

Sistem Programlama

Ders 3

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

Girdi ve çıktı işlemleri

- Dosya tanımlayıcıları
 - Dosya tanımlayıcıları kernel tarafından işlemlerin kullanmakta oldukları dosyaları tanımlayabilmek için kullanılan negatif-olmayan tam sayılardır.
 - İşlemler tarafından açılan her dosyanın bir dosya tanımlayıcısı bulunur.
- Bununla birlikte kabuk tarafından açılan üç farklı dosya tanımlayıcısı mevcuttur. Bunlar:
 - Standart girdi, standart çıktı, standart hata.
- Daha önce belirttiğimiz gibi kabukların birçoğu otomatik açılan bu dosyaları yönlendirmek için gereken yöntemleri sağlar.
 - `ls > file.list`

Girdi ve çıktı işlemler

- Önbelleksiz (unbuffered) girdi çıktı işlemler open, read, write, lseek ve close fonksiyonları ile yapılır.
 - Bu dosyalar üzerinde işlem yapabilmek için dosya tanımlayıcıları kullanılır.
- Örnek2: mycat.c

Girdi ve çıktı işlemleri

- mycat.c
 - `STDIN_FILENO` ve `STDOUT_FILENO` POSIX standartlarının bir parçasıdır.
 - Bu sabit değerler `<unistd.h>` dosyasında tanımlanmıştır.
 - Bu iki sabit aslında 0 ve 1 değerleri olarak tanımlanmıştır.
 - `read` fonksiyonu okunan byte sayısını döner
 - Okunan byte sayısı `write` fonksiyonunda kullanılıyor.
 - Girdi tamamlandığında `read` fonksiyonu 0 döner ve program sonlanır.
 - Program çalışırken `read` fonksiyonunda bir hata oluşursa -1 değeri döner.

Girdi ve çıktı işlemleri

- Standart I/O
 - Standart I/O fonksiyonları kullanarak önbellekli (buffered) girdi çıktı işlemleri yapılabilir.
 - Standart I/O fonksiyonu kullanırsanız doğru önbellek büyüklüğünü bulmak zorunda kalınmaz.
 - Ayrıca satır satır okumak için özel fonksiyonlar vardır.
 - fgets ==> tam bir satırı okur.
 - Standart I/O fonksiyonlarının en ünlüsü printf fonksiyonudur.
 - Örnek3: getcputc.c
 - getc fonksiyonu birer birer karakterleri okur ve okunan karakter putc tarafından yazılır.
 - En son byte okunduktan sonra getc EOF değeri döner ve bu değer <stdio.h> kütüphanesinde tanımlıdır.
 - Kütüphanenin kullandığı stdin ve stdout sabitleri <stdio.h> kütüphanesinde tanımlıdır.

Programlar ve işlemler

- Program
 - Program disk üzerinde bulunan çalıştırılabilir bir dosyadır.
 - Bir program çalıştığında önce hafızaya yüklenir ve kernel tarafından 6 farklı exec fonksiyonundan biri kullanılarak çalıştırılır.
- İşlemler ve işlem ID
 - Bir program çalıştığında bir işlem oluşur.
 - Bazı işletim sistemlerinde işlem yerine görev kelimesi kullanılır.
 - Unix sisteminde her işlem için ayrı işlem ID verilir. İşlem ID sayıları negatif olmayan tam sayılardır.
- Örnek4: hello.c
- İşlem kontrolü
 - İşlem kontrolü için üç temel fonksiyon kullanılır.
 - fork, exec, waitpid
 - Örnek5: shell1.c

Program ve işlemler

- shell1.c
 - Örnekte işlem kontrol fonksiyonu kullanarak standart girdiden komutları okuyan ve bu komutları çalıştıran bir program görülmektedir.
 - Kabuk benzeri programlar buna benzer şekilde oluşturulur.
 - Standart I / O fonksiyonu olan fgets fonksiyonu standart girdiden satır okumak için kullanılır.
 - Program çalışırken CTRL-D yani dosya sonu karakteri girdiğimizde fgets fonksiyonu NULL işaretleyici döner, döngü sonlanır ve işlem sonlanır.
 - fgetc fonksiyonu ile alınan satırlar yeni satır (newline) karakteri ile sonlanır ve null byte tarafından takip edilir.
 - Alınan satırdaki yeni satır karakteri null byte ile değiştirilir çünkü kullandığımız fonksiyon olan execlp fonksiyonu null byte ile biten bir argüman beklemektedir.

Programlar ve işlemler

- shell1.c
 - fork fonksiyonunu kullanarak yeni bir işlem yaratırız. Yaratılan işlem fork çağrısını yapan işlemin kopyasıdır.
 - Çağırان işlem üst işlem, yeni yaratılan işlem ise alt işlemidir.
 - Fork fonksiyonu üst işleme yeni yaratılan alt işlemin işlem numarasını dönerken, alt işlemde 0 döner.
 - Alt işlem execlp fonksiyonunu kullanarak standart girdiden girilen komutu çalıştırır.
 - Üst işlem alt işlemin sonlanmasını bekler.
 - Bekleme işlemi waitpid fonksiyonu ile yapılır ve bu fonksiyona beklenecek olan alt işlemin işlem numarası verilir.
 - Bu programın en büyük kısıtlaması şu anki versiyonunda komutlara argüman gönderemenin mümkün olmamasıdır.

Programlar ve işlemler

- İşlemci ve işlemci numarası
 - Genelde, bir işlem sadece bir kontrol işlemciği içerir.
 - Ancak bazı durumlarda birden fazla işlemi aynı anda yapmak istersek, problemin farklı parçalarını çözen işlemciler oluşturabiliriz.
 - Bir işlem içerisindeki bütün işlemciler aşağıdakileri paylaşır.
 - Adres alanı (ing: address space)
 - Dosya tanımlayıcıları
 - Stack (ing: yığın)
 - İşlem ile alakalı özellikler.
 - Her işlemci kendi yığına sahiptir. Ancak aynı işlem içerisindeki diğer işlemcilerin yığınlarına ulaşabilirler.
 - İşlemcilerin aynı hafızaya erişebildiklerinden dolayı kendi aralarında senkronize olmaları gerekmektedir.

Programlar ve işlemler

- İşlemciler, işlemci numarası ile tanımlanırlar.
 - İşlemci numarası sadece bulunduğu işlem içerisinde anlamlıdır.
- İşlemci numarası kullanarak işlem içerisinde istediğimiz işlemciğe ait komutlar çalıştırabiliriz.
- İşlemciler Unix sistemine çok sonradan eklenmiştir.

Hata işleme

- Bir Unix sistem fonksiyonun çalışması esnasında hata oluşursa
 - Genellikle negatif bir değer döner.
 - Tam sayı errno değişkenine genellikle bir değer atanır ve bu değer hata konusunda bilgi verir.
 - Örneğin open fonksiyonuna ait 15 farklı errno değeri vardır.
 - Bazı fonksiyonlar değer yerine başka bir işlem yapabilir.
 - Örneğin bir nesneye işaret eden işaretleyici dönen fonksiyonlar genellikle hata durumunda null işaretleyici dönerler.
 - Hataların errno değerleri ve açıklamaları <errno.h> dosyasında bulunur.
 - EACCESS: izinle ilgili bir hata olduğunu belirtir.
 - POSIX ve ISO C standartlarında errno değeri ve kullanımı ayrıntılarıyla açıklanmıştır.

Hata işleme

- Eğer işlem içerisinde birçok işlemcik var ise her işlemcik kendi errno kopyasına sahiptir. Bu sayede işlemcikleri birbirlerini etkilemezler.
- Bir hata olmadığı sürece errno değeri hiçbir işlem tarafından değiştirilmez.
 - Program içerisinde errno değerine sadece bir fonksiyon hata döndüğünde bakılmalıdır.
- Hiçbir fonksiyon errno değerini 0 yapmaz ve 0 değerine sahip bir hata <errno.h> dosyasında bulunmaz.

Hata işleme

- Hata mesajı yazdırılabilmesi için C standart kütüphanesinde iki fonksiyon tanımlanmıştır.
- `#include <string.h>`
- `char *strerror (int errnum);`
 - Bu fonksiyon `errnum` değerini (`errno` değeri) hatayı anlatan bir metne çevirir ve bu metne bir işaretleyici döner.
- `#include <stdio.h>`
- `void perror (const char *msg);`
 - Bu fonksiyon standart hataya o zamanki `errno` değerine bağlı olarak bir hata mesajı koyar ve döner.
 - Çıktı olarak `msg` tarafından işaret edilen metni verir, noktalı virgül ve boşluk ve `errno` ile alakalı bir hata mesajı ile takip eder.
- Örnek6: `testerror.c`

Hata işleme

- Program içerisinde oluşabilecek hataların listelendiği <errno.h> dosyasında hatalar iki farklı kategoriye ayrılabilir: ölümcül olanlar ve ölümcül olmayanlar.
- Ölümcül bir hata olduğunda işlemin kurtulma yolu yoktur.
 - Yapılabilecek tek şey bir hata mesajı yazdırmak ve işlemi sonlandırmaktır.
- Ölümcül olmayan hatalar farklı şekilde çözülebilir.
 - Ölümcül olmayan hataların birçoğu geçicidir.
 - Örneğin kaynak yetersizliği nedeniyle oluşan hatalar gibi.
 - Kaynak yetersizliği nedeniyle oluşan ölümcül olmayan hatalarda genellikle yapılan bir süre bekleyip tekrar denemektir.
 - Bazı uygulamalar üstel geri çekilme yöntemi kullanarak bekledikleri süreyi her sefer artırırlar.
 - Hangi hatalardan geri dönüşün yapılabileceği konusu programcı tarafından belirlenir.

Kullanıcı tanımlama

- Kullanıcı numarası
 - Kullanıcı numarası parola dosyasından alınır ve sistemdeki kullanıcıları bir sayısal değer kullanarak tanımlar.
 - Her kullanıcı için ayrı bir kullanıcı numarası vardır.
 - Kernel kullanıcı numarasını kullanarak bir kullanıcı bir işlem yaptığında gerekli izinleri kontrol eder.
 - Kullanıcı numarası 0 olan kullanıcı root veya superuser olarak adlandırılır.
 - Bazı işletim sistemi fonksiyonları sadece superuser tarafından çalıştırılabilir.

Kullanıcı tanımlama

- Grup numarası
 - Parola dosyasında kayıt ayrıca her kullanıcının grup numarasını tanımlar.
 - Giriş ismi oluşturulduğunda sistem tarafından ayrıca bir grup numarası tanımlanır.
 - Gruplar genellikle kullanıcıları proje çalışanları ve departman çalışanlarına ayırmak için kullanılır.
 - Grup numarasını kullanarak belli dosyalara sadece belli gruba ait kullanıcılar erişebilecek şekilde ayarlayabiliriz.
 - Grup dosyası grup isimlerini grup numaralarına eşler.
 - Genellikle bu dosya / etc / group dosyasıdır.
 - Örnek7: uidgid.c
 - Ayrıca kullanıcılara tamamlayıcı grup numaralar verilebilir.

Sinyaller

- Sinyaller işlemlere belli durumların oluştuğunu bildirmek için kullanılan bir yöntemdir.
 - Bir işlem bir sayıyı 0 ile bölmeye çalışırsa, bu işleme bir sinyal gönderilir.
- İşlem bir sinyal aldığı anda üç farklı seçeneği olur.
 - Mümkünse sinyali görmezden gelebilir.
 - Tanımlı aksiyonun olmasına izin verir.
 - Sinyal oluştuğunda yapılacak olanları bir fonksiyon ile belirler.
 - Bu sinyali yakalamak olarak adlandırılır.
 - Birçok durum sinyal oluşmasına neden olur.

Zaman değerleri

- Geleneksel olarak Unix sisteminde iki farklı zaman değeri tutulur.
 - Takvim zamanı: Bu deper 00:00:00 1 Ocak 1970'den günümüze geçen toplam saniye sayısını verir.
 - Bu zaman değeri dosya son değişme tarihinin kayıt edilmesi için kullanılır.
 - Temel veri tipi `time_t` tipinde değişken bu değeri tutar.
 - İşlem zamanı
 - Bu değer CPU zamanı olarak adlandırılır ve işlem tarafından CPU'nun ne kadar kullanıldığını hesaplar.
 - İşlem zamanı saat tiklemesi olarak hesaplanır ve her saniyede 50, 60 veya 100 saat tiklemesi olur.
 - Temel veri tipi `clock_t` tipinde değişken bu değeri tutar.

Zaman değerleri

- Bir işlemin çalışma zamanını hesapladığımızda Unix sisteminin işlemler için üç farklı zaman değerini sakladığını görürüz.
 - Saat zamanı
 - Kullanıcı CPU zamanı
 - Sistem CPU zamanı.
- Saat zamanı işlemin tamamlanması için geçen süreyi hesaplar ve bu süre o esnada sistemde çalışan diğer işlemlerden etkilenir.
 - Saat zamanını raporlamak isterseniz, o esnada sistemde başka çalışan bir aktivite olmamalıdır.

Zaman değerleri

- Kullanıcı CPU zamanı kullanıcı komutları için geçen süredir.
- Sistem CPU zamanı işlem kernel tarafından çalıştırılırken geçen süredir.
 - İşlem bir sistem çağrısı ile sistemle alakalı bir komut çalıştırdığında, kernel tarafından bu komut gerçekleştirilir. Komut gerçekleştirilirken geçen süre işlemin harcadığı süreye eklenir.
- Kullanıcı CPU zamanı ve sistem CPU zamanı toplamı CPU zamanı olarak adlandırılır.
- time komutu ile bir işlemin saat zamanı, kullanıcı zamanı ve sistem zamanı hesaplanabilir.
- `$ cd /usr / include`
- `$ time -p grep _POSIX_SOURCE */*.h > /dev/null`

Sistem çağrıları ve kütüphane fonksiyonları

- Bütün işletim sistemlerinde kullanıcı programlarının kernel üzerinde komut çalıştırabilmesi için gerekli hizmet noktaları vardır.
- Bütün Unix sürümlerinde kernel üzerinde komut çalıştırabilmek için gerekli, tüm detaylarıyla tanımlanmış sistem çağrıları mevcuttur.
 - Research UNIX System Versiyon 7: 50 sistem çağrısı.
 - 4.4BSD: 110 sistem çağrısı
 - SVR4: Yaklaşık 120 sistem çağrısı
 - Linux: 240 - 260 sistem çağrısı
 - FreeBSD: 320 sistem çağrısı

Sistem çağrıları ve kütüphane fonksiyonları

- Unix sistem çağrıları arayüzü Unix programcı el kitabının ikinci kısmında dökümante edilmiştir.
 - Tanımlar C dilinde yapılmıştır.
- Unix sistemlerinde sistem çağrılarıyla aynı isimde C kütüphanesinde fonksiyonlar bulunur.
 - Kullanıcı işlemi kütüphanedeki fonksiyonu çağırır.
 - Bu fonksiyon ilgili kernel işlemini başlatır.
 - Örneğin fonksiyon bir veya birden fazla C argümanını ilgili registerlara koyar ve bir makine komutu çalıştırarak kernel için bir yazılım interrupt oluşturur.
 - Bu ders için sistem çağrılarını C fonksiyonları olarak kabul edeceğiz.

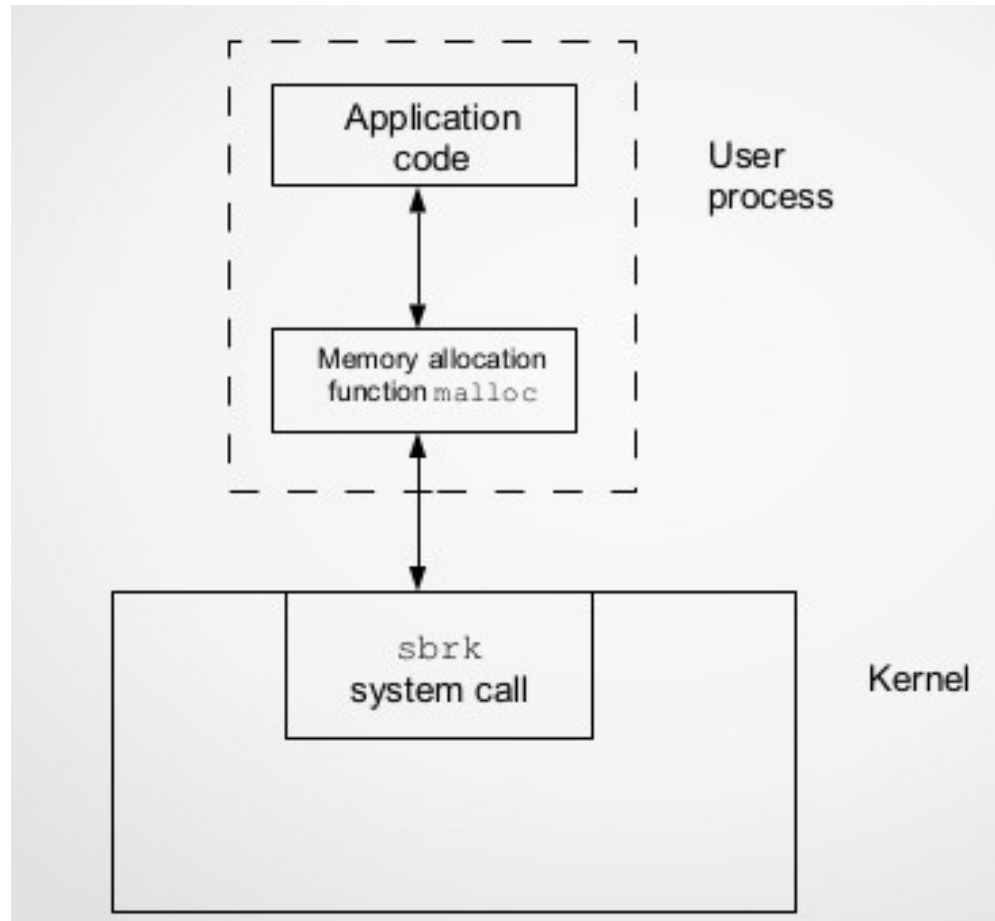
Sistem çağrıları ve kütüphane fonksiyonları

- El kitabının 3. kısmında genel kullanım için gereken fonksiyonlar bulunur.
 - Bu fonksiyonlar kernel erişimi için kullanılmaz ancak bazıları kernel sistem çağrılarını çalıştırabilir.
 - Örneğin printf bir sistem çağrısı olan write fonksiyonunu çağırır, strcpy ve atoi fonksiyonları ise kernel ile ilgili bir işlem yapmaz.
- Sistem çağrıları ve kütüphane fonksiyonları uygulama programlarına hizmet eder.
 - Kütüphane fonksiyonlarının yerine başka bir yöntem kullanabiliriz, ancak sistem çağrılarının yerine başka yöntem kullanılamaz.

Sistem çağrıları ve kütüphane fonksiyonları

- Hafıza alma için kullanılan malloc fonksiyonunu ele alalım.
- Hafıza alma ve sonrasında çöp toplama işlemleri için birçok farklı yöntem vardır.
 - Her bir yöntemin uygun olduğu program bulunabilir.
- Unix sistem çağrısı olan sbrk(2) hafıza alma için kullanılır ancak genel kullanım için uygun değildir.
 - Bu çağrıyı yapan işlemin adres alanını verilen byte sayısı kadar artırır.
 - Yeni ayrılan alanı ne yapacağı işleme kalmıştır.
 - Hafıza alanı alma için kullanılan malloc(3) fonksiyonu hafıza alma işini belli bir şekilde yapar.
 - İstersek kendi malloc fonksiyonu yazıp kullanabiliriz. Büyük ihtimal ile yazdığımız fonksiyon sbrk sistem çağrısını kullanacaktır.

Sistem çağrıları ve kütüphane fonksiyonları



Sistem çağrıları ve kütüphane fonksiyonları

- Bir uygulama isterse direk olarak sistem çağrılarını kullanabilir veya kütüphane fonksiyonlarını kullanabilir.

