lec7-malloc-string

April 9, 2018

1 Lecture 7: Malloc, String

Zewei Chu 4/9/2018

1.0.1 malloc

(dynamically) allocate some new memories in the heap

- Stack and Heap

The following program won't work, because memory allocated on stack will be deallocated after the function returns.

```
In [ ]: double* alloc_init_final_grades(int num_students){
            double scores[num_students];
            for (int i = 0; i < num_students; i++)
                scores[i] = 2.0;
            return scores;
        }
```

- Do not return a pointer to a local variable - the memory will have been deallocated by then.
- Same reason, do not return an array.
- Use malloc instead to allocate memory on the heap. malloc allocates memory in bytes and returns a pointer pointing to the beginning of the space. If memory allocation failed, a NULL pointer is returned.

```
In [1]: #include<stdio.h>
        #include<stdlib.h>

        double* alloc_init_final_grades(int num_students){
            double* scores = (double*) malloc(sizeof(double) * num_students);
            for (int i = 0; i < num_students; i++)
                scores[i] = 2.0;
            return scores;
        }

        int main(){
            double * scores = alloc_init_final_grades(10);
```

```
    scores[2] = 95;
    scores[5] = 75;
    *(scores+1) = 88.5;
    for (int i = 0; i < 10; i++)
        printf("%.2lf ", *(scores+i));
    printf("\n");
    free(scores);
    return 0;
}
```

```
2.00 88.50 95.00 2.00 2.00 75.00 2.00 2.00 2.00 2.00
```

1.0.2 Dynamically allocated 2-D array

In [37]:
```
#include<stdio.h>

void print_2d_values(float arr[][2], int height, int width){
    for (int i = 0; i < height; i++){
        for (int j = 0; j < width; j++)
            printf("%f ", arr[i][j]);
        printf("\n");
    }
}

int main(){
    float arr[2][2] = {{2.5, 6.7}, {1.2, 2.2}};
    print_2d_values(arr, 2, 2);
    printf("%f\n", arr[0][1]);
    return 0;
}
```

```
2.500000 6.700000
1.200000 2.200000
6.700000
```

In [32]:
```
#include<stdio.h>
#include<stdlib.h>
double** create_and_init_2d(int height, int width){
    double** array = (double**)malloc(height * sizeof(double*));
    int i, j;
    for (i = 0; i < height; ++i){
        array[i] = (double*)malloc(width*sizeof(double));
        for (j = 0; j < width; ++j)
            array[i][j] = 0.0;
    }
    return array;
}
```

```
        void free_2d(double** array, int height){
            for (int i = 0; i < height; ++i){
                free(array[i]);
            }
            free(array);
        }

        void print_2d_array(double** array, int height, int width){
            for (int i = 0; i < height; i++){
                for (int j = 0; j < width; ++j)
                    printf("%.2lf ", array[i][j]);
                printf("\n");
            }
        }

        int main(){
            double** array = create_and_init_2d(3,5);
            array[2][4] = 2.5;
            array[1][3] = -5.4;
            print_2d_array(array, 3, 5);
            free_2d(array, 3);
            return 0;
        }
```

```
0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 -5.40 0.00
0.00 0.00 0.00 0.00 2.50
```

1.0.3 String

- Array of characters ending in '\0' (a null character).

In [2]:
```
#include <stdio.h>
int main(){
    printf("%s\n", "abcde");
}
```

```
abcde
```

- Need to #include<string.h> for string library functions
- strcpy function to copy a string

In [3]:
```
#include <stdio.h>
#include <string.h>

int main(){
```

```
    char myString[20];
    strcpy(myString, "hello");
    printf("%s\n", myString);
    printf("%lu\n", strlen(myString));
    for (int i = 0; i < 5; ++i)
        putchar(myString[i]);
    return 0;
}
```

```
hello
5
hello
```

In [4]:
```
#include<stdio.h>
#include<string.h>

void strcpy2(char dest_str[], char source_str[]){
    int i = 0;
    while (source_str[i] != '\0'){
        dest_str[i] = source_str[i];
        i ++;
    }
    dest_str[i] = '\0';
}

int main(){
    char myString[20];
    strcpy2(myString, "hello");
    printf("%s\n", myString);
    printf("%lu\n", strlen(myString));
    for (int i = 0; i < 5; ++i)
        putchar(myString[i]);
    return 0;
}
```

```
hello
5
hello
```

- strlen to count string length

In [10]:
```
#include<stdio.h>
#include<string.h>

int strlen2(char str[]){
    int length = 0;
    while (str[length] != '\0')
        length ++;
    return length;
```

```
}

int main(){
    printf("%lu\n", strlen("hello"));
    printf("%d\n", strlen2("hello"));
    return 0;
}
```

```
5
5
```

1.0.4 command line arguments

- argc: number of arguments
- argv: arguments - an array of strings

In [41]:
```
#include<stdio.h>

int main(int argc, char* argv[]){
    printf("%d\n", argc);
    for (int i = 0; i < argc; ++i){
        printf("%s ", argv[i]);
    }
    printf("\n");
}
```

```
1
/var/folders/87/k3tmbndn0b77_wxs0bbd_n4h0000gq/T/tmpw64_cfqf.out
```