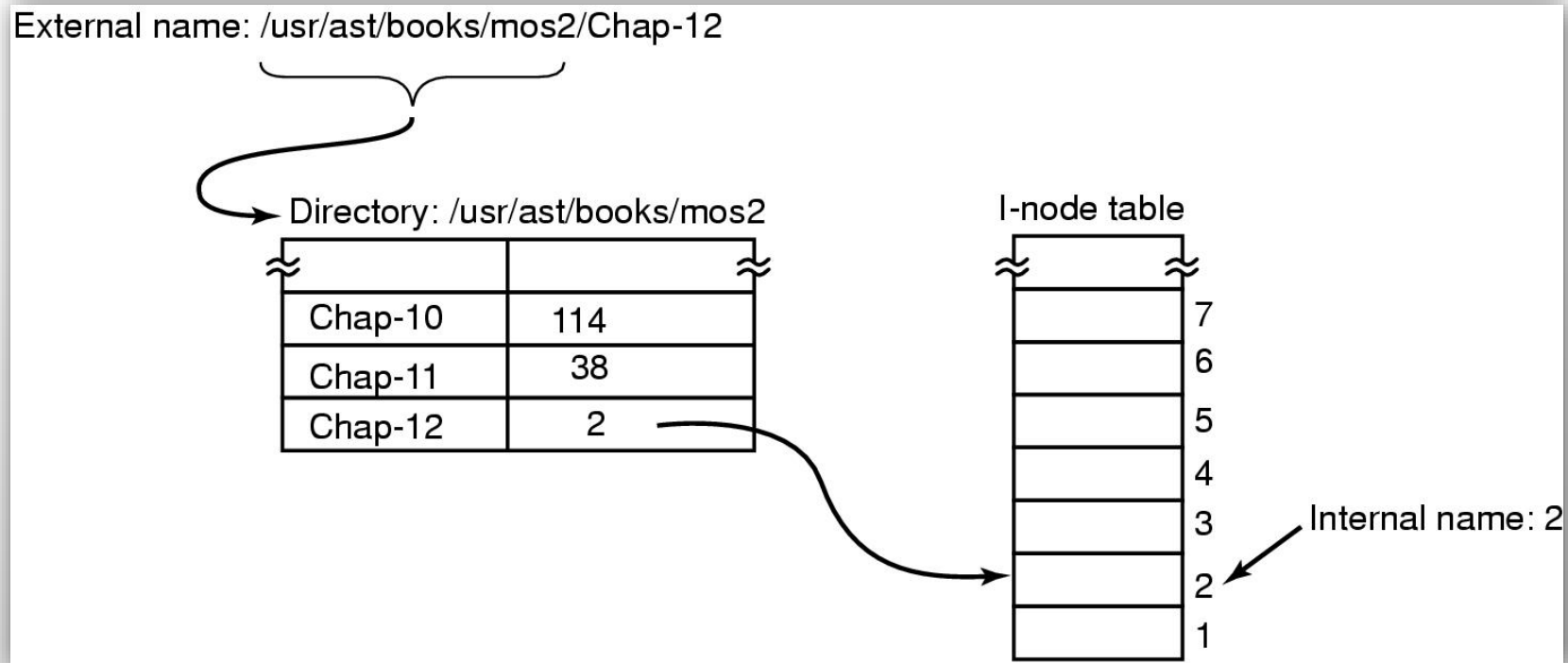
Çekirdek İş Parçacıkları

- Çekirdek düzeyinde uygulanır ve çekirdek tarafından yönetilir.
- Çekirdek ile aynı bellek alanını paylaşır, daha verimli bir zamanlama ve yürütme avantajına sahiptir.
- Düşük ek yük ve daha hızlı bağlam değiştirme gerektiren sistem görevlerini ve işlemlerini yürütmek için kullanılır.
- Kesme işleme, cihaz 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 Süresi

Erken bağlama

- Basit
- Az esnek

Geç bağlama

- Daha karmaşık
- Daha esnek



Statik ve Dinamik Yapılar

- Belirli bir PID için işlem 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

- Donanımı gizlemek
- Dolaylı yol
- Tekrar kullanılabilirlik (reusability)
- Yeniden giriş (reentrancy)
- Kaba kuvvet (brute force)
- Önce hataları kontrol edin (check for errors first)



Performans

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



Donanımı Gizleme

- CPU bağımlı koşullu derleme

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



Donanımı Gizleme

- Sözcük 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 prosedürü.

```
#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) Piksel başına 24 bitlik sıkıştırılmamış bir görüntü (b) GIF ile sıkıştırılmış aynı kısım, piksel başına 8 bit (c) Renk paleti

24 Bits

3,8,13	3,8,13	26,4,9	90,2,6
3,8,13	3,8,13	4,19,20	4,6,9
4,6,9	10,30,8	5,8,1	22,2,0
10,11,5	4,2,17	88,4,3	66,4,43

(a)

8 Bits

7	7	2	6
7	7	3	4
4	5	10	0
8	9	2	11

(b)

24 Bits

11	66,4,43
10	5,8,1
9	4,2,17
8	10,11,5
7	3,8,13
6	90,2,6
5	10,30,8
4	4,6,9
3	4,19,20
2	88,4,3
1	26,4,9
0	22,2,0

(c)



Önbelleğe Almak

- /usr/ast/mbox'ı aramak için aşağıdaki disk erişimleri gerekir:
- Kök dizin için i-node'u okuyun (i-node 1).
- Kök dizini okuyun (blok 1).
- /usr için i-node'u okuyun (i-node 6).
- /usr dizinini okuyun (blok 132).
- /usr/ast için i-node'u okuyun (i-node 26).
- /usr/ast dizinini okuyun (blok 406).



Önbelleğe Almak

■ .

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



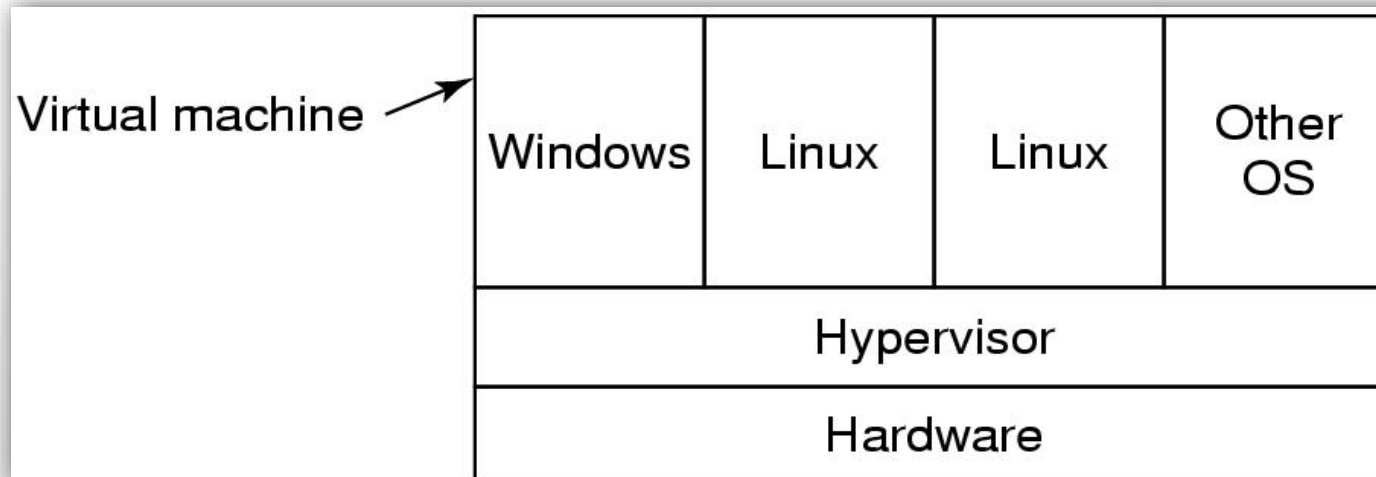
İşletim Sistemlerinde Eğilimler

- Sanallaştırma (virtualization)
- Çok çekirdekli yongalar
- Geniş adres uzaylı işletim sistemleri
- Ağ işlemleri
- Paralel ve dağıtık sistemler
- Multimedya
- Pille çalışan sistemler
- Gömülü sistemler
- Sensör düğümleri



Sanallaştırma

- Dört sanal makine çalıştıran bir hipervizör.





SON