



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

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

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.

- (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”.

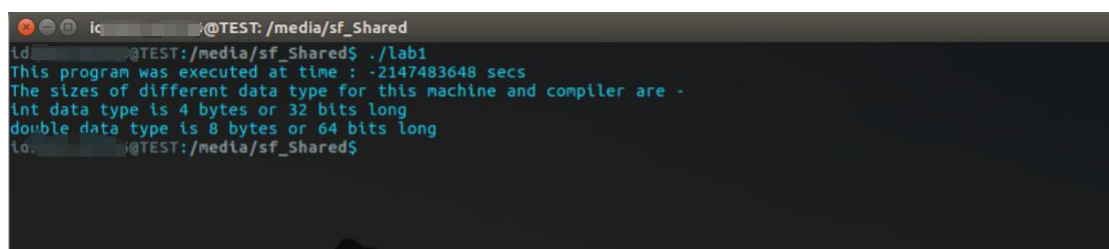
3. Lab Device or Environment

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:

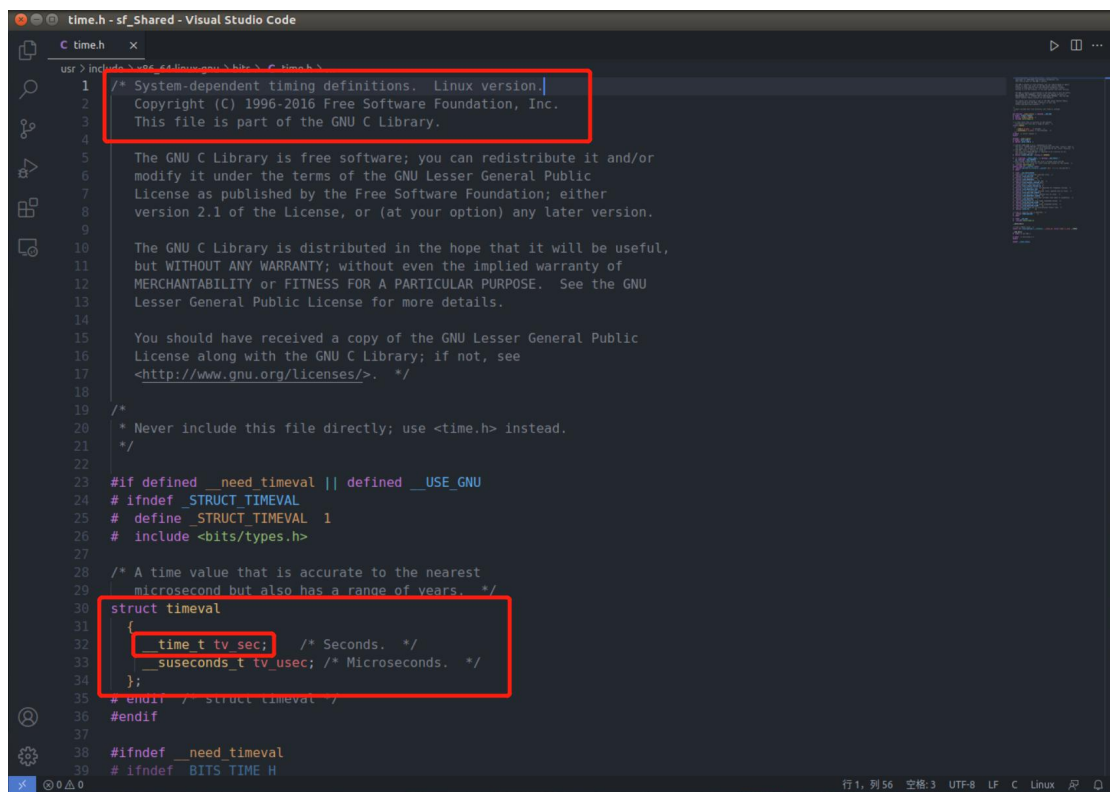


```
lq @TEST: /media/sf_Shared
ld. @TEST:/media/sf_Shared$ ./lab1
This program was executed at time : -2147483648 secs
The sizes of different data type for this machine and compiler are -
int data type is 4 bytes or 32 bits long
double data type is 8 bytes or 64 bits long
ld. @TEST:/media/sf_Shared$
```

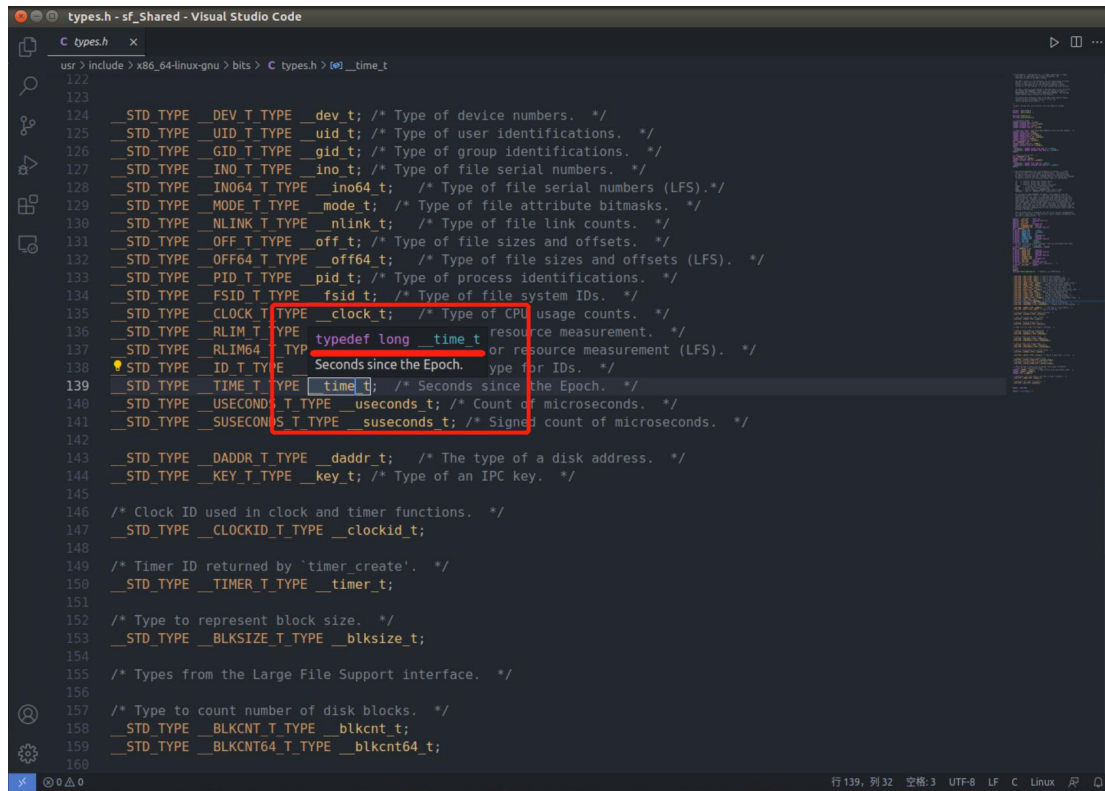
- (2) Get the positive time after changing the data type of the variable `time_stamp` from *double* to *long int*:

```
id @TEST: /media/sf_Shared
id @TEST: /media/sf_Shared$ ./lab1
This program was executed at time : 1665143514 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
id @TEST: /media/sf_Shared$
```

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

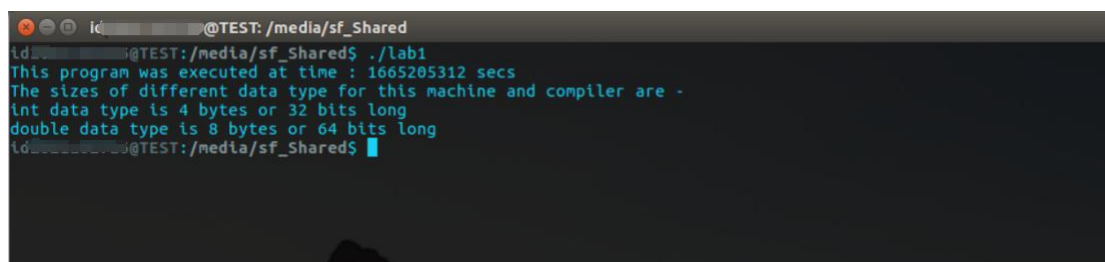


```
time.h - sf_Shared - Visual Studio Code
C time.h
usr > include <sys/time.h>
1  /* System-dependent timing definitions.  Linux version.
2  Copyright (C) 1996-2016 Free Software Foundation, Inc.
3  This file is part of the GNU C Library.
4
5  The GNU C Library is free software; you can redistribute it and/or
6  modify it under the terms of the GNU Lesser General Public
7  License as published by the Free Software Foundation; either
8  version 2.1 of the License, or (at your option) any later version.
9
10 The GNU C Library is distributed in the hope that it will be useful,
11 but WITHOUT ANY WARRANTY; without even the implied warranty of
12 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
13 Lesser General Public License for more details.
14
15 You should have received a copy of the GNU Lesser General Public
16 License along with the GNU C Library; if not, see
17 <http://www.gnu.org/licenses/>.  */
18
19 /*
20  * Never include this file directly; use <time.h> instead.
21  */
22
23 #if defined __need_timeval || defined __USE_GNU
24 # ifndef _STRUCT_TIMEVAL
25 #  define _STRUCT_TIMEVAL 1
26 #  include <bits/types.h>
27
28 /* A time value that is accurate to the nearest
29  * microsecond but also has a range of years.  */
30 struct timeval
31 {
32     time_t tv_sec; /* Seconds.  */
33     __suseconds_t tv_usec; /* Microseconds.  */
34 };
35 # endif
36 #endif
37
38 #ifndef __need_timeval
39 # ifndef _BITS_TIME_H
```



```
122
123
124 __STD_TYPE __DEV_T_TYPE __dev_t; /* Type of device numbers. */
125 __STD_TYPE __UID_T_TYPE __uid_t; /* Type of user identifications. */
126 __STD_TYPE __GID_T_TYPE __gid_t; /* Type of group identifications. */
127 __STD_TYPE __INO_T_TYPE __ino_t; /* Type of file serial numbers. */
128 __STD_TYPE __INO64_T_TYPE __ino64_t; /* Type of file serial numbers (LFS).*/
129 __STD_TYPE __MODE_T_TYPE __mode_t; /* Type of file attribute bitmasks. */
130 __STD_TYPE __NLINK_T_TYPE __nlink_t; /* Type of file link counts. */
131 __STD_TYPE __OFF_T_TYPE __off_t; /* Type of file sizes and offsets. */
132 __STD_TYPE __OFF64_T_TYPE __off64_t; /* Type of file sizes and offsets (LFS). */
133 __STD_TYPE __PID_T_TYPE __pid_t; /* Type of process identifications. */
134 __STD_TYPE __FSID_T_TYPE __fsid_t; /* Type of file system IDs. */
135 __STD_TYPE __CLOCK_T_TYPE __clock_t; /* Type of CPU usage counts. */
136 __STD_TYPE __RLIM_T_TYPE __rlim_t; /* resource measurement. */
137 __STD_TYPE __RLIM64_T_TYPE __rlim64_t; /* or resource measurement (LFS). */
138 __STD_TYPE __ID_T_TYPE __id_t; /* Seconds since the Epoch. type for IDs. */
139 __STD_TYPE __TIME_T_TYPE __time_t; /* Seconds since the Epoch. */
140 __STD_TYPE __USECOND_T_TYPE __useconds_t; /* Count of microseconds. */
141 __STD_TYPE __SUSECONDS_T_TYPE __suseconds_t; /* Signed count of microseconds. */
142
143 __STD_TYPE __DADDR_T_TYPE __daddr_t; /* The type of a disk address. */
144 __STD_TYPE __KEY_T_TYPE __key_t; /* Type of an IPC key. */
145
146 /* Clock ID used in clock and timer functions. */
147 __STD_TYPE __CLOCKID_T_TYPE __clockid_t;
148
149 /* Timer ID returned by 'timer_create'. */
150 __STD_TYPE __TIMER_T_TYPE __timer_t;
151
152 /* Type to represent block size. */
153 __STD_TYPE __BLKSIZE_T_TYPE __blksize_t;
154
155 /* Types from the Large File Support interface. */
156
157 /* Type to count number of disk blocks. */
158 __STD_TYPE __BLKCNT_T_TYPE __blkcnt_t;
159 __STD_TYPE __BLKCNT64_T_TYPE __blkcnt64_t;
160
```

(4) Output of the fixed version:



```
@TEST: /media/sf_Shared
ld: @TEST: /media/sf_Shared$ ./lab1
This program was executed at time : 1665205312 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
ld: @TEST: /media/sf_Shared$
```

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

```
id: @TEST: /media/sf_Shared
id: @TEST: /media/sf_Shared$ ./lab1_2_32bit
char          data type is 1 bytes or 8 bits long
unsigned char  data type is 1 bytes or 8 bits long
short         data type is 2 bytes or 16 bits long
unsigned short data type is 2 bytes or 16 bits long
int           data type is 4 bytes or 32 bits long
unsigned int   data type is 4 bytes or 32 bits long
long          data type is 4 bytes or 32 bits long
unsigned long  data type is 4 bytes or 32 bits long
int32_t       data type is 4 bytes or 32 bits long
uint32_t      data type is 4 bytes or 32 bits long
int64_t       data type is 8 bytes or 64 bits long
uint64_t      data type is 8 bytes or 64 bits long
char *        data type is 4 bytes or 32 bits long
float         data type is 4 bytes or 32 bits long
double        data type is 8 bytes or 64 bits long
id: @TEST: /media/sf_Shared$ ./lab1_2_64bit
char          data type is 1 bytes or 8 bits long
unsigned char  data type is 1 bytes or 8 bits long
short         data type is 2 bytes or 16 bits long
unsigned short data type is 2 bytes or 16 bits long
int           data type is 4 bytes or 32 bits long
unsigned int   data type is 4 bytes or 32 bits long
long          data type is 8 bytes or 64 bits long
unsigned long  data type is 8 bytes or 64 bits long
int32_t       data type is 4 bytes or 32 bits long
uint32_t      data type is 4 bytes or 32 bits long
int64_t       data type is 8 bytes or 64 bits long
uint64_t      data type is 8 bytes or 64 bits long
char *        data type is 8 bytes or 64 bits long
float         data type is 4 bytes or 32 bits long
double        data type is 8 bytes or 64 bits long
id: @TEST: /media/sf_Shared$
```

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):

```
1.  /*
2.  Instructions -
3.
4.  To run this program - first compile to crete an executable,
5.  then run the executable code.
6.
7.  Type the following in the Linux console/shell to compile and make an executable
8.  using the gcc complier-
9.
10. gcc lab1.c -o lab1
11.
12. To run the executable named "lab1", type the following-
13.
14. ./lab1
15. */
16.
17. #include <stdint.h>    //For int32_t
18. #include <stdio.h>     //For input/output
19. #include <sys/time.h>  //For gettimeofday() function
20. #include <stdlib.h>    //For exit() function
21.
22. int main() {
23.     int int_var;  // Tag 1
24.
25.     struct timeval this_instant;
26.     long int time_stamp;
27.
28.     FILE *my_file_pointer;
29.     if ((my_file_pointer = fopen("lab1_out.txt", "w")) == NULL) {
30.         printf("Error opening the file, so exiting\n");
31.         exit(1);
32.     }
```

```

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.
62.                                // get an output !
63.
64.     return 0;
65. }

```

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");

```

```
9.         exit(1);
10.     }
11.
12.     // Code segment for file I/O
13.     fprintf(my_file_pointer,
14.         "char          data type is %d bytes or %2d bits long\n",
15.         sizeof(char), sizeof(char) * 8);
16.     fprintf(my_file_pointer,
17.         "unsigned char data type is %d bytes or %2d bits long\n",
18.         sizeof(unsigned char), sizeof(unsigned char) * 8);
19.     fprintf(my_file_pointer,
20.         "short          data type is %d bytes or %2d bits long\n",
21.         sizeof(short), sizeof(short) * 8);
22.     fprintf(my_file_pointer,
23.         "unsigned short data type is %d bytes or %d bits long\n",
24.         sizeof(unsigned short), sizeof(unsigned short) * 8);
25.     fprintf(my_file_pointer,
26.         "int            data type is %d bytes or %2d bits long\n",
27.         sizeof(int), sizeof(int) * 8);
28.     fprintf(my_file_pointer,
29.         "unsigned int   data type is %d bytes or %2d bits long\n",
30.         sizeof(unsigned), sizeof(unsigned) * 8);
31.     fprintf(my_file_pointer,
32.         "long           data type is %d bytes or %2d bits long\n",
33.         sizeof(long), sizeof(long) * 8);
34.     fprintf(my_file_pointer,
35.         "unsigned long  data type is %d bytes or %2d bits long\n",
36.         sizeof(unsigned long), sizeof(unsigned long) * 8);
37.     fprintf(my_file_pointer,
38.         "int32_t        data type is %d bytes or %2d bits long\n",
39.         sizeof(int32_t), sizeof(int32_t) * 8);
40.     fprintf(my_file_pointer,
41.         "uint32_t        data type is %d bytes or %2d bits long\n",
42.         sizeof(uint32_t), sizeof(uint32_t) * 8);
43.     fprintf(my_file_pointer,
44.         "int64_t         data type is %d bytes or %2d bits long\n",
45.         sizeof(int64_t), sizeof(int64_t) * 8);
46.     fprintf(my_file_pointer,
47.         "uint64_t        data type is %d bytes or %2d bits long\n",
48.         sizeof(uint64_t), sizeof(uint64_t) * 8);
49.     fprintf(my_file_pointer,
50.         "char *          data type is %d bytes or %2d bits long\n",
51.         sizeof(char *), sizeof(char *) * 8);
52.     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. }
```