# Lab 1: Number Storage Lab

[Lab 1: Number Storage Lab](#lab-1-number-storage-lab)  
 [I. Introduction](#i-introduction)  
 [2. Lab Instructions or Steps](#X2e3707871fdaad114dfcbba591dd899555cc4fc)  
 [3.Lab Device and Environment](#Xf63aca51f9a082ef37ee3a066bb20a9e10f6794)  
 [3.1Environment](#Xa65688011ab8ac028e996e754c6ea2091b24c2c)  
 [3.2 Lab Device](#Xa31b8a3f0121c6d19eebdbef2035a6a6277f37c)  
 [4.Results and Analysis](#X34f2c784eab62eeb907a0b676f8585d10370ef1)  
 [4.1 Answer to A](#X509c7772988f359479863eddb15489d969c12e9)  
 [4.2 Results of B](#X0642ff15e561f944a3d64b4d8b8cdbe62b33cba)  
 [4.3 Answer to C](#X9d30db1379c4884efb335a1f72bdb209479cafc)  
 [4.4 Answer to D](#X46a1a38d57fb50e3d3a683de63b14078b6dfebc)  
 [4.5 Answer to E](#X3d90f2cc587de340b485bd78573be0d963bb235)  
 [4.6 Answer to F](#Xd0b994df5b08449163985d14fe97256b8b90161)  
 [4.7 Answer to G](#X4040450a9652950cf00addb764e4ecf3432a701)  
 [5.Appendix](#Xa718826a98947f221248872ee84a93c51a35c6d)

## I. 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

This code (file name “lab1.c”) is available from the course materials.

*Activities to do-*

* a) Using less than 4 sentences, explain what function/action, the two statements marked with “Tag 1” and “Tag 2” perform.
* b) Compile and run this program with “gcc” on the Linux machine.
* c) 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.
* d) 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?
* e) Find out the structure of type “timeval”. Is it a standard C data type or is it platform specific?
* f) Print your output files for all runs in your report. (and the fixed versions if you made changes above ). *Note: If you submit an output file with negative time, you must submit the corrected version also. Failure to do so will lead to loss of 50% of the points for this lab.*
* g) 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”. *Hint – You can add codes to the lab1.c program to do this step*.

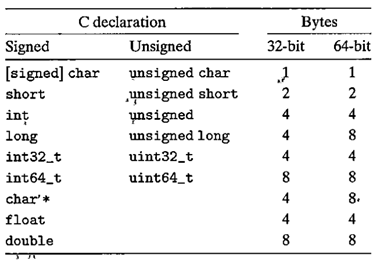


Figure 1 Typical sizes (in bytes) of basic C data types (from figure 2.3 in the textbook)

## 3.Lab Device and Environment

### 3.1Environment

root@AvA:/mnt/c/Users/XuanQuan# cat /proc/version  
Linux version 5.10.16.3-microsoft-standard-WSL2 (oe-user@oe-host) (x86\_64-msft-linux-gcc (GCC) 9.3.0, GNU ld (GNU Binutils) 2.34.0.20200220) #1 SMP Fri Apr 2 22:23:49 UTC 2021

### 3.2 Lab Device

处理器 Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz  
机带 RAM 16.0 GB (15.8 GB 可用)  
系统类型 64 位操作系统, 基于 x64 的处理器

版本 Windows 10 家庭版  
版本号 21H1  
安装日期 ‎2021/‎7/‎30  
操作系统内部版本 19043.1266

## 4.Results and Analysis

### 4.1 Answer to A

Tag1 declare an int\_val variable of type int and give it an address.

Tag2 prints the number of bytes and bits occupied by int\_val.

### 4.2 Results of B

root@AvA:/codeProject/CSAPP/Lab-1# gcc lab1.c -o lab1  
lab1.c: In function ‘main’:  
lab1.c:31:3: warning: implicit declaration of function ‘exit’ [-Wimplicit-function-declaration]  
 31 | exit(1);  
 | ^~~~  
lab1.c:31:3: warning: incompatible implicit declaration of built-in function ‘exit’  
lab1.c:18:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’  
 17 | #include <sys/time.h> //For gettimeofday() function  
 +++ |+#include <stdlib.h>  
 18 |  
lab1.c:37:65: warning: format ‘%d’ expects argument of type ‘int’, but argument 3 has type ‘double’ [-Wformat=]  
 37 | fprintf(my\_file\_pointer, "This program was executed at time : %d secs\n", time\_stamp);  
 | ~^ ~~~~~~~~~~  
 | | |  
 | int double  
 | %f  
root@AvA:/codeProject/CSAPP/Lab-1# ./lab1  
This program was executed at time : 1953063456 secs  
The sizes of different data type for this machine and compiler are -  
int data type is 4 bytes or 32 bits long  
double data type is 8 bytes or 64 bits long

Note that we get two different results in the terminal and the file.That is because there is buffering going on, and the actual output probably won't switch at the end-of-line character. I add a

fprintf(stdout, "%d", time\_stamp);

after the fprintf function and then the two function get the same results.

### 4.3 Answer to C

Yes, that is because double and int take up the different amount of space on a computer. When we convert double to int, we print out only the lower 32 bits and discard the remaining 32 bits of the double value.

Change the parameter of the print function and fprintf function from %d to %f can fix it.

fprintf(my\_file\_pointer, "This program was executed at time : %f secs\n", time\_stamp);  
printf("This program was executed at time : %f secs\n", time\_stamp);

### 4.4 Answer to D

Yes, we get a new result as 1633976974 secs and change it to the time we get 2021-10-12 02:29:34.That is the correct value.It is because the long int type is also an integer type. But there is also a risk of mistakes because in 64-bit enviroment，the long int is 8 bytes and the int is 4 bytes, and if the time value bigger than 2147483647, then it will still have mistakes in this program.To solve it, we need to change %d to %ld.

### 4.5 Answer to E

/\*  
DESCRIPTION  
 The functions gettimeofday and settimeofday can get and set the time as  
 well as a timezone. The tv argument is a timeval struct, as specified  
 in <sys/time.h>:  
\*/  
  
 struct timeval {  
 time\_t tv\_sec; /\* seconds \*/  
 suseconds\_t tv\_usec; /\* microseconds \*/  
 };

This struct is in the sys/time.h, and the sys/time.h is a date-time header file for Linux.

So it is platform specific.

### 4.6 Answer to F

the fixed versions

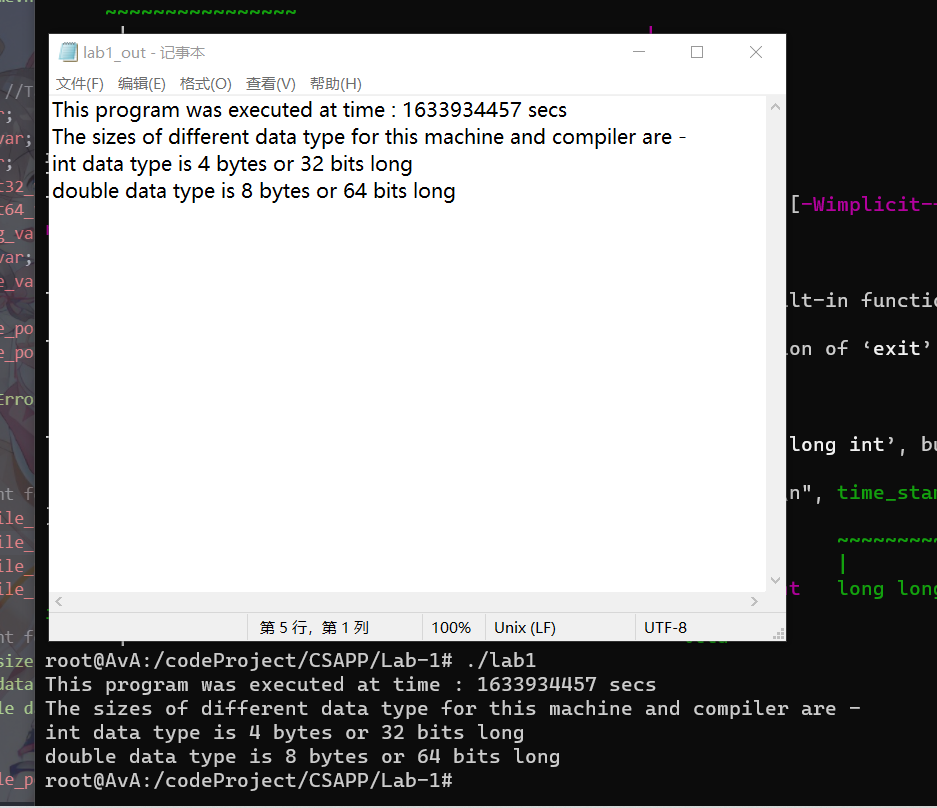
change from

double time\_stamp;  
fprintf(my\_file\_pointer, "This program was executed at time : %d secs\n", time\_stamp);  
printf("This program was executed at time : %d secs\n", time\_stamp);

to

long int time\_stamp;  
fprintf(my\_file\_pointer, "This program was executed at time : %ld secs\n", time\_stamp);  
printf("This program was executed at time : %ld secs\n", time\_stamp);

and we get



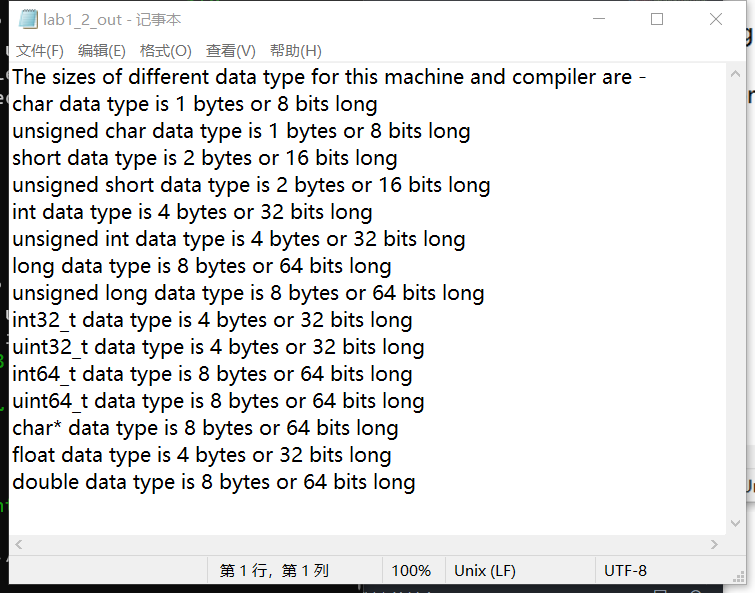
change the time\_stamp to time, we can get 2021-10-11 14:40:57.

### 4.7 Answer to G

The source code are in appendix.

Since it asked the numerical data types, we don't need to show bit and byte lengths for char\*.

And we get the lab1\_2\_out.txt.



## 5.Appendix

lab\_1\_2.c

#include <stdio.h> //For input/output  
#include <sys/time.h> //For gettimeofday() function  
#include <stdint.h> //For the int32\_t and the int64\_t  
  
int main()  
{  
  
 FILE \*my\_file\_pointer;  
 if ( (my\_file\_pointer = fopen("lab1\_2\_out.txt", "w")) == NULL)  
 {  
 printf("Error opening the file, so exiting\n");  
 exit(1);  
 }  
  
 //Code segment for file I/O  
 fprintf(my\_file\_pointer, "The sizes of different data type for this machine and compiler are -\n");  
  
 fprintf(my\_file\_pointer, "char data type is %d bytes or %d bits long\n",sizeof(char), sizeof(char)\*8 );  
 fprintf(my\_file\_pointer, "unsigned char data type is %d bytes or %d bits long\n",sizeof(unsigned char), sizeof(unsigned char)\*8 );  
 fprintf(my\_file\_pointer, "short data type is %d bytes or %d 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 %d bits long\n",sizeof(int), sizeof(int)\*8 );  
 fprintf(my\_file\_pointer, "unsigned int data type is %d bytes or %d bits long\n",sizeof(unsigned int), sizeof(unsigned int)\*8 );   
 fprintf(my\_file\_pointer, "long data type is %d bytes or %d bits long\n",sizeof(long), sizeof(long)\*8 );  
 fprintf(my\_file\_pointer, "unsigned long data type is %d bytes or %d bits long\n",sizeof(unsigned long), sizeof(unsigned long)\*8 );  
 fprintf(my\_file\_pointer, "int32\_t data type is %d bytes or %d bits long\n",sizeof(int32\_t), sizeof(int32\_t)\*8 );  
 fprintf(my\_file\_pointer, "uint32\_t data type is %d bytes or %d bits long\n",sizeof(uint32\_t), sizeof(uint32\_t)\*8 );  
 fprintf(my\_file\_pointer, "int64\_t data type is %d bytes or %d bits long\n",sizeof(int64\_t), sizeof(int64\_t)\*8 );  
 fprintf(my\_file\_pointer, "uint64\_t data type is %d bytes or %d bits long\n",sizeof(uint64\_t), sizeof(uint64\_t)\*8 );  
 fprintf(my\_file\_pointer, "char\* data type is %d bytes or %d bits long\n",sizeof(char\*), sizeof(char\*)\*8 );  
 fprintf(my\_file\_pointer, "float data type is %d bytes or %d bits long\n",sizeof(float), sizeof(float)\*8 );  
 fprintf(my\_file\_pointer, "double data type is %d bytes or %d bits long\n",sizeof(double), sizeof(double)\*8 );  
   
 fclose(my\_file\_pointer); //To close the output file, mandatory to actually get an output !  
  
 return 0;  
}