

Brief Revision on C Programming

...

OS Lab 1

Spring 2024
Feb 06

Welcome

Instructor:

Mohamed El Halaby m.alhalaby@aucegypt.edu

TAs:

Lab 1 8:30am to 11:20am	Sara Gad sara.gad@aucegypt.edu
Lab 2 11:30am to 2:20pm	Aya Hamdy aya_hamdy@aucegypt.edu

Group Work

- Pick your team, no later than next week. Unassigned students must inform the TAs by email.
- Once you have chosen your team, it won't be possible to change it.
- Teams can have at most 3 members, registered in the same lab.
- It's crucial that you attend the lab you are registered in.
- The workload of every task must be divided equally between team members.
- After submission, every team will present their work and grades will be given accordingly.

Conduct

1. AUC's attendance and academic integrity policies will be applied, so please make sure you read them carefully.
2. There will be 20% deduction for late submissions up to 48 hours beyond the deadline. After 48 hours the submission will not be accepted.

So please make sure you submit your work before the deadline by a good margin in case any problems or technical issues arise.

3. Arriving more than 10 minutes late for a session will contribute negatively to your attendance record and can potentially put you at risk of missing important remarks that are vital to completing the lab task.
4. Participation during lab sessions is highly encouraged.



```
sum = 0
for i in range(1,100000001):
    sum += i
print('Sum = ', sum)
```

Execution time:
22.4 sec.



```
int main()
{
    unsigned long long sum = 0;
    for(int i = 1; i<=100000000; i++)
    {
        sum += i;
    }
    = " << sum << "\n";
}
```

Execution time:
0.02 sec.

C/C++ is FAST !!!

C++ is **1120**
times faster

C is a Compiled Language

Compiler: Software program that converts high-level language source code into machine-specific code.

```
int main()
{
    unsigned long long sum = 0;
    for(int i = 1; i<=100000000; i++)
        sum += i;

    printf("Sum = %d", sum);
}
```

hello.c



Compiler

```
0101110101
0101010101
0100101010
1010101000
1111010101
0101010101
0101010110
```

Machine code

Command Line Arguments

- `argc` is short for “argument count”.
- Arrays don’t know their length, so `argc` will be set to the number of arguments.
- Strings are defined as `char*` (pointer to `char`), and the parameter.
- `argv` is short for “argument vector” and holds the command line arguments.

(Demo)

```
/*
 * File: hello.c
 */
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("Hello world!\n");
}
```

Arrays

```
int numbers[3];  
char letters[5];
```

```
int numbers[3] = { 1, 2, 3 };  
char letters[5] = { 'a', 'b', 'c', 'd', 'z' };
```


Strings

```
char name[32];
```

```
char name[] = "Ahmed";
```

Structured Types

```
struct point {  
    int x; int y;  
};  
  
struct point origin;  
  
origin.x = 0;  
  
origin.y = 0;
```

(Demo)

https://www.w3schools.com/c/c_structs.php

Composition:

```
struct line {  
    struct point start;  
    struct point end;  
};  
  
struct line line1;  
  
line1.start.x = 1;  
line1.start.y = 2;  
line1.end.x = 3;  
line1.end.y = 4;
```

Pointers

You can retrieve the address of the location in memory where the variable is stored, this is called a “pointer”.

If the variable is, say, an `int`, then the type of its address is “pointer to `int`”, written `int*`.

The `&` (ampersand) operator, also called the “address-of” operator, gets the address of a variable.

```
int i = 123;
```

```
int* ip = &i;
```

(Demo)

Preprocessing

- Any preprocessor command starts with pound sign (#) which forms the beginning of the preprocessor command.
- The carriage return forms the end of any preprocessor command.

(Demo)

```
#include <stdio.h>

#define message "Hello, World!\n"

int main() {

    printf(message);

    return 0;

}
```

Dynamic Memory Allocation

The C library function

```
(void *) malloc(size_t size)
```

allocates the requested memory on the heap and returns a `void` pointer to the first allocated byte, if the allocation is successful, otherwise it returns a `NULL` pointer.

- `size_t` is an unsigned integer and represents the number of bytes.

- Normally, you have to typecast the returned `void` pointer to the type that you want.
- Dynamically allocated memory created with `malloc()` doesn't get freed on their own. You must explicitly use `free()` to release the space.
- It's a good practice to set the pointer value to `NULL` once it is passed to `free` to avoid accidentally triggered undefined behaviour.

(Demo)

Task

(individual)

Write a C program that represents a book (book_title, author name & book_id) as a struct.

1. `write_data`: takes a list of book structs and a file name, then writes the structs to a file with the name given [1 point].
2. `load_data`: takes a file name on which book structs are stored and loads it in a list data structure in memory, then returns the list data structure [1 point].
3. `search_by_id` takes a list of book structs and a book_id, then returns the struct object with the given id if it exists [1 point].

In main() [2 point]:

1. Prompt the user to choose whether to:
 - a. Load data.
 - b. Enter data.
 - c. Search by id.
 - d. Exit the program
2. If a, b or c are chosen, then prompt the user to enter the name of the file.
3. If a is chosen, then use `load_data` and print the list in table format on the screen.
4. If b is chosen, then allow the user to **keep entering** book structs data till a reserved character is entered. Next, use `write_data` to write the data to a file with the name given. Note: You are NOT allowed to assume an upper limit on the number of struct objects the user will enter.
5. If d is chosen, then exit the program.

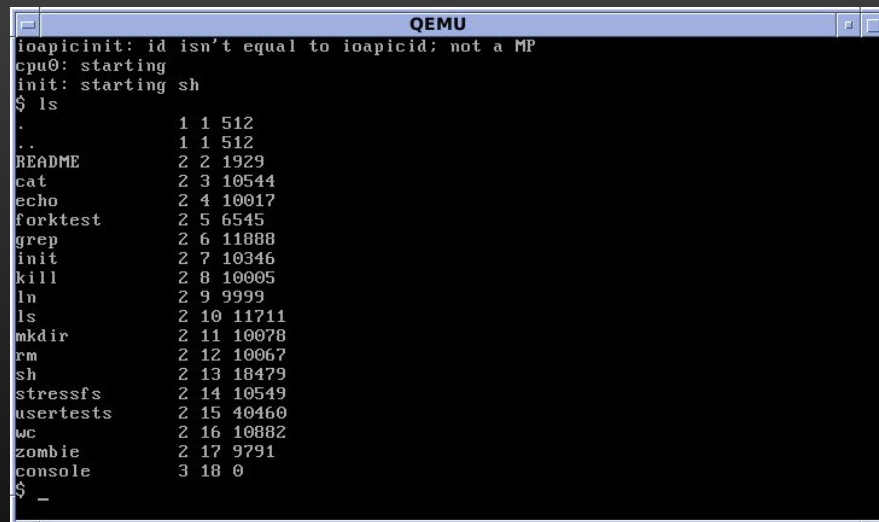
Deadline: Monday, Feb 12 11:59pm

Task

(individual)

Create a Linux VM and install XV6 (x86 version).

(Demo)



```
QEMU
ioapicinit: id isn't equal to ioapicid; not a MP
cpu0: starting
init: starting sh
$ ls
.          1 1 512
..         1 1 512
README    2 2 1929
cat        2 3 10544
echo       2 4 10017
forktest   2 5 6545
grep       2 6 11888
init       2 7 10346
kill       2 8 10005
ln          2 9 9999
ls          2 10 11711
mkdir      2 11 10078
rm          2 12 10067
sh          2 13 18479
stressfs   2 14 10549
usertests   2 15 40460
wc          2 16 10882
zombie     2 17 9791
console    3 18 0
$ -
```