APS105
Winter 2012

Jonathan Deber
jdeber -at- cs -dot- toronto -dot- edu

Lecture 19
March 5, 2012

Today

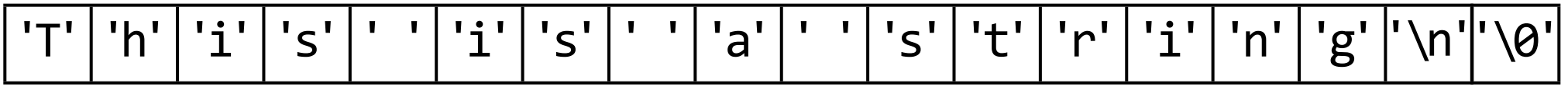
- More Strings

Strings
(recap)

```
char s[] = "This is a string\n";
```

```
char s[] = {'T','h','i','s',' ','i','s',' ','a',
            ' ','s','t','r','i','n','g','\n','\0'};
```

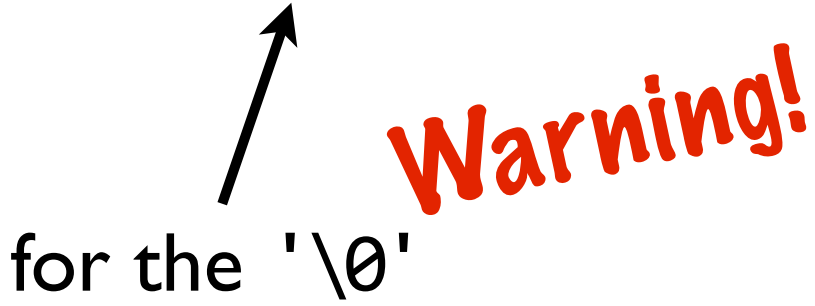
s:



C String Issues

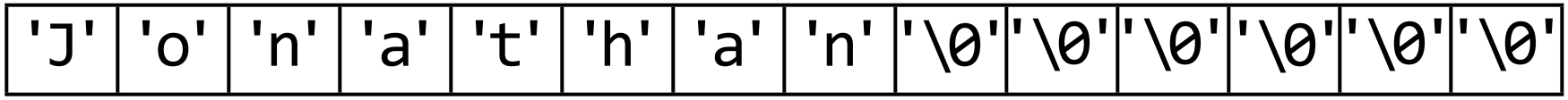
- Need to decide maximum string length in advance

```
#define STR_LEN 40
...
char s[STR_LEN + 1];
```



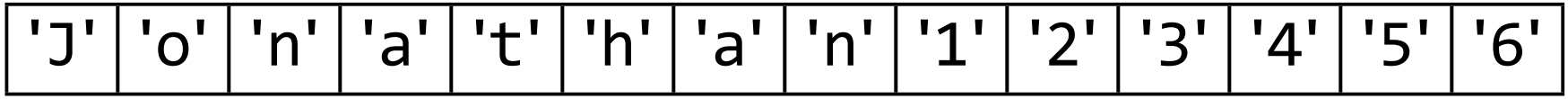
Shorter is Fine

```
#define STR_LEN 13
...
char name[STR_LEN + 1] = "Jonathan";
```




```
int a[5] = {1,2};
int a[5] = {1,2,0,0,0};
```

```
char name[STR_LEN + 1] = "Jonathan123456";
```



Legal, but not a string

C String Issues

- Strings don't know how long they are
  - Need to search for the `'\0'`
- Arrays don't know how big they are
  - Don't know if a string will fit into a given `char[]`

Strings
(not a recap)

Printing Strings

- `printf()` has a `%s` conversion specifier

```
char message1[] = "Hello everyone.\n";
printf(message1);
```

Hello everyone.

```
char message2[] = "50% Off!\n";
printf(message2);
```

Wrong

??

```
printf("%s", message2);
```

50% Off!

```
char trouble[2] = "Hi";
printf("%s", trouble);
```

??

Wrong

Reading Strings

- Can use `scanf()` with `%s` specifier
- Stops at the first whitespace character

```
char input[INPUT_LEN + 1];
scanf("%s", input);
```

Hello input: "Hello"

Hello everyone! input: "Hello"

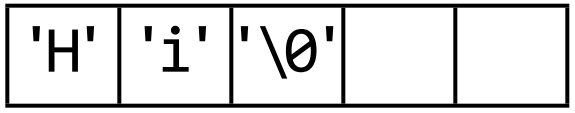
gets()

```
char *gets(char *s);
```

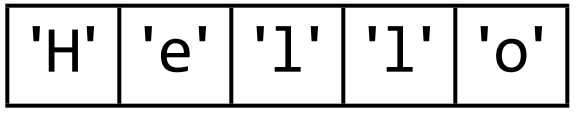
- Reads a single line, throws out `'\n'`, adds `'\0'`

```
#define INPUT_LEN 5
...

char input[INPUT_LEN + 1];
gets(input);
printf("%s\n", input);
```

Hi



Hello there



`' ' 't' 'h' 'e' 'r' 'e' '\0'`

Never Use `gets()`!

- `gets()` is inherently unsafe

You are not allowed to use `gets()` on assignments and tests

fgets()

```
char *fgets(char *s, int size, FILE *stream);
```

- Reads a single line of at most `size - 1` characters, keeps the `'\n'` (if room), adds `'\0'`
- The stream is where to read from, for us it's always `stdin`

```
char *fgets(char *s, int size, FILE *stream);


#define INPUT_LEN 5
...

char input[INPUT_LEN + 1];
gets(input);
printf("%s\n", input);
```

```
char *fgets(char *s, int size, FILE *stream);


#define INPUT_LEN 5
...

char input[INPUT_LEN + 1];
fgets(input, INPUT_LEN + 1, stdin);
printf("%s\n", input);
```

String Processing

- It's just an array...

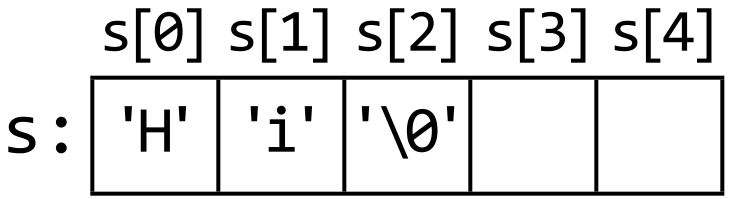
```
int countNumSpaces(const char s[], int n)
{
    int counter = 0;

    for (int i = 0; i < n; i++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```

```
int countNumSpaces(const char s[], int n)
{
    int counter = 0;

    for (int i = 0; i < n; i++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```

```
char s[MAX_LEN + 1];

fgets(s, MAX_LEN + 1, stdin);

int numSpaces = countNumSpaces(s, MAX_LEN + 1);

printf("There are %d spaces.\n", numSpaces);
```

String Processing

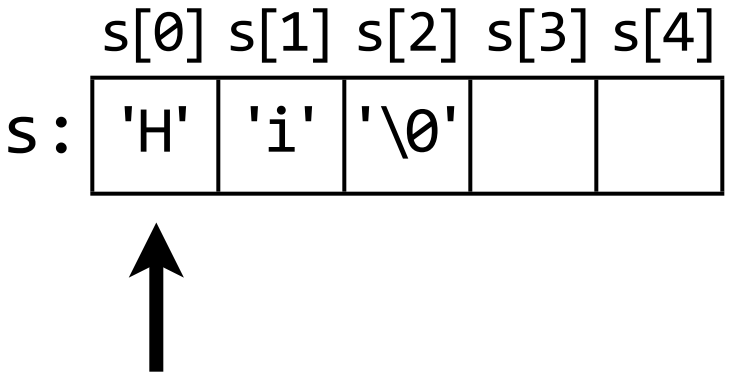
- It's just an array... but there is a difference
- We know there's a `'\0'` at the end



String Processing

- It's just an array... but there is a difference
- We know there's a `'\0'` at the end

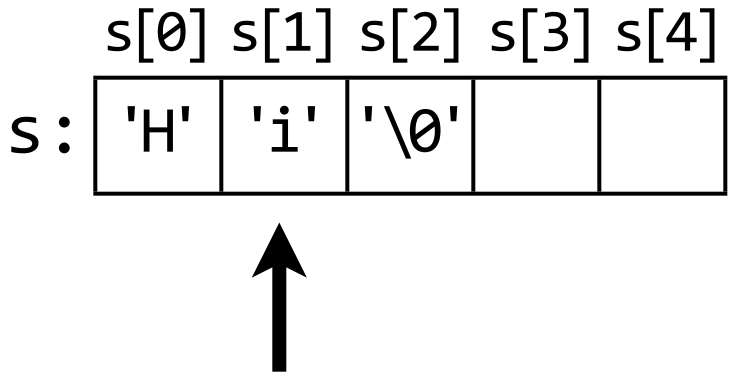
```
for (int i = 0; i < n; i++)
```



String Processing

- It's just an array... but there is a difference
- We know there's a `'\0'` at the end

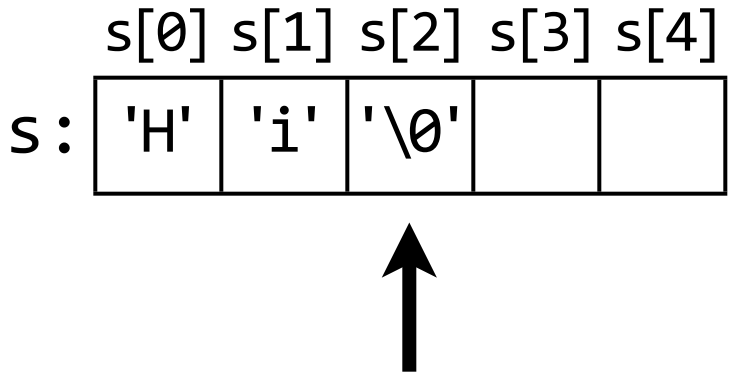
```
for (int i = 0; i < n; i++)
```



String Processing

- It's just an array... but there is a difference
- We know there's a `'\0'` at the end

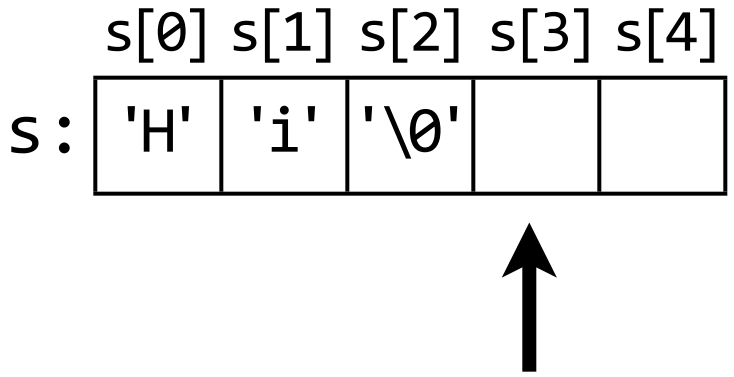
```
for (int i = 0; i < n; i++)
```



String Processing

- It's just an array... but there is a difference
- We know there's a `'\0'` at the end

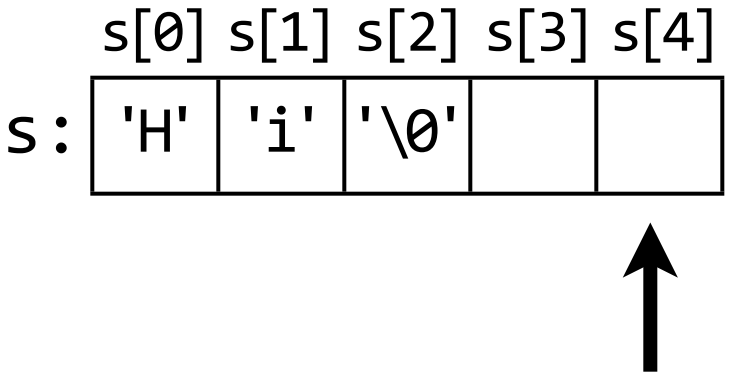
```
for (int i = 0; i < n; i++)
```



String Processing

- It's just an array... but there is a difference
- We know there's a `'\0'` at the end

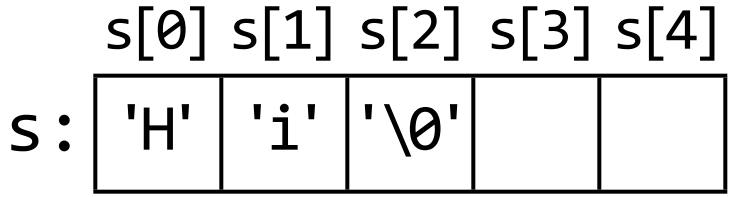
```
for (int i = 0; i < n; i++)
```



String Processing

- It's just an array... but there is a difference
- We know there's a `'\0'` at the end

```
for (int i = 0; i < n; i++)
```

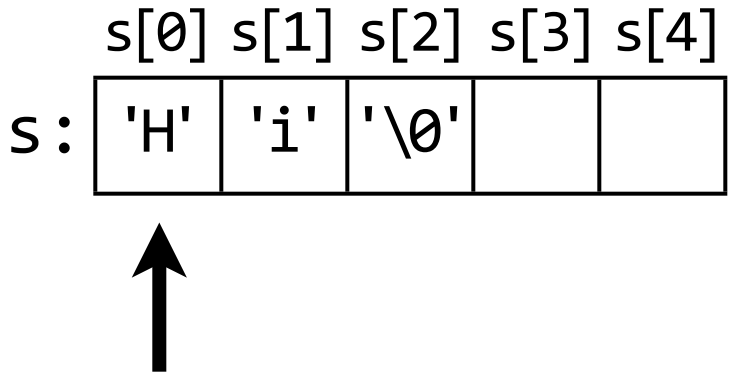



```
for (int i = 0; s[i] != '\0'; i++)
```

String Processing

- It's just an array... but there is a difference
- We know there's a `'\0'` at the end

```
for (int i = 0; i < n; i++)
```

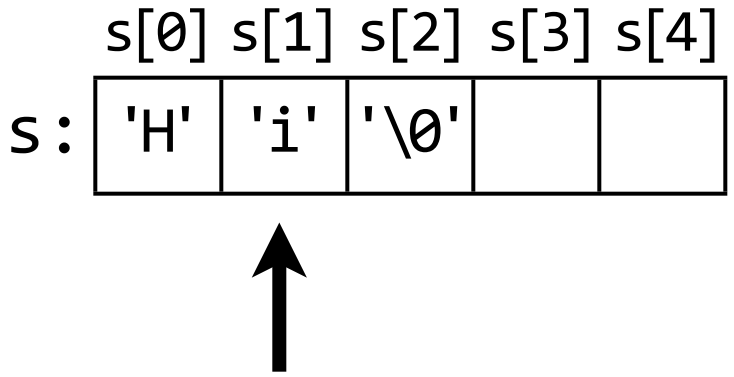



```
for (int i = 0; s[i] != '\0'; i++)
```

String Processing

- It's just an array... but there is a difference
- We know there's a `'\0'` at the end

```
for (int i = 0; i < n; i++)
```

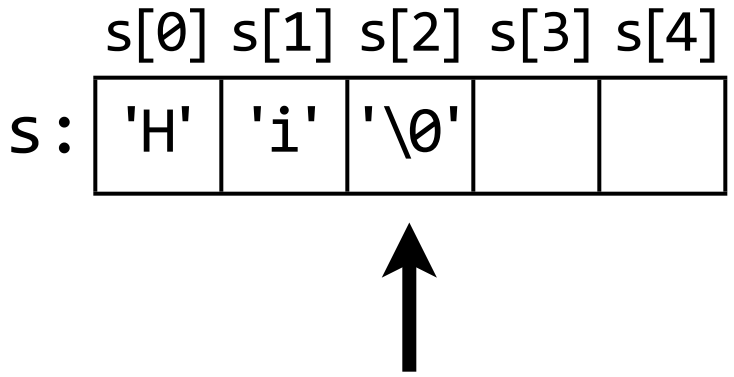



```
for (int i = 0; s[i] != '\0'; i++)
```

String Processing

- It's just an array... but there is a difference
- We know there's a `'\0'` at the end

```
for (int i = 0; i < n; i++)
```




```
for (int i = 0; s[i] != '\0'; i++)
```

```
int countNumSpaces(const char s[], int n)
{
    int counter = 0;

    for (int i = 0; i < n; i++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```

```
int countNumSpaces(const char s[], int n)
{
    int counter = 0;

    for (int i = 0; s[i] != '\0'; i++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```

```
int countNumSpaces(const char s[])
{
    int counter = 0;

    for (int i = 0; s[i] != '\0'; i++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```

```
int countNumSpaces(const char s[])
{
    int counter = 0;

    for (int i = 0; s[i] != '\0'; i++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```

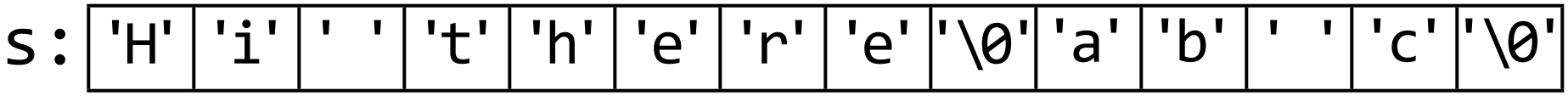
```
char s[MAX_LEN + 1];

fgets(s, MAX_LEN + 1, stdin);

int numSpaces = countNumSpaces(s);

printf("There are %d spaces.\n", numSpaces);
```

```
int countNumSpaces(const char s[], int n)
{
    int counter = 0;

    for (int i = 0; i < n; i++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}

int countNumSpaces(const char s[])
{
    int counter = 0;

    for (int i = 0; s[i] != '\0'; i++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```

Slower

Wrong

How many space characters are in this string?



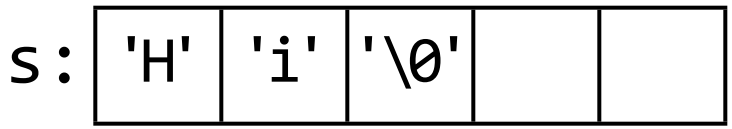
1?

2?

String Processing

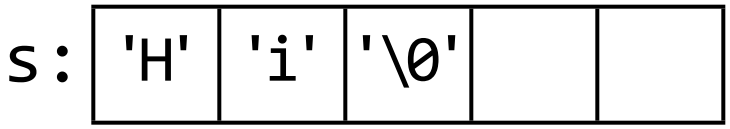
- When you're *reading* characters from a string, you don't need the size of the array
- Instead, you care about the location of the `'\0'`

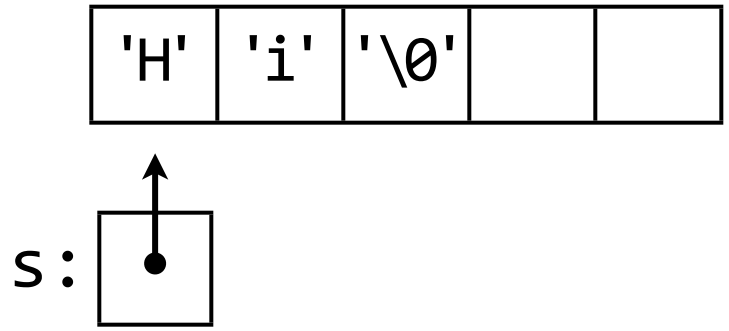
- When you're *writing* characters to a string, you do need the size of the array



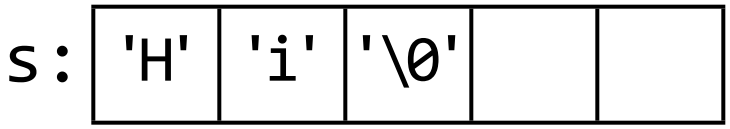
Pointer Processing

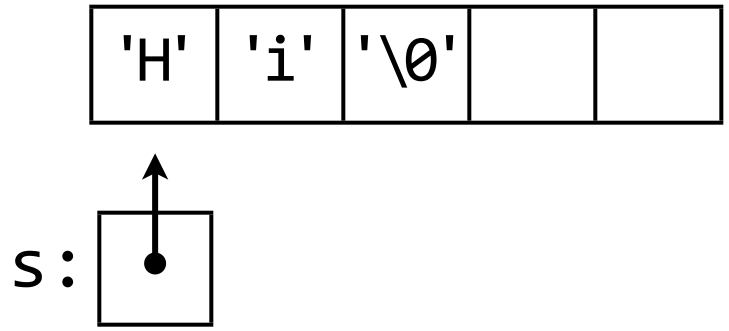
```
int countNumSpaces(const char s[])
{
    int counter = 0;

    for (int i = 0; s[i] != '\0'; i++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```



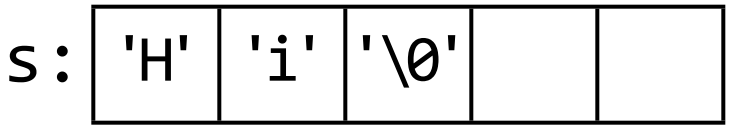


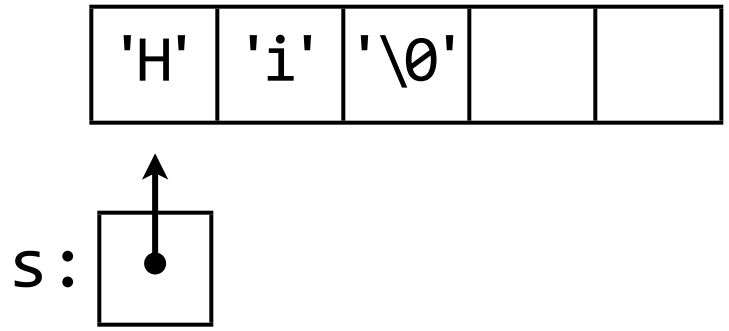

```
int countNumSpaces(const char *s)
{
    int counter = 0;

    for (int i = 0; s[i] != '\0'; i++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```



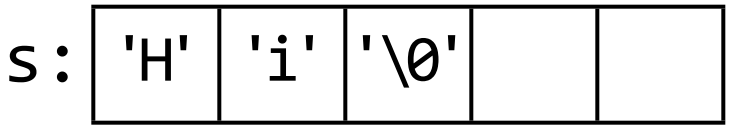


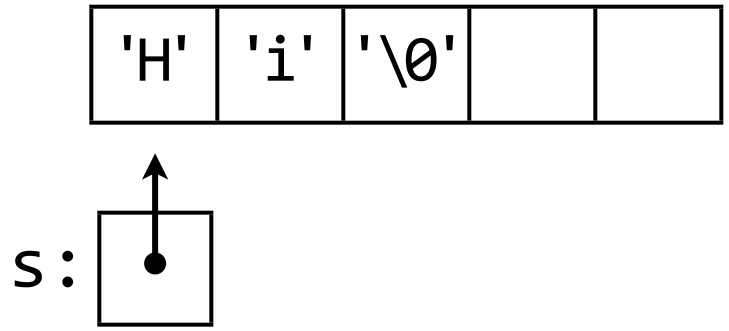

```
int countNumSpaces(const char *s)
{
    int counter = 0;

    for (int i = 0; *s != '\0'; i++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```



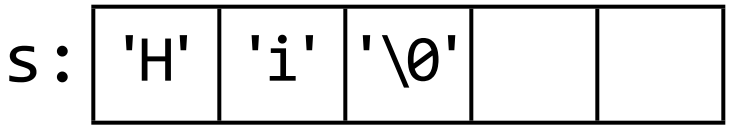


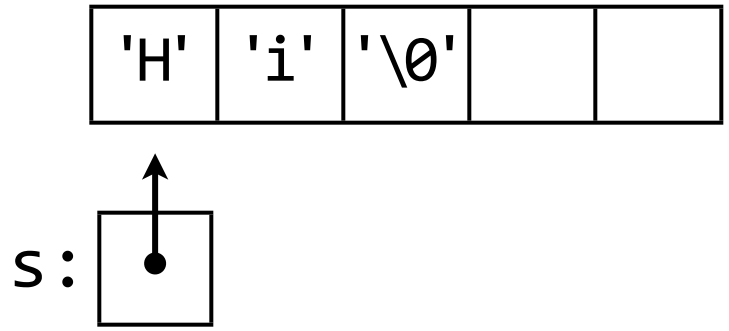

```
int countNumSpaces(const char *s)
{
    int counter = 0;

    for (int i = 0; *s != '\0'; s++)
    {
        if (s[i] == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```






```
int countNumSpaces(const char *s)
{
    int counter = 0;

    for (int i = 0; *s != '\0'; s++)
    {
        if (*s == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```






```
int countNumSpaces(const char *s)
{
    int counter = 0;

    for ( ; *s != '\0'; s++)
    {
        if (*s == ' ')
        {
            counter++;
        }
    }

    return counter;
}
```

`string.h`

- The C string library
- `#include <string.h>` (no `-l` needed)
- Provides a bunch of useful string operations
- Works around some C limitations

```
char str1[MAX_LEN + 1];
char str2[MAX_LEN + 1];

str1 = "Hello";
```
Error

```
str1 = {'H','e','l','l','o','\0'};
```
Error

```
int a[4];
a = {2, 4, 6, 8};
```
Error

```
char str3[MAX_LEN + 1] = {'H','i', '\0'};


str1 = str3;
```
Error

Copying Arrays

- This will not work:

```
int a[4] = {1, 2, 3, 4};
int b[4];

b = a;
```

Error

```
int i = 8;
int j;

j = i;
```

Copying Arrays

```
int a[4] = {1, 2, 3, 4};
int b[4];

for (int i = 0; i < 4; i++)
{
    b[i] = a[i];
}
```

Why Can't You Copy Arrays?

- Remember an array name by itself is short for "the address of the first element of the array"

```
int a[] = {1, 2, 3};
int *p = a;      <⟷>   int *p = &a[0];
```

```
str1 = str2;     <⟷>   &str1[0] = &str2[0];
```

Error