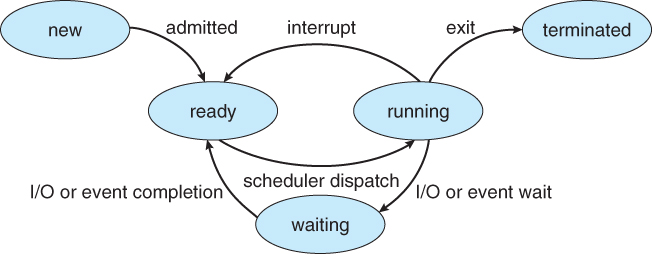
* Batch system – jobs
* Time-shared systems – user programs or tasks

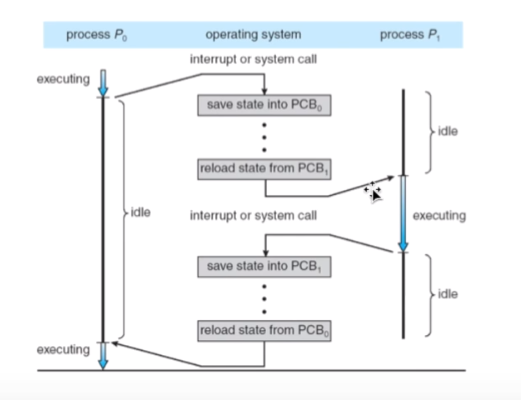
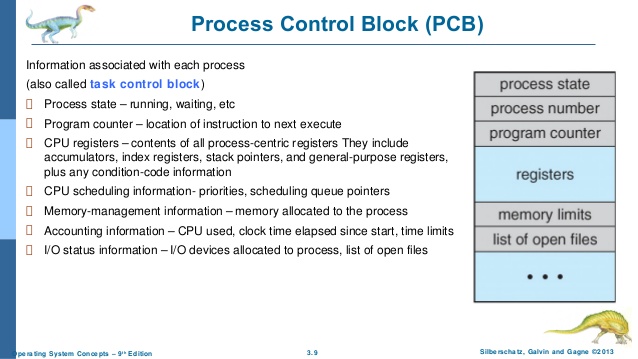
Bir islemcide ayni anda sadece 1 islem(process) calisabilir. Ayni anda program calisma durumunda aslinda bir tanesi calisiyor sonra o kesilip oteki calisiyor falan bu sekilde ilerliyor fakat aradaki gecisler o kadar hizli ki biz ayni anda calisiyor zannediyoruz. Cok islemcili sistemler olabilir. Ve ayni zamanda multithreading yapisi kullanilip bu sekilde islemler yaptirilabilir.

Bir programi derlediniz ve calistirdiniz. Programiniz RAM de yasiyor ve surekli CPU da sira bekliyor. CPU’ya alindikca calisip gorevini yerine getiriyor.

Diagram of Progress State

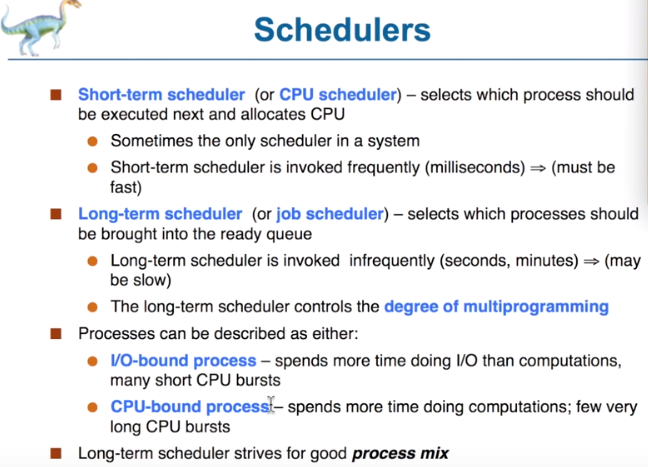
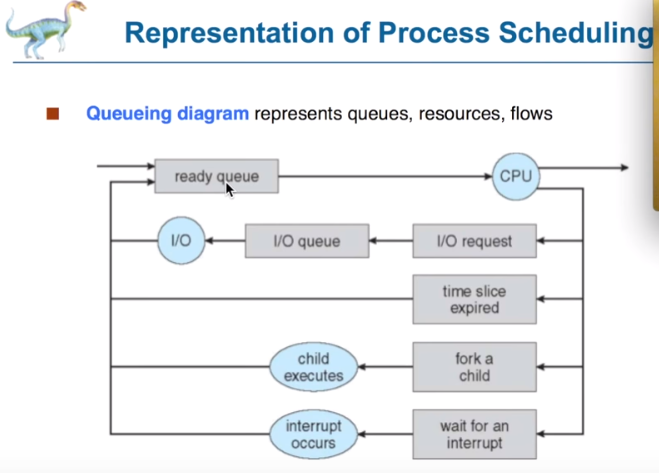
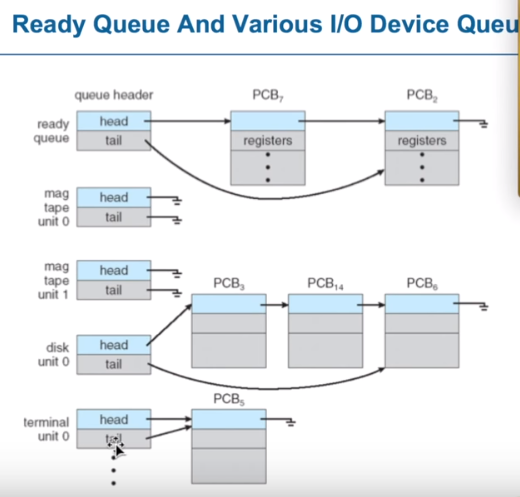


**Process Control Block (PCB)** – Process’in kunyesi gibi dusunulebilir. Onun boyle tanitim karti gibi o an ne yapiyor bu process ne ediyor process id’si ne falan bir suru bilgiye ulasmamizi saglayan blocklardir.



Process’I bir kol olarak dusunursek thread parmaklardir. Thread olabilmesi icin process olmasi gereklidir. Process olmasi icin thread gerekli degildir. Tek processli isletim sistemi – MS-DOS.

**Process Scheduling**



**I/O-bound process** = Yedek alma islemi mesela bol bol diskle hasir nesir olunacak. (Long-term)

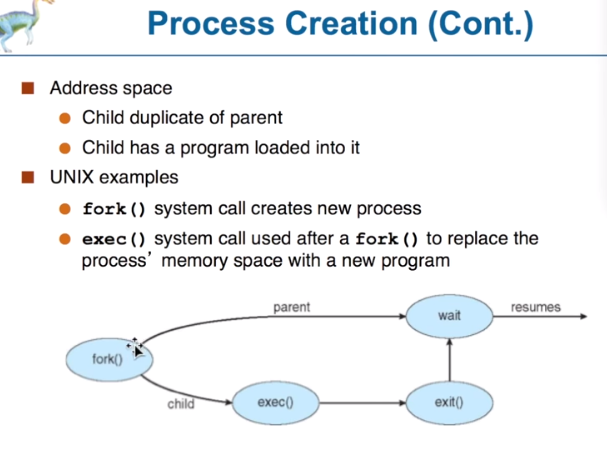
**CPU-bound process** = Oyun oynamak mesela CPU yu mesgul edecek. Mesela oyun oynarken bilgisayar size uyari gonderip bir dakika bir yedek alip sonra oyununuza devam edeceksiniz demiyor. (Short-term)

Medium-term scheduler = Mesela bir process geliyor ready queue’ye ve bekliyor. CPU onun calistirmaya basladiktan sonra mesela swap in veya swap out yapabilir. (Swapping). Yani su anlama geliyor veriyi RAM’de degil de diskte (virtual memory) saklayabilir bir sureligine ve daha sonra sirasi geldiginde yine RAM’e gelip ordan da CPU ile calisir.

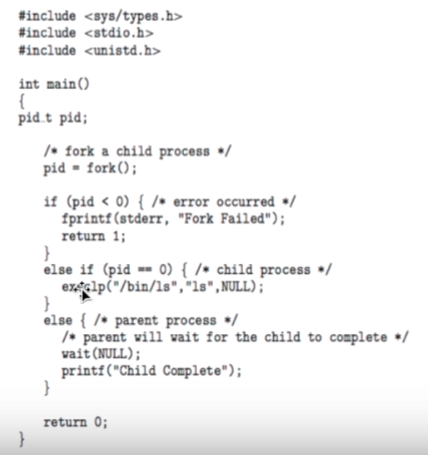
Long-term icin tum RAM bu islemlere ayrilir. Short-term icin RAM kucuk parcalara ayrilip tum hepsi ayni anda calistirilir. Medium-term ise Diski RAM gibi kullanip processlerin ayni anda calismasini saglar.

Siz telefonunuza bir uygulama yukluyorsunuz ve bu bellegin storage kisminda duruyor. Daha sonra siz uygulamayi calistirdiginizda bellegin RAM icin yani CPU’nun kullanabilecegi sekilde ayrilmis kismina geciyor uygulama ve bu sekilde calisiyor.

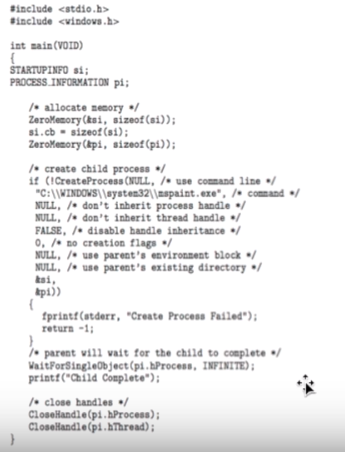
**Context-Switching** = Iste processlerin CPU tarafindan calistirilmasi sonra durumun kesilmesi kaydedilmesi ve bir baska processin calistirilmasi.



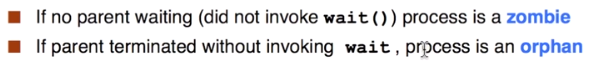
**LINUX ORNEGI**



**WINDOWS ORNEGI (Cok onemli degil ☺)**



Programimiz return ettiginde aslinda process biter ama return oncesinde exit() fonksiyonu ile terminate edebiliriz.



**Cascading Termination** – Mesela process agaci var iste processler, cocuklari falan buyuk bir yapi siz bu processleri kapatmak istediginizde en alt seviyeden baslayip kapatilir ve en son en ilk calistirilan en baba process kapatilir iste buna **cascading** denir.

Mesela siz excelde calisiyorsunuz bir suru tablolariniz var falan simdi excel’in kendisi zaten bir process olusturdugunuz tablolar falan da child lar simdi eger siz excel’i tablolari falan kapatmadan kapatmaya calisirsaniz isletim sistemi tarafindan sorun olabilir o yuzden tek tek dosyalari kapatip sonra excel’I kapatmaniz cok daha mantikli olur.

**Multiprocess Architecture** – Mesela Google Chrome’da acilan her yeni sekme yeni bir process olur. Bununla ilgili sikayet vardir hatta cok fazla kullaniyor bilgisayari kasiyor tarzinda. Diger browserlar thread kavrami ile bu islemi saglarken Chrome her acilan sekmeyi yeni bir process olarak olusturuyor. Browser ise ana processdir.

