October 10, 2017

October 10, 2017 11:27 AM

Run-time stack

- small than heap
- local variables & return address

at the re turn, variables are gone

---

array in rts

```
int array[4];

sizeof(array) //== sizeof(int)*4

foo(array);
```

```
void foo(int array[]) //
    sizeof(array) //!= sizeof(int)*4
```

sizeof(array) // != sizeof(int)*4

void foo(int * array)

```
int array[4];
int size = sizeof(array) / sizeof(int);

foo(array, size);
```

```
int main(void) {
    int array[1024*1024*1024];
```

---

malloc() allocates memory on the heap.

free()

void *

```
int * buffer = (int*) malloc (1000000);
. . .
free (buffer);
```

strdup ()

$0

unit - '0'

'0' - '0' => 0

'1' - '0' => 1

---

malloc()

[? ?]

```
char * str = (char*) malloc(100);
memset (str, 0 , 100);
```

0 ↑ value to set

100 ↑ number of bytes

```
char*str = (char*) calloc(100);
```

```
char * str = NULL;
str = (char *) realloc (str, 100);
```

```
char * str=(char*)malloc (100000);
...
```

```
str = realloc (str, 100000);
```
str becomes NULL!

```
char * tmp = realloc (str, ...)
if(tmp != NULL)
```

str = tmp

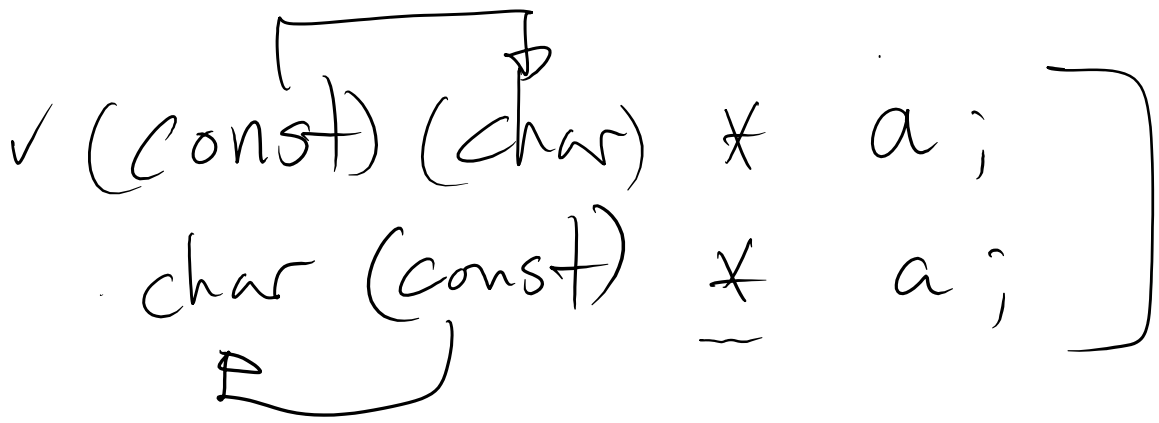
```
char * ptr = NULL;
```

= 0

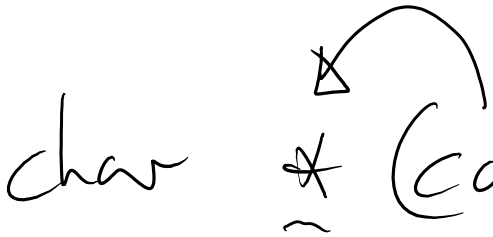
```
if ( ptr )
  // if ( NULL != ptr )

if ( !ptr )
  // if ( NULL == ptr )
```

'const'



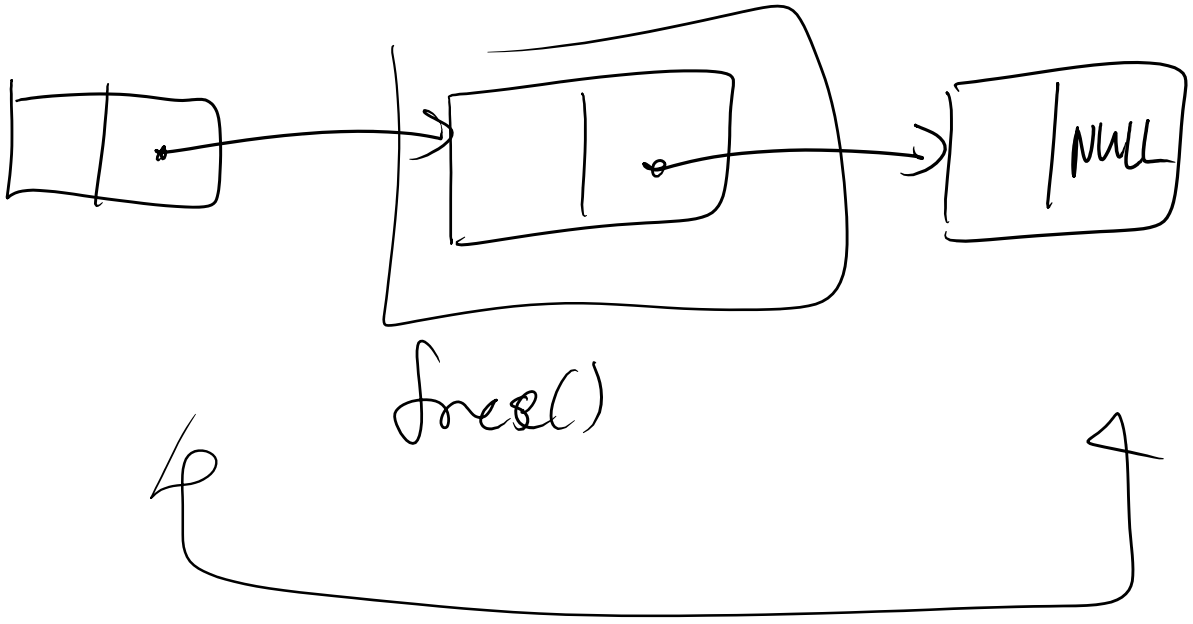

```
a = &data; // works
*a = '0'; // not work
```




```
a = &data; // this is not
             work
*a = '0'; // this works
```

✓ char const * const a;

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




```
struct node * ptr =
    (struct node *) malloc(
        sizeof(struct node));
```

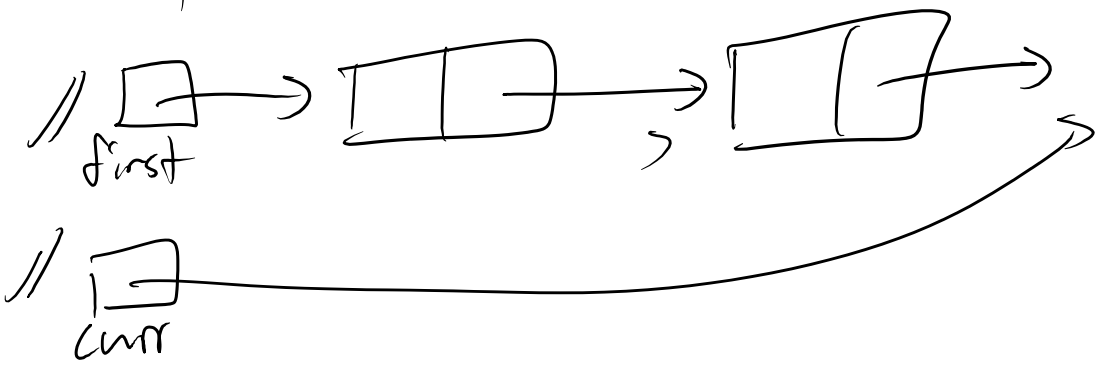
```
struct node * first = NULL;

void add(int data) {
    struct node * newNode;
    newNode = (struct node *) malloc(
        sizeof(struct node) );

    if(newNode) {
        newNode -> data = data;
        (*newNode).next = first;

        first = newNode;
    }
}
```

```
void printAll() {
    struct node * curr = first;
    while (NULL != curr) {
        printf(...);
        curr = curr->next;
    }
}
```




```
struct node* search(int valuelookingfor) {
    struct node * curr = first;
    while( curr && curr->data
        != valuelookingfor) {
        curr = curr->next;
```

```
    }
    return curr;
}
```

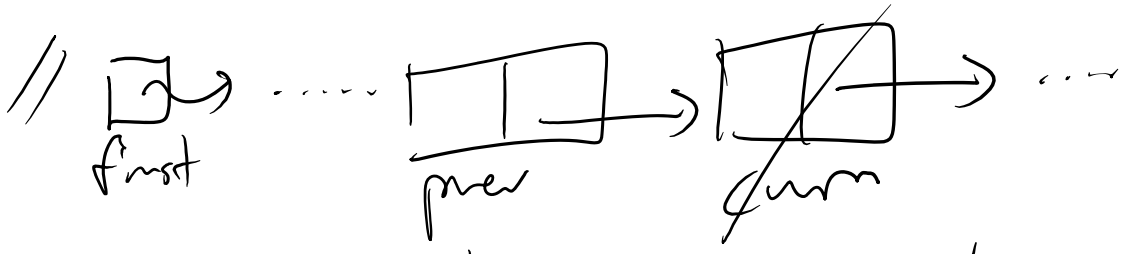
```
void remove ( int value ) {
    struct node * prev = NULL;
    struct node * curr = first;

    while (curr && curr->data != value)
    {
        prev = curr;
        curr = curr->next;
    }

    if(NULL == curr) {
    }
    else if( NULL == prev) {
        // the data was found at
```

```
// the data was found at
// the head
```


```
first = curr->next;
free(curr);
}
else {
```




```
prev->next = curr->next;
free(curr);
}
}
```

```
void clear() {
    struct node* curr = first;
    struct node* prev = NULL;
    while(NULL != curr) {
        prev = curr;
        curr = curr->next;
        free(prev);
    }
    first = NULL;
}
```