APS105 Lecture 24

Last lecture: Input/Output Strings

Today: Recap on getStringSafely function and introduce the string library #include <string.h>

-DEMO-

```
char * getStringSafely (char *s, int size){
    int i=0;
    char c;
    while (i<size-1 && (c=getchar())!='\n'){
        s[i]=c;
        i=i+1;
    }
    s[i]='\0';
    return s;
}
```

not size-1 since user can enter '\n' before they use up size-1 characters

s is of type (char *)

```
int main() {
    char s[6];
    printf("user entered %s", getStringSafely(s, 6));
    return 0;
}
```

is returning char *

As you've seen so far, there is no data type called string in C. But thankfully there is a standard library for strings in C, which has handy functions.

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

String functions typically start with str.

For example

I - Function to find length

Prototype: int size_t strlen(const char *);

size_t: another term for unsigned int, but since we don't deal with unsigned int, think of it as int

const: Character array can't be modified inside the function. So contents of array are fixed; if altered we get compile-time error

```
char s[6];
printf("%d", strlen(s));
```

This will print 0, since s is not initialized and it happens that it was initialized with 10 in 1st element.

```
char s[6] = "hi";
printf("%d", strlen(s));
```

This will print 2, not counting '10' character.

Let's try implementing strlen,

```
int stringlength(const char * s){
    int count = 0;
    while(*(s+i) != '10'){
        count++;
    }
    return count;
}
```

s[i] != '10' or this is more readable

*(s+i) != '10' → can also be written as s[i]

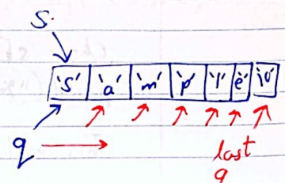
if s[i] is null → zero → false

if s[i] is not null → not zero → true

Another way →

Instead of incrementing a count, we increment a pointer to the array.

```
int stringLength (const char *s) {
    const char *q = s;
    while (*q != '\0') {
        q++;
    }
    return q - s;
}
```



If s has address 2 then last q has address 8.
q - s = 8 - 2 = 6 # of chars without null

Note: const char *s means contents of array can't be changed, but the pointer variable s can hold a different address.

II function that copies a string from one value to another.

```
char *strcpy (char *dest, const char *src);
```

But why use it? Can't I just do this

```
char s[6] = "Hello";
char d[6] = s;
```

→ to initialize array, you need to do it "Hello" this way

dis not assignable. CANNOT change it.

```
d = s;
```

→ remember d is an address not pointer variable.

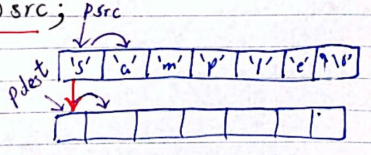
Here, strcpy comes into use

```
char * stringCpy(char *dest, const char * src){
    int i=0;
    while ( src[i] != '\0' ){
        dest[i] = src[i];
        i++;
    }
    dest[i] = '\0';
    return dest;
}
```

very Important!

If we use pointers

```
char * stringCpy ( char *pdest, const char * psrc){
    const char * t = pdest;
    while ( *psrc != '\0'){
        *pdest = *psrc;
        pdest++;
        psrc++;
    }
    *pdest = '\0';
    return t;
}
```



Is this safe? NO
What happens if pdest doesn't have enough space?