Sistem Programlama

Ders 10

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

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
  - cc -std=c99 hello.c
  - echo $?

atexit fonksiyonu

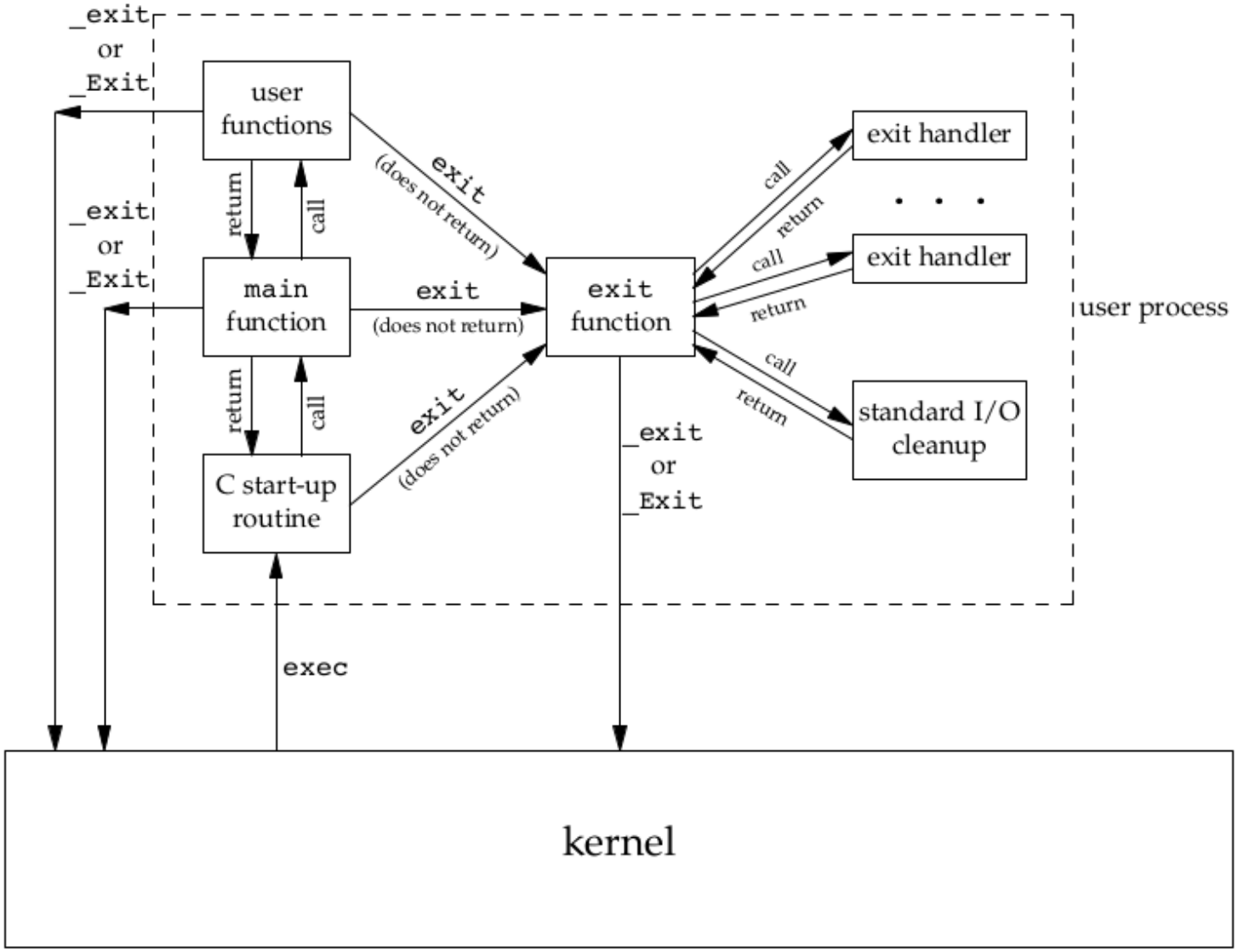
- ISO C standartlarına göre 32 taneye kadar exit tarafından otomatik olarak çağırılacak fonksiyon tanımlayabiliriz.
- Bu fonksiyonlar çıkış denetleyicisi olarak adlandırılır ve atexit fonksiyonu ile kayıt edilir.

```
#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

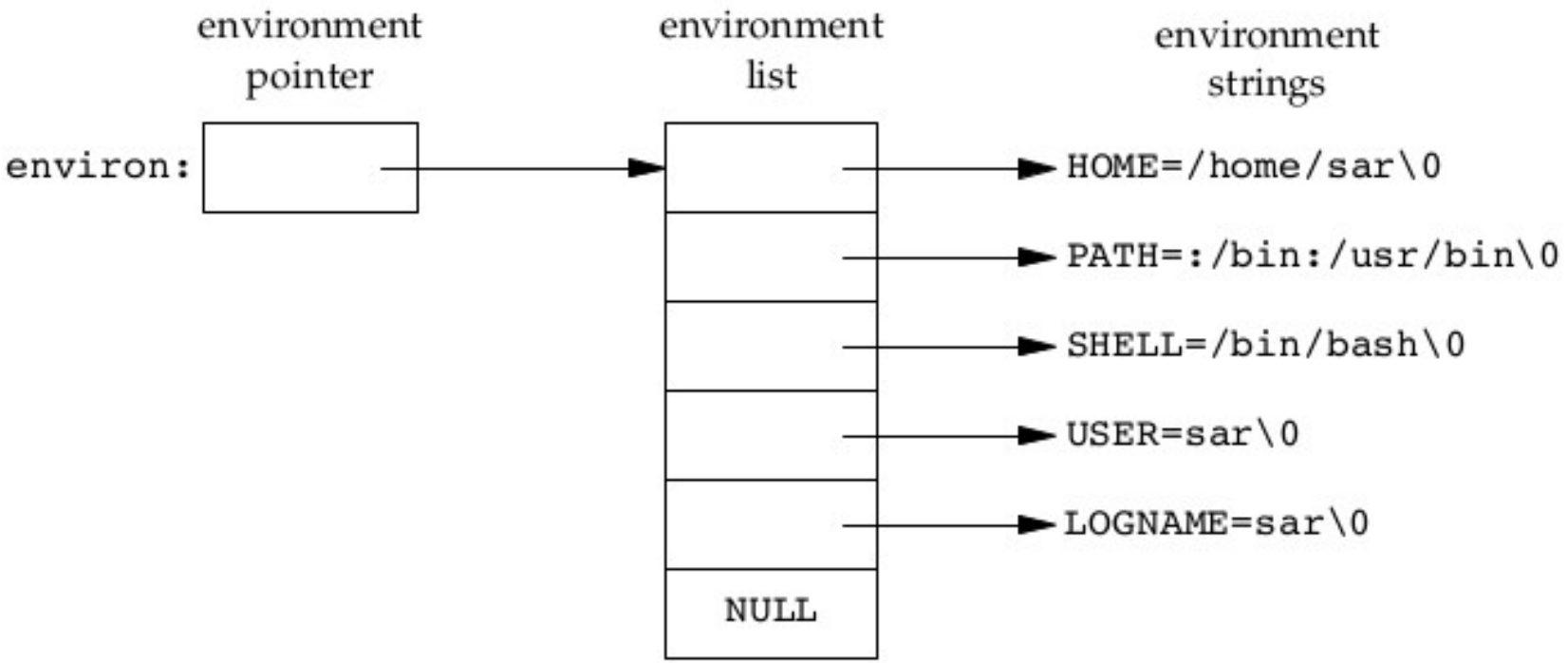
- Bir program çalıştırıldığında emir satırı argümanları programa gönderilebilir.
- Örnek23

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.

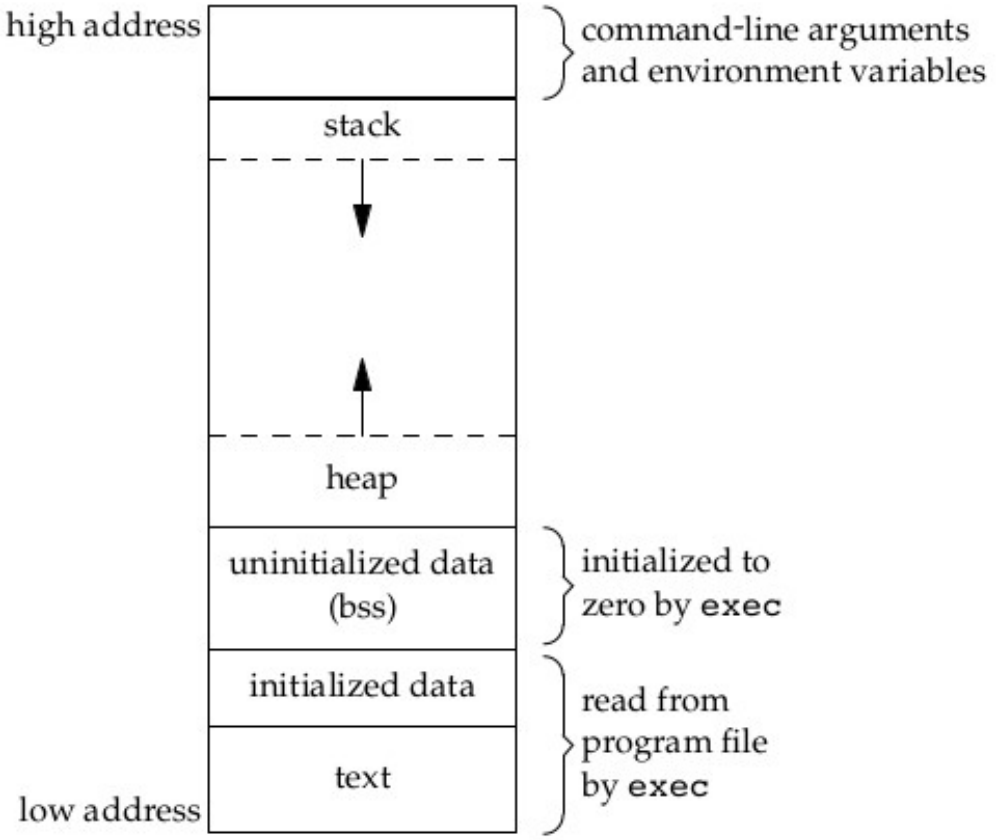
Bir C programının hafıza yapısı

- Geleneksel olarak bir C programı aşağıda belirtilen parçalardan oluşur
  - Metin bölümü, CPU tarafından çalıştırılan makine kodunu içerir.
    - Metin bölümü genellikle paylaşılabilir olur. Bu sayede sık kullanılan programlar tekrar tekrar hafızaya yüklenmez.
  - Değeri atanmış veri bölümü
    - Bir fonksiyon dışında aşağıda şekilde atanmış parametreler bu bölümde saklanır.
      - `int maxcount = 99;`
  - Değeri atanmamış bölümü (ayrıca bss bölümü olarak bilinir)
    - Bir fonksiyon dışında aşağıdaki şekilde atanmış parametreler bu bölümde saklanır.
      - `long sum[1000];`
    - Ayrılan verinin içerisine program çalıştırıldığında kernel tarafından 0 veya null işaretçi atanır.

Bir C programının hafıza yapısı

- Geleneksel olarak bir C programı aşağıdaki yapılardan oluşur.
  - Stack (yığın)
    - Otomatik değişkenler ile birlikte her fonksiyon çağrısı için bazı bilgiler bu kısımda saklanır.
      - Her fonksiyon çağrısında, fonksiyon bittiğinde nereye dönüleceği, çağıranın çevre değişkenleri ile ilgili bazı değişkenler, makine registerları gibi, yığın kısmında saklanır.
      - Yeni çağırılan fonksiyonun otomatik ve geçici değişkenleri için yığında yer ayrılır.
  - Heap
    - Dinamik olarak oluşturulan değişkenler burada saklanır.

Bir C programının hafıza yapısı



size /bin/sh

Hafıza alma

- ISO C üç farklı hafıza alma fonksiyonu tanımlamıştır.
  - malloc
    - Belirtilen byte sayısı kadar hafıza alır.
    - Ayrılan hafızanın ilk değeri belirsizdir.
  - calloc
    - Belirtilen büyüklükte olan nesnelerden belirtilen sayıda miktarı için hafıza ayırır.
    - Ayrılan yerdeki bütün bitler 0 değeri alır.
  - realloc
    - Daha önce alınmış bir alanı arttırır veya azaltır.
    - Daha önce ayrılmış alanın başka bir yere taşınması gerekebilir.
    - Yeni ayrılan bölgedeki değerler belirsizdir.

Hafıza alma

```
#include <stdlib.h>

void *malloc(size_t size);

void *calloc(size_t nobj, size_t size);

void *realloc(void *ptr, size_t newsize);
```

Dönüş: OK ise null olmayan işaretçi, hata ise NULL

```
void free(void *ptr);
```

- Üç fonksiyon tarafından dönen işaretçi düzgün şekilde sıralanmıştır. Bu sayede nesneler tarafından kullanılabilir.
- void* işaretçi döndükleri için cast yapmadan başka bir işaretçi tipine eşitlenebilir.
- free fonksiyonu alınmış olan alanın geri verilmesini sağlar.
  - Geri verilen alan genellikle kullanılabilir alana geri iade edilir, böylece alloc fonksiyonları tarafından alınabilir.
- realloc fonksiyonu daha önce ayrılmış alanı arttırır veya azaltır.
  - Fonksiyonun en son argümanı yeni alanın büyüklüğüdür.

Hafıza alma

- Hafıza alma işlemler genellikle sbrk (2) sistem çağrısı kullanılarak yapılır.
  - Bu sistem çağrısı heap kısmında ayrılmış alanı arttırır veya azaltır.
- Dinamik olarak ayrılmış alanın sonundan sonraya veya başından önceye yazmak iç kayıtlara veya diğer dinamik olarak ayrılan nesnelere zarar verebilir.
  - Birçok sistemde istenilenden bir miktar fazla yer ayrılılr. Bu alan iç kayıt tutulması için kullanılır.
- Örnek24.