3 İPLİKLER

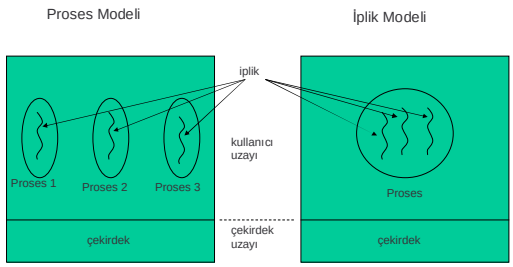
Giriş

- geleneksel işletim sistemlerinde her prosesin
  - özel adres uzayı ve
  - tek akış kontrolü var.
- aynı adres uzayında birden fazla akış kontrolü gerekebilir
  - aynı adres uzayında çalışan paralel prosesler gibi

İplik Modeli

- iplik = hafif proses
- aynı adres uzayını paylaşan paralel prosesler benzeri
- aynı proses ortamında birden fazla işlem yürütme imkanı
- iplikler tüm kaynakları paylaşır:
  - adres uzayı, bellek, açık dosyalar, ...
- çoklu iplikli çalışma
  - iplikler sıra ile koşar

İplik Modeli



İplik Modeli

- iplikler prosesler gibi birbirinden bağımsız değil:
  - adres uzayı paylaşır
    - global değişkenleri de paylaşırlar
    - birbirlerinin yığınını değiştirebilir
    - koruma yok çünkü:
      - mümkün değil
      - gerek yok

İplik Modeli

- iplikların paylaştıkları:
  - adres uzayı
  - global değişkenler
  - açık dosyalar
  - çocuk prosesler
  - bekleyen sinyaller
  - sinyal işleyiciler
  - kullanıcı bilgileri
- her bir ipliğe özel:
  - program sayacı
  - saklayıcılar
  - yığın
  - durum

İplik Modeli

- işler birbirinden büyük oranda bağımsız ise ⇒ proses modeli
- işler birbirine çok bağlı ve birlikte yürütülüyorsa ⇒ iplik modeli
- iplik durumları = proses durumları
  - koşuyor
  - bloke
    - bir olay bekliyor: dış olay veya bir başka ipliği bekler
  - hazır

İplik Modeli

- her ipliğin kendi yığını var
  - yığında çağrılmış ama dönülmemiş yordamlarla ilgili kayıtlar ve yerel değişkenler
  - her iplik farklı yordam çağrıları yapabilir
    - geri dönecekleri yerler farklı ⇒ ayrı yığın gerekli

İplik Modeli

- prosesin başta bir ipliği var
- iplik kütüphane yordamları ile yeni iplikler yaratır
  - örn: `thread_create`
    - parametresi: koşturacağı yordamın adı
- yaratılan iplik aynı adres uzayında koşar
- bazı sistemlerde iplikler arası anne – çocuk hiyerarşik yapısı var
  - çoğu sistemde tüm iplikler eşit

İplik Modeli

- işi biten iplik kütüpane yordamı çağrısı ile sonlanır
  - örn: `thread_exit`
- zaman paylaşımı için zamanlayıcı yok
  - iplikler işlemciyi kendileri bırakır
    - örn: `thread_exit`
- iplikler arası
  - senkronizasyon ve
  - haberleşme olabilir

İplik Modeli

- ipliklerin gerçeklenmesinde bazı sorunlar:
  - örn. UNIX'te `fork` sistem çağrısı
    - anne çok iplikli ise çocuk proseste de aynı iplikler olacak mı?
    - olmazsa doğru çalışmayabilir
    - olursa
      - örneğin annedeki iplik giriş bekliyorsa çocuktaki de mi beklesin?
      - giriş olunca her ikisine de mi yollansın?
    - benzer problem açı ağ bağlantıları için de var

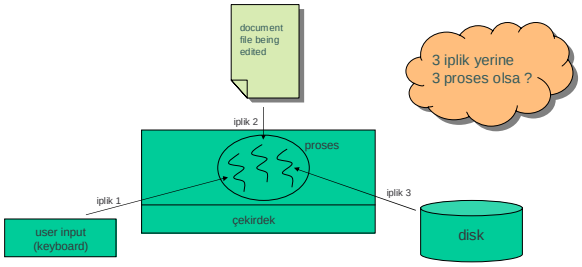
İplik Modeli

- *('sorunlar' devam)*
  - bir iplik bir dosyayı kapadı ama başka iplik o dosyayı kullanıyordu
  - bir iplik az bellek olduğunu farkedip bellek almaya başladı
    - işlem tamamlanmadan başka iplik çalıştı
    - yeni iplik de az bellek var diye bellek istedi
    - ⇒ iki kere bellek alınabilir
- çözümler için iyi tasarım ve planlama gerekli

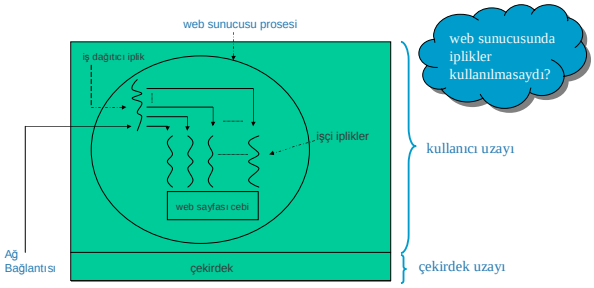
İpliklerin Kullanımı

- neden iplikler?
  - bir proses içinde birden fazla işlem olabilir
    - bazı işlemler bazen bloke olabilir; ipliklere bölmek performansı arttırır
  - ipliklerin kendi kaynakları yok
    - yaratılmaları / yok edilmeleri proseslere göre kolay
  - ipliklerin bazıları işlemciye yönelik bazıları giriş-çıkış işlemleri yapıyorsa performans artar
    - hepsi işlemciye yönelikse olmaz
  - çok işlemcili sistemlerde faydalı

İplik Kullanımına Örnek – 3 İplikli Kelime İşlemci Modeli



İplik Kullanımına Örnek – Web Sitesi Sunucusu



İplik Kullanımına Örnek – Web Sitesi Sunucusu

İş dağıtıcı iplik kodu

```
while TRUE {
    sıradaki_isteği_al(&tmp);
    işi_aktar(&tmp);
}
```

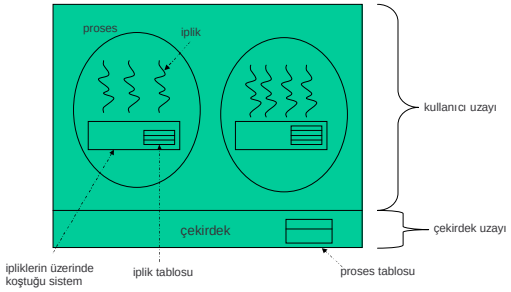
İşçi ipliklerin kodu

```
while TRUE {
    iş_bekle(&tmp);
    sayfayı_cepte_ara(&tmp,&sayfa);
    if (sayfa_cepte_yok(&sayfa)
        sayfayı_diskten_oku(&tmp,&sayfa);
    sayfayı_döndür(&sayfa);
}
```


İpliklerin Gerçeklenmesi

- iki türlü gerçekleme mümkün
  - kullanıcı uzayında
  - çekirdek uzayında
- hibrid bir gerçekleme de olabilir

İpliklerin Kullanıcı Uzayında Gerçeklenmesi



İpliklerin Kullanıcı Uzayında Gerçeklenmesi

- çekirdeğin ipliklerden haberi yok
- çoklu iplik yapısını desteklemeyen işletim sistemlerinde de gerçeklenebilir
- ipliklerin üzerinde koştuğu sistem
  - iplik yönetim yordamları
    - örn. `thread_create`, `thread_exit`, `thread_yield`, `thread_wait`, ...
    - iplik tablosu
      - program sayacı, saklayıcılar, yığın işaretçisi, durumu, ...

İpliklerin Kullanıcı Uzayında Gerçeklenmesi

- iplik bloke olacak bir işlem yürüttüyse
  - örneğin bir başka ipliğin bir işi bitirmesini beklemek
  - bir rutin çağırır
  - rutin ipliği bloke durum sokar
  - ipliğin program sayacı ve saklayıcı içeriklerini iplik tablosuna saklar
  - sıradaki ipliğin bilgilerini tablodan alıp saklayıcılara yükler
  - sıradaki ipliği çalıştırır
  - hepsi yerel yordamlar ⇒ sistem çağrısı yapmaktan daha hızlı

İpliklerin Kullanıcı Uzayında Gerçeklenmesi

- avantajları:
  - ipliklerin ayrı bir iş sıralama algoritması olabilir
  - çekirdekte iplik tablosu yeri gerekmiyor
  - tüm çağrılar yerel rutinler ⇒ çekirdeğe çağrı yapmaktan daha hızlı

İpliklerin Kullanıcı Uzayında Gerçeklenmesi

- Problemler:
  - bloke olan sistem çağrılarının gerçeklenmesi
    - iplik doğrudan bloke olan bir sistem çağrısı yapamaz ⇒ tüm iplikler bloke olur
    - sistem çağrıları değiştirilebilir
      - işletim sisteminin değiştirilmesi istenmez
      - kullanıcı programlarının da değişmesi gerekir
    - bazı sistemlerde yapılan çağrının bloke olup olmayacağını döndüren sistem çağrıları var
      - sistem çağrılarına ara-birim (wrapper) yazılır
      - önce kontrol edilir, bloke olunacaksa sistem çağrısı yapılmaz, iplik bekletilir

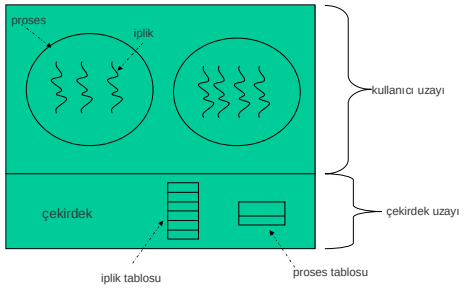
İpliklerin Kullanıcı Uzayında Gerçeklenmesi

- *(problemler devam)*
  - sayfa hataları
    - programın çalışması gereken kod parçasına ilişkin kısım ana bellekte değilse
      - sayfa hatası olur
      - proses bloke olur
      - gereken sayfa ana belleğe alınır
      - proses çalışabilir
    - sayfa hatasına iplik sebep olduysa
      - çekirdek ipliklerden habersiz
      - tüm proses bloke edilir

İpliklerin Kullanıcı Uzayında Gerçeklenmesi

- *(problemler devam)*
  - iş sıralama
    - iplik kendisi çalışmayı bırakmazsa diğer iplikler çalışamaz
      - altta çalışan sistem belirli sıklıkta saat kesmesi isteyebilir
        - » ipliklerin de saat kesmesi ile işi varsa karışır
  - çok iplikli çalışma istendiği durumlarda sıkça bloke olan ve sistem çağrısı yapan iplikler olur
    - çekirdek düzeyinde işlemek çok yük getirmez çekirdeğe

İpliklerin Çekirdek Uzayında Gerçeklenmesi



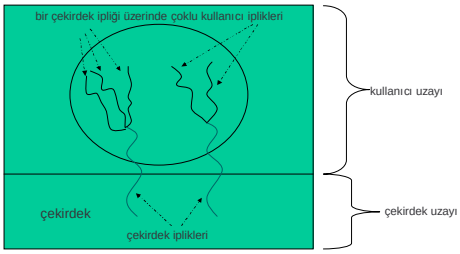
İpliklerin Çekirdek Uzayında Gerçeklenmesi

- çekirdek ipliklerden haberdar
- iplik tablosu çekirdekte
- yeni iplik yaratmak için çekirdeğe sistem çağrısı
- ipliği bloke edebilecek tüm çağrılar çekirdeğe sistem çağrısı
- işletim sistemi hangi ipliğin koşacağına karar verir
  - aynı prosesin ipliği olmayabilir

İpliklerin Çekirdek Uzayında Gerçeklenmesi

- bloke olan sistem çağrılarının yeniden yazılması gerekmez
- sayfa hatası durumu da sorun yaratmaz
  - sayfa hatası olunca çekirdek aynı prosesin koşabilir baka ipliği varsa çalıştırır
- sistem çağrısı gerçekleme ve yürütme maliyetli
  - çok sık iplik yaratma, yoketme, ... işlemleri varsa vakit kaybı çok

İpliklerin Hibrit Yapıda Gerçeklenmesi



İpliklerin Hibrit Yapıda Gerçeklenmesi

- çekirdek sadece çekirdek düzeyi ipliklerden haberdar
- bir çekirdek düzeyi iplik üzerinde birden fazla kullanıcı düzeyi iplik sıra ile çalışır
- kullanıcı düzeyi iplik işlemleri aynı şekilde