CS 214 Recitation(Sec. 6)

Zuohui Fu
Ph.D. Department of Computer Science
Office hour: Mon 2pm-3pm
Email: zf87 AT cs dot rutgers dot edu
10/12/2017

Topics

- HW3 solution
- Review for Midterm

HW3

- 0. What are the differences between strlen and sizeof a string in C? Why?
- 1. Write the function: replace(char string[], char from[], char to[])

  which finds the string from in the string string and replaces it with the string to. You may assume that from and to are the same length. For example, the code:

```
char string[] = "recieve";
replace(string, "ie", "ei");
```

should change string to "receive".

HW3-Cont.

- 2. Write a short program to read two lines of text, and concatenate them using strcat. Since strcat concatenates in-place, you'll have to make sure you have enough memory to hold the concatenated copy. For now, use a char array which is twice as big as either of the arrays you use for reading the two lines. Use strcpy to copy the first string to the destination array, and strcat to append the second one.

Solution to HW3 Q1

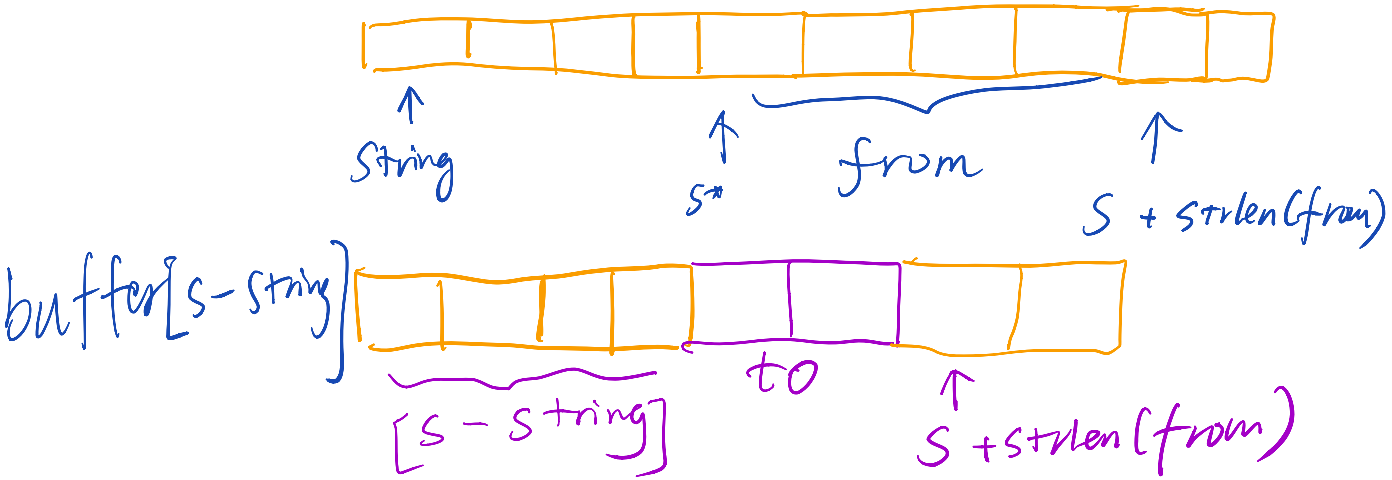
- strncpy()
- strstr()
- sprintf()

Solution to HW3 Q1

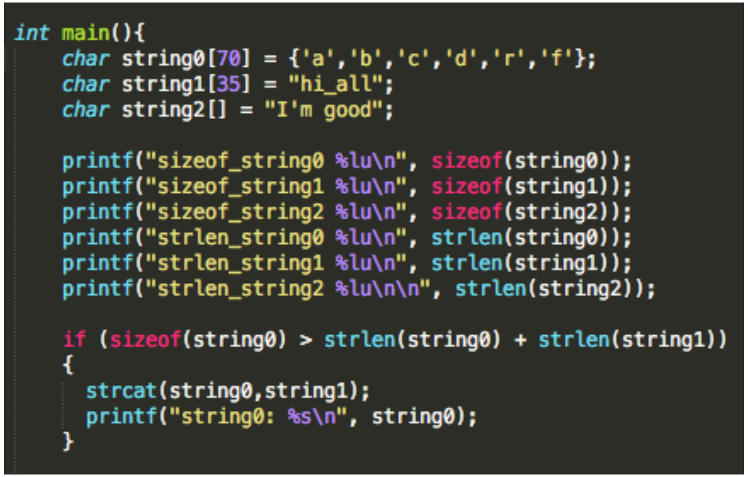
can't do strcpy("abc", "abc");

- if I do replace(string1,"ei","ei");

  why segmentation fault? by s = strstr(string, from)



Solution to HW3 Q2



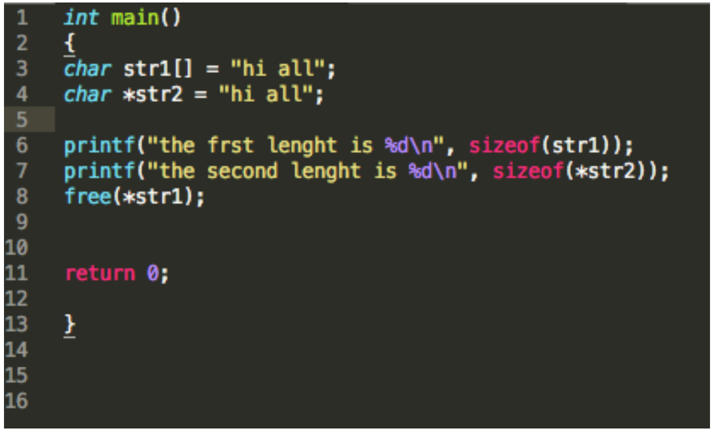
```c
int main(){
    char string0[70] = {'a','b','c','d','r','f'};
    char string1[35] = "hi_all";
    char string2[] = "I'm good";

    printf("sizeof_string0 %lu\n", sizeof(string0));
    printf("sizeof_string1 %lu\n", sizeof(string1));
    printf("sizeof_string2 %lu\n", sizeof(string2));
    printf("strlen_string0 %lu\n", strlen(string0));
    printf("strlen_string1 %lu\n", strlen(string1));
    printf("strlen_string2 %lu\n\n", strlen(string2));

    if (sizeof(string0) > strlen(string0) + strlen(string1))
    {
      strcat(string0,string1);
      printf("string0: %s\n", string0);
    }
```

```
sizeof_string0 70
sizeof_string1 35
sizeof_string2 9
strlen_string0 6
strlen_string1 6
strlen_string2 8

string0: abcdrfhi_all
sizeof_string0 70
strlen_string0 12
```

Should know how strlen and sizeof work.

sizeof

- What does sizeof return for one of them?



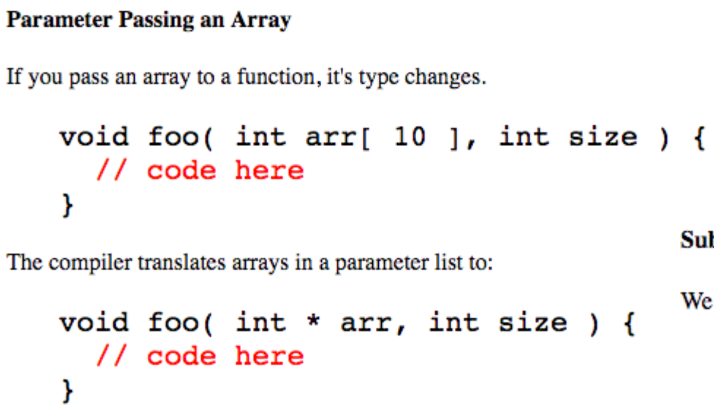
```
int main()
{
char str1[] = "hi all";
char *str2 = "hi all";

printf("the frst lenght is %d\n", sizeof(str1));
printf("the second lenght is %d\n", sizeof(*str2));
free(*str1);


return 0;

}
```

```
sizeof( int ) ==> 4
sizeof( short ) ==> 4
```

Think about it

- What is malloc? How is it different than calloc. Once memory is malloced how can I use realloc?
- What is the & operator? How about *?

Pointer



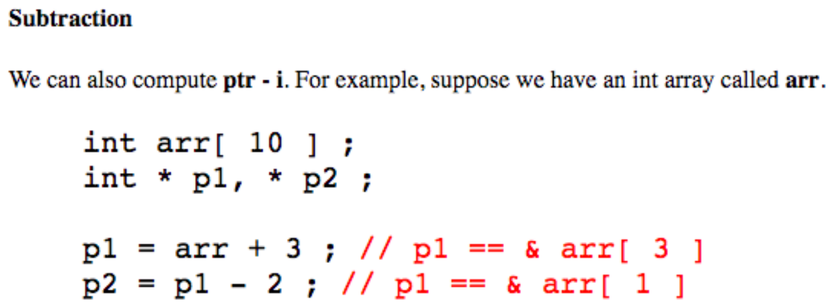
Parameter Passing an Array

If you pass an array to a function, it's type changes.

```
void foo( int arr[ 10 ], int size ) {
  // code here
}
```

The compiler translates arrays in a parameter list to:

```
void foo( int * arr, int size ) {
  // code here
}
```

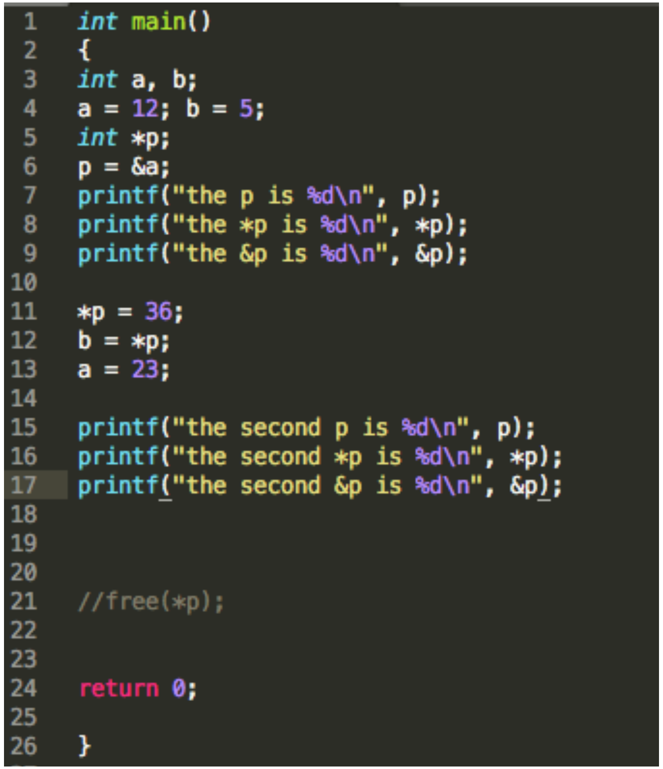


Subtraction

We can also compute **ptr - i**. For example, suppose we have an int array called **arr**.

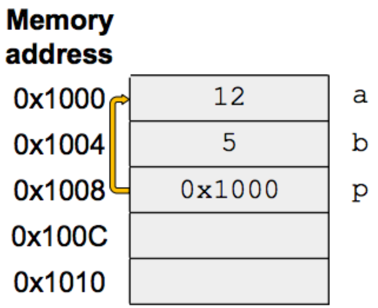
```
int arr[ 10 ] ;
int * p1, * p2 ;

p1 = arr + 3 ; // p1 == & arr[ 3 ]
p2 = p1 - 2 ; // p1 == & arr[ 1 ]
```

Pointer

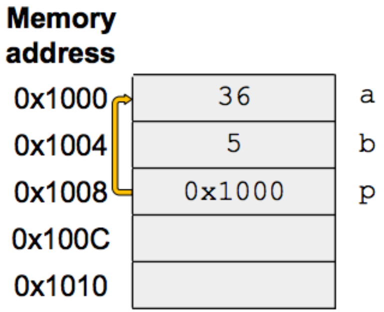


```c
int main()
{
int a, b;
a = 12; b = 5;
int *p;
p = &a;
printf("the p is %d\n", p);
printf("the *p is %d\n", *p);
printf("the &p is %d\n", &p);

*p = 36;
b = *p;
a = 23;

printf("the second p is %d\n", p);
printf("the second *p is %d\n", *p);
printf("the second &p is %d\n", &p);



//free(*p);


return 0;

}
```

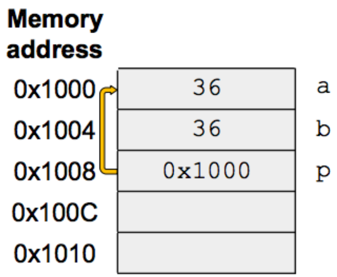
Pointer



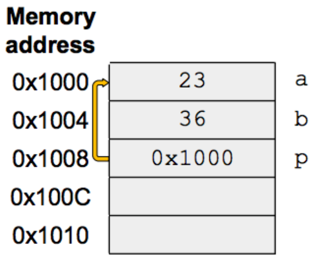
```
int a, b;
a = 12; b = 5;

int *p;
p = &a;
```



```
int a, b;
a = 12; b = 5;

int *p;
p = &a;
*p = 36;
```



```
int a, b;
a = 12; b = 5;

int *p;
p = &a;
*p = 36;
b = *p;
```



```
int a, b;
a = 12; b = 5;

int *p;
p = &a;
*p = 36;
b = *p;
a = 23;
```

Position of Type Qualifiers

- const int i;

  /* means i is a const int */
  /* i cannot be modified */
  /* a value can be assigned to i by using an initializer */
- const int *p1;

  /* means p1 is a pointer to a const int */
  /* p1 can be modified, but the int pointed to by p1 cannot be modified */
- int *const p2;

  /* means p2 is a const pointer to an int */
  /* p2 cannot be modified, but the int pointed to by p2 can be modified */
- const int *const p3;

  /* means p3 is a const pointer to a const int */
  /* neither p3 nor the int pointed to by p3 can be modified */

Topics you should review

- C Strings representation
- C Strings as pointers
- char p[]vs char* p
- Simple C string functions (strcmp, strcat, strcpy)
- sizeof char
- sizeof x vs x*
- Heap memory lifetime
- Calls to heap allocation
- Deferencing pointers
- Address-of operator
- Pointer arithmetic
- String duplication
- String truncation
- double-free error
- String literals
- Print formatting.
- memory out of bounds errors
- static memory
- fileio POSIX vs. C library
- C io fprintf and printf
- POSIX file IO (read, write, open)
- Buffering of stdout
- Debug

Good Luck to your Mid-Exam~