

# Sistem Programlama

---

DR. ÖĞR. ÜYESİ ABDULLAH SEVİN

# İçerik

---

a) String işlemler

- <http://web.eecs.utk.edu/~jplank/plank/classes/cs360/360/notes/Strings-In-C/index.html>

# String-strcat()

---

`char *strcat(char *s1, const char *s2);`

- ❑ Strcat(), s1 ve s2'nin her ikisinin de boş/null ile sonlandırılmış dizeler olduğunu varsayar.
- ❑ Strcat() daha sonra s2'yi s1'in sonuna birleştirir.
- ❑ Strcat(), s1'de bu ekstra karakterleri tutmak için yeterli alan olduğunu varsayar. Aksi takdirde, ayırmadığınız hafızayı ezmeye başlarsınız.

```
UNIX> bin/strcat
Give
Give Him
Give Him Six!
UNIX>
```

```
/* Using strcpy() and strcat() to create the string "Give Him Six!" incrementally. */
#include <stdio.h>
#include <string.h>

int main()
{
    char givehimsix[15];

    strcpy(givehimsix, "Give");
    printf("%s\n", givehimsix);
    strcat(givehimsix, " Him");
    printf("%s\n", givehimsix);
    strcat(givehimsix, " Six!");
    printf("%s\n", givehimsix);
    return 0;
}
```

# String-strcat()

---

□ src/strcat2.c'ye bakalım. Çıktının neden böyle olduğunu açıklayabilir misiniz?

```
UNIX> bin/strcat2
give: 0xbfffe060 him: 0xbfffe050 six: 0xbfffe040
Give Him Six!
deh T.J. Houshmandzadeh Six!
deh Help! T.J. Houshmandzadeh Help! Six!
UNIX>
```

□ \*adres yerleşimleri değişmiş

```
UNIX> bin/makej 50
|||||
UNIX> bin/strcat3 50
|||||
UNIX>
```

# String-strcat()

- ❑ C tarzı string işlemleri, C++ tarzı string işlemlere göre biraz daha zordur.
- ❑ Örneğin, belirli sayıda `j` içeren bir dizi oluşturmak istediğinizi varsayalım.
- ❑ C'de diziyi tahsis etmek için önce `malloc()`'u çağırmanız gerektiğinden bu biraz daha zordur.

```

/* Trying to use strcat() like C++ string concatenation. */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char **argv)
{
    char *s;
    int i;
    int n;

    if (argc != 2) { fprintf(stderr, "usage: strcat3 number\n"); exit(1); }

    n = atoi(argv[1]);
    s = (char *) malloc(sizeof(char)*(n+1));
    strcpy(s, "");

    for (i = 0; i < n; i++) strcat(s, "j"); /* Here's the strcat() call, wh

    printf("%s\n", s);
    return 0;
}

```

```
/* Create a string with a given number of j's by using string concatenation. */

#include <iostream>
#include <cstdio>
#include <cstdlib>
using namespace std;

int main(int argc, char **argv)
{
    int i, n;
    string s;

    if (argc != 2) { fprintf(stderr, "usage: makej number\n"); exit(1); }
    n = atoi(argv[1]);

    for (i = 0; i < n; i++) s += "j";    // Here is the string concatenation.
    cout << s << endl;
    return 0;
}
```



# String-strcat()

---

❑ Gerçekten büyük bir sayı üzerinde deneyelim.

❑ Burada, standart çıktıyı /dev/null'a yönlendirelim,

```
UNIX> time sh -c "bin/makej 1000 > /dev/null"
0.002u 0.004s 0:00.01 0.0% 0+0k 0+0io 0pf+0w # Blink of an eye.
UNIX> time sh -c "bin/makej 10000 > /dev/null"
0.002u 0.004s 0:00.00 0.0% 0+0k 0+0io 0pf+0w # Blink of an eye.
UNIX> time sh -c "bin/makej 100000 > /dev/null"
0.004u 0.004s 0:00.01 0.0% 0+0k 0+0io 0pf+0w # Blink of an eye.
UNIX> time sh -c "bin/strcat3 1000 > /dev/null"
0.002u 0.004s 0:00.00 0.0% 0+0k 0+0io 0pf+0w # Blink of an eye.
UNIX> time sh -c "bin/strcat3 10000 > /dev/null"
0.039u 0.004s 0:00.04 75.0% 0+0k 0+0io 0pf+0w # A little slower
UNIX> time sh -c "bin/strcat3 100000 > /dev/null"
3.468u 0.005s 0:03.47 99.7% 0+0k 0+0io 0pf+0w # Nearly 100 times slower!
UNIX>
```

❑ >

❑ Redirection-Yönlendirmeler, ileride detaylı olarak bahsedilecek

❑ /dev/null

❑ Her bir Linux sisteminde bulunan özel bir dosyadır. Ancak, diğer birçok sanal dosyanın aksine, okumak yerine yazmak için kullanılır. /dev/null'a yazdığınız her şey atılır (çöp kutusu gibi). Bir UNIX sisteminde boş aygıt olarak bilinir.

# String-strcat()

- ❑ Sorunu görüyor musunuz? C++ dizesi, dizinin uzunluğunu korur, bu nedenle birleştirme hızlıdır.
- ❑ Aksine, strcat() her çağrıda dizgenin sonunu bulmak zorundadır, bu da programı  $O(n^2)$  yapar.
- ❑ String'in sonunun nerede olduğunu bildiğimiz için düzeltebiliriz.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char **argv)
{
    char *s;
    int i;
    int n;

    if (argc != 2) { fprintf(stderr, "usage: strcat4 number\n"); exit(1); }

    n = atoi(argv[1]);
    s = (char *) malloc(sizeof(char)*(n+1));
    strcpy(s, "");

    for (i = 0; i < n; i++) strcat(s+i, "j"); /* The only changed line */

    printf("%s\n", s);
    return 0;
}
```

```
UNIX> bin/strcat4 50 jooooooooooooooooooooooooooooooooooooooooooooooooooooo
UNIX> time sh -c "bin/strcat4 100000 > /dev/null"
0.003u 0.004s 0:00.01 0.0% 0+0k 0+0io 0pf+0w
UNIX>
```

- **command > file**: Sends standard output to **<file>**
- **command < input**: Feeds a command input from **<input>**

```
$ command1 | command2 | command3
```

\*Bir komutu diğerine "iletmek" için

# String-strlen()

---

- Strlen(), s'nin boş sonlandırılmış bir dize olduğunu varsayar. Boş karakterden önceki karakter sayısını döndürür. Yani uzunluğunu!

```
#include <stdio.h>
#include <string.h>

int main()
{
    char give[5];
    char him[5];
    char six[5];

    strcpy(give, "Give");
    strcpy(him, "Him");
    strcpy(six, "Six!");

    printf("%s %s %s\n", give, him, six);
    printf("%ld %ld %ld\n", strlen(give), strlen(him), strlen(six));
    return 0;
}
```

UNIX> **bin/strlen**  
Give Him Six!  
4 3 4

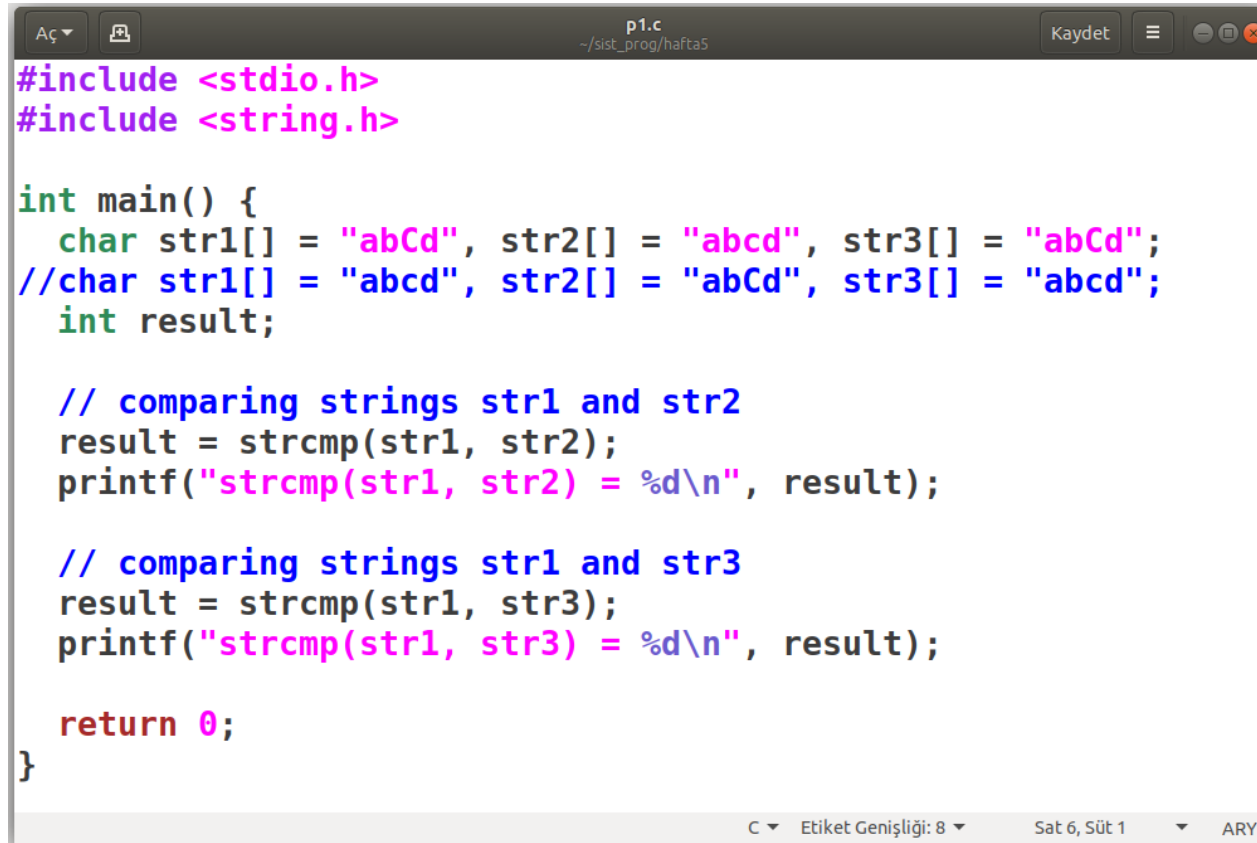


# String-strcmp() ve strncmp()

---

- ❑ **Strcmp()**, iki string ifadenin sözlüksel bir karşılaştırmasını gerçekleştir.
  - ❑ *Eşitlerse 0,*
  - ❑ *s1 s2'den küçükse negatif,*
  - ❑ *aksi takdirde pozitif bir sayı döndürür.*
- ❑ Oldukça fazla strcmp() kullanacağız, çünkü iki stringi karşılaştırmanın en kolay yolu bu.
- ❑ **Strncmp()**, boş karaktere henüz ulaşılmadıysa, n karakterden sonra karşılaştırmayı durdurur

# String-strcmp() ve strncmp()



```
p1.c
~/sist_prog/hafta5
Kaydet

#include <stdio.h>
#include <string.h>

int main() {
    char str1[] = "abCd", str2[] = "abcd", str3[] = "abCd";
    //char str1[] = "abcd", str2[] = "abCd", str3[] = "abcd";
    int result;

    // comparing strings str1 and str2
    result = strcmp(str1, str2);
    printf("strcmp(str1, str2) = %d\n", result);

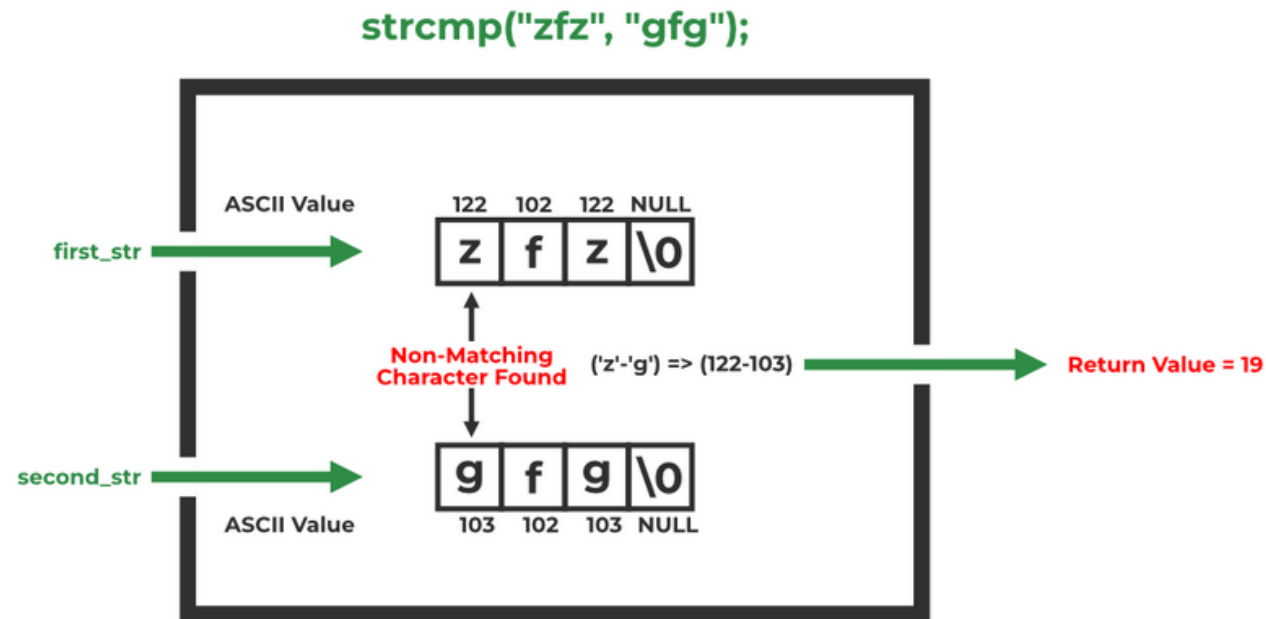
    // comparing strings str1 and str3
    result = strcmp(str1, str3);
    printf("strcmp(str1, str3) = %d\n", result);

    return 0;
}
```

C Etiket Genişliği: 8 Sat 6, Süt 1 ARY

# String-strcmp() ve strncmp()

---



<https://www.geeksforgeeks.org/strcmp-in-c/>

# String-strncmp() ve strncmp()

---

```
#include <stdio.h>
#include <string.h>

int main () {
    char str1[20];
    char str2[20];
    int result;

    strcpy(str1, "hello");
    strcpy(str2, "hello WORLD");

    //This will compare the first 4 characters
    result = strncmp(str1, str2, 4);

    if(result > 0) {
        printf("ASCII value of first unmatched character of str1 is greater than str2");
    } else if(result < 0) {
        printf("ASCII value of first unmatched character of str1 is less than str2");
    } else {
        printf("Both the strings str1 and str2 are equal");
    }

    return 0;
}
```

# String-strchr()

**char** \*strchr(const char \*s, int c);

❑ **Strchr()**, C string'lerinde tek karakterler için "bulma" işlemidir.

❑ c bir tamsayıdır, ancak bir karakter olarak ele alınır. Strchr(), s cinsinden c'ye eşit karakterin ilk geçtiği yere bir işaretçi döndürür. s, c'yi içermiyorsa, NULL döndürür.

```
/* Use strchr() to determine if each line of standard input has a space. */  
  
#include <stdio.h>  
#include <string.h>  
  
int main()  
{  
    char line[100];  
    char *ptr;  
  
    while (fgets(line, 100, stdin) != NULL) {  
        ptr = strchr(line, ' ');  
        if (ptr == NULL) {  
            printf("No spaces\n");  
        } else {  
            printf("Space at character %ld\n", ptr-line);  
        }  
    }  
    return 0;  
}
```

UNIX> **bin/strchr Jim**

No spaces

**Jim Plank** Space at character 3

**James Plank** Space at character 5

**HI!** Space at character 0

**HI!!** Space at character 0 <

**CNTL-D>**

UNIX>

# String-strchr()

□ Biraz modifikasyon ile bütün karakterlerin yerlerini gösterebiliriz;

```
#include <stdio.h>
#include <string.h>

int main()
{
    char line[100];
    char *ptr;

    while (fgets(line, 100, stdin) != NULL) {
        ptr = strchr(line, ' ');
        if (ptr == NULL) {
            printf("No spaces\n");
        } else {
            while (ptr != NULL) {
                printf("Space at character %ld\n", ptr-line);
                ptr = strchr(ptr+1, ' ');
            }
        }
    }
    return 0;
}
```

```
UNIX> bin/strchr2
Jim
No spaces
Jim Plank
Space at character 3
Jim Plank
Space at character 3
Space at character 4
    Give Him Six!!!
Space at character 0
Space at character 1
Space at character 6
Space at character 7
Space at character 8
Space at character 12
Space at character 13
Space at character 14
<CNTL-D>
UNIX>
```



# String-scanf()

---

- ❑ `scanf()`, `printf()` gibidir. Ancak parametreleri uçbirime yazmak yerine uçbirimden okur (veya standart girdi ne ise).
- ❑ `scanf()`'in insanların kafasını karıştırdığı nokta, C'de referans değişkenleri olmamasıdır, dolayısıyla işaretçiler kullanmanız gerekir.
- ❑ Biçim dizgisine `"%d"` yazarsanız, `scanf()` bir tamsayı okuyacaktır. Geçmeniz gereken parametre, okumak istediğiniz tamsayıya bir işaretçidir.
- ❑ Tamsayı için depolama alanı mevcut olmalıdır. `scanf()`, tamsayıyı standart girdiden okuyacak ve tamsayının dört baytını dolduracaktır.

# String-scanf()

□ scanf()'i çağırdığımızda, standart girdiden bir tamsayı okumasını ve bu dört baytı o tamsayı ile doldurmasını söylüyorum. Scanf() yaptığı başarılı okumaların sayısını döndürür.

```
/* Read a single integer from standard input using scanf. */

#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i;

    if (scanf("%d", &i) == 1) {
        printf("Just read i: %d (0x%x)\n", i, i);
    } else {
        printf("Scanf() failed for some reason.\n");
    }
    exit(0);
}
```

```
UNIX> bin/scanf1
10
Just read i: 10 (0xa)
UNIX> bin/scanf1
Fred
Scanf() failed for some reason.
UNIX> bin/scanf1
15.999999999999999
Just read i: 15 (0xf)
UNIX> bin/scanf1
-15.999999999999999
Just read i: -15 (0xfffffff1)
UNIX> bin/scanf1
<CNTL-D>
Scanf() failed for some reason.
UNIX> echo "" | bin/scanf1
Scanf() failed for some reason.
UNIX> echo 15fred | bin/scanf1
Just read i: 15 (0xf)
UNIX>
```

# String-scanf()

---

- ❑ Derlenir (her ne kadar bazı derleyiciler hatalı olduğunu anlayabilir ve size uyarabilir).
- ❑ Hatanın ortaya çıkıp çıkmaması bir şans meselesidir.

```
int main()
{
    int *i;

    printf("i = 0x%lx\n", (unsigned long) i);
    if (scanf("%d", i) == 1) {
        printf("Just read i: %d (0x%x)\n", *i, *i);
    } else {
        printf("Scanf() failed for some reason.\n");
    }
    exit(0);
}
```

```
UNIX> echo 10 | bin/scanf2
i = 0x7fff5fc01052
Bus error
UNIX>
```

# String-scanf()

---

- ❑ Aşağıdaki program (src/scanf3.c), standart girdiden bir dize okumak ve ardından tek tek karakterleri yazdırmak için scanf()'i kullanır:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char s[10];
    int i;

    if (scanf("%s", s) != 1) exit(0);

    for (i = 0; s[i] != '\0'; i++) {
        printf("Character: %d: %3d %c\n", i, s[i], s[i]);
    }
    exit(0);
}
```

```
UNIX> echo "Jim-Plank" | bin/scanf3
Character: 0:  74 J
Character: 1: 105 i
Character: 2: 109 m
Character: 3:  45 -
Character: 4:  80 P
Character: 5: 108 l
Character: 6:  97 a
Character: 7: 110 n
Character: 8: 107 k
UNIX>
```

# String-scanf()

---

- ❑ String'ler ile Scanf() sorunludur. Özellikle, 10'dan fazla karakter içeren bir string girdiğinizde ne olacağını düşünün.
- ❑ Bellek, tıpkı yukarıdaki "T. J. Houshmanzadeh" ile strcpy() ve strcat() örneklerinde olduğu gibi zorlanacaktır.
- ❑ Örneğin bin/scanf3'e 80.000 'j' karakterli bir dizi gönderelim:

```
UNIX> bin/makej 80000 | bin/scanf3  
Segmentation fault: 11  
UNIX>
```

# String-Sscanf()

- ❑ Sscanf(), ilk parametresi olarak ek bir string alması ve standart girdi yerine bu string'den "okuması" dışında, tıpkı scanf() gibidir.
- ❑ Yaptığı doğru eşleşmelerin sayısını döndürür. Bu nedenle, stringleri tamsayılara ve double değişkene dönüştürmek için oldukça uygundur.

```
#include <stdio.h>

int main()
{
    char buf[1000];
    int i, h;
    double d;

    while (fgets(buf, 1000, stdin) != NULL) {
        if (sscanf(buf, "%d", &i) == 1) {
            printf("When treated as an integer, the value is %d\n", i);
        }
        if (sscanf(buf, "%x", &h) == 1) {
            printf("When treated as hex, the value is 0x%x (%d)\n", h, h);
        }
        if (sscanf(buf, "%lf", &d) == 1) {
            printf("When treated as a double, the value is %lf\n", d);
        }
        if (sscanf(buf, "0x%x", &h) == 1) {
            printf("When treated as a hex with 0x%x formatting, the value is 0x%x (%d)\n", h, h);
        }
        printf("\n");
    }
}
```

UNIX> bin/sscanf1

10

When treated as an integer, the value is 10

When treated as hex, the value is 0x10 (16)

When treated as a double, the value is 10.000000

55.9

When treated as an integer, the value is 55

When treated as hex, the value is 0x55 (85)

When treated as a double, the value is 55.900000

.5679

When treated as a double, the value is 0.567900

a

When treated as hex, the value is 0xa (10)

0x10

When treated as an integer, the value is 0

When treated as hex, the value is 0x10 (16)

When treated as a double, the value is 16.000000

When treated as a hex with 0x%x formatting, the value is 0x10 (16)

UNIX>



# String-Strdup()

---

- ❑ Bellek ayırarak dizenin bir kopyasını oluşturur.
- ❑ Malloc()'u çağırdığı için, eğer kopyayla işiniz bittiyse, bellek sızıntılarını önlemek için sonrasında free()'i çağırmanız gerekir.

```
char *strdup(const char *s);
```

```
char *strdup(const char s)
{
    return strcpy(malloc(strlen(s)+1), s);
}
```

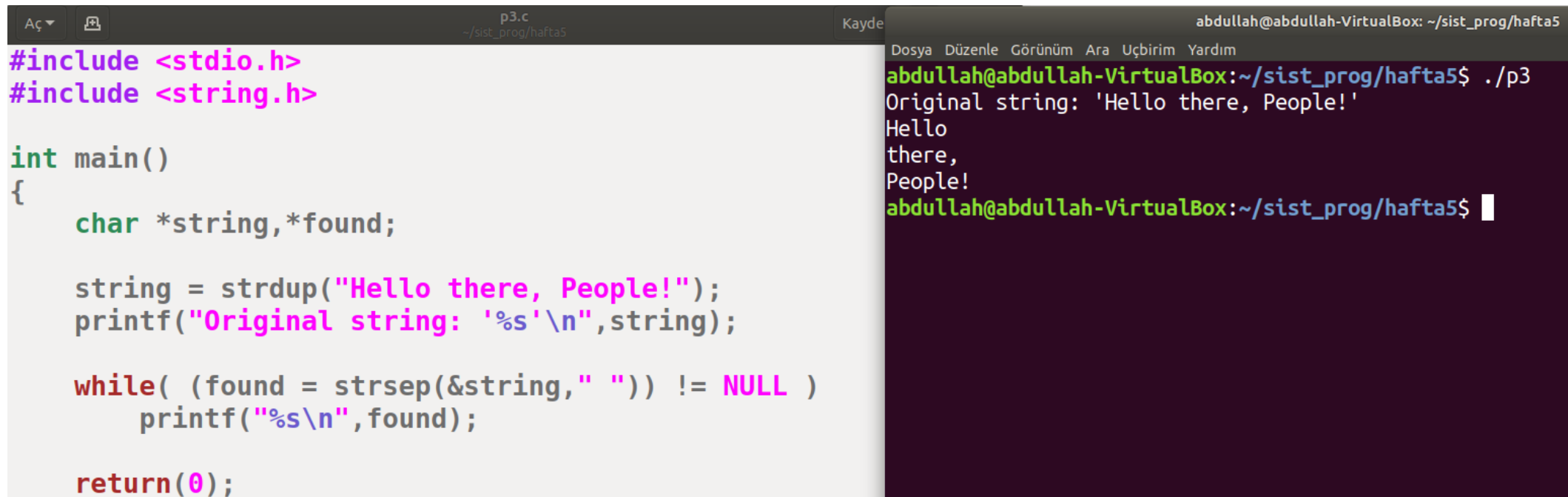
# Diğerleri

---

- **strchr()** bir karakterin son geçtiği yeri bulur
- **strstr()** bir alt-string (Substring) bulur.
- **strcasestr()** bir alt-string (Substring) bulur ancak büyük/küçük harf durumunu yok sayar.
- **strsep()**, sınırlayıcılarla dizeleri ayırmanıza yardımcı olur.
- **strncpy()** kısıtlı bir strcpy yapar.
- **memcpy()**, belleğin bir bölgesini diğerine kopyalar.
- **memcmp()**, belleğin iki bölgesinin bayt bayt karşılaştırmasını yapar.
- **bzero()**, her bayt sıfır olacak şekilde bir bellek bölgesi ayarlar.

# Strsep()

---



The image shows a C program in a text editor and its execution in a terminal. The program, named p3.c, uses the `strsep()` function to split a string. The terminal output shows the original string and the words extracted by the function.

```
p3.c
~/sist_prog/hafta5
Aç Kayde

#include <stdio.h>
#include <string.h>

int main()
{
    char *string,*found;

    string = strdup("Hello there, People!");
    printf("Original string: '%s'\n",string);

    while( (found = strsep(&string," ")) != NULL )
        printf("%s\n",found);

    return(0);
}
```

```
abduallah@abduallah-VirtualBox: ~/sist_prog/hafta5
Dosya Düzenle Görünüm Ara Uçbirim Yardım
abduallah@abduallah-VirtualBox:~/sist_prog/hafta5$ ./p3
Original string: 'Hello there, People!'
Hello
there,
People!
abduallah@abduallah-VirtualBox:~/sist_prog/hafta5$
```

# Strstr()

```
int main()
{
    // Take any two strings
    char s1[] = "Home Sweet Home";
    char s2[] = "Sweet";
    char* p;

    // Find first occurrence of s2 in s1
    p = strstr(s1, s2);

    // Prints the result
    if (p) {
        printf("String found\n");
        printf("First occurrence of string '%s' in '%s' is '%s'", s2, s1, p);
    } else
        printf("String not found\n");

    return 0;
}
```

```
abduallah@abduallah-VirtualBox: ~/sist_prog/hafta5
Dosya Düzenle Görünüm Ara Uçbirim Yardım
abduallah@abduallah-VirtualBox:~/sist_prog/hafta5$ ./p4
String found
First occurrence of string 'Sweet' in 'Home Sweet Home' is 'Sweet Home'
abduallah@abduallah-VirtualBox:~/sist_prog/hafta5$
```