暨南大學
JINAN UNIVERSITY

Undergraduate Lab Report

Course Title: Experiment of Computer Organization

Course No: 60080014

Student Name: _____

Student No: _____

School: International School

Department: _____

Major: Computer Science & Technology

Instructor: SUN Heng

Academic Year:202~202,Semester : 1st [√] 2nd[ ]

Academic Affairs Office of Jinan University

Date (dd/mm/yyyy)_____

Computer Organization Lab List

Student Name:_____ Student No:_____

ID	Lab Name	Type
1	Number Storage Lab	Individual
2	Manipulating Bits	Individual
3	Simulating Y86-64 Program	Individual
4	Performance Lab	Team
5	A Simple Real-life Control System	Team
6	System I/O	Individual

Undergraduate Lab Report of Jinan University

Course Title Experiment of Computer Organization Evaluation

Lab Name Number Storage Lab Instructor SUN Heng

Lab Address

Student Name Student No

College International School

Department Major CST

Date / / Afternoon

1. Introduction

This lab extends the section *Information Storage* (chapter 2) in the textbook. It will immerse you in a problem context, which will require you to apply the theoretical concepts taught in the C language course. This will help you to develop basic skills needed to become a device level system developers and digital system designer. While going through this lab concentrate on developing a good understanding of the problem context, as it is a fundamental requirement for becoming a skilled system designer or application developer.

2. Lab Instructions or Steps

(1) Using less than 4 sentences, explain what function/action, the two statements marked with “Tag 1” and “Tag 2” perform.

(2) Compile and run this program with “gcc” on the Linux machine.

(3) When you ran your code, did you get the time executed to be negative? If yes, why did that happen? (Since time cannot be negative). How could you fix this? Figure out how to fix it and do the necessary modifications.

(4) Change the data type of the variable time_stamp from *double* to *long int*. Is there a change in the values reported? If so, which of the values is the correct value? Why is there a difference?

(5) Find out the structure of type "timeval". Is it a standard C data type or is it platform specific?

(6) Print your output files for all runs in your report. (and the fixed versions if you made changes above).

(7) Write a C program to get bit and byte lengths for all the following C numerical data types according to the following table. The new C code file should have name "lab1_2.c" and it should generate an output file named "lab1_2_out.txt".

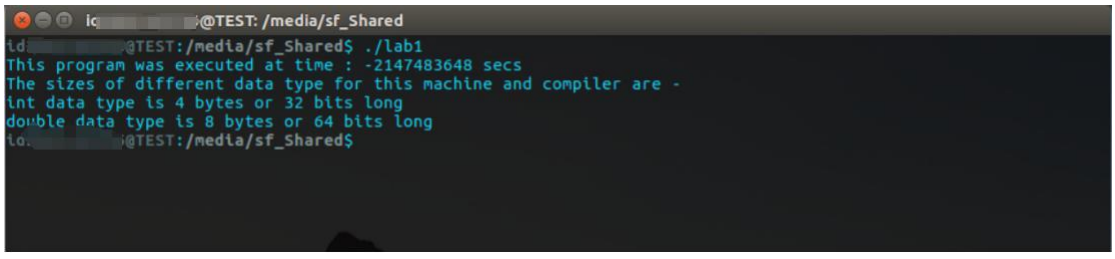
3. Lab Device or Environment

Ubuntu 16.04 (64-bit) with AMD Ryzen 9 5900HS CPU @ 3.30GHz and 4GB memory on virtual machine (Oracle VM VirtualBox)

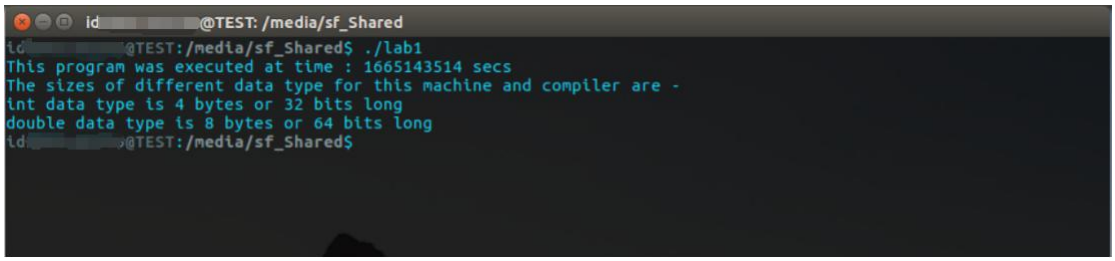
4. Results and Analysis

Result:

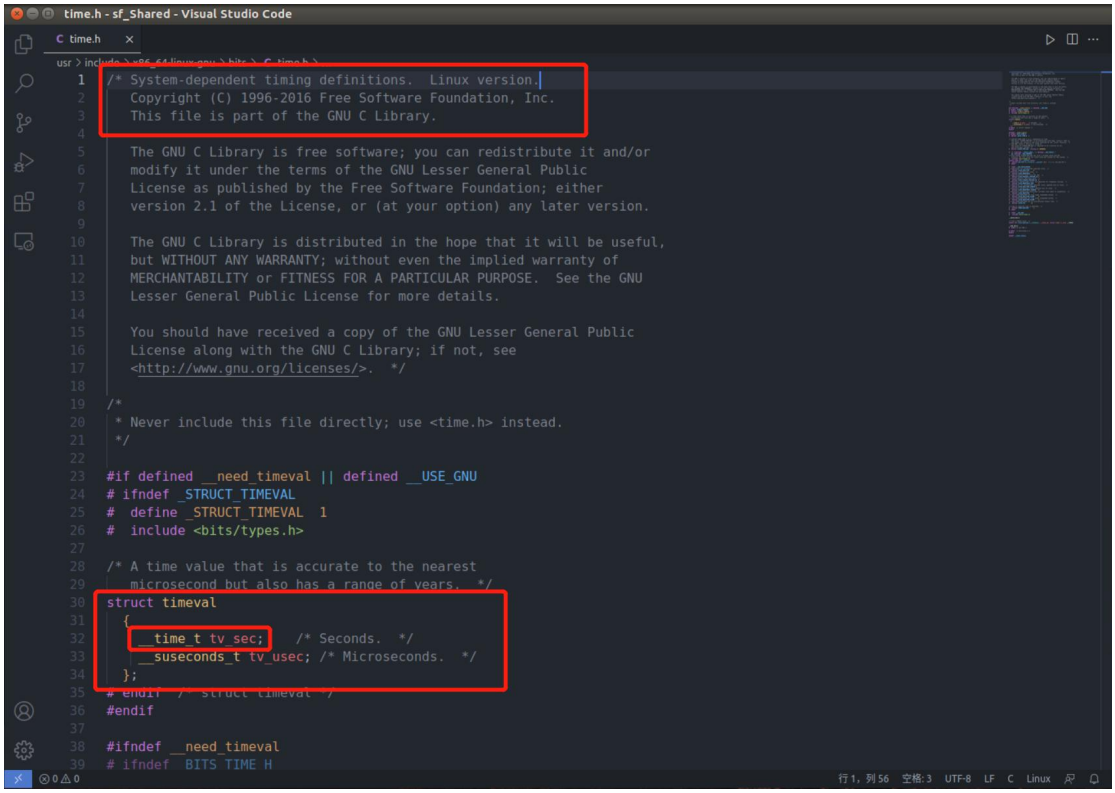
(1) Get the time executed to be negative:

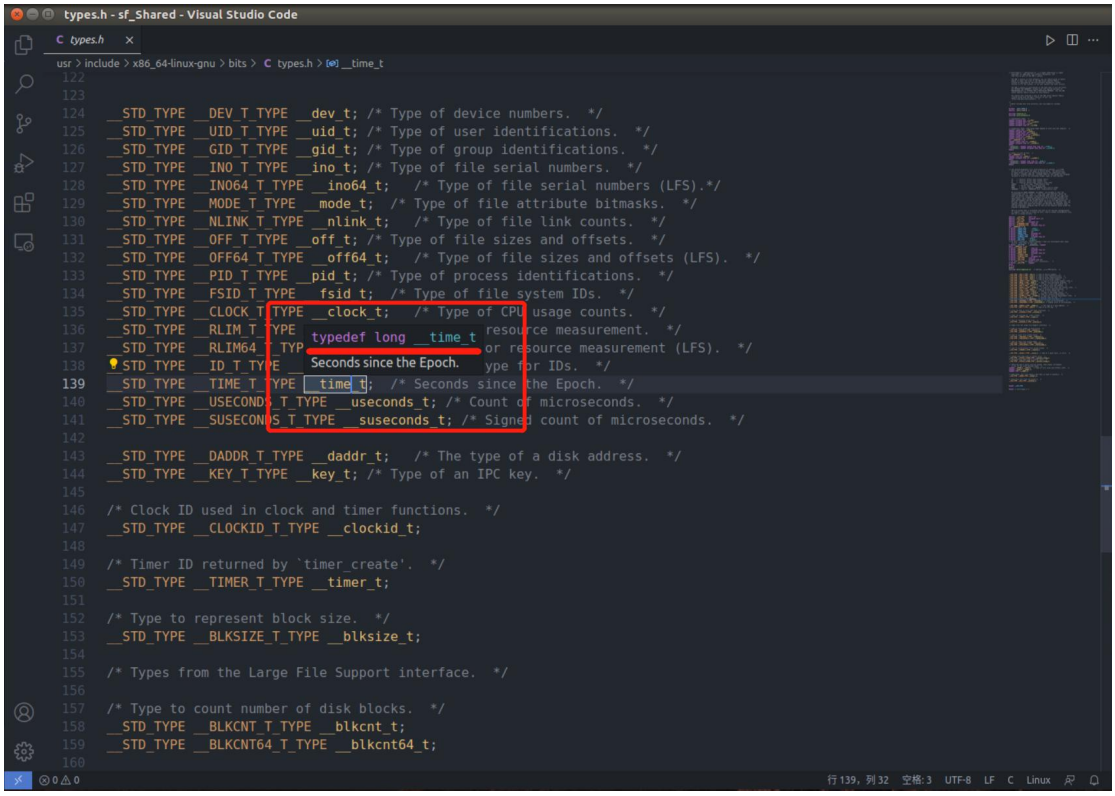


(2) Get the positive time after changing the data type of the variable time_stamp from double to long int:

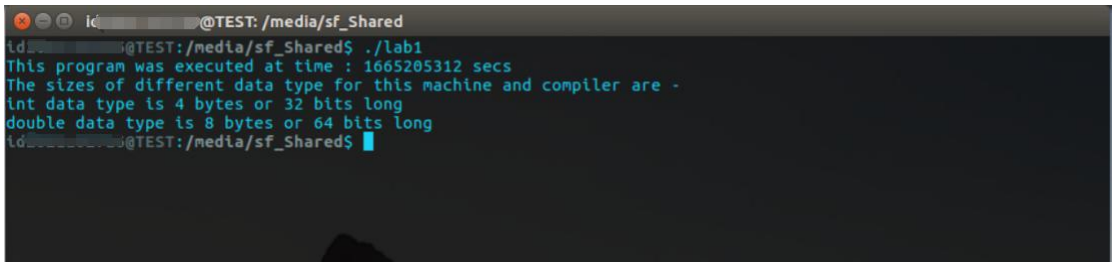


(3) Find out the structure of type “timeval” and the original data type of time_t:

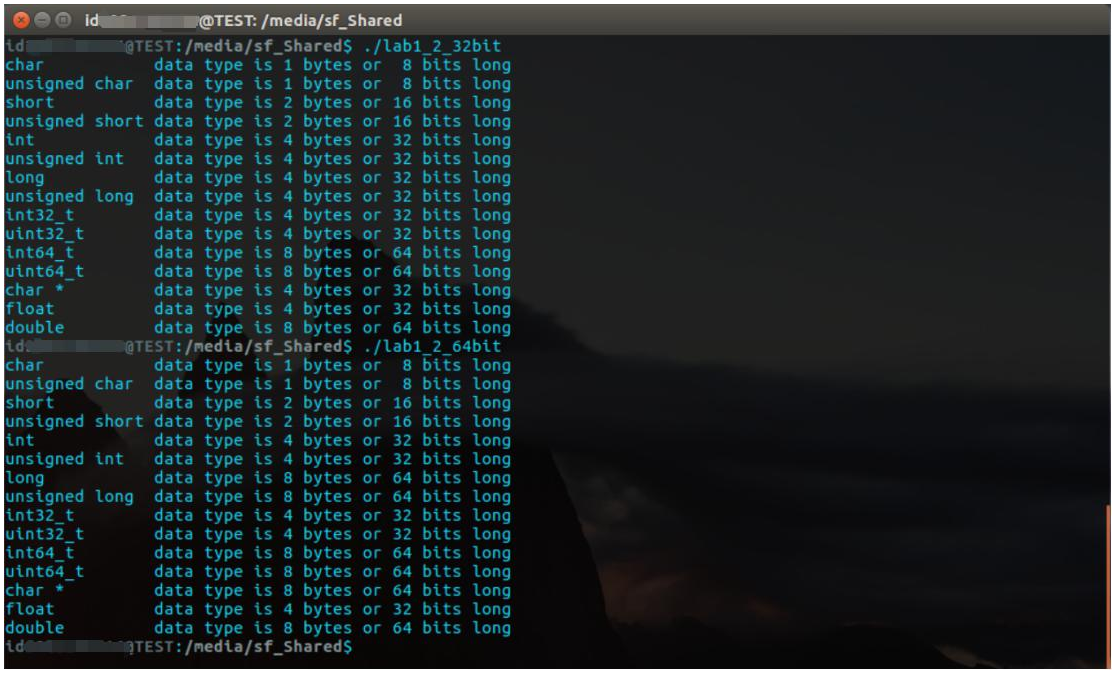




(4) Output of the fixed version:



(5) Output of Lab1_2(32-bit version & 64-bit version):



Analysis:

(1) Tag 1: The purpose of this statement is to create an *int* variable int_var.

(2) Tag 2: The purpose of this statement is to print the amount of memory (in bits and bytes) used by the variables created by the Tag 1 statement in the standard output stream (screen).

(3) When you ran your code, did you get the time executed to be negative? If yes, why did that happen? How could you fix this?

Yes. When I tried to find out the reason why it happens, I found 2 ways to make the time executed to be negative.

(a) The first way is if we use a *double* type to store tv_sec in timeval, and we don't cast it to *int* type before we print it and we print it as %d,

then because %d truncates the data and prints it as an *int* type, and a *double* type has a different permutation than an *int* type, it is possible that the highest bit of *int* type is 1 and the data will be printed as a negative error value.

(b) Another way is to convert a variable from *double* type to an *int* type when the value stored in it is larger than the maximum value that *int* can represent. This will cause int to find no integer approximation to represent, resulting in a negative error value.

I think the reason for this code problem is (b). To solve this problem, I change the data type where tv_sec is stored to the matching *long int* and change the output placeholder to %ld.

(4) Change the data type of the variable time_stamp from *double* to *long int*. Is there a change in the values reported? If so, which of the values is the correct value? Why is there a difference?

The value went from negative to positive. The positive value output by using *long int* is correct. Since the data storage mode between *double* and *long int* is different, and long int is the matching data type of tv_sec, using *long int* to store and output will get the correct result.

(5) Find out the structure of type “timeval”. Is it a standard C data type or is it platform specific?

I have placed the result above. From the time.h file we know that the type “timeval” is a system dependent timing definition type, and the original data type of timeval::tv_sec is long int.

5. Appendix (Program Code)

Lab1_1.c(modified version):

```
/*
Instructions -

To run this program - first compile to crete an executable,
then run the executable code.

Type the following in the Linux console/shell to compile and make an executable
using the gcc complier-

gcc lab1.c -o lab1

To run the executable named "lab1", type the following-

./lab1
*/

#include <stdint.h>    //For int32_t
#include <stdio.h>     //For input/output
#include <sys/time.h>  //For gettimeofday() function
#include <stdlib.h>    //For exit() function

int main() {
    int int_var;  // Tag 1

    struct timeval this_instant;
    long int time_stamp;

    FILE *my_file_pointer;
    if ((my_file_pointer = fopen("lab1_out.txt", "w")) == NULL) {
        printf("Error opening the file, so exiting\n");
        exit(1);
    }
```

```
33.
34.     gettimeofday(&this_instant, 0);
35.     time_stamp = this_instant.tv_sec;
36.
37.     // Code segment for file I/O
38.     fprintf(my_file_pointer, "This program was executed at time : %ld secs\n",
39.             time_stamp);
40.
41.     fprintf(my_file_pointer,
42.             "The sizes of different data type for this machine and compiler "
43.             "are -\n");
44.     fprintf(my_file_pointer, "int data type is %d bytes or %d bits long\n",
45.             sizeof(int_var), sizeof(int_var) * 8);
46.     fprintf(my_file_pointer, "double data type is %d bytes or %d bits long\n",
47.             sizeof(double), sizeof(double) * 8);
48.
49.     // Code segment for console I/O, this can be used instead of the file I/O
50.     printf("This program was executed at time : %ld secs\n", time_stamp);
51.
52.     printf(
53.         "The sizes of different data type for this machine and compiler are "
54.         "-\n");
55.     printf("int data type is %d bytes or %d bits long\n", sizeof(int_var),
56.            sizeof(int_var) * 8);  // Tag 2
57.     printf("double data type is %d bytes or %d bits long\n", sizeof(double),
58.            sizeof(double) * 8);
59.
60.     fclose(my_file_pointer);  // To close the output file, mandatory to actually

61.                               // get an output !
62.
63.     return 0;
64. }
```

Lab1_2.c:

```
1.  #include <stdint.h>  //For int32_t
2.  #include <stdio.h>   //For input/output
3.  #include <stdlib.h>  //For exit() function
4.
5.  int main(void) {
6.      FILE *my_file_pointer;
7.      if ((my_file_pointer = fopen("lab1_2_out.txt", "w")) == NULL) {
8.          printf("Error opening the file, so exiting\n");
```

```
        exit(1);
    }

    // Code segment for file I/O
    fprintf(my_file_pointer,
            "char           data type is %d bytes or %2d bits long\n",
            sizeof(char), sizeof(char) * 8);
    fprintf(my_file_pointer,
            "unsigned char  data type is %d bytes or %2d bits long\n",
            sizeof(unsigned char), sizeof(unsigned char) * 8);
    fprintf(my_file_pointer,
            "short          data type is %d bytes or %2d bits long\n",
            sizeof(short), sizeof(short) * 8);
    fprintf(my_file_pointer,
            "unsigned short data type is %d bytes or %d bits long\n",
            sizeof(unsigned short), sizeof(unsigned short) * 8);
    fprintf(my_file_pointer,
            "int            data type is %d bytes or %2d bits long\n",
            sizeof(int), sizeof(int) * 8);
    fprintf(my_file_pointer,
            "unsigned int   data type is %d bytes or %2d bits long\n",
            sizeof(unsigned), sizeof(unsigned) * 8);
    fprintf(my_file_pointer,
            "long           data type is %d bytes or %2d bits long\n",
            sizeof(long), sizeof(long) * 8);
    fprintf(my_file_pointer,
            "unsigned long  data type is %d bytes or %2d bits long\n",
            sizeof(unsigned long), sizeof(unsigned long) * 8);
    fprintf(my_file_pointer,
            "int32_t        data type is %d bytes or %2d bits long\n",
            sizeof(int32_t), sizeof(int32_t) * 8);
    fprintf(my_file_pointer,
            "uint32_t       data type is %d bytes or %2d bits long\n",
            sizeof(uint32_t), sizeof(uint32_t) * 8);
    fprintf(my_file_pointer,
            "int64_t        data type is %d bytes or %2d bits long\n",
            sizeof(int64_t), sizeof(int64_t) * 8);
    fprintf(my_file_pointer,
            "uint64_t       data type is %d bytes or %2d bits long\n",
            sizeof(uint64_t), sizeof(uint64_t) * 8);
    fprintf(my_file_pointer,
            "char *         data type is %d bytes or %2d bits long\n",
            sizeof(char *), sizeof(char *) * 8);
    fprintf(my_file_pointer,
```

```
53.             "float          data type is %d bytes or %2d bits long\n",
54.             sizeof(float), sizeof(float) * 8);
55.     fprintf(my_file_pointer,
56.             "double         data type is %d bytes or %2d bits long\n",
57.             sizeof(double), sizeof(double) * 8);
58.
59.     // Code segment for console I/O, this can be used instead of the file I/O
60.     printf("char           data type is %d bytes or %2d bits long\n",
61.            sizeof(char), sizeof(char) * 8);
62.     printf("unsigned char  data type is %d bytes or %2d bits long\n",
63.            sizeof(unsigned char), sizeof(unsigned char) * 8);
64.     printf("short          data type is %d bytes or %2d bits long\n",
65.            sizeof(short), sizeof(short) * 8);
66.     printf("unsigned short data type is %d bytes or %2d bits long\n",
67.            sizeof(unsigned short), sizeof(unsigned short) * 8);
68.     printf("int            data type is %d bytes or %2d bits long\n",
69.            sizeof(int), sizeof(int) * 8);
70.     printf("unsigned int   data type is %d bytes or %2d bits long\n",
71.            sizeof(unsigned), sizeof(unsigned) * 8);
72.     printf("long           data type is %d bytes or %2d bits long\n",
73.            sizeof(long), sizeof(long) * 8);
74.     printf("unsigned long  data type is %d bytes or %2d bits long\n",
75.            sizeof(unsigned long), sizeof(unsigned long) * 8);
76.     printf("int32_t        data type is %d bytes or %2d bits long\n",
77.            sizeof(int32_t), sizeof(int32_t) * 8);
78.     printf("uint32_t       data type is %d bytes or %2d bits long\n",
79.            sizeof(uint32_t), sizeof(uint32_t) * 8);
80.     printf("int64_t        data type is %d bytes or %2d bits long\n",
81.            sizeof(int64_t), sizeof(int64_t) * 8);
82.     printf("uint64_t       data type is %d bytes or %2d bits long\n",
83.            sizeof(uint64_t), sizeof(uint64_t) * 8);
84.     printf("char *         data type is %d bytes or %2d bits long\n",
85.            sizeof(char *), sizeof(char *) * 8);
86.     printf("float          data type is %d bytes or %2d bits long\n",
87.            sizeof(float), sizeof(float) * 8);
88.     printf("double         data type is %d bytes or %2d bits long\n",
89.            sizeof(double), sizeof(double) * 8);
90.
91.     fclose(my_file_pointer);
92.
93.     return 0;
94. }
```