< ERRNO.H >

* C Programlama dili standart n kütüphanesinde yer alan errno.h başlık dosyası hata kodları tanımlayan makrolar içerir.
* C Standart kütüphanesi bu makrolardan sadece EDOM, ERANGE ve EILSEQ makrolarının tanımlanmasını gerektirir.
* Dosyada yer alan errno değeri, int bir değişken olarak işlem görür kütüphane fonksiyonları bir hata tespit ettiklerinde, hata numarasını errno değişkenine atarlar Program başlangıcında , değişken değeri sıfırdır. Kütüphane fonksiyonları sadece sıfırdan büyük değerleri bu değişkene atar.
* Errno.h değerlerin errno tanımlarını içerir. Ancak içinde errno.h verilen tanımların tümü 32 bit Windows işletim sistemlerinde kullanılmaz içindeki değerlerden errno.h bazıları , UNIX işletim sistemi ailesi ile uyumluluğu korumak için mevcuttur. Errno 32 bit Windows işletim sistemindeki değerler , UNIX sistemlerindeki değerlerinin errno bir alt kümesidir.

Değerin errno, Windows işletim sisteminden bir sistem çağrısı tarafından döndürülen gerçek hata koduyla aynı olması gerekmez. Gerçek işletim sistemi hata koduna erişmek için bu değeri içeren değişkenini \_doserrno kullanın.

Aşağıdaki errno değerler desteklenir;

|  |  |  |
| --- | --- | --- |
| SABİT | DESCRİPTİON | DEĞER |
| E2BIG | Bağımsız değişken listesi çok uzun. | 7 |
| EACCES | İzin reddedildi. Dosyanın izin ayarı belirtilen erişime izin vermiyor. Dosyaya (veya bazı durumlarda dizine) dosyanın öznitelikleriyle uyumsuz bir şekilde erişme girişiminde bulunuldu.  Örneğin, açık olamayan bir dosyadan okunmaya çalışıldığında hata oluşabilir. Ya da yazmak için var olan salt okunur dosyayı açma veya dosya yerine dizin açma girişiminde bulunabilirsiniz. MS-DOS işletim sistemi 3.0 ve üzeri sürümleri altında, EACCES bir kilitleme veya paylaşım ihlali de gösterilebilir.  Hata, bir dosyayı veya dizini yeniden adlandırma veya mevcut dizini kaldırma girişiminde de oluşabilir. | 13 |
| EAGAIN | Artık işlem yok veya yeterli bellek yok ya da iç içe yerleştirme düzeyi üst sınırına ulaşıldı. Başka işlem yuvası olmadığından veya yeterli bellek olmadığından veya iç içe yerleştirme üst sınırına ulaşıldığından yeni işlem oluşturma girişimi başarısız oldu. | 11 |
| EBADF | Hatalı dosya numarası. İki olası neden vardır: 1) Belirtilen dosya tanımlayıcısı geçerli bir değer değil veya açık bir dosyaya başvurmuyor. 2) Salt okunur erişim için açılmış bir dosyaya veya cihaza yazma girişiminde bulunuldu. | 9 |
| EBUSY | Cihaz veya kaynak meşgul. | 16 |
| ECHILD | Oluşturulan işlem yok. | 10 |
| EDEADLK | Kaynak kilitlenmesi oluşabilir. | 36 |
| EDEADLOCK | EDEADLK Eski Microsoft C sürümleriyle uyumlulukla aynı. | 36 |
| EDOM | Matematik bağımsız değişkeni. Matematik işlevinin bağımsız değişkeni işlevin etki alanında değildir. | 33 |
| EEXIST | Dosyalar var. Zaten var olan bir dosya oluşturma girişiminde bulunuldu.  Örneğin, \_O\_CREAT ve \_O\_EXCL bayrakları bir \_open çağrıda belirtilir, ancak adlandırılmış dosya zaten vardır. | 17 |
| EFAULT | Hatalı adres. | 14 |
| EFBIG | Dosya çok büyük. | 27 |
| EILSEQ | Geçersiz bayt dizisi (örneğin, bir MBCS dizede). | 42 |
| EINTR | Kesilen işlev. | 4 |
| EINVAL | Geçersiz bağımsız değişken. Bir işlevin bağımsız değişkenlerinden biri için geçersiz bir değer verildi. Örneğin, bir dosya işaretçisi konumlandırılırken kaynak için verilen değer (çağrısıyla fseek) dosyanın başlangıcından öncedir. | 22 |
| EIO | G/Ç hatası. | 5 |
| EISDIR | Bir dizindir. | 21 |
| EMFILE | Çok fazla açık dosya var. Başka dosya tanımlayıcısı olmadığından başka dosya açılamaz. | 24 |
| EMLINK | Çok fazla bağlantı var. | 31 |
| ENAMETOOLONG | Dosya adı çok uzun. | 38 |
| ENFILE | Sistemde çok fazla dosya açık. | 23 |
| ENODEV | Böyle bir cihaz yok. | 19 |
| ENOLCK | Kullanılabilir kilit yok. | 39 |
| ENOEXEC | Exec biçimi hatası. Yürütülebilir olmayan veya yürütülebilir dosya biçimi geçersiz olan bir dosyayı yürütme girişiminde bulunuldu. | 8 |
| ENOENT | Böyle bir dosya veya dizin yok. Belirtilen dosya veya dizin yok veya bulunamıyor. Bu ileti, belirtilen bir dosya mevcut olmadığında veya yolun bir bileşeni varolan bir dizini belirtmediği zaman oluşabilir. | 2 |
| ENOMEM | Denenen işleç için yeterli bellek yok. Örneğin, bir alt işlemi yürütmek için yeterli bellek olmadığında veya bir \_getcwd çağrıdaki ayırma isteği karşılanmadığında bu ileti oluşabilir. | 12 |
| ENOSPC | Cihazda yer kalmadı. Cihazda yazma için daha fazla alan yoktur (örneğin, disk dolu olduğunda) | 28 |
| ENOSYS | İşlev desteklenmiyor. | 40 |
| ENOTDIR | Dizin değil | 20 |
| ENOTEMPTY | Dizin boş değil. | 41 |
| ENOTTY | Uygunsuz G/Ç denetimi işlemi. | 25 |
| ENXIO | Böyle bir cihaz veya adres yok. | 6 |
| EPERM | İşleme izin verilmiyor. | 1 |
| EPIPE | Bozuk boru. | 32 |
| ERANGE | Sonuç çok büyük. Matematik işlevinin bağımsız değişkeni çok büyük olduğundan sonuç kısmen veya toplam anlam kaybına neden olur. Bu hata, bir bağımsız değişken beklenenden büyük olduğunda (örneğin, için bağımsız değişkeni \_getcwd beklenenden uzun olduğunda) diğer işlevlerde *buffer* de oluşabilir. | 34 |
| EROFS | Salt okunur dosya sistemi. | 30 |
| ESPIPE | Geçersiz arama. | 29 |
| ESRCH | Böyle bir işlem yok. | 3 |
| EXDEV | Cihazlar arası bağlantı. Bir dosyayı farklı bir cihaza taşıma girişiminde bulunuldu (işlevi kullanılarak rename ). | 18 |
| STRUNCATE | Dize kopyalama veya birleştirme işlemi, kesilen bir dizeye neden oldu. Bkz. [\_TRUNCATE](https://learn.microsoft.com/tr-tr/cpp/c-runtime-library/truncate?view=msvc-170). | 80 |

***Posix ;***

POSIX (Portable Operating System Interface for Unix, [ˈpɒzɪks]), UNIX için "taşınabilir işletim sistemi arabirimi". Bir işletim sistemi hizmeti kümesi tanımlayan Elektrik ve Elektronik Mühendisleri Enstitüsü (IEEE) standardı. POSIX standardını kullanan programlar bir sistemden diğerine kolayca taşınabilir.

POSIX ile uyumluluk için aşağıdaki değerler desteklenir:

|  |  |  |
| --- | --- | --- |
| ***SABİT*** | ***DESCRİPTİON*** | ***DEĞER*** |
| EADDRINUSE | Adres kullanımda. | 100 |
| EADDRNOTAVAIL | Adres kullanılamıyor. | 101 |
| EAFNOSUPPORT | Adres ailesi desteklenmiyor. | 102 |
| EALREADY | Bağlantı zaten devam ediyor. | 103 |
| EBADMSG | Hatalı ileti. | 104 |
| ECANCELED | İşlem iptal edildi. | 105 |
| ECONNABORTED | Bağlantı durduruldu. | 106 |
| ECONNREFUSED | Bağlantı reddedildi. | 107 |
| ECONNRESET | Bağlantı sıfırlama. | 108 |
| EDESTADDRREQ | Hedef adres gerekli. | 109 |
| EHOSTUNREACH | Konağa ulaşılamıyor. | 110 |
| EIDRM | Tanımlayıcı kaldırıldı. | 111 |
| EINPROGRESS | İşlem devam ediyor. | 112 |
| EISCONN | Zaten bağlı. | 113 |
| ELOOP | Çok fazla sembolik bağlantı düzeyi var. | 114 |
| EMSGSIZE | İleti boyutu. | 115 |
| ENETDOWN | Ağ kapalı | 116 |
| ENETRESET | Ağ sıfırlama. | 117 |
| ENETUNREACH | Ağa ulaşılamıyor. | 118 |
| ENOBUFS | Arabellek alanı yok. | 119 |
| ENODATA | Kullanılabilir ileti yok. | 120 |
| ENOLINK | Bağlantı yok. | 121 |
| ENOMSG | mesaj yok. | 122 |
| ENOPROTOOPT | Protokol seçeneği yok. | 123 |
| ENOSR | Akış kaynağı yok. | 124 |
| ENOSTR | Akış değil. | 125 |
| ENOTCONN | Bağlı değil. | 126 |
| ENOTRECOVERABLE | Durum kurtarılamaz. | 127 |
| ENOTSOCK | Yuva değil. | 128 |
| ENOTSUP | Desteklenmez. | 129 |
| EOPNOTSUPP | İşlem desteklenmiyor. | 130 |
| EOTHER | Diğer. | 131 |
| EOVERFLOW | Değer çok büyük. | 132 |
| EOWNERDEAD | Sahibi öldü. | 133 |
| EPROTO | Protokol hatası. | 134 |
| EPROTONOSUPPORT | Protokol desteklenmiyor. | 135 |
| EPROTOTYPE | Yanlış protokol türü. | 136 |
| ETIME | Akış zaman aşımı. | 137 |
| ETIMEDOUT | Zaman aşımına uğradı. | 138 |
| ETXTBSY | Metin dosyası meşgul. | 139 |
| EWOULDBLOCK | İşlem engel olur. | 140 |

\_doserrno ; Program yürütme sırasında ayarlanan hata kodlarını tutan genel makrolar ve görüntülenmek üzere hata kodlarının dize eşdeğerleri.

* SÖZDİZİMİ;

|  |
| --- |
| C |
| * # define errno (\*\_errno ( ) ) * #define doserrno (\*\_\_doserrno ( ) ) * #define \_sys\_errlist (\_\_sys\_errlist ( ) ) * #define \_sys\_nerr (\*\_\_sys\_nerr ( ) ) |

\_doserrno Hem hem de errno, program başlatma sırasında çalışma zamanı tarafından 0 olarak ayarlanır. errno , sistem düzeyinde bir çağrıdaki bir hatada ayarlanır. errno Bunu ayarlayan son çağrının değerini tuttuğundan, bu değer başarılı çağrılar tarafından değiştirilebilir. Bir hatada ayarlanan errno çalışma zamanı kitaplığı çağrıları başarılı olduğunda temizlenemez errno . Ayarlayabilen bir aramadan hemen önce arayarak \_set\_errno(0) her zaman temizleyin errno ve aramadan hemen sonra kontrol edin.

Bir hatada, errno sistem çağrısı tarafından döndürülen hata koduyla aynı değere ayarlanmış olması gerekmez. G/Ç işlemleri için, \_doserrno kodların işletim sistemi hata kodu eşdeğerlerini depolar errno . G/Ç olmayan işlemlerin çoğu için değeri \_doserrno ayarlanmamıştır.

Her errno değer, işlevlerden biri kullanılarak yazdırılabilen veya veya işlevlerinden biri [perror](https://learn.microsoft.com/tr-tr/cpp/c-runtime-library/reference/perror-wperror?view=msvc-170)[strerror](https://learn.microsoft.com/tr-tr/cpp/c-runtime-library/reference/strerror-strerror-wcserror-wcserror?view=msvc-170)[strerror\_s](https://learn.microsoft.com/tr-tr/cpp/c-runtime-library/reference/strerror-s-strerror-s-wcserror-s-wcserror-s?view=msvc-170) kullanılarak bir dizede depolanabilen bir hata iletisiyle **\_sys\_errlist** ilişkilendirilir. perror ve strerror işlevleri, hata bilgilerini işlemek için dizisini ve **\_sys\_nerr** içindeki **\_sys\_errlist** öğelerin sayısını kullanır. **\_sys\_errlist**. ve **\_sys\_nerr** öğesine **\_sys\_errlist**  doğrudan erişim, kod güvenliği nedeniyle kullanım dışıdır. Burada gösterildiği gibi genel makrolar yerine daha güvenli, işlevsel sürümleri kullanmanızı öneririz:

|  |  |
| --- | --- |
| GENEL MAKRO | İŞLEVSEL EŞDEĞERLER |
| * \_doserrno | * \_get\_doserrno, \_set\_doserrno |
| * errno | * \_get\_errno, \_set\_errno |
| * \_sys\_errlist,   \_sys\_nerr | * Strerror\_s, \_strerror\_s,   \_wcserror\_s, \_\_wcserror\_s |

çağrılarak [\_matherr](https://learn.microsoft.com/tr-tr/cpp/c-runtime-library/reference/matherr?view=msvc-170) ayarlanan kitaplık errno matematik yordamları. Matematik hatalarını farklı şekilde işlemek için, başvuru açıklamasına \_matherr göre kendi yordamınızı yazın ve olarak adlandırın \_matherr.

Tüm errno değerler içinde <errno.h> önceden tanımlanmış sabitlerdir ve UNIX ile uyumludur. Iso C99 standardında yalnızca ERANGE, EILSEQve EDOM belirtilir. \_doserrno, \_sys\_errlistve \_sys\_nerr makroları Microsoft uzantılarıdır.

\_MATHERR

* Matematik hatalarını işler.
* SÖZDİZİMİ;

|  |
| --- |
| C |
| * İnt \_matherr (struct\_exception \*except); |

Parametreler; Hata bilgilerini içeren yapıya yönelik işaretçi.

DÖNDÜRÜLEN DEĞER ;

\_matherr hata göstermek için 0 veya başarıyı göstermek için sıfır olmayan bir değer döndürür:

* 0 döndürürse \_matherr , bir hata iletisi görüntülenebilir ve errno uygun bir hata değerine ayarlanır.
* Sıfır olmayan bir değer döndürürse \_matherr , hiçbir hata iletisi görüntülenmez ve errno değişmeden kalır.

İşlev, \_matherr matematik kitaplığının kayan nokta işlevleri tarafından oluşturulan hataları işler. Bir hata algılandığında bu işlevler çağrılır \_matherr . Bu etkileşim [, derleyicinin kayan nokta modundan](https://learn.microsoft.com/tr-tr/cpp/build/reference/fp-specify-floating-point-behavior?view=msvc-170) veya [kayan nokta denetim sözcüğünden](https://learn.microsoft.com/tr-tr/cpp/c-runtime-library/reference/control87-controlfp-control87-2?view=msvc-170) etkilenmez. Bir kitaplık işlevi olduğundan \_matherr , matematik [iç işlevleri](https://learn.microsoft.com/tr-tr/cpp/intrinsics/compiler-intrinsics?view=msvc-170) onu çağırmaz.

Özel hata işleme için farklı bir tanımı \_matherr sağlayabilirsiniz. C çalışma zamanı kitaplığının (CRT) dinamik olarak bağlı sürümünü kullanıyorsanız, istemci yürütülebilir dosyasındaki varsayılan \_matherr yordamı kullanıcı tanımlı bir sürümle değiştirebilirsiniz. Ancak, CRT DLL'nin DLL istemcisindeki varsayılan \_matherr  yordamı değiştiremezsiniz.

Matematik yordamında bir hata oluştuğunda, \_matherr  bağımsız değişken olarak bir \_exception tür yapısına (içinde <math.h>tanımlanan) işaretçisi ile çağrılır.

Yapı \_exception aşağıdaki öğeleri içerir.

|  |
| --- |
| C |
| struct \_exception  {    int type; // exception type - see below  char\* name; // name of function where error occurred  double arg1; // first argument to function  double arg2; // second argument (if any) to function  double retval; // value to be returned by function  }; |

Üye **type** matematik hatasının türünü belirtir. bu, içinde <math.h>tanımlanan aşağıdaki değerlerden biridir:

|  |  |
| --- | --- |
| MAKRO | DESCRİPTİON |
| \_DOMAIN | Bağımsız değişken etki alanı hatası |
| \_SING | Bağımsız değişken tekilliği |
| \_OVERFLOW | Taşma aralığı hatası |
| \_PLOSS | Kısmi anlam kaybı |
| \_TLOSS | Toplam anlam kaybı |
| \_UNDERFLOW | Sonuç temsil edilemeyecek kadar küçük. (Bu koşul şu anda desteklenmiyor.) |

Yapı üyesi name , hataya neden olan işlevin adını içeren null olarak sonlandırılan bir dizenin işaretçisidir. Yapı üyeleri arg1 ve arg2 hataya neden olan değerleri belirtin. Yalnızca bir bağımsız değişken verilirse, içinde arg1 depolanır.

Verilen hata için varsayılan dönüş değeri şeklindedir retval. Dönüş değerini değiştirirseniz, hatanın gerçekten oluşup oluşmadığını belirtmelidir.

**TİME.H;**

C Programlama Dili standart kütüphanesinde yer alan **time.h** başlık dosyasında tarih ve saat işlemleri ile ilgili fonksiyon, makro ve değişken tanımlamaları yer almaktadır

Tarih ve zaman ile ilgili işleri yapmak istediğimizde, standart C kütüphanesinden time.h başlık fonksiyonu çağırmalıyız. time.h başlık fonksiyonu, tarih ve zamanla ilgili işleri yapan fonksiyonları içerir (bkz. time.h fonksiyonları). Bu fonksiyonlardan bazılarını ele alacağız

## **gmtime()**

Python’da (ve başka programlama dillerinde), zaman-tarih hesaplamalarında ‘zamanın başlangıcı’ (EPOCH) diye bir kavram bulunur. ‘Zamanın başlangıcı’, bir işletim sisteminin, tarih hesaplamalarında sıfır noktası olarak aldığı tarihtir. Kullandığınız işletim sisteminin hangi tarihi ‘zamanın başlangıcı’ olarak kabul ettiğini bulmak için şu komutu verebilirsiniz:

>>> time.gmtime(0)

Buradan şu çıktıyı alıyoruz:

time.struct\_time(tm\_year=1970, tm\_mon=1, tm\_mday=1, tm\_hour=0,

tm\_min=0, tm\_sec=0, tm\_wday=3, tm\_yday=1, tm\_isdst=0)

Bu, struct\_time adlı özel bir veri tipidir. Bu veri tipi içindeki niteliklere şu şekilde ulaşabilirsiniz:

epoch = time.gmtime(0)

**>>>** epoch.tm\_year *#yıl*

1970

**>>>** epoch.tm\_mon *#ay*

1

**>>>** epoch.tm\_mday *#gün*

1

Demek ki zamanın başlangıcı 1 Ocak 1970 tarihi olarak alınıyormuş… İşte bilgisayarımız, içinde bulunduğumuz zaman ve saati, bu başlangıç zamanından bu yana geçen saniyeleri hesaplayarak bulur.

gmtime() fonksiyonunu parametresiz olarak kullandığınızda, o anda içinde bulunduğunuz tarih ve saat bilgisini elde edersiniz.

time.struct\_time(tm\_year=2014, tm\_mon=12, tm\_mday=10, tm\_hour=12, tm\_min=5, tm\_sec=33, tm\_wday=2, tm\_yday=344, tm\_isdst=0)

Ancak bu çıktı, özellikle saat kısmı konusunda her zaman doğru olmayabilir. Çıktının birkaç saat saptığını görebilirsiniz.

## **time()**

time() fonksiyonu, [epok](https://docs.python.org/3/library/time.html" \l "epoch)’tan itibaren, o anda içinde bulunduğumuz ana kadar geçen toplam saniye miktarını verir:

**>>>** time.time()

1418213083.726988

Elde ettiğiniz bu değeri, gmtime() fonksiyonunu kullanarak anlamlı bir tarih değerine dönüştürebilirsiniz:

**>>>** time.gmtime(time.time())

time.struct\_time(tm\_year=2014, tm\_mon=12, tm\_mday=10,

tm\_hour=12, tm\_min=9, tm\_sec=19, tm\_wday=2, tm\_yday=344,

tm\_isdst=0)

Ancak bu çıktı da özellikle saat kısmında sapmalara uğrayabilir.

## **localtime()**

Tıpkı gmtime() fonksiyonundan olduğu gibi, anlık tarih ve zaman bilgisini bir struct\_time nesnesi olarak almak için localtime() fonksiyonunu da kullanabiliriz. Bu fonksiyon bize yerel saati doğru bir şekilde verecektir:

**>>>** time.localtime()

time.struct\_time(tm\_year=2014, tm\_mon=12, tm\_mday=10,

tm\_hour=14, tm\_min=24, tm\_sec=21, tm\_wday=2, tm\_yday=344, tm\_isdst=0)

Bu nesnenin içindeki yıl, ay ve gün gibi bilgilere tek tek nasıl erişebileceğinizi biliyorsunuz.

## **asctime()**

Başta da söylediğimiz gibi, time modülü, datetime modülüne benzer. Bunların aynı işi gören çeşitli fonksiyonları vardır. Bir örnek verelim.

Hatırlarsanız, bugünün tarihini bir karakter dizisi olarak almak için datetime modülünü şu şekilde kullanabiliyorduk:

**>>> import** **datetime**

**>>>** an = datetime.datetime.now()

**>>>** datetime.datetime.ctime(an)

'Wed Dec 10 13:56:22 2014'

Yukarıdaki işlemi time modülünün asctime() fonksiyonunu kullanarak da yapabiliriz:

**>>> import** **time**

**>>>** time.asctime()

'Wed Dec 10 13:58:31 2014'

asctime() fonksiyonu tercihe bağlı bir parametre de alabilir. İsterseniz bu fonksiyona 9 öğeli bir demet veya bir struct\_time nesnesi verebilirsiniz.

Yukarıda, gmtime() fonksiyonunun bir struct\_time nesnesi ürettiğini öğrenmiştik. Dolayısıyla bu nesneyi asctime() fonksiyonuna parametre olarak verebilirsiniz:

**>>>** time.asctime(time.gmtime())

'Wed Dec 10 12:14:29 2014'

**>>>** time.asctime(time.gmtime(0))

'Thu Jan 1 00:00:00 1970'

Aynı şekilde localtime() fonksiyonunun da bize bir struct\_time() nesnesi verdiğini biliyoruz. Dolayısıyla bu fonksiyon da asctime() fonksiyonuna parametre olarak verilebilir:

**>>>** time.asctime(time.localtime())

'Wed Dec 10 14:28:05 2014'

Veya, sırasıyla yıl, ay, gün, saat, dakika, saniye, haftanın günü, yılın günü, gün ışığından yararlanma durumu değerini içeren bir demet de oluşturabilir, daha sonra bunu asctime() fonksiyonuna parametre olarak verebilirsiniz:

**>>>** demet = (2014, 5, 27, 13, 45, 23, 0, 0, 0)

**>>>** time.asctime(demet)

Ancak özellikle haftanın günü, yılın günü ve gün ışığından yararlanma durumu bilgilerini doğru tahmin etmek zor olduğu için, bu demeti elle oluşturmanızı pek tavsiye etmem.

## **strftime()**

Hatırlarsanız datetime modülünü anlatırken, datetime sınıfı içindeki strftime() adlı bir fonksiyondan söz etmiştik. Bu fonksiyonun, tarih-saat bilgisi içeren karakter dizilerini manipüle edebilmemizi sağladığını biliyorsunuz.

Bu fonksiyonu şöyle kullanıyorduk:

**>>> import** **datetime**

**>>>** an = datetime.datetime.now()

**>>>** datetime.datetime.strftime(an, '**%c**')

'10.12.2014 14:57:48'

İşte yukarıdaki işlemi, time modülünün strftime() fonksiyonunu kullanarak biraz daha pratik bir şekilde gerçekleştirebiliriz:

**>>> import** **time**

**>>>** time.strftime('**%c**')

'10.12.2014 14:58:02'

Yıkarıdaki örnek Strftime() örneği

**%a:**hafta gününün kısaltılmış adı

**%A:**hafta gününün tam adı

**%b:**ayın kısaltılmış adı

**%B:**ayın tam adı

**%c:**tam tarih, saat ve zaman bilgisi

**%d:**sayı değerli bir karakter dizisi olarak gün

**%j:**belli bir tarihin, yılın kaçıncı gününe denk geldiğini gösteren 1-366 arası bir sayı

**%m:**sayı değerli bir karakter dizisi olarak ay

**%U:**belli bir tarihin yılın kaçıncı haftasına geldiğini gösteren 0-53 arası bir sayı

**%y:**yılın son iki rakamı

**%Y:**yılın dört haneli tam hali

**%x:**tam tarih bilgisi

**%X:**tam saat bilgisi

***import******locale***

*locale.setlocale(locale.LC\_ALL, 'turkish')*

## **strptime()**

time modülünün strptime() fonksiyonunun yaptığı iş, datetime modülünün datetime sınıfının strptime() fonksiyonunun yaptığı işe çok benzer:

**>>> import** **datetime**

**>>>** t = '27 Mayıs 1980'

**>>>** tarih = datetime.datetime.strptime(t, '**%d** %B %Y')

**>>>** tarih

datetime.datetime(1980, 5, 27, 0, 0)

Burada ‘27 Mayıs 1980’ tarihini, strptime() fonksiyonu yardımıyla bir datetime nesnesine dönüştürdük. Aynı şeyi şu şekilde de yapabiliriz:

**>>> import** **time**

**>>>** t = '27 Mayıs 1980'

**>>>** tarih = time.strptime(t, '**%d** %B %Y')

**>>>** tarih

time.struct\_time(tm\_year=1980, tm\_mon=5, tm\_mday=27,

tm\_hour=0, tm\_min=0, tm\_sec=0, tm\_wday=1, tm\_yday=148,

tm\_isdst=-1)

Gördüğünüz gibi, time modülünün strptime() fonksiyonu datetime modülü içindeki strptime() fonksiyonunun aksine bir struct\_time nesnesi veriyor.

## **sleep()**

sleep() fonksiyonu, time modülünün en sık kullanılan araçlarından bir tanesidir. Bu fonksiyonu kullanarak kodlarımızın işleyişini belli sürelerle kesintiye uğratabiliriz.

Basit bir örnek verelim:

**>>> for** i **in** range(10):

**...**  time.sleep(1)

**...**  print(i)

Bu kodları çalıştırdığınızda, 0’dan 10’a kadar olan sayılar ekrana basılırken her bir sayı arasına 1’er saniyelik duraklamalar eklendiğini göreceksiniz. Eğer arzu ederseniz bu süreyi 1 saniyenin de altına çekebilirsiniz:

**>>> for** i **in** range(10):

**...**  time.sleep(0.5)

**...**  print(i)

Gördüğünüz gibi, sleep() fonksiyonuna 0.5 parametresini vererek, duraklama süresinin 500 milisaniye olmasını sağladık.

time modülünün sleep() fonksiyonunu, kodlarınız arasına duraklama eklemek istediğiniz her durumda kullanabilirsiniz.

**FONKSİYONLAR;**

|  |  |
| --- | --- |
| **FONSİYON ADI** | **AÇIKLAMA** |
| asctime asctime\_s (C11) | char\*asctime (const struct tm \*tmptr);  kendisine parametre olarak geçirilen tm zaman yapısı içindeki ınt değişken değerleri ile oluşturduğu yıl,ay,gün ve saat bilgilerini içeren karakter dizisini geri döndürür. |
| clock | clock\_t clock(void);  programın çalışmaya başlamasından itibaren geçen saat tik sayısını geri döndürür. |
| ctime ctime\_s (C11) | char\* ctime(const time\_t\*timer); kendisine parametre olarak geçirilen time\_t nesnesinin lokal zamana ait bilgilerini içeren karakter dizisini geri döndürür. |
| difftime | double difftime (time\_t time\_last,time\_t time\_ first);  ilk ve ikinci parametreleri (time\_t ) arasındaki zaman farkını saniye olarak geri döndürür. |
| gmtime gmtime\_s (C11) | struct tm\* gmtime(const time\_t\*timer);  parametre olarak geçirilen time\_t değerini kullanarak tm yapısını evrensel saat (UTC) değerlerine göre doldurarak geri döndürür. |
| localtime localtime\_s(C11) | struct tm\*localtime(const time\_t\*timer);  parametre olarak geçirilen time \_t değerini kullanarak tm yapısını local zamana göre doldurarak geri döndürür. |
| mktime | time\_t mktime (struct tm\*tmptr);  parametre olarak geçirilen tm yapı değişenlerini local zamana göre time\_t değişkenine çevirerek geri döndürür. |
| strftime | size\_t strftime (char\*buf,size\_t\*bufsz,const char\*format,const struct tm\*tmptr);  dördüncü parametresindeki tm yapı değerlerini üçüncü parametresine uygun olarak yapılandırarak ilk parametreye atar. |
| time | time\_t time(time\_t\*timer);  1 ocak 1970 tarihinden beri geçen zamanı saniye olarak hesaplayarak geri döndürür. |
| timespec\_get (C11) | İnt timespec\_get (struct timespec\*ts,int base); takvim zamanını belirli bir zaman dilimine göre geri döndürür. |

**VERİLER;**

|  |  |  |
| --- | --- | --- |
| **VERİ TÜRÜ ADI** | **DEĞER** | **AÇIKLAMA** |
| clock\_t | long | İşlemci çalışma zamanını yüklemek için kullanılan değişken değer. |
| time\_t | \_int32/\_int64 | Takvim zamanını yüklemek için kullanılan değişken değer. |
| timespec(C11) | - | Zaman saniye ve nano saniye olarak göterir. |
| tm | struct tm { int tm\_sec ; /\* saniye: 0-59 \*/ int tm\_min; /\* dakika: 0-59 \*/ int tm\_hour; /\* saat: 0-23 \*/ int tm\_mday; /\* ayın günü: 1-31 \*/ int tm\_mon; /\*aylar ocak’tan itibaren: 0-11 \*/ int tm\_year; /\* yıllar 1990’dan itibaren \*/ int tm\_wday; /\* hafta gün sayısı Pazar’dan itibaren ( 0-6 ) \*/ int tm\_yday; /\* yıl gün sayısı1 ocak’tan itibaren : 0-365 \*/ int tm\_isdst; /\* +1 günışığından faydalanma zamanı, 0 GFZ yok, -1 bilgi yok \*/ }; | Zaman bilgilerini depolamak için kullanılan yapıdır. |

**MAKRO SABİTLER;**

|  |  |
| --- | --- |
| **SABİT ADI** | **AÇIKLAMA** |
| COLOCKS\_PER\_SEC | İşlemcinin bir saniyedeki tik sayısını gösterir. |

**<stdio.h>**

C Programlama Dili'nde giriş ve çıkış işlemleri, **stdio.h** başlık dosyasında yer alan C standart giriş ve çıkış kütüphanesindeki fonksiyon, değişken, sabit ve diğer değerler kullanılarak yapılır. Bu kitaplık, klavyeler, yazıcılar, terminaller gibi fiziksel aygıtlarla veya sistem tarafından desteklenen diğer dosya türleriyle çalışmak için akış adı verilenleri kullanır.

Bu kütüphaneyi kullanarak, bilgisayardaki klavye, ekran, disk sürücü veya port'lara erişim sağlayarak işlem yapabiliriz. Bilgisayar bileşenlerine erişim için mantıksal bir arabirim görevi yapan **Akış** adı verilen bir kavram kullanılır.

**FONKSİYONLAR**

|  |  |
| --- | --- |
| **FONKSİYON ADI** | **AÇIKLAMA** |
| [clearerr](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/clearerr) | void clearerr(FILE \*stream);  Verilen dosya akışı için EOF ve hata göstergelerini siler. |
| [fclose](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fclose) | int fclose(FILE \*stream);  Verilen dosya akışını kapatır. Tüm tamponlar silinir. |
| [feof](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/feof) | int feof(FILE \*stream);  Parametrede gösterilen dosya akışının sonuna erişilip erişilmediğini kontrol eder |
| [ferror](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/ferror) | int ferror(FILE \*stream);  Verilen dosya akışı için hata göstergesini kontrol eder. |
| [fflush](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fflush) | int fflush(FILE \*stream);  Verilen dosya akışı için hata göstergesini kontrol eder. |
| [fgetc getc](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fgetc) | int fgetc(FILE \*stream);  Verilen dosya akışı içinde, bir sonraki karakteri (unsigned char) okur ve konum göstergesini bir ileri taşır. |
| [fgetpos](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fgetpos) | İnt fgetpos(FILE \*stream, fpos\_t \*pos); Akışın aktif dosya konumunu alır ve pos değişkenine yazar |
| [fgets](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fgets) | char\* fgets(char \*str, int count, FILE \*stream);  Üçüncü parametrede verilen dosya akışından en fazla ikinci parametre değerinin 1 eksiği kadar karakter okur ve okuduğu karakterleri ilk parametrede ile gösterilen karakter dizisine atar. |
| fopen fopen\_s (C11) | FILE\* fopen(const char \*filename, const char \*mode); Verilen dosya adını mode değerine uygun olarak açar. |
| fprintf fprintf\_s (C11) | int fprintf(FILE \*stream, const char \*format, ...);  Bir karakter dizisi içinde format tanımlayıcıları ile tanımlanan değerleri bir dosya akışına yazar |
| [fputc putc](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fputc) | int fputc(int char, FILE \*stream);  Verilen karakteri (unsigned char) akış içine yazar ve konum göstergesini bir ileri taşır. |
| [fputs](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fputs) | int fputs(const char \*str, FILE \*stream);  Verilen karakter dizisini, karakter dizisi sonunda bulunan NULL değer hariç, akışa yazar. |
| [fread](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fread) | size\_t fread(void \*ptr, size\_t size, size\_t count, FILE \*stream);  Verilen akıştan okuduğu değeri ptr ile gösterilen işaretçi adresine yazar. |
| freopen freopen\_s (C11) | FILE\* freopen(const char \*filename, const char \*mode, FILE \*stream);  Yeni bir dosya adını verilen açık akışla ilişkilendirir ve aynı zamanda akıştaki eski dosyayı kapatır. |
| fscanf fscanf\_s (C11) | int fscanf(FILE \*stream, const char \*format, ...);  Bir dosya akışından okuduğu değerleri değişkenlere atar |
| [fseek](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fseek) | int fseek(FILE \*stream, long offset, int origin);  Dosya akışına ait dosya konum göstergesini origin parametre değerinde yer alan değere göre offset parametre değeri ile gösterilen değere ayarlar. |
| [fsetpos](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fsetpos) | int fsetpos(FILE \*stream, const fpos\_t \*pos);  Verilen akışın dosya konumunu verilen konuma ayarlar. |
| [ftell](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/ftell) | long ftell(FILE \*stream);  Verilen akışın geçerli dosya konumunu döndürür |
| [fwrite](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/fwrite) | size\_t fwrite(void \*ptr, size\_t size, size\_t count, FILE \*stream);  Fonksiyona geçirilen ptr adresindeki değeri verilen akışa yazar. |
| [getchar](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/getchar) | int getchar(void);  Klavyeden girilen bir karakteri okur |
| gets gets\_s (C11) | char\* gets(char \*str);  Klavyeden girilen bir karakter dizisi okur. |
| [perror](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/perror) | void perror(const char \*str);  Oluşan hatayı ekrana yazar. |
| printf printf\_s (C11) | int printf(const char \*format, ...);  Bir karakter dizisi içinde format tanımlayıcıları ile tanımlanan değerleri ekrana yazar. |
| [putchar](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/putchar) | int putchar(int char);  Bir karakteri ekrana yazar. |
| [puts](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/puts) | int puts(const char \*str);  Bir karakter dizisini ekrana yazar |
| [remove](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/remove) | int remove(const char \*filename);  Adı verilen dosyayı siler. |
| [rename](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/rename) | Adı verilen dosyanın adını değiştirir. |
| [rewind](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/rewind) | int rename(const char \*oldname, const char \*newname);  Verilen akışın dosya konumunu dosyanın başına alır. |
| scanf scanf\_s (C11) | int scanf(const char \*format, ...);  Klavyeden okuduğu değerleri değişkenlere atar. |
| [setbuf](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/setbuf) | void setbuf(FILE \*stream, char \*buffer);  Bir dosya akışı için ara bellek ayarlaması yapar. |
| [setvbuf](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/setvbuf) | int setvbuf(FILE \*stream, char \*buffer, int mode, size\_t size);  Bir dosya akışı için ara bellek düzenlemesi yapar. |
| snprintf (C99) snprintf\_s (C11) | int snprintf(char \*restrict buffer, size\_t bufsize, const char \*restrict \*format, ...);  Bir karakter dizisi içinde format tanımlayıcıları ile tanımlanan değerleri boyutu belirli bir karakter dizisi içine yazar. |
| sprintf sprintf\_s (C11) | int sprintf(char \*buffer, const char \*format, ...);  Bir karakter dizisi içinde format tanımlayıcıları ile tanımlanan değerleri bir karakter dizisi içine yazar |
| sscanf sscanf\_s (C11) | int sscanf(const char \*buffer, const char \*format, ...);  Bir karakter dizisinden okuduğu değerleri değişkenlere atar. |
| tmpfile tmpfile\_s (C11) | FILE\* tmpfile(void);  Güncelleme modunda (wb+) geçici bir ikili sistem dosyası açar. |
| tmpnam tmpnam\_s (C11) | char\* tmpnam(char \*filename);  Mevcut olmayan geçici bir dosya oluşturarak adını döndürür. |
| [ungetc](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/ungetc) | int ungetc(int char, FILE \*stream);  Verilen karakteri (unsigned char) tanımlanan akışa geri gönderir. |
| vfprintf vfprintf\_s (C11) | int vfprintf(FILE \*stream, const char \*format, va\_list arg);  Yapılandırılmış çıkışı bir argüman listesi kullanarak dosya akışına gönderir. |
| vfscanf (C99) vfscanf\_s (C11) | int vfscanf(FILE \*stream, const char \*format, va\_list arg);  Bir dosya akışından okuduğu verileri bir argüman listesi içinde tanımlanan değişken adreslerine aktarır. |
| vprintf vprintf\_s (C11) | int vprintf(const char \*format, va\_list arg);  Yapılandırılmış çıkışı bir argüman listesi kullanarak standart çıkışa (stdout) gönderir. |
| vscanf (C99) vscanf\_s (C11) | int vscanf(const char \*format, va\_list arg);  Klavyeden okuduğu verileri bir argüman listesi içinde tanımlanan değişken adreslerine aktarır |
| vsnprintf (C99) [vsnprintf\_s (C11)](https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio/vsnprintf) | int vsnprintf(char \*restrict buffer, size\_t bufsz, const char \*restrict format, va\_list arg);  Yapılandırılmış çıkışı bir argüman listesi kullanarak boyutu belirli bir karakter dizisine yazar. |
| vsprintf vsprintf\_s (C11) | int vsprintf(char \*buffer, const char \*format, va\_list arg);  Yapılandırılmış çıkışı bir argüman listesi kullanarak bir karakter dizisine yazar. |
| vsscanf (C99) vsscanf\_s (C11) | int vsscanf(const char \*buffer, const char \*format, va\_list arg);  Bir karakter dizisinden okuduğu verileri bir argüman listesi içinde tanımlanan değişken adreslerine aktarır. |

**VERİLER;**

|  |  |
| --- | --- |
| **VERİ TÜRÜ ADI** | **AÇIKLAMA** |
| FILE | Bir dosya ile ilgili giriş veya çıkış işlemleri yapmak için, dosya veya akış hakkında bilgiler içeren bir veridir. |
| fpos\_t | Dosyanın konumunu göstermek için kullanılan long veya long olarak tanımlanan bir veri türüdür. |
| size\_t | Unsigned integer olarak tanımlanan bir veri türüdür. |

**MAKRO DEĞİŞKENLER;**

|  |  |
| --- | --- |
| **DEĞİŞKEN ADI** | **AÇIKLAMA** |
| stdin | Standart giriş akışını gösteren bir işaretçi tanımlayan bir makrodur |
| stdout | Standart çıkış akışını gösteren bir işaretçi tanımlayan bir makrodur |
| stdin | Standart hata akışını gösteren bir işaretçi tanımlayan bir makrodur. |

**MAKRO SABİTLER;**

|  |  |
| --- | --- |
| **SABİT ADI** | **AÇIKLAMA** |
| BUFSIZ | Setbuf() fonksiyonu tarafından kullanılan arabelleğin boyutunu belirleyen bir int değeridir |
| EOF | Dosya sonu durumunu belirten bir negatif int bir değerdir |
| FILENAME\_MAX | Açılan bir dosya adını yüklemek için kullanılacak bir char dizinin azami boyutunu belirler |
| FOPEN\_MAX | Eşzamanlı olarak açılabilecek dosya sayısını gösterir. |
| \_IOFBF | Bir kısaltma (Giriş/Çıkış Tamamen Tamponlu - Input/Output Fully Buffered) olup 0x0000 değerini taşır. Açık bir akış için tamamen tamponlanmış giriş ve çıkış talebi için setvbuf() fonksiyonuna geçirilebilecek bir tamsayıdır |
| \_IOLBF | Bir kısaltma (Giriş/Çıkış Satır Tamponlu - Input/Output Line Buffered) olup 0x0040 değerini taşır. Açık bir akış için satır olarak tamponlanmış giriş ve çıkış talebi için setvbuf() fonksiyonuna geçirilebilecek bir tamsayıdır. |
| \_IONBF | Bir kısaltma (Giriş/Çıkış Tamponlanmamış - Input/Output Not Buffered) olup 0x0004 değerini taşır. Açık bir akış için tamponlanmamış giriş ve çıkış talebi için setvbuf() fonksiyonuna geçirilebilecek bir tamsayıdır. |
| L\_tmpnam  L\_tmpnam\_s (C11) | Tmpnam() fonksiyonu tarafından oluşturulan geçici bir dosya adını yüklemek için kullanılan bir char dizinin boyutunu gösterir.  Tmpnam\_s() fonksiyonu tarafından oluşturulan geçici bir dosya adını yüklemek için kullanılan bir char dizinin boyutunu gösterir. |
| NULL | Boş işaretçi sabiti olan bir makrodur. |
| SEEK\_CUR | Geçerli dosya konumuna göre konumlandırma talebi için fseek() fonksiyonuna geçirilebilecek bir int değerdir |
| SEEK\_END | Dosya sonuna konumuna göre konumlandırma talebi için fseek() fonksiyonuna geçirilebilecek bir int değerdir. |
| SEEK\_SET | Dosya başına göre konumlandırma talebi için fseek() fonksiyonuna geçirilebilecek bir int değerdir |
| TMP\_MAX  TMP\_MAX\_S (C11) | Tmpnam() fonksiyonu tarafından oluşturulan azami benzersiz dosya adı sayısını gösterir.  Tmpnam\_s() fonksiyonu tarafından oluşturulan azami benzersiz dosya adı sayısını gösterir. |

**MELİKE HANLI**

**OKUL NO:222703016**