Sistem Programlama

Ders 13

Doç. Dr. Mehmet Dinçer Erbaş
Bolu Abant İzzet Baysal Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

İşlem kontrolü

- Bu bölümde Unix sistemi tarafından işlemlerin nasıl kontrol edildiğini öğreneceğiz. Bu konu aşağıda belirtilen işlemleri içerir:
  - Yeni işlemlerin oluşturulması
  - Işlemlerin çalışması
  - Işlemlerin sonlanması
- Ayrıca işlemlere ait farklı kullanıcı numaraları, gerçek, efektif ve kaydedilmiş, ve bunların işlem kontrolü araçlarına etkilerini öğreneceğiz.

İşlem tanımlayıcıları

- Her işlem kendine ait bir işlem numarasına (process id) sahiptir.
  - Bu sayı negatif olmayan bir tam sayıdır.
- Her işlemin işlem numarası kendine aittir. Bu sebeple uygulamalar bu numarayı kullanarak sadece işleme ait tanımlayıcılar oluşturabilir.
  - Örneğin bazı uygulamalar, işlem numarasını dosya isminin parçası olarak kullanır ve böylece sadece işleme ait dosya isimleri oluşturur.
- Bu numaralar emsalsiz olmasına rağmen tekrar kullanılabilir.
  - Bir işlem sonlandığında, işlemin numarası sonradan oluşturulacak işlemlere verilebilir.
- Unix sistemlerinde genellikle işlem numaralarının tekrar kullanılmasını geciktirecek algoritmalar bulunur.

İşlem kontrolü

- Sistemden sisteme farklılık göstermekle birlikte, Unix sistemlerinde bazı özel işlemler mevcuttur.
- İşlem numarası 0 genellikle görev zamanlayıcısı işlemidir (scheduler).
  - Bu işlem ayrıca swapper olarak bilinir.
- İşlem numarası 1 genellikle init işlemidir.
  - init işlemi başlatma sırasında kernel tarafından çalıştırılır ve işletim sisteminin parçalarını çalıştıran işlemdir.
  - init işlemini başlatan program eski Unix versiyonlarında / etc / init programıyken, yeni versiyonlarda / sbin / init dosyasıdır.
  - Kernel yüklendikten sonra init işlemi başlatılır ve Unix sisteminin parçaları bu işlem tarafından başlatılır.
  - init işlemi asla sonlanmaz.
    - Birazdan göreceğimiz üzere yetim kalan işlemlerin üst işlemi (parent process) init işlemi olur.

İşlem kontrolü

- İşlem numarasından başka, işlemlere ait başka tanımlayıcılar mevcuttur. Bu tanımlayıcılara aşağıdaki fonksiyonlar ile ulaşılabilir.

```
#include <unistd.h>

pid_t getpid(void);
```

Dönüş: Çağıran işlemin işlem numarası

```
pid_t getppid(void);
```

Dönüş: Çağıran işlemin üst işleminin numarası

```
uid_t getuid(void);
```

Dönüş: Çağıran işlemin gerçek kullanıcı numarası

```
uid_t geteuid(void);
```

Dönüş: Çağıran işlemin efektif kullanıcı numarası

```
gid_t getgid(void);
```

Dönüş: Çağıran işlemin gerçek grup numarası

```
gid_t getegid(void);
```

Dönüş: Çağıran işlemin efektif grup numarası

fork fonksiyonu

- Çalışmakta olan bir işlem fork fonksiyonunu kullanaak yeni bir işlem oluşturabilir.

```
#include <unistd.h>

pid_t fork(void);
```

Dönüş: OK ise alt işlemde 0, üst işlemde alt işlemin işlem numarası, hata ise -1.

- fork fonksiyonu tarafından yeni oluşturulan işleme alt işlem denir.
- Bu fonksiyon bir kere çağırılır ancak iki kere dönüş yapar.
- Dönüş değerlerinde fark şu şekildedir: Alt işlemde dönüş değeri 0 olur. Üst işlemde alt işlemin işlem numarası döner.
  - Bu şekilde bir işlem birden fazla alt işlem oluşturabilir.
  - Bir işlemin sadece bir üst işlemi olabilir ve üst işlemin işlem numarasına getppid fonksiyonu ile ulaşabilir.

fork fonksiyonu

- Hem üst işlem hem de alt işlem fork fonksiyon çağrısını takip eden komutları çalıştırmaya devam eder.
  - Alt işlem, üst işlemin kopyasıdır. Alt işlem, üst işlemin veri alanı, heap ve yığın kısımlarının kopyasını alır.
  - Burada unutulmaması gereken alt işlemin yeni bir kopyaya sahip olduğudur. Yani alt işlem ile üst işlem hafızanın bu bölümlerini paylaşmaz.
  - Alt işlem ve üst işlem metin bölümünü paylaşır.
  - Notlar
    - Birçok sistemde üst işlemin veri, yığın ve heap kısımları hemen kopyalanmaz. Çünkü genellikle fork fonksiyonunu exec çağrısı takip eder.
    - Bu bölümler alt ve üst işlem tarafından paylaşılır ve korunma durumları “sadece-okunur” olarak ayarlanır.
    - Her iki işlemden biri bu bölümleri değiştirmek isterse, değiştirilen bölümün kopyası oluşturulur (fork1.c).

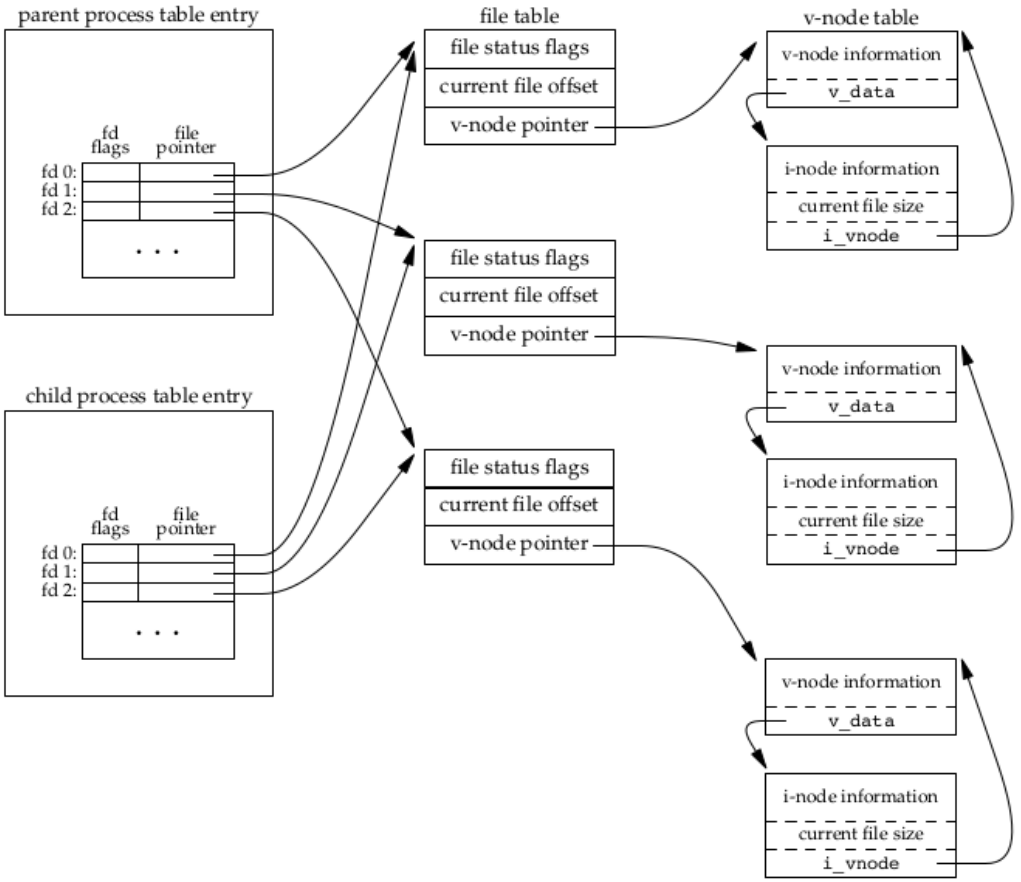
fork fonksiyonu

- Genellikle, üst işlemin mi alt işlemin mi önce çalışmaya başlayacağını bilemeyiz.
  - Kernel tatafından kullanılan planlama (scheduling) algoritması buna karar verir.
- Örneğimizde üst işlemi 2 saniye uyutmuştuk. Ancak bu bile alt işlemin önce çalışacağını garanti etmez.
- Örneğimizde standart çıktıya yazarken sondaki null byte yazılmaması için önbellek büyüklüğünden 1 çıkardık.
- Ayrıca I/O fonksiyonları ile fork arasındaki ilişkiye dikkat edelim.
  - Hatırlayacağınız üzere standart çıktı teminale yönlenmişse satır önbellekli, aksi takdirde tam önbelleklidir.

fork fonksiyonu

- Dosya paylaşımı
  - Örneğimizde görüldüğü üzere üst işlemde standart çıktıyı bir dosyaya yönlendirdiğimizde, alt işlem için de standart çıktı dosyaya yönlendi.
  - Bu durum fork ile yeni işlem açıldığında geçerlidir.
    - Açık olan dosya belirteçleri kopyalanır.
  - Bir işlemin standart girdi, standart çıktı ve standart hatayı açık tuttuğunu ve bu işlemin fork fonksiyonunu çağırdığını farzedelim.

fork fonksiyonu



fork fonksiyonu

- Eğer hem üst işlem hem de alt işlem, üst işlemin alt işlemi beklemesi benzeri herhangi bir senkronizasyon olmadan, aynı dosya belirteçlerini kullanarak yazma işlemi yaparlarsa, iki işlemin çıktıları birbirine karışır.
- Böyle karışıklıkları engelemek için fork sonrası dosya belirteçleri için iki farklı yöntem kullanılır.
  - Üst işlem alt işlemin tamamlanmasını bekler. Bu durumda üst işlemin dosya tanımlayıcıları ile ilgili bir işlem yapmasına gerek kalmaz. Alt işlem bittiğinde gerçekleştirdiği yazma ve okuma işlemlerine uygun olarak dosya ofseti yenilenir.
  - Üst ve alt işlemler kendi yollarına giderler. Bu durumda fork sonrası üst işlem ihtiyaç duymadığı dosya belirteçlerini kapar ve alt işlem de aynısını yapar. Bu sayede kimse diğerinin açık dosya belirteçleri ile ilgili işlem yapmaz. Bu senaryo ağ sağlayıcılarında sıklıkla görülür.

fork fonksiyonu

- Açık dosyalar dışında aşağıda belirtilen özellikler üst işlemden alt işleme geçer.
  - Gerçek kullanıcı numarası, gerçek grup numarası, efektif kullanıcı numarası, efektif grup numarası
  - Destekleyici grup numarası
  - Işlem grup numarası
  - Oturum numarası
  - Kontrol eden terminal
  - set-user-id ve set-group-id bayrakları
  - Mevcut çalışma klasörü
  - Kök klasör

fork fonksiyonu

- Açık dosyalar dışında aşağıda belirtilen özellikler üst işlemden alt işleme geçer (devam).
  - Dosya modu oluşturma maskesi
  - Sinyal maskesi
  - Açık dosya belirteçlerinin close-on-exec bayrağı
  - Çevre
  - Eklenmiş paylaşılan hafıza bölümleri
  - Hafıza yönlendirmeleri
  - Kaynak limitleri

fork fonksiyonu

- Üst ve alt işlem arasındaki farklar şunlardır:
  - fork fonksiyonunun dönüş değeri.
  - Işlem numaraları
  - Her iki işlemin üst işlem numaraları farklıdır.
  - Alt işlemin tms_utime, tms_stime, tms_cutime ve tms_cstime değerleri 0 olarak ayarlanır.
  - Üst işlemin dosya kilitleri alt işleme geçmez.
  - Üst işlemin bekleyen alarmları alt işleme geçmez.
  - Üst işlemin bekleyen sinyalleri alt işleme geçmez.

fork fonksiyonu

- Fork fonksiyonun başarısız olması iki sebebe bağlıdır.
  - Sistemde aşırı sayıda işlem oluşturulmuşsa.
  - Işlemin oluşturan gerçek kullanıcı numarasına ait sistem limitinin aşılması.
- fork fonksiyonun iki kullanımı vardır
  - Bir işlem kendisini kopyalayarak üst ve alt işlemlerin işleme ait kodun farklı kısımlarını çalıştırması için. Bu durum ağ sağlayıcıları için çoklukla kullanılır. Üst işlem kullanıcılardan istek bekler. Bir istek geldiğinde, üst işlem fork çağırır ve alt işlem istenilen isteği gerçekleştirir. Üst işlem ise bir sonraki isteği bekler.
  - Bir işlem başka bir programı çalıştırmak istediğinde. Bu durum kabuk için geçerlidir. Bu durumda oluşturulan alt işlem fork sonrası exec fonksiyonu çağırır.

vfork fonksiyonu

- vfork fonksiyonun çağrı şekli ve dönüş değerleri fork fonksiyonu ile aynıdır.
- vfork fonksiyonu, yeni oluşturulan işlemin amacı exec fonksiyonu çağırmak olduğunda kullanılabilir.
- vfork, fork fonksiyonunda olduğu gibi yeni bir işlem oluşturur. Ancak üst işlemin adres alanı alt işleme kopyalanmaz. Alt işlem exec fonksiyonu çağırana kadar üst işlemin adres alanında çalışır.
- vfork kullanıldığında alt işlemin exec veya exit çağırana kadar önce çalışması garantidir.
- vfork1.c.

exit fonksiyonları

- İşlem sonlandırmak için sekiz farklı yol vardır.
- Normal sonlandırma beş farklı şekilde yapılır.
  - main fonksiyonun return yapılması
  - exit çağırmak
  - _exit veya _Exit çağırmak
  - Son işlemciğin başlangıç rutininden return yapması
  - Son işlemciğin pthread_exit çağırması.
- Anormal sonlandırma üç farklı şekilde gerçekleşir.
  - Abort çağırılması
  - Bir sinyal alınması
  - Son işlemciğin iptal isteğine karşılık vermesi.
- Her ne şekilde sonlanırsa sonlansın, kernel tarafından sonuçta aynı kod çalıştırılır. Kernel kod açık olan dosya belirteçleri kapar, kullanılan hafızayı serbest bırakır ve diğer işlemleri yapar.

exit fonksiyonları

- Önceki slaytta belirtilen her durumda, sonlanan işlemin üst işlemine ne şekilde sonlandığı konusunda bilgi vermelidir.
  - Belirtilen üç exit fonksiyonu ile, argüman olarak verilen çıkış değeri ile yapılır.
  - Anormal sonlanma durumunda, kernel sonlanma nedenini açıklayan bir sonlanma durumu oluşturur.
  - Her durumda, işlemin üst işlemi wait ve waitpid fonksiyonlarını kullanarak alt işlemin sonlanma durumunu öğrenebilir.
  - _exit fonksiyonu sonuç olarak çağrıldığında kernel çıkış değerini sonlanma değerine dönüştürür.

exit fonksiyonları

- Eğer bir işlemin üst işlemi kendinden önce sonlanırsa
  - init işlemi, üst işlemi sonlanan işlemin üst işlemi haline gelir.
  - Bu durumda işlem init tarafından evlat edinilmiş olur.
- Eğer alt işlem üst işleminden önce sonlanırsa
  - Eğer alt işlem tamamiyle ortadan kaybolursa, üst işlem hazır olup istediğinde sonlanan alt işlemin sonlanma durumu hakkındaki bilgiye ulaşamaz.
  - Bu sebeple kernel sonlanan işlemler hakkında az bir bilgiyi saklar. Bu sayede istediğinde sonlanan işlemin üst işlemi wait ve waitpid fonksiyonları ile sonlanan işlem hakkında bilgi alabilir.
    - En azından saklanan bilgiler şunları içerir: sonlanan işlemin işlem numarası, sonlanma durumu, harcanan CPU zamanı.
    - Kernel sonlanan işlemin kullandığı hafızayı tamamiyle geri alabilir ve açık olan dosyaları kapatabilir.
  - Unix teminolojisinde, sonlanan ancak üst işlem tarafından sonlanma durumu okunmayan işlemler, zombi olarak adlandırılır.

exit fonksiyonları

- init tarafından evlat edinilen bir işlem sonlanırsa
  - init sonlanan işlem için wait fonksiyonunu çağırır ve sonlanan işlemin sonlanma durumu okunur.

wait ve waitpid fonksiyonları

- Bir işlem sonlandığında, normal veya anormal olarak, kernel bu işlemin üst işlemine SIGCHLD sinyali gönderir.
  - Bu sinyal kernel tarafından üst işleme gönderilen asenkron haber biçimidir.
- Üst işlem bu sinyali görmezden gelebilir veya sinyal gelince çalışacak bir fonksiyon tanımlanabilir.
  - Sinyal için tanımlanan varsayılan işlem görmezden gelmektir.
- wait veya waitpid fonksiyonu çağıran işlem:
  - Eğer bütün alt işlemleri çalışmakta ise bloklanır.
  - Eğer işlemin sonlanmış ve sonlanma durumu okunmasını bekleyen bir alt işlemi var ise, fonksiyon bu alt işlemin sonlanma durumuna döner.
  - Eğer işlemin alt işlemi yok ise, hata ile döner.
- Eğer bir işlem SIGCHLD sinyali alıp wait fonksiyonu çağırırsa, fonksiyonun hemen dönüş yapması beklenir. Ancak sinyal almadan bu fonksiyon çağırılırsa, bloklanabilir.

wait ve waitpid fonksiyonları

```
#include <sys/wait.h>

pid_t wait(int *statloc);

pid_t waitpid(pid_t pid, int *statloc, int options);
```

Dönüş: OK ise işlem numarası, hata ise 0 veya -1.

- İki fonksiyon arasındaki farklar:
  - wait fonksiyonu bir alt işlem sonlanıncaya kadar çağıran işlemi bloklayabilir. waitpid ise bloklamayı engeleyecek bir opsiyona sahiptir.
  - waitpid fonksiyonu ilk sonlanan alt işlemi beklemez. Beklediği alt işlemin seçilmesi için birkaç opsiyonu vardır.
- Eğer bir alt işlem sonlanmış ve zombi olarak bekliyorsa, wait fonksiyonu bu alt işlemin çıkış durumu ile döner.
  - Böyle bir alt işlem yoksa bir alt işlem sonlanana kadar çağıran işlem bloklanır.

wait ve waitpid fonksiyonları

- Eğer bir işlem birden fazla alt işleme sahipse, wait fonksiyonu bu alt fonksiyonlar biri sonlandığı dönüş yapar.
  - wait fonksiyonun dönüş değeri olan işlem numarasını kullanarak hangi alt işlemin sonlandığını söyleyebiliriz.
- Her iki fonksiyon statloc isminde int tipinde işaretçi alır.
  - Bu işaretçiye sonlanan alt işlemin sonlanma durumu yazılır.
  - Ancak null işaretçi gönderirseniz sonlanma durumu kayıt edilmez.
- Bir sonraki slaytta gösterilen 4 ayrı makro kullanılarak işlemin sonlanması konusunda bilgi alınabilir. Belirtilen 4 ayrı makro girdi olarak işlemin sonlanma durumunu alır.
  - Sonlanma durumunun içerisinde çıkış değeri mevcuttur.

wait ve waitpid fonksiyonları

Makro	Tanım
WIFEXITED(status)	Normal sonlanmış bir alt işlem için dönüş yapılmışsa doğru olur. Bu dumda aşağıdaki makro ile alt işlemin çıkış değeri öğrenilir. WEXITSTATUS (status)
WIFSIGNALED (status)	Bir sinyal alınması sonrası anormal şekilde sonlanmış bir alt işlem için dönüş yapılmışsa doğru olur. Bu durumda aşağıdaki makro ile sonlanmaya sebep olan sinyal öğrenilebilir. WTERMSIG (status) Aşağıdaki makrı ile core dosyası oluşup oluşmadığı kontrol edilir. WCOREDUMP(status)
WIFSTOPPED (status)	Durdurulmuş bir alt işlem için dönüş yapılmışsa doğru olur. Alt işlemi durduran sinyal numarasını öğrenmek için aşağıdaki makro kullanılabilir. WSTOPSIG (status)
WIFCONTINUED (status)	İş durdurma sonrası tekrar başlayan bir alt işlem için dönüş yapılmşsa doğru olur.

wait1.c

wait ve waitpid fonksiyonları

- Belli bir alt işlemi beklemek istersek, POSIX.1 bunu sağlamak için waitpid fonksiyonunu tanımlamıştır
- pid argümanının değerine göre belli alt işlemler beklenebilir.
  - pid == 1
    - Herhangi alt işlem beklenir. wait ile aynı şekilde çalışır.
  - pid > 0
    - İşlem numarası pid olan alt işlem beklenir.
  - pid == 0
    - Grup numarası çağıran işlem ile aynı olan bir alt işlem beklenir.
  - pid < 0
    - Grup numarası pid değerinin mutlak değerine eşit olan bir alt işlem beklenir.

wait ve waitpid fonksiyonları

- wait fonksiyonun tek hata durumu çağıran işlemin alt işlemi olmaması durumudur (bir diğer hata durumu ise fonksiyon çağrısının bir sinyalle sonlandırılmasıdır).
- waitpid fonksiyonu farklı hatalar verebilir. Örneğin beklenilen işlemin olmaması gibi, verilen işlem grup numarasının bulunmaması veya verilen grup numarasının işlemin alt işlemine ait olmaması gibi.
- waitpid fonksiyonunun çalışması kontrol edebilmek için options parametresi kullanılabilir. Bu argüman ya 0 değerini alır yada aşağıda belirtilen sabitlerin or işlemine sokulması ile oluşturulur.

wait ve waitpid fonksiyonları

Sabit	Tanım
WCONTINUED	Sistem iş kontrolünü destekliyorsa, bununla ilgili bir opsiyondur.
WNOHANG	Waitpid işlem numarası verilen alt işlem sonlanmamışsa bloklamaz. Bu durumda dönüş değeri 0 olur.
WUNTRACED	Sistem iş kontrolünü destekliyorsa, bununla ilgili bir opsiyondur.

wait ve waitpid fonksiyonları

- waitpid fonksiyonu wait fonksiyonunda olmayan özellikler içerir.
  - waitpid fonksiyonu ile belli bir işlemi bekleyebiliriz. wait ise sonlanan herhangi bir alt işlemin sonlanma durumuna dönüş yapar.
  - waitpid ile wait fonksiyonunun bloklama yapmayan versiyonunu çalıştırabiliriz.
    - Bazı durumlarda bu özellik yararlı olabilir.
  - waitpid fonksiyonunun iş yönetimi ile ilgili opsiyonları mevcuttur.
- fork2.c