KOCAELİ ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

Linux Ağ Yönetimi

12. Hafta – Processlerin Yaşam Döngüsü

Yrd. Doç. Dr. A. Burak İNNER

Kocaeli Üniversitesi Bilgisayar Mühendisliği
Yapay Zeka ve Benzetim Sistemleri Ar-Ge Lab.
http://yapbenzet.kocaeli.edu.tr

PROCESS(Süreç) OLUŞTURMA

Fork-Exec Komutları

- Yeni bir süreç oluşturma 2 aşamada gerçekleşir;
  - Fork:

    Bu aşamada yeni süreç(alt süreç) oluşturulur ve atalarının özelliklerini miras alır.
  - Execute:

    Bu aşamada yeni süreç(alt süreç) atalarından ayrılır ve kendi işlevselliğiyle devam eder.

Fork

Execute

Fork

- ‘Fork’ komutunun işi = Yeni bir süreç (alt süreç) oluşturmak ve onu çalıştırmak için ihtiyaç duyduğu şeylerle hazırlamaktır

- Bu komut ayrıca şu görevleri de içerir;
  - Yeni bir pid(process id) atama,
  - Çekirdekte(Kernel) ilişkili yapılar oluşturur,
  - ppid, pgid, sid ve diğer tüm nitelikleri ayarlar,
  - Sürecin çalıştırılacağı zaman dilimlerinin verilmesi için, çekirdek zamanlayıcı sırasına eklenir,
  - Ana bellek alanını (üst işlemle ilişkili tüm bellek alanları) alt süreç bellek alanına kopyalar,
  - Diğer ana süreç kaynaklarını kopyala,
    - Normal dosya işleyicileri(Normal file handlers),
    - Soket işleyicileri,
    - Paylaşılan hafıza(Shared Memory) işleyicileri,
    - ….

Exec

- "Exec"'in görevi = kendilerine atanan işleri yapması için alt süreçleri tek başlarına göndermektir.

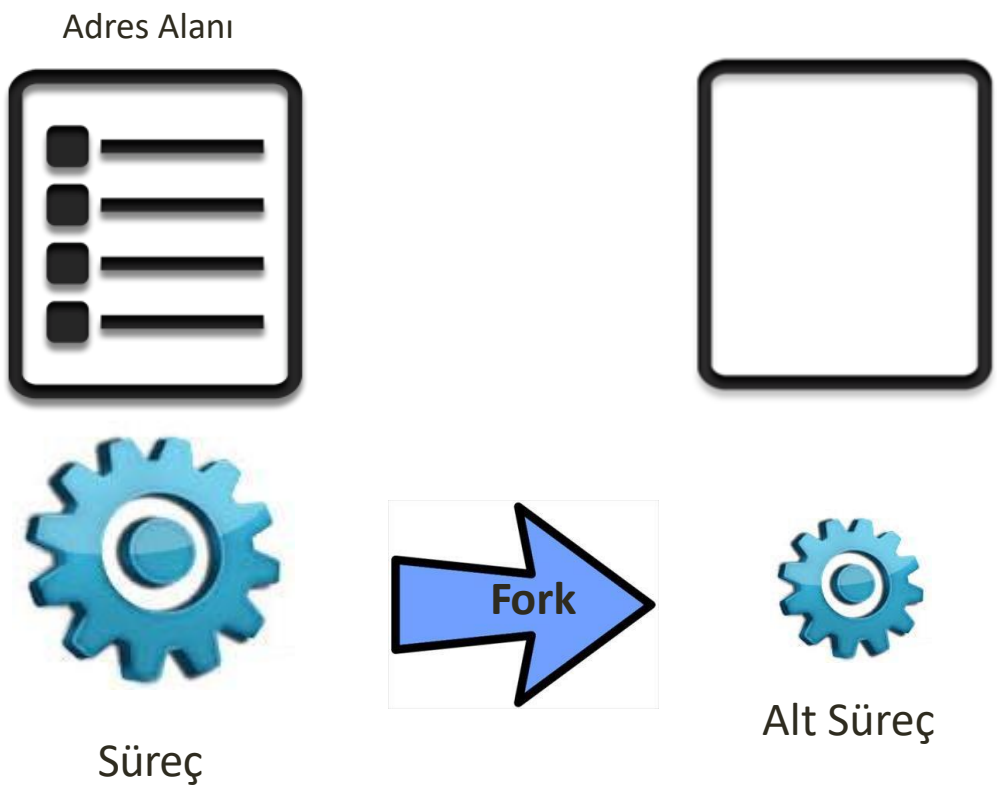
- Bu, alt süreçlerin şu özelliklerle birlikte yükleneceği anlamına gelir,
  - Ana sürecininki yerine kendi boş alanını kullanır,
  - Ana süreç kaynaklarına erişimi kaybolur,
  - Çalışacak olan yeni program için bir işaretçi(pointer)
- Bu komut alt süreçleri ana süreçlerden ayırır.
- Alt süreçler artık kendi hafıza adreslerini ve kendisine ayrılan/oluşturulan sistem kaynaklarını kullanarak , kendi başlarına çalışırlar.

Süreç Oluşturma: Ana Süreç

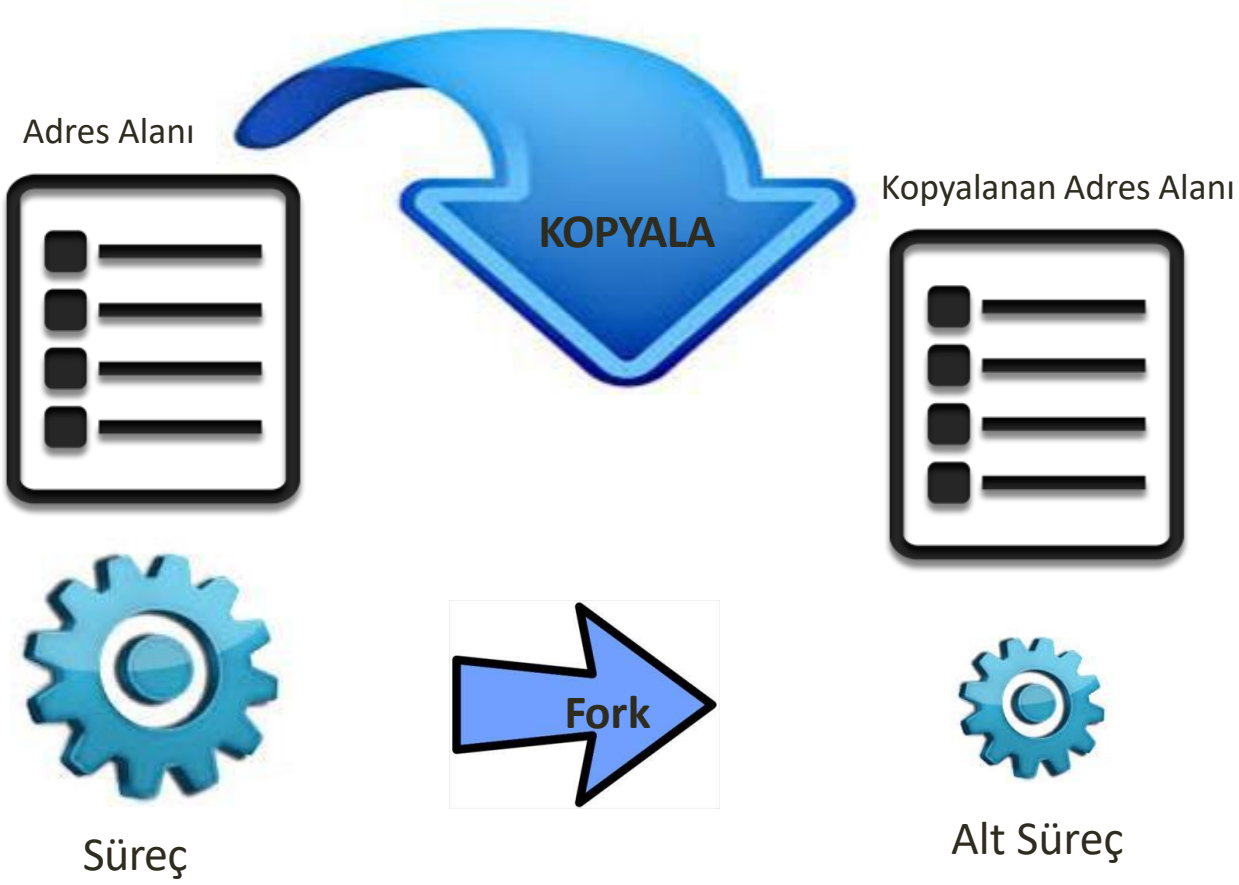
Adres Alanı

Süreç

Süreç Oluşturma: Fork Komutu



Süreç Oluşturma:Adres Alanını Kopyalama



Süreç Oluşturma: Exec Komutu

Adres Alanı

Süreç

Yeni Adres Alanı

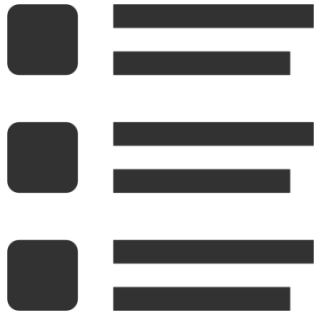
Alt Süreç

Süreç Oluşturma: Ana ve Alt Süreç

Adres Alanı

Süreç

Yeni Adres Alanı



Alt Süreç

Süreç(Process) Vs. İş Parçacığı(Thread)

Süreç vs. İş Parçacığı

- Hem süreçler hem de iş parçacıkları, çekirdekte zamanlayıcı yardımıyla paralel çalışmayı (çok görevli) sağlayan işlerdir.

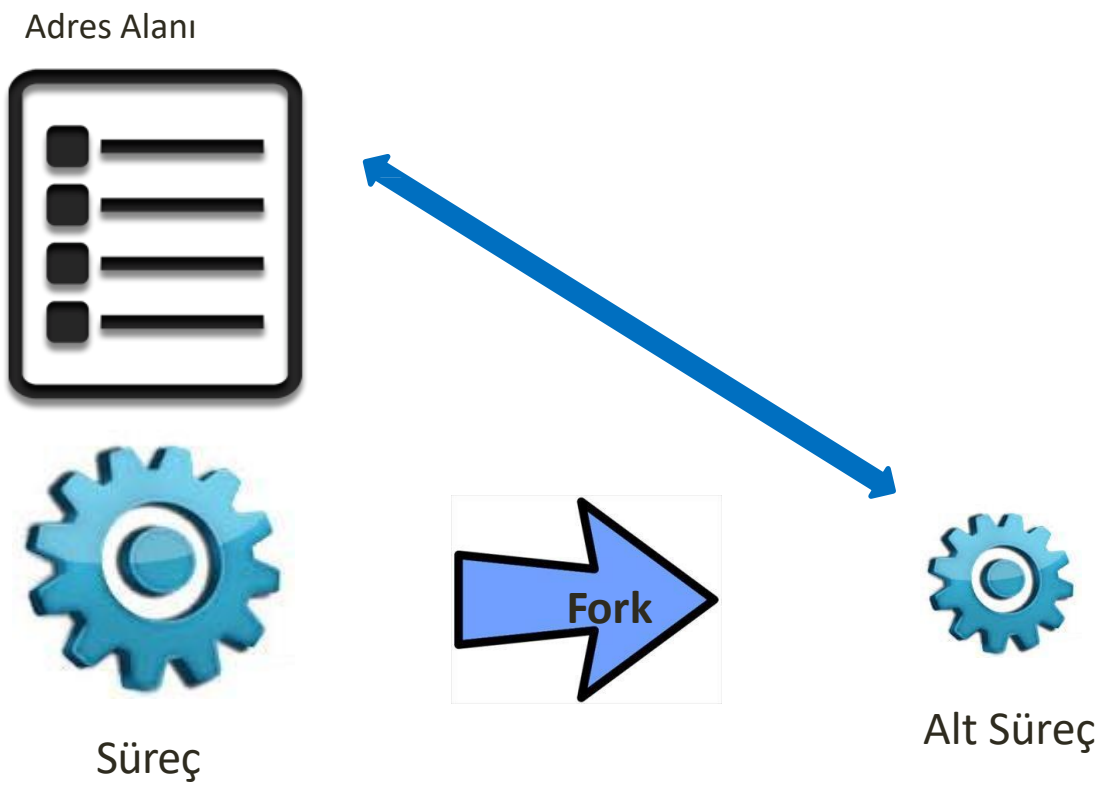
- Her iki durumda da, Linux çekirdeği, başka birilerine atanmadan önce süreç/iş parçacığı için yürütülecek zaman dilimleri atar.
- Linuxta iş parçacığı, özel farklılıkları olan bir süreçtir:
  - İş parçacığı oluşturulurken, adres alanları (ve bazı diğer kaynakları) ana süreçten kopyalanmaz. Ana süreç ile paylaşırlar.
  - Bu iş parçacığının ana süreç ile aynı adres alanını kullandığı ve kendisine özel adres alanının oluşturulmadığı anlamına gelir.
- Bunun dışında iş parçacıkları süreçlerle aynıdır.
  - Örneğin çekirdek zamanlayıcı, bağımsız olarak her iş parçacığı ile ilgilenir.
- Yani, aynı süreç tarafından oluşturulan iş parçacıkları, ana sürecin aynı adres alanını ve kaynaklarını paylaşan birden fazla süreçtir.

İş Parçacığı Oluşturma:Ana Süreç

Adres Alanı

Süreç

İş Parçacığı Oluşturma:Fork Prosedürü



Süreç Sonlandırılması

Süreç Sonlandırılması

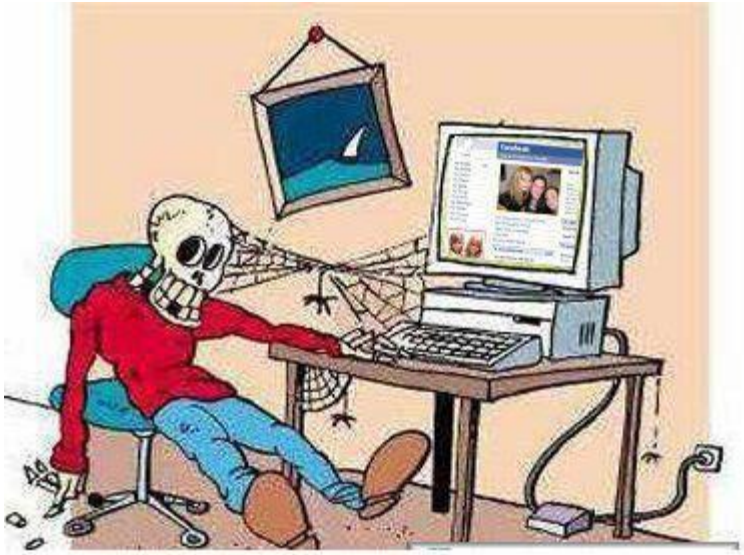
- Bir süreç sonlandırılması 3 şekilde gerçekleşebilir,
  - Normal sonlandırma (işlemin sonuna kadar),
  - Kendisini sonlandırmasına neden olacak bir Sinyal alma,
  - Çekirdek tarafından sonlandırılma
- Süreç sonlandırıldığında, kaynaklarının bir kısmı henüz serbest bırakılmaz, ve durumu Zombi durumuna gelir.
- Ana süreç, alt süreçlerinin gerekli kaynak temizlemelerini beklemelidir.
- Ana süreç gerekli temizlemesini yaptıktan sonra, süreç artık Zombi durumunda olmaz.
- Bu yüzden her sürecin yaşayan bir ana süreci olmalıdır. Eğer ana süreç sonlanırsa, alt süreçler ***init*** sürecinde yeniden ana süreç edinmelidirler.

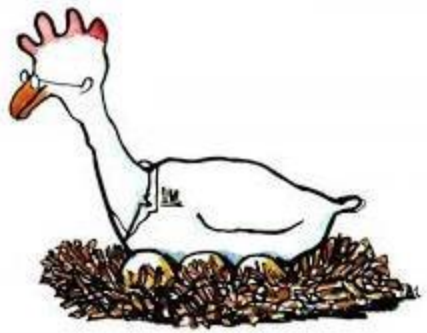
Süreç Durumları

Çalışan

Durmuş

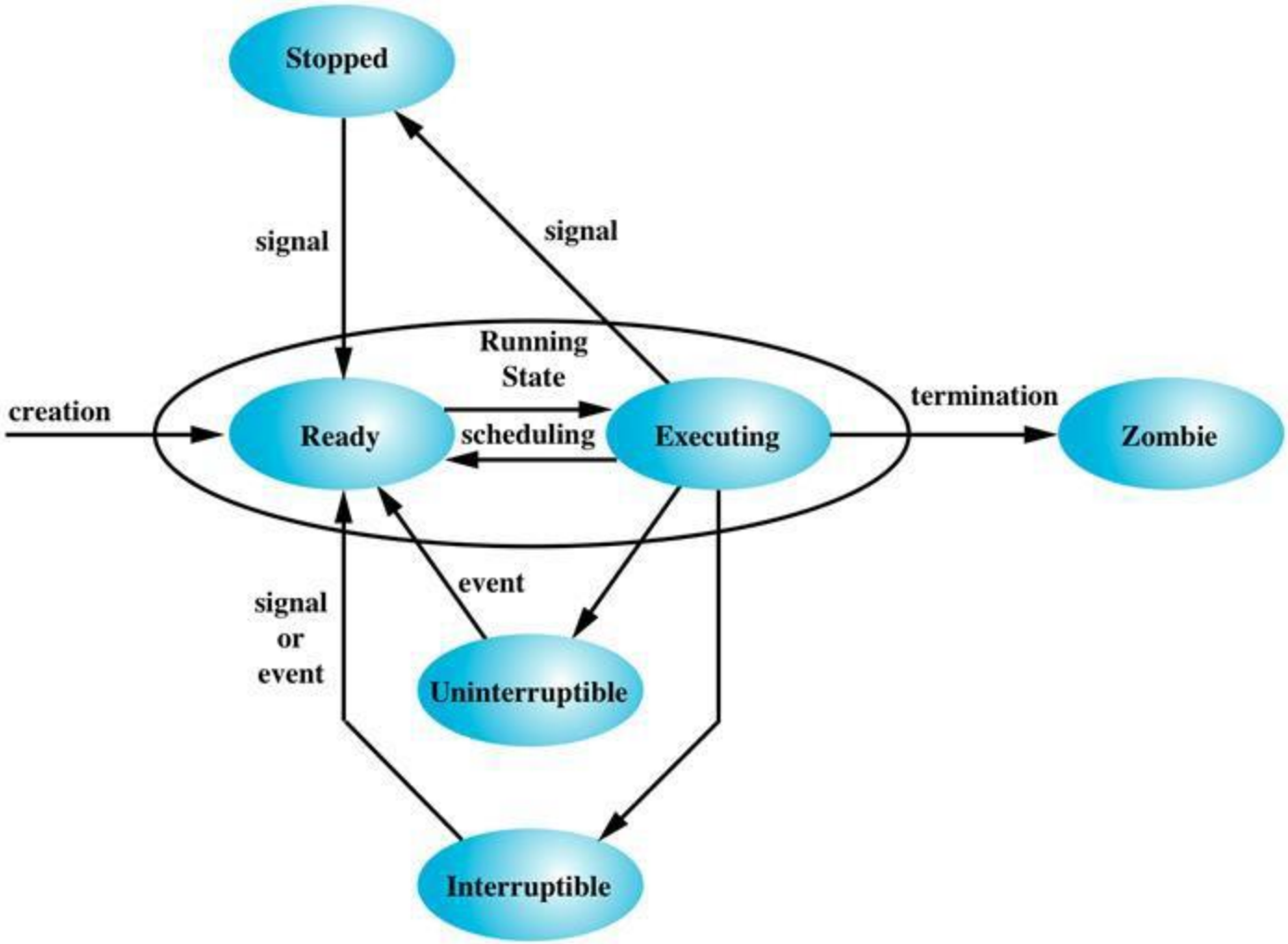
İş Bekleyen



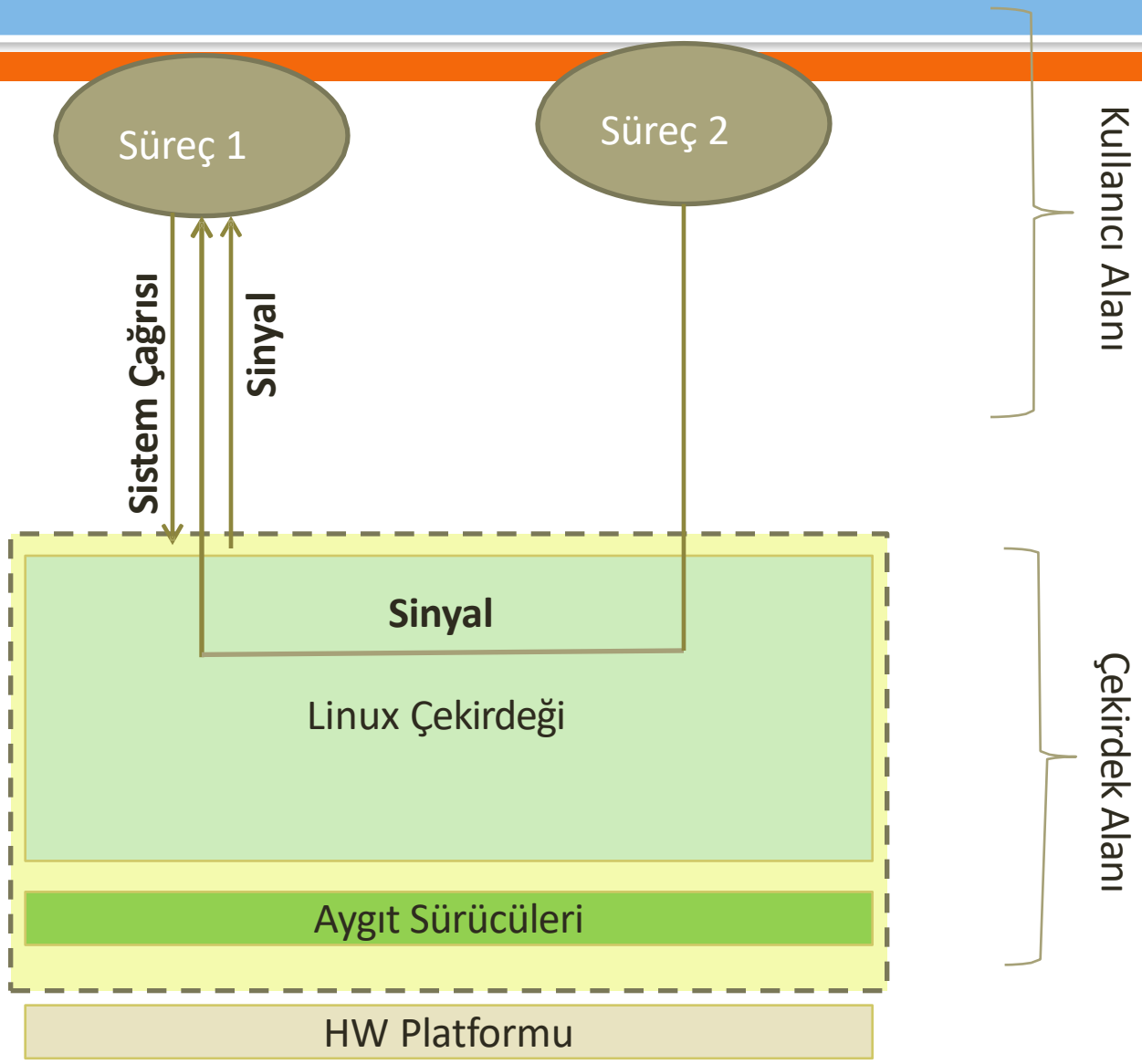


Zombi

Process Durumları



Sinyaller


Süreç 1
Süreç 2
Sistem Çağrısı
Sinyal
Kullanıcı Alanı
Sinyal
Linux Çekirdeği
Aygıt Sürücüleri
Çekirdek Alanı
HW Platformu

Sinyal Nedir ??

- Linux un iki düzlemi vardır:
  - Çekirdek düzlemi: Sistem ve donanım kaynaklarına erişilen düzlem.
  - Kullanıcı düzlemi: Uygulamaların çalışabileceği düzlem.
- Kullanıcı uygulamalarının çekirdeğe direk erişim adresleri yoktur.
- Kullanıcı uygulamaları çekirdek ile iletişim kurabilmek için **Sistem Çağrıları**na ihtiyaç duyarlar.
- Peki çekirdek, kullanıcı uygulamasına bir istek/komut/bildirim/… göndermeye ihtiyaç duyarsa ??
- Bu durumda kullanılan mekanizma **Sinyaller**dir.
- Yani Sinyaller, çekirdeğin kullanıcı uygulamaları ile iletişimini sağlayan mekanizmadır.
- Sinyaller ayrıca kullanıcı uygulamalarının kendi arasındaki iletişimde de kullanılır.(Bir uygulamanın diğer uygulamaya istek/komut gönderme ihtiyacı olduğunda)

Sinyal Nedir ??

- Sinyaller, Linux Çekirdeği tarafından kullanıcı düzleminde çalışan bir süreçle iletişim kurmak için kullanılan iletişim kanalıdır.
- Bunlar çekirdek tarafından bir süreci öldürmek için, durdurmak için, devam ettirmek için, zamanlayıcının dolduğunu bildirmek için, vb…
- Ayrıca sinyaller kullanıcı düzeyinde çalışan diğer süreçler tarafından da gönderilebilir. Örneğin,
  - Kullanıcı çalışan uygulamayı ***Ctrl-c*** tuşlarıyla sonlandırmak istediğinde, veya ***Ctrl+z*** tuşlarıyla durdurmak istediğinde
  - Kullanıcı arka plandaki durmuş olan uygulamayı devam ettirmek için "***bg***" komutunu kullandığında, veya ön plandakini devam ettirmek için "***fg***" komutunu kullandığında
  - Kullanıcı çalışan uygulamayı sonlandırmak için "***kill***" komutunu kullandığında
- Tüm bu durumlarda, sinyal gerekli görevi yerine getirmek için hedef uygulamaya gönderilir.
- Diğer kullanıcı uygulamalarından gönderildiğinde, önce sistem çağrısıyla çekirdeğe aktarılır ve bu da istenen işleme geri gönderilir.

Sinyaller

- Sinyaller hem Unix Sistemlerinde hem de POSIX Standardında tanımlanmıştır
- Sinyaller için bir tablo vardır. Her sinyalin,
  - Sinyal numarası
  - Sinyal ismi (‘SIG’ ile başlayan) vardır.
- (1-31) arasındaki sinyaller normal sinyallerdir.
- 31’den büyük olan sinyaller “**Gerçek Zamanlı Sinyaller**” olarak adlandırılır.

Bir Sürece Sinyal Gönderme (kill Komutu)

$ kill <pid>
$ kill -<Sinyal Numarası> <pid>
$ kill <Sinyal Numarası> <pid>

- Kill komutu istenen sürece bir sinyal gönderir

- Bununla birlikte, komut adının 'kill' olmasına rağmen, farklı amaçlara sahip olan ve yalnızca işlemleri öldürmek için kullanılmayan her türlü sinyal için kullanılır.
- Sinyal belirtilmemişse, varsayılan olarak SIGTERM sinyali gönderilir.
- Örnek:
  $ kill -9 1185
  $ kill SIGKILL 1185

Aynı komutlar !!!!

“öldürmek(kill)” İçin Diğer Yollar

Süreci İsmine Göre Seçme(killall Komutu)

$ killall <Sinyal> <komut adı>

- ***Killall*** komutu, belirtilen komutu çalıştıran tüm proseslere istenen sinyali gönderir.

  $ killall chrome
- Bu yöntem pid'sini aramadan bir süreç seçmek için kullanışlıdır.
- Ayrıca çalışan programın tüm örneklerine sinyal göndermek için de kullanışlıdır.
- Sinyal belirlenmezse, **SIGTERM** devreye girer.
- Örnek:

  $ killall -9 gedit (Bütün "gedit" örneklerine SIGKILL sinyali gönderir)

Diğer Kriterlere Göre Süreç Seçme (pkill Komutu)

$ pkill <sinyal> -u <kullanıcıadı>
$ pkill < sinyal > -P <ana süreç>
$ pkill < sinyal > <komut düzeni>
$ pkill < sinyal > -t <Terminal ismi>

- ***pkill*** komutu ilgili farklı ölçütlere uyan süreçlere istenen komutu gönderir.
- Örneğin:
  - Tom kullanıcısına sahip tüm süreçlere SIGTERM sinyali gönder.
    $ pkill -u tom
  - tty1 terminalinde çalışan tüm süreçlere SIGSTOP sinyali gönder.
    $ pkill SIGSTOP -t tty1
  - pid'si 1107 olan ana sürecin tüm alt süreçlerine SIGTERM sinyali gönder.
    $ pkill –P 1107

GUI Süreçlerini Mouse ile Öldürme (xkill Komutu)

$ xkill

- xkill komutu, kullanıcıya öldürmek istediği süreci mouse ile öldürme izni sağlar.
- Komut yazılır, daha sonra öldürülmek istenen ekrana arayüzden tıklanır.

Askıdaki GUI Uygulamalarıyla Başa Çıkma

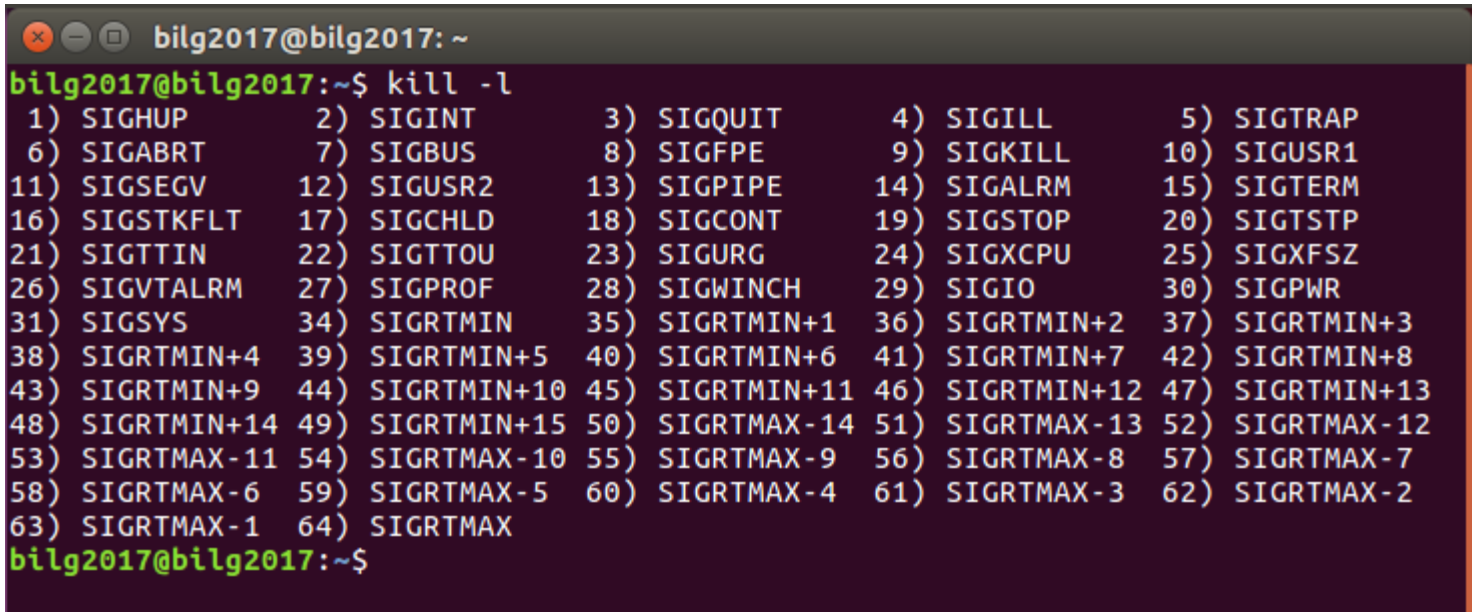
- Eğer GUI uygulaması askıdaysa, ve onu öldürmen gerekliyse ama askıda ekrandan dolayı öldüremiyorsan:
  - Linux sanal terminali via‘ ya geç,

    [Ctrl] [Alt] [F1]
  - Sanal terminalden, ilgili “kill” komutu gir,

    $ killall -9 chrome
  - İşlem bittikten sonra, tty-7‘ e geri dön

    [Alt] [F7]

Sinyal Numaralarının Listelenmesi



Gerçek Zamanlı Sinyaller

- Numarası 31'den büyük olan sinyallerdir.
- Ön tanımlı fonksiyonları yoktur ve kullanıcı/geliştiricilerin kullanması için bırakılmıştırlar.
- Programcı tarafından tanımlanırlar.

Sinyal Göndermek için Tetikleyiciler

Klavye Kısayol Tuşları

- Bazı sinyaller kullanıcıların kısayol tuşlarını kullanmasıyla tetiklenirler.
- Çekirdek bu kısayol tuşlarını algıladığında, ilgili sinyali çalışan sürece gönderir(veya çalışan süreç grubuna)
- Örnek:
  - ***Ctrl-c*** ***SIGINT*** komutunu tetikler
  - ***Ctrl-z*** ***SIGTSTP*** komutunu tetikler
  - ***Ctrl-*** ***SIGQUIT*** komutunu tetikler

Diğer Süreçlerden İstekler

- Bir süreç, çekirdeği, bir sistem çağrısı yaparak başka bir sürece sinyal göndermek üzere tetikleyebilir
- Bunun için örnek, kullanıcı terminalden komutlar göndermesidir.

  $ kill (Sinyali göndererek tetiklenir, ***SIGTERM*** varsayılan olarak)

  $ fg (Sinyali göndererek tetiklenir, ***SIGCONT***)

  $ bg (Sinyali göndererek tetiklenir, ***SIGCONT***)
- Bir süreç, başka bir sürece sinyal gönderilmesini de bazı özel fonksiyon API'ları vasıtasıyla programatik olarak tetikleyebilir

Donanım Olayı

- Donanım işlevi sürece sinyal göndererek çekirdeği tetikleyebilir.
- Buna örnek olarak: Çekirdek modemdeki bağlantı kesintisini algıladığında, oturumdaki bütün süreçlere ***SIGHUP*** sinyali gönderir.

İlgili Bir Sürecin Ölümü

- Bir süreç sonlandığında, çekirdek şu sinyaller gönderir,
  SIGHUP sinyali bütün alt süreçlere, ana sürecin ölümünü bildirmek üzere gönderilir.
  - ***SIGCHLD*** sinyali ana sürece, sonlanmış olan alt süreçleri bildirmek üzere gönderilir.

Hata ve İstisnalar

- Bazı süreç istisnaları, sürece sinyal gönderilmesi için çekirdeği tetikler.
- Örnek:
  - Eğer süreç illegal bir komut çalıştırmaya çalışırsa, çekirdek sürece ***SIGILL*** sinyalini göndermek üzere tetiklenir.
  - Eğer süreç Kayan Nokta Hatası'ndan dolayı zarar görürse, çekirdek ***SIGFPE*** sinyali gönderir.
  - Eğer süreç boruya(pipe) yazmaya çalışıyorsa, diğer yandan da dinlemiyorsa, çekirdek ***SIGPIPE*** sinyali gönderir.
  - Eğer süreç Boş İşaretçiye(Null Pointer) erişirse(veya diğer hafıza hataları), çekirdek ***SIGSEGV*** sinyali gönderir.
  - Eğer süreç arka planda standart giriş/çıkış a okuma/yazma yapmaya çalışıyorsa, çekirdek ***SIGTTIN***/***SIGTTOUT*** sinyali gönderir.
  - Eğer süreç bir hatadan çıkmak için ***abort()*** kullanırsa, çekirdek ***SIGABRT*** sinyali gönderir.

Çekirdek Tarafından Bildirim

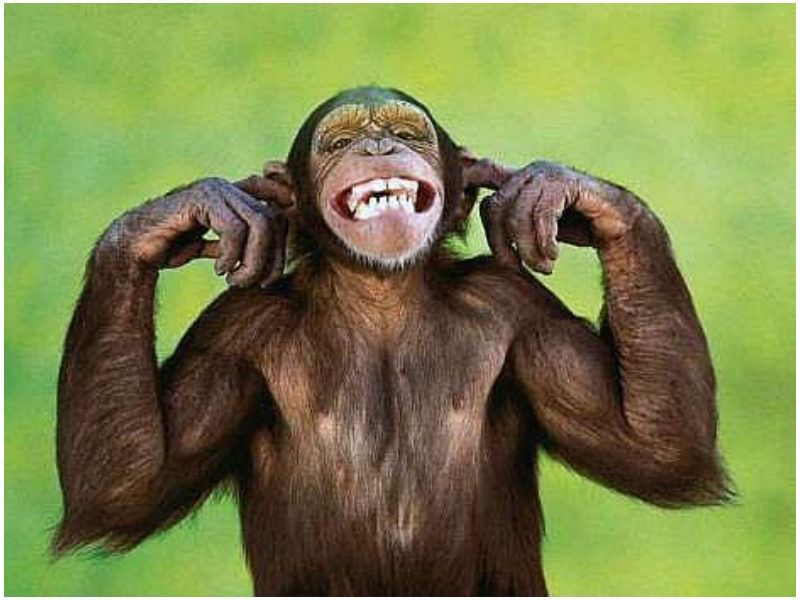
- Bazen, bazı olaylar gerçekleşirken süreç çekirdekten bildirim isteyebilir.
- Normalde, süreç bu bildirim için bekler.
- Çekirdek bu bildirimi bir sinyal aracılığıyla ulaştırır
- Örnek:
  - Çekirdek zamanlayıcının tükendiğini bildirmek için ***SIGALRM*** ve ***SIGVTALRM*** sinyallerini kullanır.
  - Süreç bir dosya işleyicisine erişildiğinde çekirdekten bildirim ister.

Süreçlerin Sinyallere Cevabı

Varsayılan Davranış

- Bir sinyal sürece eriştiğinde, varsayılan eylemi gerçekleştirir.
- Her sinyalin kendi varsayılan davranışı açıklamasında yer almaktadır.
- Örnek:
  - Bazı sinyallerin varsayılan sonlandırma eylemi vardır. (hassas sonlandırma). Bu ***SIGTERM***, ***SIGPIPE***, ***SIGUSR1***, ***SIGUSR2*** gibi birçok sinyal için geçerlidir.
  - Bazı sinyallerin varsayılan olarak süreçler tarafından göz ardı edilme eylemleri vardır. ***SIGCHLD*** gibi.
  - Bazı sinyaller süreçlere kendi varsayılan davranışını değiştirme izni vermez. Bu da sadece izin verilen, varsayılan davranışı yaptırma zorunluluğu kılar. Süreçleri durmaya zorlayan ***SIGSTOP*** komutu ve süreci derhal durdurma komutu ***SIGKILL*** da buna dahildir.

Sinyali Göz Ardı Etme



- Bir süreç belirli sinyalleri göz ardı edecek şekilde ayarlanabilir. Bu sinyaller alındığında buna göre bir işlem gerçekleştirilmez.
- Bazı sinyallerin göz ardı edilmesi mümkün değildir. ***SIGSTOP*** ve ***SIGKILL*** gibi

Sinyali Bloklama

- Bir süreç bazı sinyalleri bloklayacak şekilde ayarlanabilir.
- Bir sinyali bloklamak, sonra işletilmek üzere sıraya koymak anlamına gelir.(bloklama kalktığında tekrar çalışabilir)
- Bu durum, süreç kesilmemesi gereken kodlar çalıştırırken oluşur.
- Her sürecin hangi sinyali bloklayacağını belirten bir maskesi vardır.
- Sadece belirli bir tipte sinyal bloklanabilir.
- Son kural gerçek zamanlı sinyaller için geçerli değildir.(numarası 31 den büyük olanlar)
- Bazı sinyaller bloklanamaz.(***SIGSTOP*** ve ***SIGKILL***)

Sinyali Yakalama

- Bir süreç varsayılan işleyicisi çalışmadan önce sinyali yakalar, ve özel işleyicisine gönderir.
- Varsayılan olarak göz ardı edilen sinyaller için daha kullanışlı bir durumdur.(***SIGCHLD***)
- Bazı sinyaller yakalanamaz (***SIGSTOP*** ve ***SIGKILL***)

Popüler Sinyaller

Hang-Up Sinyali (1) SIGHUP

- Kapatma sinyali, çekirdek tarafından, oturum kapatıldığında bir tty oturumunda çalışan süreçleri bildirmek için kullanılır. Şu sebeplerden oluşabilir,
  - Oturum modeminin bağlantısının kesilmesi,
  - Makineye bağlanmak için kullanılan bağlantının kesilmesi
- Genellikle daemon süreçlerinin yapılandırma dosyalarını yeniden okumasını istemek için kullanılır
- Bu sinyalin süreçler için varsayılan eylemi sonlandırmadır.

Interrupt(Kesme) Sinyali(2) SIGINT

- Kesme sinyali kullanıcı ***Ctrl-c*** kısayol tuşunu kullandığında tetiklenir.
- Bu çekirdeğin o an çalışan süreç grubuna ***SIGINT*** sinyalini göndermesine sebep olur.
- Bu sinyalin süreçler için varsayılan eylemi sonlandırmadır.

Çıkış Sinyali (3) SIGQUIT

- Çıkış sinyali kullanıcının ***Ctrl-*** kısayol tuşlarını kullanmasıyla tetiklenir.
- Çalışan bütün süreç gruplarına ***SIGQUIT*** sinyali göndermesi için çekirdeği tetikler.
- Varsayılan eylemi çıkış ve çöplük dosyası üretmedir.(derleme işlemi için uygundur)

İllegal Komut Sinyali (4) SIGILL

- İllegal komut sinyali, süreç doğrulanmayan bir komut gerçekleştirdiğinde tetiklenir.
- Kodda veya yığın belleğindeki hafıza bozulmalarında gerçekleşebilir.
- Varsayılan eylemi sonlandırma ve çöplük dosyası üretmedir.

İptal Sinyali (6) SIGABRT

- Abort sinyali, programın yanlış bir çıkışını belirtmek için ***abort()*** fonksiyonunu çağırdığında, çekirdek tarafından sürece gönderilir.
- Varsayılan eylemi sonlandırma ve çöplük dosyası üretmedir.

Tuzak Sinyali (5) SIGTRAP

- Tuzak sinyali genelde derleyicilerde ve program izleyicilerinde kullanılır.

Bus(Yol) Hata Sinyali (7) SIGBUS

- Veri yolu hata sinyali, hafızaya yanlış girme girişiminde bulunan bir program hatası tarafından tetiklenir
- Buna bellek erişimindeki hizalama hataları sebep olabilir.
- Varsayılan eylem sonlandırma ve çöplük dosyası oluşturmadır.

Kayan Nokta Hatası(8) SIGFPE

- Bir süreç kayan nokta komutu çalışırken istisna oluşursa, çekirdek tarafından sürece kayan nokta hata sinyali gönderilir.
- Varsayılan eylemi sonlandırma ve çöplük dosyası oluşturmadır.

Öldürme(Kill) Sinyali(9) SIGKILL

- Bir, başka süreçten gelen istekle açıkça öldürülür.
- İstek, işlemi temizleme şansı vermeden derhal ve zorla öldürmektir
- Bir süreç bu sinyali göz ardı edemez, bloklayamaz veya yakalayamaz.

Durdurma Sinyali(17 &18) SIGSTOP & SIGTSTP

- ***SIGTSTP*** (terminal durdurma) ***Ctrl-z*** tuşlarıyla tetiklenir.
- Bu sinyal sürecin(veya süreç grubunun) operasyonu askıya almasına sebep olur.
- ***SIGSTOP*** sinyali ***SIGTSTP*** sinyali ile aynı etkiye sahiptir. Tek fark ***SIGSTOP*** komutunun gözardı edilememesi/bloklanamaması/yakalanamamasıdır.
- SIGSTOP ve SIGTSTP komutlarından biri ile durdurulan süreçler, ***SIGCONT*** komutunu beklerler.

Sonlandırma Sinyali(15) SIGTERM

- Sinyal belirtilmediğinde, sonlandırma sinyali, ***kill*** komutu için varsayılan sinyaldir
- ***SIGTERM*** sinyalinin varsayılan eylemi, süreci temizlendikten sonra hassasça sonlandırmaktır.

Kaynakça

- Ahmed ElArabawy, Linux for Embedded Systems for Arabs

Teşekkürler.

Dersin Sonu

Kocaeli Üniversitesi Bilgisayar Mühendisliği
Yapay Zeka ve Benzetim Sistemleri Ar-Ge Lab.
http://yapbenzet.kocaeli.edu.tr/