

İŞLETİM SİSTEMLERİ

İşletim Sistemi Çeşitleri:

- Batch OS
- Time Sharing OS
- Distributed OS
- Network OS
- Real Time OS
- Multi Programming OS
- Processing / Testing OS

İşletim Sistemi Fonksiyonları:

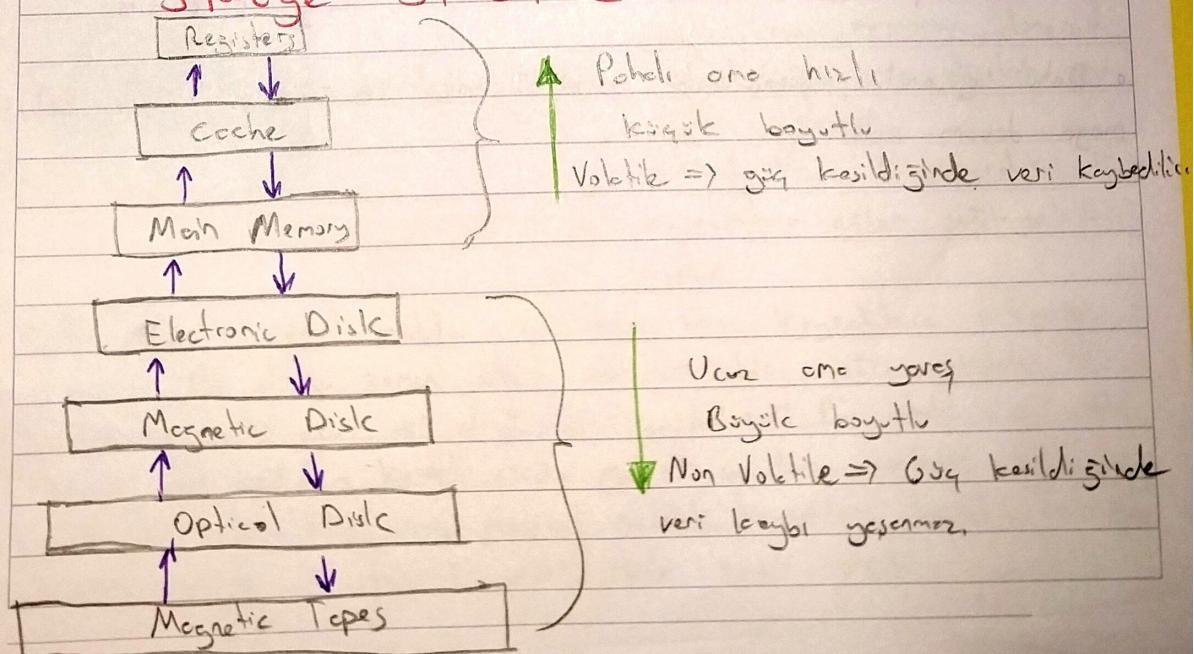
- Kullanıcı ile donanım arasında arayüz yazılımıdır.
- Kaynakları yönetir
- Bellek, giventik gibi alanları yönetir.

Bootstrap Program: Bilgisayar çalışmaya başladığında çalışın ilk program.

- ROM'de depolanır.
- OS'yi yüklemeyi ve başlatmayı direkt sağlar.

Interrupt: Donanımlardan ve de yazılımlardan gelen sinyalle process'in devam etmesi sağlar.

Storage Structure



I/O Structure

* Boyutlu boyutlu verileri taşımak için verimsiz bir yöntemdir.
Bunu önemlilik için Direct Memory Access (DMA) kullanılır.

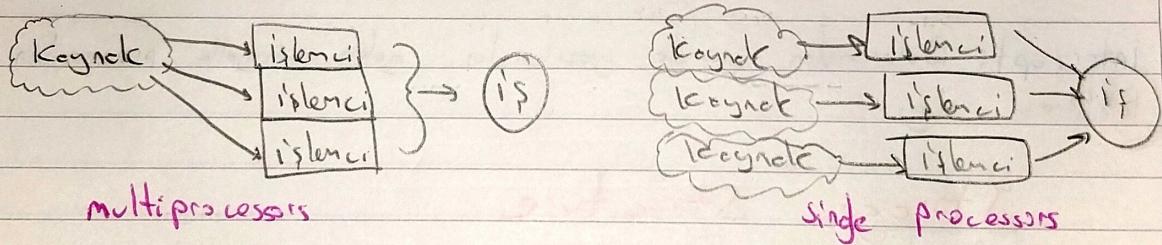
Computer System Architecture

1. Single Processors System: Tek bir one işlemci genel amaçlı talimatları ve kullanıcıdan gelen talimatları çalıştırır.

2. Multiprocessors System: İki ya da daha fazla işlemci için işin iletiminde bulandırılarak, computer bus paylaşarak bir ya da daha fazla görevi yerine getirir.

Avantajları:

- Sistemin performansında artı. Birden fazla işlemci olduğu için bir işi parçel olarak yapanlarında iş daha kolay şekilde gerçekleştirilebilir.
- Daha ekonomik. Fazla işlemciler işi yaparken genellikle kaynakları paylaşarak kullanır.



- Daha güvenlidir. İşlemcilerden biri çökmez ise sistem hâlâ çalışmaya devam eder.

3. Clustered Systems: Çoklu işlemciler gibi birden fazla sistemini bir araya getirip bir işi yapmasıdır. Çoklu işlemcilerden farklı; işlemcilerden değil, temanlanmış sistemlerden oluşuyor olmasıdır.

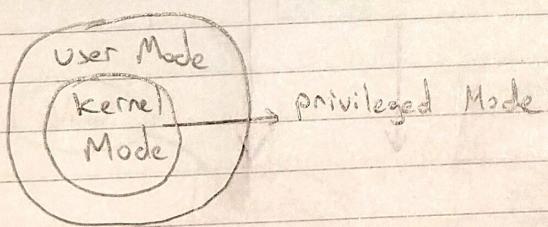
Operating System Architecture

1. Multiprogramming: Multiprogramming işleri tek tekde CPU kullanımını ortttır. Tüm kaynakların etkili bir şekilde kullanılmaması sağlanır, ancak kullanıcı ile bilgisayar sistemi arasında etkileşimde bulunmaz.

2. Multitasking (Time Sharing):

- CPU çok fazla işi ardışık olarak yapmak için kullanılır.
- Bu değişimler (switches) çok hızlı bir şekilde gerçekleşir. Böylece kullanıcılar her bir program çalışırken etkileşimde bulunabilir.
- Time sharing interaktif bir sistemdir. Bu sayede kullanıcı ile sistem arasında iletişim sağlanır. (Multiprogramming'den farklı budur)
- Time shared sistem birden fazla kullanıcıyı bilgisayarın simultane bir şekilde kullanmasına izin verir.

System Cells



User Mode: Program user mode'da Çalıştırılabilir memory, doncüm gibi kaynaklara direkt erişimi yoktur.

Kernel Mode: Bilgisayardaki tüm kaynaklar erişim sağlayabilen mode'dır.

- Bir program genelde modda çalışırken bazı kaynaklara ihtiyaç duyabildir. Bu olduğu zaman işletim sisteme bir çağrı gönderilir. Bu kaynaklara ihtiyaçım var diye istekte bulunur. Bu istek sonucunda program user mode'den kernel mode geçer ve bu context switch odu verilir. Mod değişikten sonra program gereklili kaynakları kullanır. Ardından sistem çağrı ile tekrar user mode geçip yapılır.

System Calls Özellikleri

1. Process Control • create process, terminate process

- end, abort

- load, execute

2. File Manipulation • create file, delete file

- open, close, read, write

3. Device Manipulation request device, release device

Structures of Operating System

Simple Structure

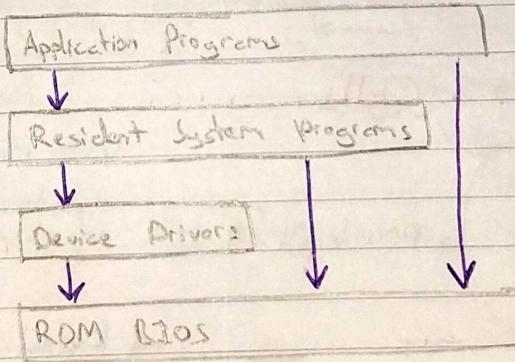
Simple yapı çekirdeği minimum düzende tutar ve diğer işlevlerin çoğunu user mode içerisinde sunuculara tesir.

Avantajları

- Daha modülerdir. Yeni hizmet eklenene
ye de değiştirmek kolaydır.
- Hizmetler çekirdekteki izole
edildiği için güvenlidir.
- Simple yapılar daha kolay tarama.

Desavantajları

- Hizmetlerin çekirdekteki sınırlı
çalışması performans iyileşime yol
acabiltir.
- Simple yapılar, birçok düşük
hizmetin etkileşimi nedeniyle kor-
meliğe yol açabilir.



Monolith Structures

Bu yapı işletim sistemi çekirdeğini ve hizmetlerini tek bir boyut ve birlesik bir yazılım olarak sağlar. Çekirdek içinde her türlü hizmeti sağlar. Bu nedenle işletim sistemi bileşenleri doğrudan çekirdekte içinde bulunur.

Avantajları

- Hızlı işlem. Tüm işlember çekirdek içinde bulunur. İşlemler arasındaki iletişim ve geçişler hızlidır.
- Monolith yapıları genellikle deha öz kaynak kullanır.
- Tek bir yazılım parçası olduğu için deha öz içermeye eğilimlidir.

Değerlendirme

- Bir bileşen hatası tüm çekirdeği çökerebilir.
- Bileşen sorunları, Genelleştirme zordur.
- Tanınlabilirliği düşüldür.

Process State

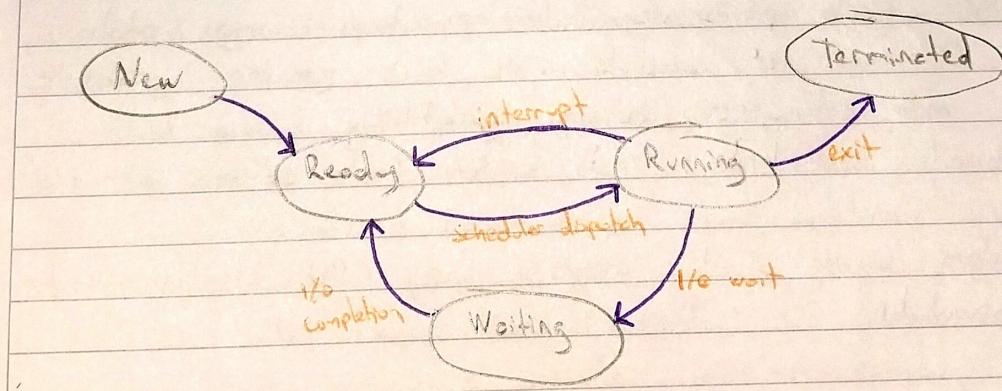
New: Proses oluşturuldu.

Running: Telsizler eşleştiriliyor.

Waiting: Proses bir olay bekliyor.

Ready: Proses işlemciye geçmek için hazır.

Terminated: Proses çalışmaya bitirdi.



Process Control Block

Process Control Block (PCB), bir işletim sistemi içindeki her işlem için oluşturulan bir veri yapısıdır. PCB, işlemi izlemek, yönetmek ve gerektiğinde işlere geçiş yapmak için kullanılır.

Process ID (PID): PCB'in en temel bilgisi işlemin kimliğidir. Her işlem benzersiz PID ile tanımlanır. İsmi diğer işlemlerden ayırmak temel yolu olarak kullanılır.

Register State: İşlemi处理ettiğinizde işlemci kayıtlarının durumunu sözler. Bu, işlemeye geçiş yaparken işlemi aynı anda devam ettirebilmeyi sağlar.

Program Counter: İşlemi gerçekleştirdiğinizde, işlemcinin hangi komutun işaretini olduğunu bilmesi gereklidir. Program counter, işlemci tarafından işlenecek bir sonraki komutun adresini işaretler.

CPU Registers: PCB, işlemci kayıtlarının işaretini sözler. Bu kayıtlar işlemi durumunu temsil eder.

Memory Management Information: PCB, işlem için ayrılmış bellek bölgelerinin başlangıç adresini ve boyutunu işaretler.

Process State: Process running, waiting, ready, veya terminated gibi farklı durumlarla değişimlidir. PCB, işlemi geçici durumunu izler.

Process Scheduling

* CPU kaynaklarını boş harcamamak için CPU sürekli işlem yapmak durumundadır.

* CPU birden fazla process ile ilgilenirken hepsini aynı anda çalışmaz. O process'ler arasında gide hızla geçişler yaparak her biri aynı anda çalışmamayı garantiler.

* Tek işlemcili sistemlerde, osb birden fazla çalışan process bulunur.

* Eğer sistem birden fazla işlemeli ise, CPU bölgeleri arasında belli olmalıdır.

Context Switch

* Interrupt gerçekleştiği anda sistemin o anda context'ı kaybetmesi gereklidir. Böylece interrupt tamamlandıkten sonra CPU kaldığı yerden çalışmaya devam edebilir.

Analogi: Odeñinde kitap okurken yem odasına girdim iken seslenir. Bu nedenle sonuc kitap okumamız interrupt edilir. Çember onnenizin seslenip deha öncelikli olduğunu. Bunu yaparken devamlı bir sayfaya tırış, yemektiğinizdir. Böylece işiniz bittiğinde okumaya kaldığınız yerden devam edebilirsiniz.

* Context switch ile CPU bir process'ten ötekine geçiş yapar. Bir process'in state durumu kaybedilir, diğer process'in state durumu restore edilir.

Process Operasyonları

Process Creation: Yeni bir process yaratıldığında olası iki senaryo vardır.

- 1-) Parent process çocuklarından başımsız şekilde çalışmaya devam eder.
- 2-) Parent process çocuk process'ları tamamlayana kadar bekler.

Ayrıca yeni bir process oluşturulduğunda yeni process'in adreslemesi için de iki seçenek vardır.

- 1-) Child process, parent process'ı copyeder. (Aynı program ve data'ları kurtarır).
- 2-) Child process, yeni bir programı oluşturur.

Process Termination: Parent process kendi çocuklarını çeşitli sebeplerle sonlandırabilir.

- Child töhrsiz edilmiş veya kaynak sınırları aşmıştır.
- Görevi tamamladıktan sonra child process'e geriye kalmadığında
- Parent process sonlandırdığında o process'in çocukları da sonlandırılır.

Interprocess Communication

Process'ler eşzamanlı olarak iki şekilde gerçekleştirilebilir.

1-) Independent Process: Bu process'ler diğer process'lar tarafından etkileşime ya da onları etkileyenler. Tek birbirine etkileşebilirler.

2-) Cooperating Processes: Bu process'ler diğerini etkileyebilir ya da etkileşebilirler.

Veri paylaşım bir process cooperating process'tır.

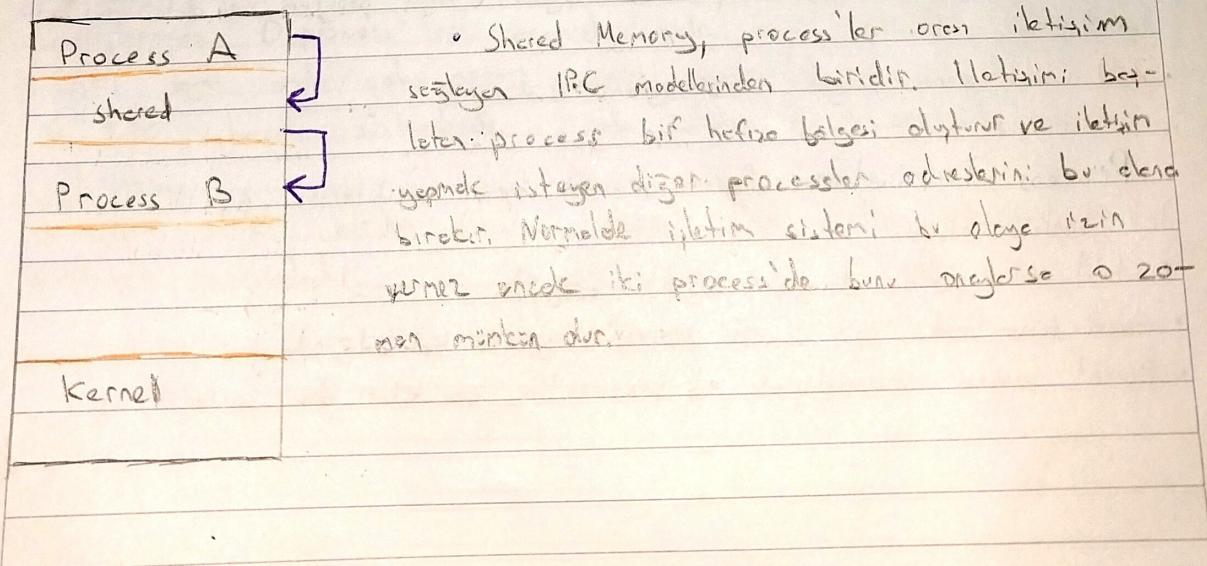
- Process birlükteligiye izin veren birkaç neden vardır:

- Information sharing (Bilgi Paylaşımı)
- Computation speedup (Sistemin hızlanması sağlar)
- Modularity (Test'i parçalara bölmek)
- Conveinice (Koşullu sağlar)

- Cooperating processes interprocess communication (IPC) adı verilen mekanizmeye ihtiyaç duyar. Bu seyede veri ve bilgi paylaşımı sağlanır.

- İki ana çeşitli IPC modeli bulunur.
- Shared Memory
- Message Passing

Shared Memory Systems



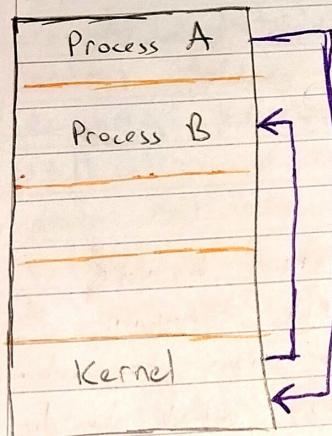
Producer Consumer Problem: Bu problemden üretici bilgiyi üretir ve tüketici bu bilgiyi tüketmeye çalışır. Sorun ise üretici ve tüketici eşzamنı derek kullanmaktadır. Çünkü tüketici önce üretilmiş olan, hazırladığı bilgiyi kullanır. Henüz üretilmemiş bilgiyi kullanamaz. Bu problemin çözüm ortak hafızasıdır (shared memory). Üretici ve tüketicinin aynı anda kullanabilmesi için, üretici tarafından sürekli olarak bilgi ile doldurulan ve tüketici tarafından tüketilebilen **buffer of items** adı verilen yapıya ihtiyaç duyuyor.

Buffer, pazarlanan belki de belgesinde yer almaktadır. İki çeşit buffer tipi bulunmaktadır.

- Unbound Buffer (Sınırsız Tempon)
- Bounded Buffer (Sınırlı Tempon)

İki durumda da eğer tüketilecek bir bilgi yoksa tüketici beklerken zorundadır. Unbound Buffer farklı derek, herhangi bir sınır olmadan üretici bilgi üretebilir,

Message Passing Systems



Message Passing Systems, pazarlanan bellek olmadan processler arası iletişim icra edilmesine olanak sağlar.

Bu sistem özellikle network ile bağlantılı bilgi yapılarının iletişiminde faydalıdır.

Bu yöntem **send()** ve **receive()** olmak üzere iki operasyon içerir. Prosesler tarafından gönderilen bu mesajlar **fixed** veya **variable** size olabilir.

Fixed size mesajlar sisteme implementasyonları kolay, programlamaları zor mesajlardır. Çünkü programlama yaparken sürekli direkt baytleri okuduğu için, kontrol etmek gereklidir.

Dogruluk İletisimi: İletinin kurmak isteyen herbir process, göndericisinin ve alıcının adresini biliridir.

- Send (P, message) Process P'ye mesaj gönderir.
- receive (Q, message) Process Q'den mesaj alır.

Bu iletişim linki şeninin birleşik özellikleri:

- Bağlantı, iletişim kurmak isteyen herbir process arasında otomatik olarak sağlanır.
- İki process sadece iki process arasında kurulur.
- Adreslerde simetri vardır. İki process'te birbirinin adresini biliir.

Dogruluk iletişimini bitirme esasında esimetrik iletişimdir.

- Send (P, message)
- receive (id, message)

Bu iki iletişim (simetrik ve esimetrik) dezavantajları sınırlı, modülerlidir.

Dolegli İletisim: Gönderenin ve alıcının mesajları mailbox' den gelir. Her bir mailbox özel belirteci vardır.

- İki process sadece paylaşımlı mailbox ile iletişim kurabilir.
- Send (A, message) Mailbox A' ye mesaj gönderir.
- receive (A, message) Mailbox A'dan mesaj alır.

Soketler

- Soketler client-server sistemlerinde kullanılan iletişim birimidir.
- Soketler, bir port numarası bağlı IP adresiyle tanımlanır.
- 1024 altındaki tüm port numaraları standart servis programlarına ayrılmıştır. Bu nedenle port numarası 1024'ü aşkın olmalıdır.

146.86.5.20

161.25.19.8

Soket

146.86.5.20: 1625

Host

Soket

161.25.19.8: 80

web server

Remote Procedure Calls (RPC)

Yerel bir bilgisayardaki processler shared memory ya da message passing yöntemi ile iletişim kururlar. Farklı bilgisayardaki processler ise RPC ile iletişim kurur.

- IPC rekenizmasına oldukça benzer

Threads

İş parçası (thread), bir bilgisayar programının içinde aynı zaman diliminde çalışan bağımsız yürütme birimidir. Bir process içinde birden fazla thread olabilir ve bu thread'lar aynı bellek alanını paylaşır. Thread'lar, paralel veya eşzamanlı çalışma sağlamak, çatışma 가능성ını gidermek ve performansı artırmak amacıyla kullanılır.

- Birden fazla thread'e sahip processler (multithread) birden fazla görevi yerine getirebilir.

Faydalari:

responsiveness: Multithread programlarında bir process işlem görürken aynı anda başka bir process'le paylaşılabilir. Örneğin olsak bir web sayfasında genetikken aynı anda müzik dinlenebilir.

resource sharing: Tüm kaynaklar (code, data, files) threadler arasında paylaştırılarak kullanılır. Böylece sistem daha etkili kullanılmış olur.

economy: Kaynaklar paylaştırıldığı için daha ekonomiktir.

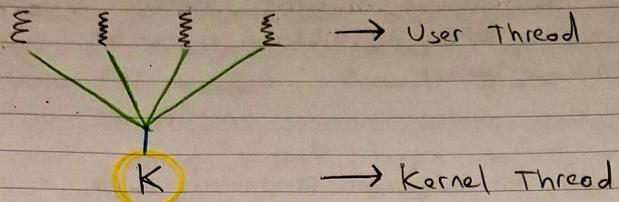
Utilization of multiprocessor architectures: Çok işlemeli bir yapıda multi-thread'ler her bir işlemci de paylaşılır. Örneğin olsak 4 thread ve 4 işlemeli bir sisteme her bir thread bir işlemci de işlen görür. Bu şekilde görev daha kısa sürede tamamlanabilir.

Multithreading Models

User Threads: Kullanıcı etrafındaki oluturları ve yönetilen thread'lerdir. İşletim sistemi çekirdeği tarafından bilmez.

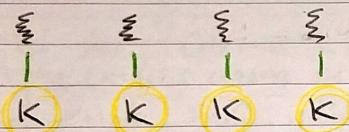
Kernel Threads: İşletim sistemi çekirdeği tarafından oluşturulur. Çekirdeğe tarafından bilinir.

Many-to-One-Model



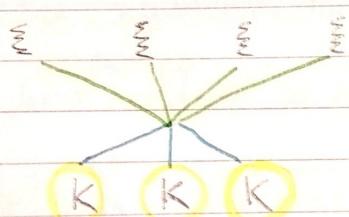
- ✓ Thread yönetimi kullanıcı düzeyinde yapılır bu seyede etkilidir.
- ✗ Eğer tek bir thread blokeleşse tüm sistem blokeleşir.
Çünkü sadece bir thread kernel' o erişim sağlayabilir.

One-to-One Model



- ✓ Bir thread blokeleşirse sistem çalışmaya devam eder. Çünkü diğer diğer thread'ler görevi devam ettirebilir.
- ✓ Çok işlenici bir sisteme her bir thread tek bir işlevi geliştirebilir.
- ✗ Yeni bir user-thread oluşturmak için one-to-one bir kernel thread oluşturmak gereklidir. Bu da sistemin maliyetini artırmır.

Many-to-Many Model



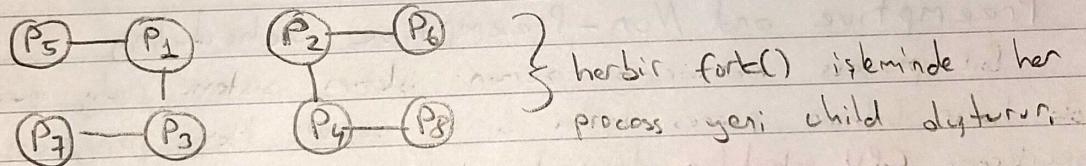
- ✓ Thread blokeleşirse sistem çalışmaya devam eder.
- ✓ Geliştiriciler gerektiği kadar yeni thread oluşturabilir.

fork() ve exec() Fonksiyonları

- fork(), yeni bir process oluşturur.
- exec(), kullanıldığı process'in içeriğini değiştirir. Yeni process oluşturmez.

```
int main()
{
    fork();
    fork();
    fork();
    printf("Merhaba dünya! PID = %d " getpid());
}
```

Kodun çıktısı
Merhaba dünya! PID = 5824
Merhaba dünya! PID = 5825



Toplam process sayısı = $2^n \rightarrow n = \text{fork sayisi}$

ex1.c

```
printf("PID ex1 = getpid();\n"
char args[] = {'Mehmet', 'Dünya'};\nexecv("./ex1.c", args);\nprintf('Back to ex1');
```

ex2.c

```
printf('ex2 dosyasında?');\nprintf("PID ex2 =
```

CPU Scheduling

Process execution, CPU execution ve I/O wait state'ları arasında sürekli bir döngü halindedir.

CPU burst → I/O burst → CPU burst → I/O burst →

Preemptive and Non-Preemptive Scheduling

Bir process çalıştığı zaman, işletim sistemi ready queue deki herhangi bulunan process'lerden birini seçer. Hangi programın seçileceği **CPU scheduler** tarafından belirlenir.

CPU zamanlaması konusunda dört durumda gerçekleşir:

- 1-) Bir process running state durumundan waiting state durumuna geçiş yapanında (I/O waiting).
- 2-) Bir process running state'den ready state geçtiğiinde (interrupt).
- 3-) Waiting state'den ready state geçtiğiinde (I/O terminasyonu).
- 4-) Bir process sonlandığında.

1. ve 4. durumlarda scheduling işin seçeneği yoktur. Yani bir process (ready queye dek bir hizmet bulmuyorsa) execution işin seçilebilir. Günlük CPU işin maksimum utilization (kullanım) sağlanmak şartındadır.

1. ve 4. durumlarda seçenek yoktur. Bu da **nonpreemptive** zamanlama,

2. ve 3. durumlarda seçenek vardır. Bu da **preemptive** zamanlama denir.