Sistem Programlama

Ders 12

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

Unix standartları

- Unix uygulamaları mümkün olduğunca farklı Unix sistemlerinde çalışabilecek şekilde dizayn edilmiştir.
- Ancak 1980’ler ile birlikte birçok Unix sistemi oluşmuş ve uygulamalarda farklılıklar ortaya çıkmıştır.
- Bu durum Unix sisteminin ortaya çıkış nedenlerinden biri olan farklı sistemlerde çalışabilme üzelliğine aykırıdır.
- Bu sebeple birçok kullanıcı sistemin standartlarının belirlenmesi konusunda istekte bulunmuştur.

Unix standartları

- 1989 senesinde C dilinin ANSI standartları kabul edilmiştir.
- Bu standartlar ayrıca ISO ve IEC tarafından 1990 yılında kabul edilmiştir.
  - ANSI: American National Standards Institute
  - ISO: International Organization for Standardization
  - IEC: International Electrotechnical Commission
- C standartları günümüzde ISO/IEC tarafından takip edilmektedir.
  - Bu gurpların amacı C dilinde yazılan programların birçok farklı işletim sisteminde düzgün şekilde çalışmasına olanak verecek standartların geliştirilmesidir.
- Bu standartlar sadece dile ait yazım ve anlam kurallarını belirlemez, ayrıca standart kütüphaneleri ve bu kütüphanelerde bulunması gereken fonksiyonları belirtir.
- ISO C kütüphanesi tanımlanan header dosyaları ile 24 farklı alana ayrılabilir.
  - Bakınız: ISO C headers.

Unix standartları

Header	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10	Description
<assert.h>	•	•	•	•	verify program assertion
<complex.h>	•	•	•	•	complex arithmetic support
<ctype.h>	•	•	•	•	character classification and mapping support
<errno.h>	•	•	•	•	error codes (Section 1.7)
<fenv.h>	•	•	•	•	floating-point environment
<float.h>	•	•	•	•	floating-point constants and characteristics
<inttypes.h>	•	•	•	•	integer type format conversion
<iso646.h>	•	•	•	•	macros for assignment, relational, and unary operators
<limits.h>	•	•	•	•	implementation constants (Section 2.5)
<locale.h>	•	•	•	•	locale categories and related definitions
<math.h>	•	•	•	•	mathematical function and type declarations and constants
<setjmp.h>	•	•	•	•	nonlocal goto (Section 7.10)
<signal.h>	•	•	•	•	signals (Chapter 10)
<stdarg.h>	•	•	•	•	variable argument lists
<stdbool.h>	•	•	•	•	Boolean type and values
<stddef.h>	•	•	•	•	standard definitions
<stdint.h>	•	•	•	•	integer types
<stdio.h>	•	•	•	•	standard I/O library (Chapter 5)
<stdlib.h>	•	•	•	•	utility functions
<string.h>	•	•	•	•	string operations
<tgmath.h>	•	•	•	•	type-generic math macros
<time.h>	•	•	•	•	time and date (Section 6.10)
<wchar.h>	•	•	•	•	extended multibyte and wide character support
<wctype.h>	•	•	•	•	wide character classification and mapping support

IEEE Posix

- Posix standartları IEEE tarafından geliştirilmiştir.
  - IEEE: Institute of Electrical and Electronics Engineers
  - Posix: Portable Operating System Interface
- 1003.1 İşletim sistemi arayüz standartları bir işletim sisteminin diğer programlara sağlaması gereken olanakları tanımlar.
  - Bu standartlar çoğu bilgisayar üreticisince kabul edilmiştir.
  - Eğer bu olanaklar sağlanıyorsa, belirtilen işletim sistemi POSIX uyumludur.
  - Bu standartlar sadece Unix veya Unix türevleri ile sınırlı değildir.
  - Bazı üreticiler kendi işletim sistemlerini Posix uyumlu hale getirmiştir.
- 1003.1 standartları bir arayüz tanımlar, bu arayüzün arka planında çalışan işlemleri tanımlamaz.

Unix standartları

Header	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10	Description
<aio.h>	•	•	•	•	asynchronous I/O
<cpio.h>	•	•	•	•	cpio archive values
<dirent.h>	•	•	•	•	directory entries (Section 4.22)
<dlfcn.h>	•	•	•	•	dynamic linking
<fcntl.h>	•	•	•	•	file control (Section 3.14)
<fnmatch.h>	•	•	•	•	filename-matching types
<glob.h>	•	•	•	•	pathname pattern-matching and generation
<grp.h>	•	•	•	•	group file (Section 6.4)
<iconv.h>	•	•	•	•	codeset conversion utility
<langinfo.h>	•	•	•	•	language information constants
<monetary.h>	•	•	•	•	monetary types and functions
<netdb.h>	•	•	•	•	network database operations
<nl_types.h>	•	•	•	•	message catalogs
<poll.h>	•	•	•	•	poll function (Section 14.4.2)
<pthread.h>	•	•	•	•	threads (Chapters 11 and 12)
<pwd.h>	•	•	•	•	password file (Section 6.2)
<regex.h>	•	•	•	•	regular expressions
<sched.h>	•	•	•	•	execution scheduling
<semaphore.h>	•	•	•	•	semaphores
<strings.h>	•	•	•	•	string operations
<tar.h>	•	•	•	•	tar archive values
<termios.h>	•	•	•	•	terminal I/O (Chapter 18)
<unistd.h>	•	•	•	•	symbolic constants
<wordexp.h>	•	•	•	•	word-expansion definitions

Unix standartları

Header	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10	Description
<arpa/inet.h>	•	•	•	•	Internet definitions (Chapter 16)
<net/if.h>	•	•	•	•	socket local interfaces (Chapter 16)
<netinet/in.h>	•	•	•	•	Internet address family (Section 16.3)
<netinet/tcp.h>	•	•	•	•	Transmission Control Protocol definitions
<sys/mman.h>	•	•	•	•	memory management declarations
<sys/select.h>	•	•	•	•	`select` function (Section 14.4.1)
<sys/socket.h>	•	•	•	•	sockets interface (Chapter 16)
<sys/stat.h>	•	•	•	•	file status (Chapter 4)
<sys/statvfs.h>	•	•	•	•	file system information
<sys/times.h>	•	•	•	•	process times (Section 8.17)
<sys/types.h>	•	•	•	•	primitive system data types (Section 2.8)
<sys/un.h>	•	•	•	•	UNIX domain socket definitions (Section 17.2)
<sys/utsname.h>	•	•	•	•	system name (Section 6.9)
<sys/wait.h>	•	•	•	•	process control (Section 8.6)

Single Unix spesifikasyonu opsiyonel header dosyaları ile Posix standartlarını genişletir.

Unix standartları

- Limitler
  - Unix sistemlerinde birçok özel sayı ve sabit tanımlanmıştır
  - İki farklı limit tipine ihtiyaç vardır. Bunlar:
    - Derleme zamanı limitleri
      - Örneğin bir short int değişkenin alabileceği maksimum değer.
    - Çalışma zamanı limitleri
      - Örneğin bir dosya isminde en fazla kaç karekter olabilir.
    - Derleme zamanı limitleri header dosyalarında saklanır ve programlar bu dosyaları ekleyerek bu limitleri kullanabilir.
    - Çalışma zamanı limitlerine erişebilmek için işlemler bir fonksiyon çağırırlar ve limit değerine erişirler.

Unix standartları

- Limitlerle ilgili aşağıda belirtilen problemler gözlemlenebilir.
  - Bazı limitler bir sistemde sabit olarak oluşturulmuş ve header dosyasında bulunurken, başka bir sisemde farklı değere sahip ve bir çalışma zamanı fonksiyonu ile değeri öğreniliyor olabilir.
    - Örneğin SVR4 öncesinde, System V sadece 14 karekterlik dosya isimlerine izin verirken, BSD-tabanlı sistemler 255 karektere kadar dosya isimlerine izin vermektedir.
    - Günümüzde kullanılan çoğu Unix sisteminde birden fazla dosya sistemi tipi desteklenmekte ve her sistemin kendine ait limitleri bulunmaktadır. Bu sebeple çalışma sırasında belli limitler dosyanın hangi sistemin parçaşı olduğuna göre değişebilmektedir.

Unix standartları

- Limitler
  - Bu problemleri çözmek için, üç farklı limit tipi tanımlanmıştır.
    - Derleme zamanı limitleri (header dosyalarında)
    - Dosya veya klasör ile ilgili olmayan çalışma zamanı limitleri (sysconf fonksiyonu)
    - Dosya veya klasör ile ilgili çalışma zamanı limitleri (pathconf ve fpathconf fonksiyonları).

Unix standartları

- sysconf, pathconf, and fpathconf fonksiyonları
- **`long sysconf (int name)`**
  - bir dosya veya klasör ile ilgili olmayan çalışma zamanı limitlerini öğrenmek için kullanılır.
- **`long pathconf (const char *pathname, int name)`**

  ve
- **`long fpathconf (int filedes, int name)`**
  - Bir dosya veya klasör ile ilgili çalışma zamanı limitlerini öğrenmek için kullanılır

Unix standartları

- ISO C limitleri
  - ISO C tarafından tanımlanan bütün limitler derleme-zamanı limitleridir.
  - Bir sonraki slaytta <limits.h> dosyasında tanımlanan sabitler görülmektedir.
    - Üçüncü sütunda ISO C tarafından kabul edilen minimum değerler görülmektedir.
      - Bu değere göre 16-bit tam sayılarla birin tamamlayıcısı (one’s complement) işlemlerin yapılmasına olanak verir.
    - Dördüncü sütun 32-bit tam sayılarla ikinin tamamlayıcısı (two’s complement) işlemleri yapan Linux sistemleri için değerlerdir.

Unix standartları

Name	Description	Minimum acceptable value	Typical value
CHAR_BIT	bits in a char	8	8
CHAR_MAX	max value of char	(see later)	127
CHAR_MIN	min value of char	(see later)	–128
SCHAR_MAX	max value of signed char	127	127
SCHAR_MIN	min value of signed char	–127	–128
UCHAR_MAX	max value of unsigned char	255	255
INT_MAX	max value of int	32,767	2,147,483,647
INT_MIN	min value of int	–32,767	–2,147,483,648
UINT_MAX	max value of unsigned int	65,535	4,294,967,295
SHRT_MAX	max value of short	32,767	32,767
SHRT_MIN	min value of short	–32,767	–32,768
USHRT_MAX	max value of unsigned short	65,535	65,535
LONG_MAX	max value of long	2,147,483,647	2,147,483,647
LONG_MIN	min value of long	–2,147,483,647	–2,147,483,648
ULONG_MAX	max value of unsigned long	4,294,967,295	4,294,967,295
LLONG_MAX	max value of long long	9,223,372,036,854,775,807	9,223,372,036,854,775,807
LLONG_MIN	min value of long long	–9,223,372,036,854,775,807	–9,223,372,036,854,775,808
ULLONG_MAX	max value of unsigned long long	18,446,744,073,709,551,615	18,446,744,073,709,551,615
MB_LEN_MAX	max number of bytes in a multibyte character constant	1	6

Dosya zamanları

- Her dosya için üç zaman bilgisi tutulur:

Bilgi	Tanım	Etkileyen fonksiyon örneği	ls (1) opsiyonu
st_atime	Dosyanın son erişilme zamanı	read	-u
st_mtime	Dosya verisinin son değiştirilme zamanı	write	Varsayılan
st_ctime	i-node durumunun son değişme zamanı	chmod, chown	-c

Dosya zamanları

- utime fonksiyonu: Dosya erişim ve değiştirilme zamanı utime fonksiyonu ile değiştirilebilir.

```
#include <utime.h>

int utime(const char *pathname, const struct utimbuf *times);
```

Dönüş: OK ise 0, hata ise -1.

  - Eğer zaman değeri NULL verilirse, dosya erişim ve dosya değişim zamanı şu anki zamana eşitlenir.
    - Bu işlem için dosyanın sahibi olmalısınız veya dosya üzerinde yazma hakkınız olmalı.
  - Eğer zaman değeri NULL değil ise, timeval timbuf yapısı içerisinde verilen değere göre dosya erişim ve dosya değişim zamanları ayarlanır.
    - Bu işlem için dosyanın sahibi olmalısınız. Dosyaya yazma hakkı yeterli değildir.
  - Her iki işlem sonucunda stc_time değeri şu anki zaman ayarlanır.
  - Örnek19.

mkdir ve rmdir fonksiyonları

- Klasörler mkdir fonksiyonu ile oluşturulur.

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

int mkdir(const char *path, mode t mode);
```

Dönüş: OK ise 0, hata ise -1.

```
#include <unistd.h>

int rmdir(const char *pathname);
```

Dönüş: OK ise 0, hata ise -1

  - Eğer klasöre işaret eden link sayısı 0 olursa (fonksiyon çalıştıktan sonra) ve başka bir işlem klasörü açık tutmuyorsa, klasör silinir.
  - Bu işlemin yapılabilmesi için klasörün boş olması gerekir (sadece . ve .. olmalı)

Klasörleri okumak

- Bu fonksiyonlar kullanılarak bir klasörün içeriğine erişilebilir.

```
#include <dirent.h>

DIR *opendir(const char *filename);
```

Dönüş: OK ise işaretçi, hata ise NULL

```
struct dirent *readdir(DIR *dp);
```

Dönüş: OK ise işaretçi, hata ve klasör sonunda NULL

```
void rewinddir(DIR *dp);

int closedir(DIR *dp);
```

Dönüş: OK ise 0, hata ise -1.

```
long telldir(DIR *dp);
```

Dönüş: dp işaretçisi ile alakalı klasördeki bulunulan pozisyon.

```
void seekdir(DIR *dp, long loc);
```

Klasörleri okumak

- dirent yapısının içeriği şu şekildedir:

```
struct dirent {
    ino_t d_ino;                        // i-node numarası
    char d_name[NAME_MAX + 1];      // NULL ile sonlanan dosya ismi
}
```

- ls1.c

chdir, fchdir ve getcwd fonksiyonları

- Çağıran işlemin mevcut çalışma klasörünü chdir ve fchdir fonksiyonları ile değiştirebiliriz.

```
#include <unistd.h>

int chdir(const char *pathname);

int fchdir(int filedes);
```

Dönüş: OK ise 0, hata ise -1.

  - mycd.c.

- Çağıran işlemin mevcut çalışma klasörünü getcwd fonksiyonu ile öğrenebiliriz.

```
#include <unistd.h>

char *getcwd(char *buf, size_t size);
```

Dönüş: OK ise buf, hata ise NULL.

İşlem çevresi

- main fonksiyonu
  - Bir C programı çalışmaya main fonksiyonundan başlar.
  - main fonksiyonunun prototipi şu şekildedir.

```
int main(int argc, char *argv[])
```

  - argc komut satırından verilen argüman sayısıdır.
  - argv ise bu argümanlara işaret eden işaretçiler dizisidir.
  - ANSI C ve POSIX.1 standartlarına aşağıdaki durum garantilenmiştir.

```
argv[argc] == NULL
```

  - C programı kernel tarafından başlatılır (exec fonksiyonlarında biriyle)
  - Kernel tarafından main fonksiyonunun (veya hangi noktadan başlangıç tanımlandıysa) başlatılması için gerekli ayarlar yapılır.

İşlem sonlandırma

- 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.

Exit fonksiyonları

```
#include <stdlib.h>
void exit(int status);
void _Exit(int status);
#include <unistd.h>
void _exit(int status);
```

- _exit ve _Exit
  - Hemen kernel'e dönerler.
  - _exit POSIX.1 de tanımlıdır.
  - _Exit ISO C99 da tanımlıdır.
  - Unix sisteminde ikisi aynı işlemi yapar.
- exit önce bazı temizlik işlemleri yapar daha sonra dönüş yapar.
  - Açık olan veri akışları için fclose fonksiyonu çalışır.
- Bu fonksiyonlar çıkış değeri alırlar.

Exit fonksiyonları

- Çoğu Unix sistem kabuklarında işlemlerin çıkış değerlerini incelemek mümkündür.
- Aşağıda belirtilen durumlarda çıkış değeri tanımsızdır:
  - Exit fonksiyonları çıkış değersiz çalıştırılırsa
  - main fonksiyonu bir dönüş değeri ile sonlanmazsa
  - main fonksiyonu bir tam sayıya dönecek şekilde tanımlanmamışsa
- main fonksiyonu integer dönüşlü tanımlanmış ve main başka bir noktada dönüş yapıyorsa, bu işlemin çıkış değer 0 olur.
- main fonksiyonunda bir tam sayıya dönüş yapmakla exit fonksiyonunu aynı değerle çalıştırmak aynı anlama gelir.
  - exit(0) ile return(0) aynıdır.
- Örnek21
  - echo $?

atexit fonksiyonu

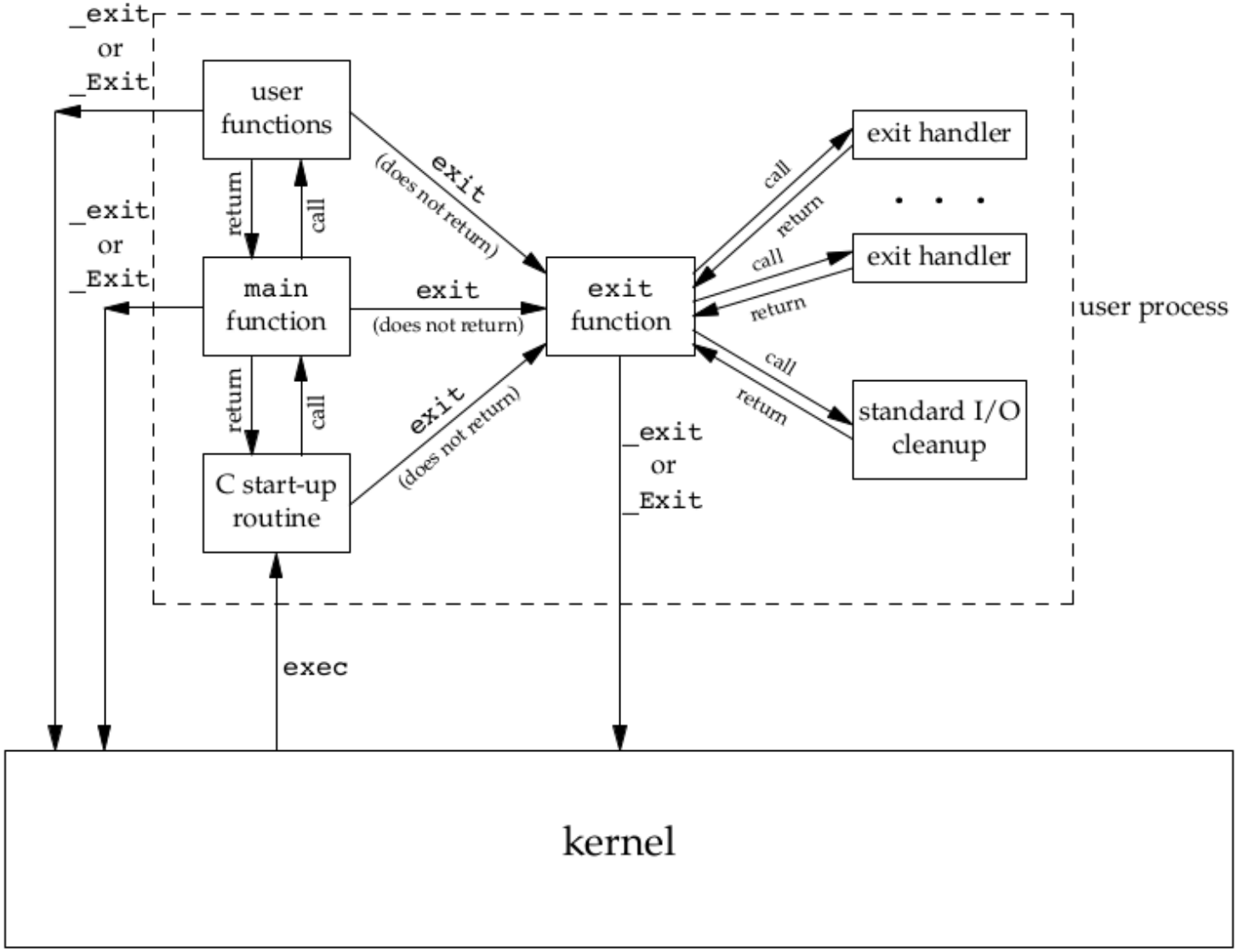
- ISO C standartlarına göre 32 taneye kadar exit tarafından otomatik olarak çağırılacak fonksiyon tanımlayabiliriz.
- Bu fonksiyonlar çıkış denetleyicisi olarak adlandırılır ve atexit fonksiyonu ile kayıt edilir.

```
#include <stlib.h>

int atexit(void (*func) (void));
```

- exit fonksiyonu kayıt edilen fonksiyonları ters sıradan çağırır.
- Bu fonksiyonlar birden fazla kez kayıt edilebilir.
- Örnek22

atexit fonksiyonu



Emir satırı argümanları

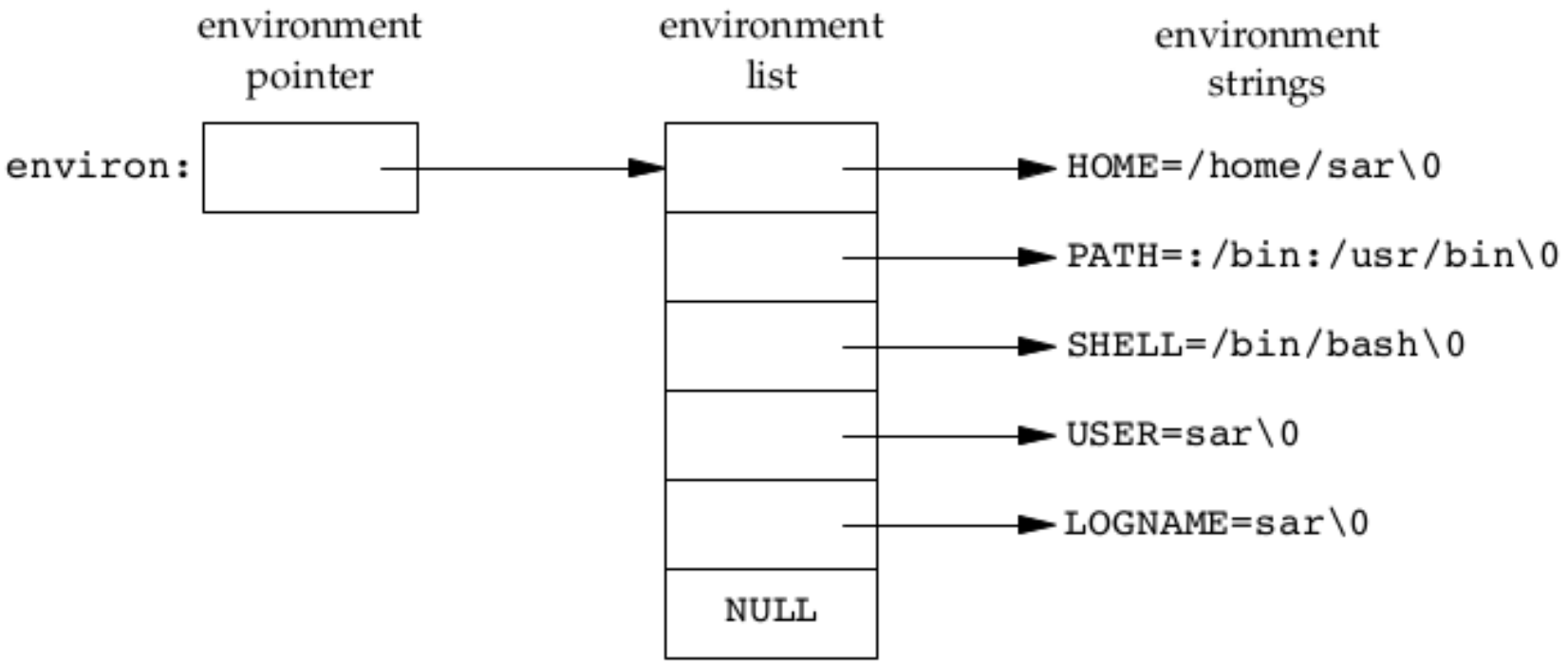
- Bir program çalıştırıldığında emir satırı argümanları programa gönderilebilir.
- Örnek23

Çevre değişkenleri listesi

- Her programa bir çevre değişkenleri listesi (İng: Environment list) verilir.
- Argüman listesinde olduğu gibi çevre değişkenleri listesi, her biri bir başka çevre değişkenine işaret eden, işaretçiler dizidisir.
- Bu işaretçilerin listesi environ global değişkeninde saklanır.

```
extern char **environ;
```

Çevre değişkenleri listesi



Kullanım şekli NAME=value şeklindedir.
Aşağıdaki şekilde çevre değişkenleri listesine ulaşabiliriz.

```
int main(int argc, char *argv[], char *envp[]);
```

environ değişkenini kullanmak daha iyi bir seçimdir.