Operating systems

Submitted by R.S and O.A

Clarification:

1. We used for this assignment, Arch Linux so some commands will be different we added explanation for everything below.
2. Every Question has its own directory.
3. The Makefile that is “outside” is the “Makefile of makefiles” it enters all the directory and calls for the makefile in them. It also can clean all using every makefile clean.
4. Git-Link

Question_1

I used coredumpctl, because it's very easy to use and when the core dumped it been saved as .zst file to
/var/lib/systemd/coredump/core.file.1000."number".zst
So instead of unzip every time, I used coredumpctl to shorten process.

what is coredumpctl?
This is a tool that is being used to retrieve and process core dumps and metadata which were saved by system-coredumps

what is gdb?
Gdb is a gnu DeBugger, it lets you see what is going on inside you program.

What is the purpose of -g flag?
This flag is used to generate "debugging" information when you compile the C program, the "debugging" information is extra information to the executable, like names of variables, line number and more used for easier debug experience.

What is ddd?
ddd stands for Data display Debugger. Is is a graphical front-end for command -line debuggers such as GDB, it provies user friendly interface.
It's key features are:

1. Graphical interface
2. Source code Display
3. Data Visualization
4. Breakpoint management
5. Interactive debugging

I created 3 executable programs file1, file2 and file3:

First file- Stack overflow main func calls it self.
Second file- division by zero.
Third file- undefined memory usage.

I used at the start "make all" to create all the executables.
$ make all

first file (Stack overflow)
using -g flag to compile
$./file1 #to run the program (create a segfault)
$ coredumpctl gdb file1
we reterive the coredump and then apply gdb

```
 24s  ••/Assigment_1/Question_1  main
 ./file1
fish: Job 1, './file1' terminated by signal SIGSEGV (Address boundary error)
 114ms  ••/Assigment_1/Question_1  main
 coredumpctl gdb file1
          PID: 266298 (file1)
          UID: 1000 (sibo)
          GID: 1000 (sibo)
       Signal: 11 (SEGV)
    Timestamp: Sat 2024-05-18 19:55:51 IDT (4s ago)
  Command Line: ./file1
   Executable: /home/sibo/Desktop/Year_2_B/Maharahot_hafhala/Assigment_1/Question_1/file1
 Control Group: /user.slice/user-1000.slice/session-2.scope
         Unit: session-2.scope
        Slice: user-1000.slice
      Session: 2
    Owner UID: 1000 (sibo)
      Boot ID: 0303a84d1a8341699e25834ac7ef7617
   Machine ID: 5719fd5c473742dcaeb38bb8df19b895
     Hostname: archlinux
      Storage: /var/lib/systemd/coredump/core.file1.1000.0303a84d1a8341699e25834ac7ef7617.266298.1716051351000000.zst (present)
  Size on Disk: 381.9K
      Message: Process 266298 (file1) of user 1000 dumped core.

               Stack trace of thread 266298:
               #0  0x000064df78a60122 n/a (/home/sibo/Desktop/Year_2_B/Maharahot_hafhala/Assigment_1/Question_1/file1 + 0x1122)
               #1  0x000064df78a60127 n/a (/home/sibo/Desktop/Year_2_B/Maharahot_hafhala/Assigment_1/Question_1/file1 + 0x1127)
               #2  0x000064df78a60127 n/a (/home/sibo/Desktop/Year_2_B/Maharahot_hafhala/Assigment_1/Question_1/file1 + 0x1127)
               #3  0x000064df78a60127 n/a (/home/sibo/Desktop/Year_2_B/Maharahot_hafhala/Assigment_1/Question_1/file1 + 0x1127)
```

```
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x000064df78a60122 in main () at file1.c:5
--Type <RET> for more, q to quit, c to continue without paging--c
5           main();
(gdb) where
#0  0x000064df78a60122 in main () at file1.c:5
(gdb)
```

We can see that the line that create the segmenation fault is line 7 which executed main();
Also in the first image we can see that we can see the Stack calls.

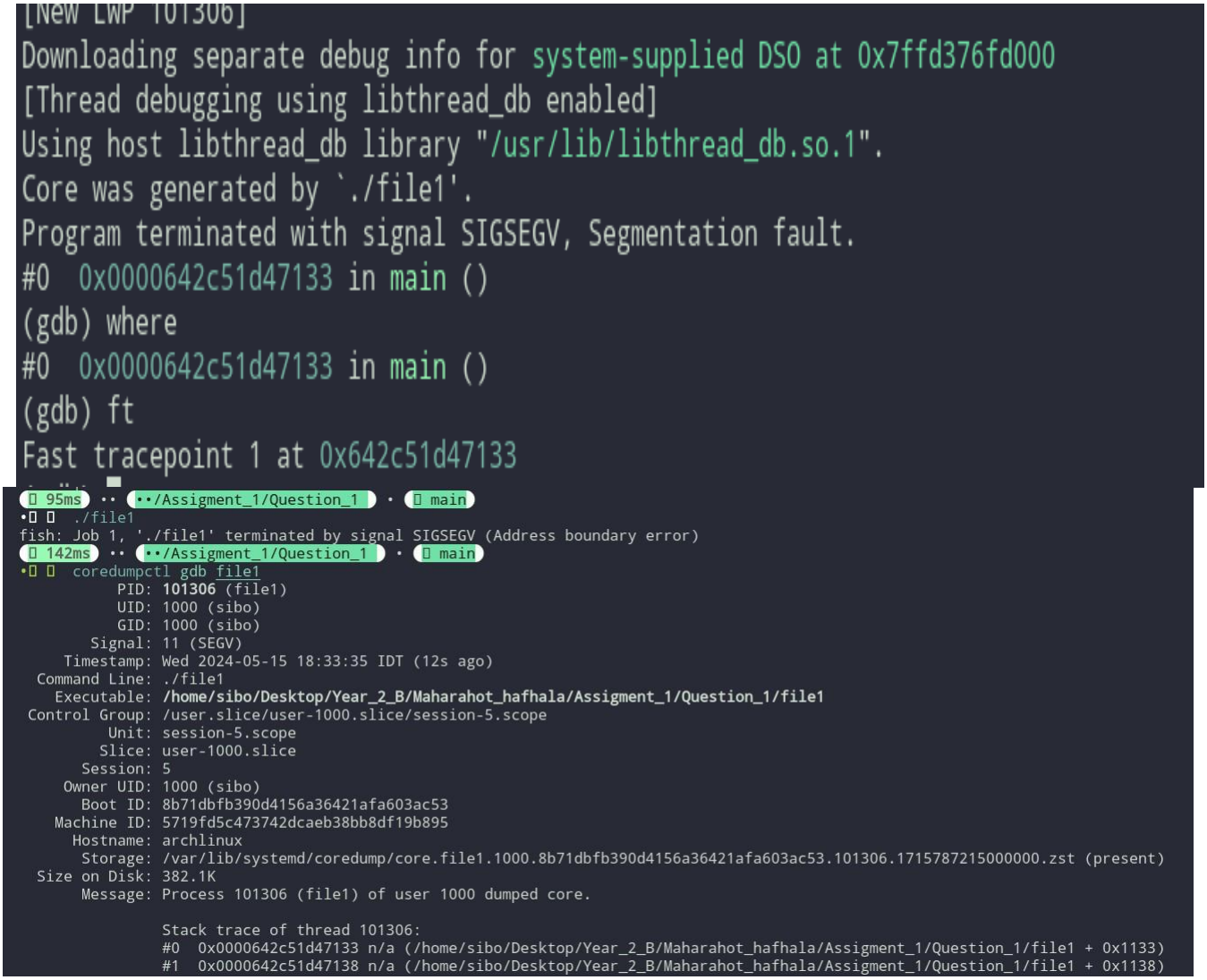
$ ddd file1 /var/lib/system/coredump/core.file1.1000. “the number of the core dump”.zst

```
 103ms  ~/Assigment_1/Question_1  main
 ./file1
fish: Job 1, './file1' terminated by signal SIGSEGV (Address boundary error)
 120ms  ~/Assigment_1/Question_1  main
 ddd file1 /var/lib/systemd/coredump/core.file1.1000.8b71dbfb390d4156a36421afa603ac53.105384.1715787640000000.zst
```

Visual interface:

We can see that line 5 is the problem (that causes the “Stack overflow”)

Without -g (just shows that there’s a segfault but doesn’t give more information)



ddd doesn’t open.

Second file (division by zero)

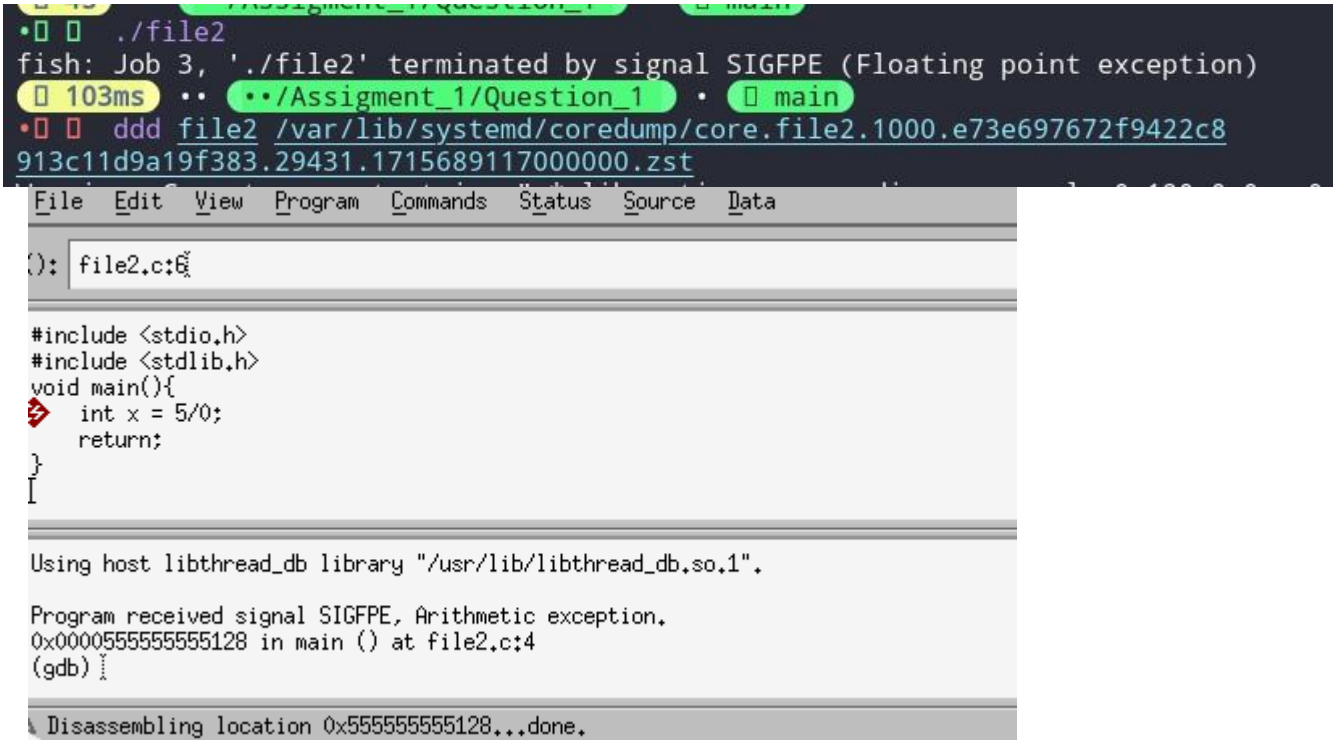
Using -g flag to compile
$ ./file2 #to run he program (creates sefault)
$ coredumpctl gdb file2

```
 1m40s  ••/Assigment_1/Question_1  main
•  ./file2
fish: Job 1, './file2' terminated by signal SIGFPE (Floating point exception)
 105ms  ••/Assigment_1/Question_1  main
•  coredumpctl gdb file2
          PID: 107440 (file2)
          UID: 1000 (sibo)
          GID: 1000 (sibo)
       Signal: 8 (FPE)
    Timestamp: Wed 2024-05-15 18:45:13 IDT (34s ago)
  Command Line: ./file2
    Executable: /home/sibo/Desktop/Year_2_B/Maharahot_hafhala/Assigment_1/Question_1/file2
 Control Group: /user.slice/user-1000.slice/session-5.scope
          Unit: session-5.scope
         Slice: user-1000.slice
       Session: 5
     Owner UID: 1000 (sibo)
       Boot ID: 8b71dbfb390d4156a36421afa603ac53
    Machine ID: 5719fd5c473742dcaeb38bb8df19b895
      Hostname: archlinux
       Storage: /var/lib/systemd/coredump/core.file2.1000.8b71dbfb390d4156a36421afa603ac53.107440.1715787913000000.zst (present)
  Size on Disk: 17.7K
       Message: Process 107440 (file2) of user 1000 dumped core.

                Stack trace of thread 107440:
                #0  0x0000598138360128 n/a (/home/sibo/Desktop/Year_2_B/Maharahot_hafhala/Assigment_1/Question_1/file2 + 0x1128)
                #1  0x0000726f28fcbc88 n/a (libc.so.6 + 0x25c88)
                #2  0x0000726f28fcbd4c __libc_start_main (libc.so.6 + 0x25d4c)
                #3  0x0000598138360045 n/a (/home/sibo/Desktop/Year_2_B/Maharahot_hafhala/Assigment_1/Question_1/file2 + 0x1045)
                ELF object binary architecture: AMD x86-64
```

```
This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.archlinux.org>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0x7fff679bb000
--Type <RET> for more, q to quit, c to continue without paging--c
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/libthread_db.so.1".
Core was generated by `./file2'.
Program terminated with signal SIGFPE, Arithmetic exception.
#0  0x0000598138360128 in main () at file2.c:4
4           int x = 5/0;
(gdb) where
#0  0x0000598138360128 in main () at file2.c:4
```

It shows that there's Arithemetic excetion in line 4, which is "int x = 5 / 0 ;"

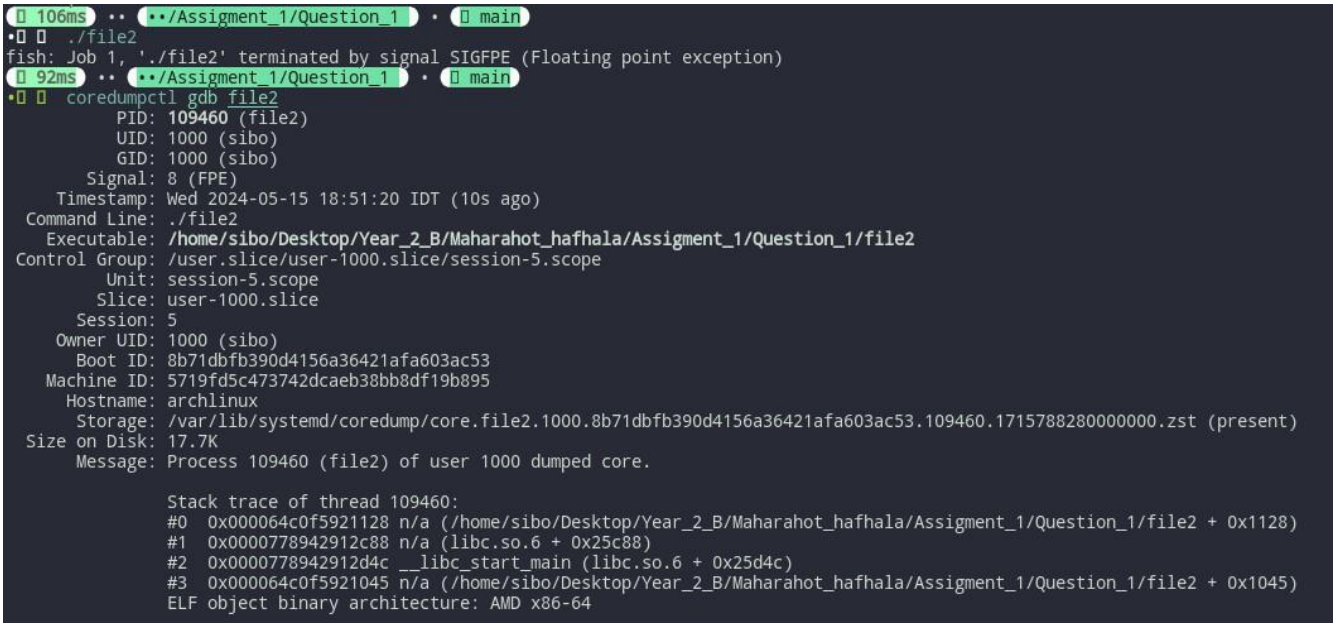
$ ddd file2 /var/lib/system/coredump/core.file2.1000. “ the number of the core dump”.zst



Also shows that there’s Arithmetic exception line 4, also been pointed out by the red pointer.

This is without -g

Shows that there’s exception but doesn’t specify where.



ddd doesn’t open.

file3 (undefined memory usage)
Using -g flag to compile
$ ./file3 #to run the program(creates segfault)
$ coredumpctl gdb file3
$ ddd file3 /var/lib/system/coredump/core.file3.1000.”number”.zst

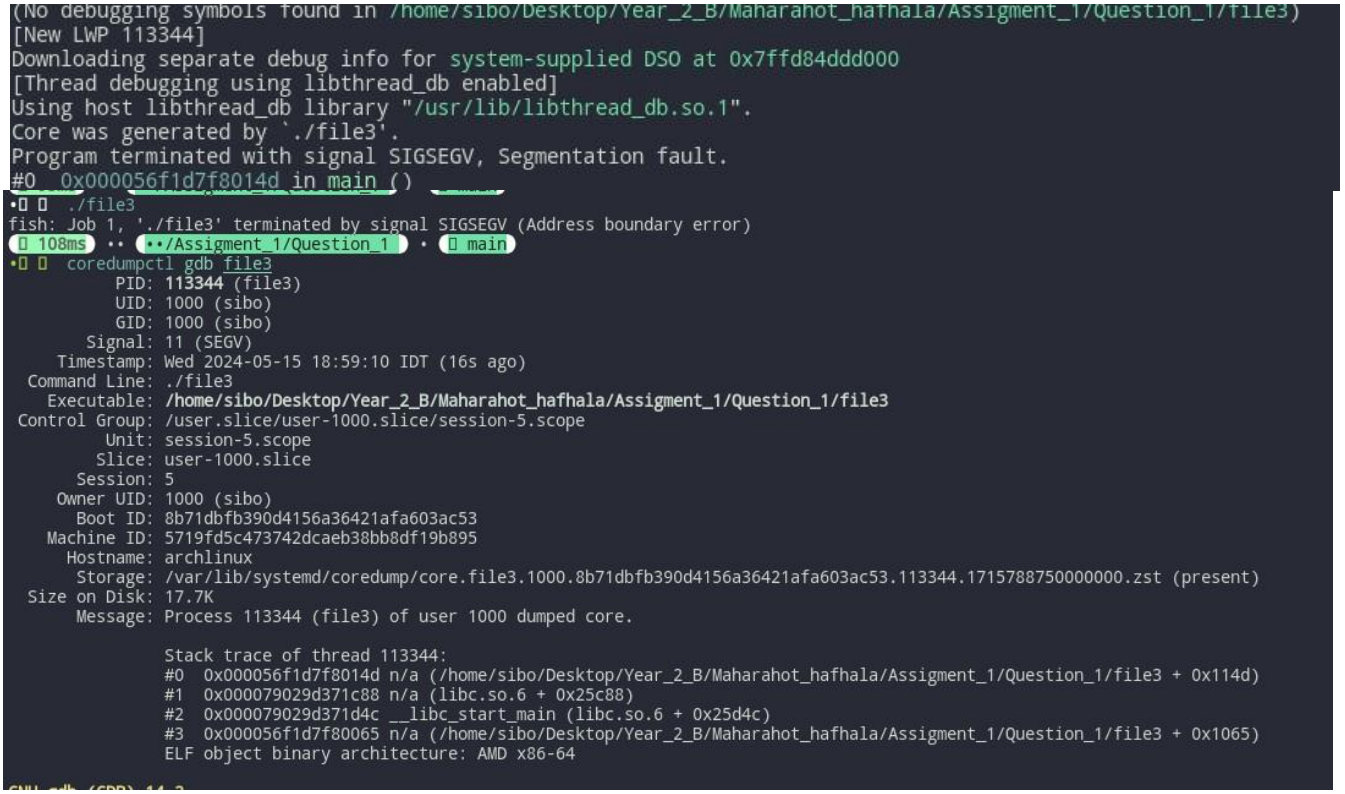
```
This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.archlinux.org>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/libthread_db.so.1".
Core was generated by `./file3'.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x000062de66ee814d in main () at file3.c:6
6           *ptr = 7;
(gdb) where
#0  0x000062de66ee814d in main () at file3.c:6
```

```
./file3
fish: Job 1, './file3' terminated by signal SIGSEGV (Address boundary error)
 80ms  ••  ••/Assigment_1/Question_1  •  main
 coredumpctl gdb file3
          PID: 111412 (file3)
          UID: 1000 (sibo)
          GID: 1000 (sibo)
       Signal: 11 (SEGV)
    Timestamp: Wed 2024-05-15 18:54:58 IDT (8s ago)
  Command Line: ./file3
   Executable: /home/sibo/Desktop/Year_2_B/Maharahot_hafhala/Assigment_1/Question_1/file3
 Control Group: /user.slice/user-1000.slice/session-5.scope
          Unit: session-5.scope
         Slice: user-1000.slice
       Session: 5
     Owner UID: 1000 (sibo)
       Boot ID: 8b71dbfb390d4156a36421afa603ac53
    Machine ID: 5719fd5c473742dcaeb38bb8df19b895
      Hostname: archlinux
       Storage: /var/lib/systemd/coredump/core.file3.1000.8b71dbfb390d4156a36421afa603ac53.111412.1715788498000000.zst (present)
  Size on Disk: 17.8K
       Message: Process 111412 (file3) of user 1000 dumped core.

                Stack trace of thread 111412:
                #0  0x00005be96f41f14d n/a (/home/sibo/Desktop/Year_2_B/Maharahot_hafhala/Assigment_1/Question_1/file3 + 0x114d)
                #1  0x00007aa1e79b9c88 n/a (libc.so.6 + 0x25c88)
                #2  0x00007aa1e79b9d4c __libc_start_main (libc.so.6 + 0x25d4c)
                #3  0x00005be96f41f065 n/a (/home/sibo/Desktop/Year_2_B/Maharahot_hafhala/Assigment_1/Question_1/file3 + 0x1065)
                ELF object binary architecture: AMD x86-64
```

```
File  Edit  View  Program  Commands  Status  Source  Data
(): main

#include <stdio.h>
#include <stdlib.h>
void main(){
    //accessing undifined memory location
    int *ptr = NULL;
    *ptr = 7;
    printf("%d", *ptr); // won't get executed
    return;
}

To quickly display variable values,
double-click on the variable name.
Show Tip of the D                    ng DDD
Run
Interrupt
Step  Stepi
Next  Nexti
Until  Finish
Cont  Kill
Up  Down
Undo  Redo
Edit  Make
Close  Next Tip

Using host libthread_db library "/usr/lib/libthread_db.so.1".

Program received signal SIGSEGV, Segmentation fault.
0x000055555555514d in main () at file3.c:6
```

We can see in the above image the line in which we accessed the undefined memory.

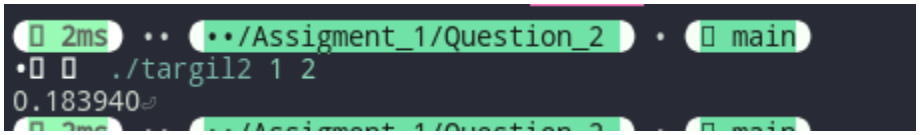
without -g flag



ddd doesn’t open.

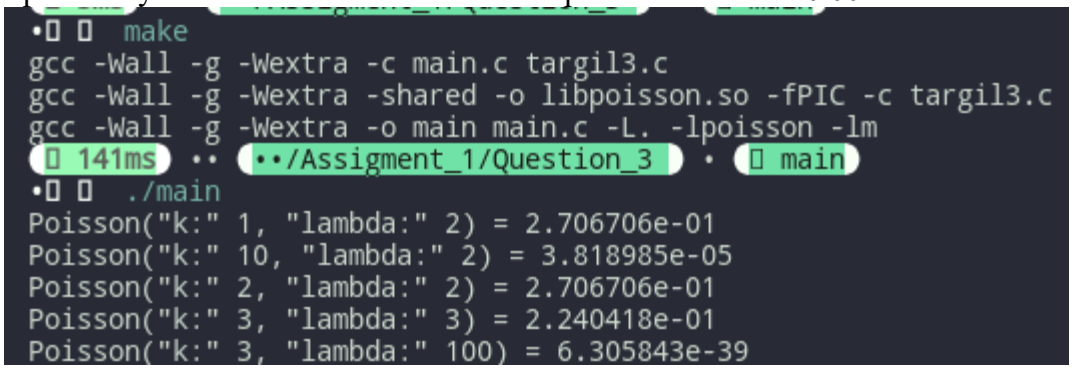
Question_2

the code for the function with the make file in git.
We give it two arguments first is Delta and the second is k.
This is how we run it and the output.
We used -lm flag to compile The C program with the gnu compiler. It instructs the linker to include the math library.
Example for the code output:
(I printed with %Lf)



Question_3:

I used the -shared flag on targil3.c to create the shared library (dynamic) “libpoisson.so ” in the main executable “main”.
Also used again the -lm flag.
I printed by %Le instead of %lf so the fifth print won’t show “0.00…”



Question_4

Input method

```
void getInputForAdjacencyMatrix(int row, int col, int graph[row][col]) {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            scanf("%d", &graph[i][j]);
        }
    }
}
```

I added if and checks to verify that the input is correct

My test cases are:

1. A valid input (see that it's excuted all the code without the checks)
2. Graph with too many edges
3. Graph with negative weights
4. Invalid start index "-1"
5. Not eligible rows and colomns
6. Non int source

To run the code write

$ make coverage: $all

1. Chmod +x test.sh (after the first time not needed)
    - # chmod: is a command used to change the permissions of files and directories.
    - # +x: This adds the execute permission to the file.
    - # test.sh: This is the name of the file whose permissions are being modified.
2. It makes dijkstra (exe file)
3. It runs the $ ./test.sh (shell script)
4. $ gcov $(srcs) (gcov on dijkstra.c)
5. $ cat dijkstra.c.gcov (to see which lines been executed and how much times)

In the following image you can see that it executed 100% of the code.

```
gcov dijkstra.c
File 'dijkstra.c'
Lines executed:100.00% of 56
Creating 'dijkstra.c.gcov'

Lines executed:100.00% of 56
cat dijkstra.c.gcov
        -:    0:Source:dijkstra.c
        -:    0:Graph:dijkstra.gcno
        -:    0:Data:dijkstra.gcda
        -:    0:Runs:6
        -:    1:#include <limits.h>
        -:    2:#include <stdbool.h>
        -:    3:#include <stdio.h>
        2:    4:int minDistance(int dist[], bool sptSet[], int len)
        -:    5:{
        -:    6:        int min = INT_MAX, min_index;
        8:    7:        for (int v = 0; v < len; v++)
        6:    8:                if (sptSet[v] == false && dist[v] <= min)
        -:    9:                        min = dist[v], min_index = v;
        -:   10:
        2:   11:        return min_index;
```

Question_5

What is void srand(seed)?

Is a function in C and cpp, it initializes the random number generator with a seed value. This seed value determines the starting po9int for the sequence of pedudo-random number generated by rand(). By providing the same seed you can ensure to get the same “random” result.

What is gprof?

Is a performance analysis tool for Unix-like operating systems. It is used for profiling application written in C or cpp.

Gprof generated a report detailing the time spent in each function, as well as the call graph showing how functions are related to each other.

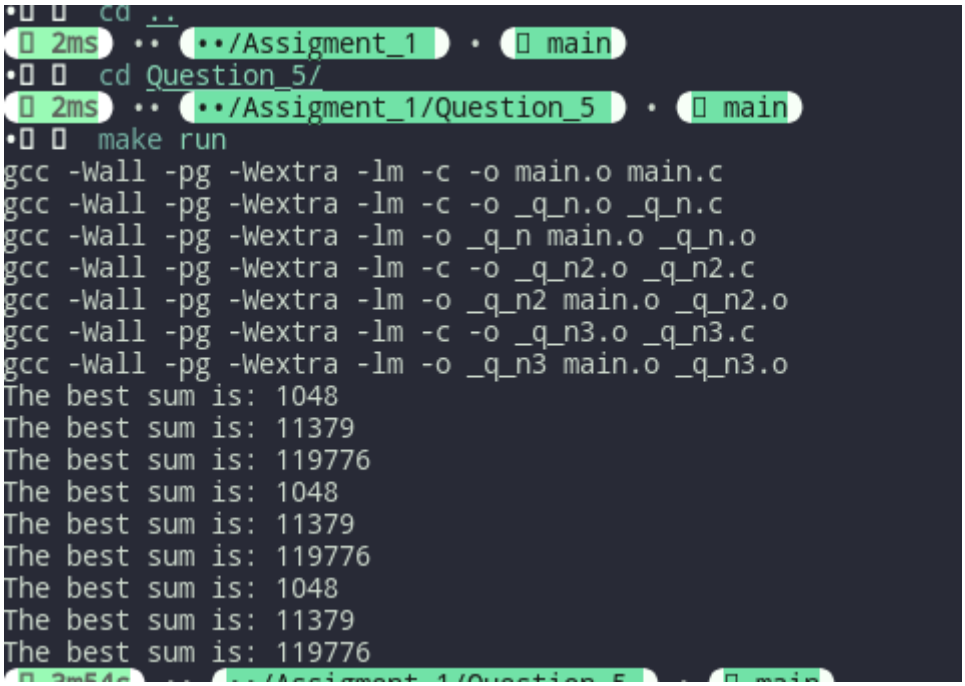
I then use rand() to generate a number%75 (max is 74 min is zero) and substruct it by 25 so there’s a good chance that the Array will contain negative number. (in _q_n {1,2,3}.c)

To run all the profiles one after one, you can use- $ make run

It enters 420 as seed and all the $$size that been defined for this example they are 100 1,000 10,000 as asked.

To clean all the files that have been just created you can you- $make clean, if you also want to delete the profiles you can undocumented line 37 (deletes all the files listed in the path “profiles”)

Example of how to use it



Question_6

Users Commands

- **What is cat (1)?**

  Cat stands for "concatenate", it's purpose is to concatenate and display the contents of files
- **What is grep (1)?**

  grep is a command line for searching plan-text data sets for lines that match a regex (regular expression). It's useful for parsing through large files or outputs and extracting relevant lines.
- **What is awk (1)?**

  The awk command is a Linux tool and programming language that allows users to process and manipulate data and produce formatted reports
- **What is sed (1)?**

  Short for stream editor, it's a command line utility for parsing and transforming text. Commonly used for performing search-and-replace operations and text manipulations.
- **What is cut (1)?**

  'cut' is a command-line utility for cutting out sections of each line of a file or input stream it's often used to extract columns from a stream.

Process Image:

Process image refers to the executable binary and its associated data and resources.

System calls:

- **What is fork (2)?**
  Creates a new process by duplicating the calling process. The new process, is called child process, is an exact copy of the calling process except for few details. (return value, Process Id and more)
- **What is pipe (2)?**
  Creates a unidirectional communication channel (pipe) between two processes. It's allowing the output of one process be an input for other process
- **What is execve (2)?**
  "execve" loads and exexcutres a new program in the current process. It replaces the current process image with a new one. The new program becomes the running program for the process.
- **What is Dup2 (2)?**
  It duplicates an open file descriptor to another file descriptor. Often used for file descriptor manipulation, such as redirecting input outputs (streams)

This creates input so you could easily check if findphon works $ make input

If there's two people with the same first name, you can add the last name and it will filter the exact person. We have been instructed to assume there's no two people with the same exact name with different phone numbers.

$ ./findPhone "first name" "last name"

We created 5 Processes (forks):

1. **Cat into inputPipe (read the phonebook)**
2. **filter the filterPipe after redirecting (using grep)**
3. **Apply the second filter and then assign it into filterPipe (optional if we got in total 3 args)**
4. **Replace spaces with # and then commas with spaces (using one sed :))**
5. **Using awk to output the second field (Phone number)**

Example of how to use it and how it works