

Linux digital assignment-2

Name: v.meghana

Regno: 17mis1187

Gdb debugging:

Example: recursive function : gcd of two numbers

Code:

```
#include <stdio.h>

int gcd(int, int);

int main()
{
    int a, b, result;

    printf("Enter the two numbers to find their GCD: ");
    scanf("%d%d", &a, &b);
    result = gcd(a, b);
    printf("The GCD of %d and %d is %d.\n", a, b, result);
}

int gcd(int a, int b)
{
    while (a != b)
    {
        if (a > b)
        {
            return gcd(a - b, b);
        }
    }
}
```

```

    }
else
{
    return gcd(a, b - a);
}
}
return a;
}

```

Output:

The compilation takes through gcc line by line and storing the output in the “test”. and later using the run command for executing the program.

```

linuxmint@linuxmint-VirtualBox:~$ gedit gcd.c
linuxmint@linuxmint-VirtualBox:~$ gcc -g gcd.c -o test
linuxmint@linuxmint-VirtualBox:~$ gdb test
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from test...done.
(gdb) run
Starting program: /home/linuxmint/test
Enter the two numbers to find their GCD: 223 34
The GCD of 223 and 34 is 1.
[Inferior 1 (process 2298) exited normally]

```

The break command is used to insert a pause in between the program.

backtrace(bt) is a summary of how your program got where it is. It shows one line per frame, for many frames, starting with the currently executing frame (frame zero)

continue command is to resume normal execution by clicking “continue“ or “c”. gdb will run until your program ends, your program crashes or gdb encounters a break point.

Next command : if the line to be executed is a function call , gdb will step into that function and start executing its code one line at a time.

```

(gdb) b 10
Breakpoint 1 at 0x555555554742: file gcd.c, line 10.
(gdb) run
Starting program: /home/linuxmint/test

Breakpoint 1, main () at gcd.c:10
10      scanf("%d%d", &a, &b);
(gdb) b gcd
Breakpoint 2 at 0x5555555547b5: file gcd.c, line 17.
(gdb) un
Ambiguous command "un": undisplay, unset, until.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/linuxmint/test

Breakpoint 1, main () at gcd.c:10
10      scanf("%d%d", &a, &b);
(gdb) continue
Continuing.
Enter the two numbers to find their GCD: 234
34

Breakpoint 2, gcd (a=234, b=34) at gcd.c:17
17      while (a != b)

```

```

Breakpoint 2, gcd (a=234, b=34) at gcd.c:17
17      while (a != b)
(gdb) bt
#0  gcd (a=234, b=34) at gcd.c:17
#1  0x000055555555476d in main () at gcd.c:11
(gdb) n
19          if (a > b)
(gdb) n
21          return gcd(a - b, b);
(gdb) n

```

```

Breakpoint 2, gcd (a=200, b=34) at gcd.c:17
17      while (a != b)
(gdb) frame 1
#1  0x00005555555547d7 in gcd (a=234, b=34) at gcd.c:21
21          return gcd(a - b, b);
(gdb) frame 2
#2  0x000055555555476d in main () at gcd.c:11
11      result = gcd(a, b);
(gdb) █

```

Memory debugging:

syntax: x/nfu address

n- Number Of Lines

f- Format

u- byte/word

the values for u can be in unsigned,hexadecimal,gaintwords,bytes.

The values for format are binary,octal,unsigned,string,float,unsigned etc.

Example: in the above code:

x/3uh : requesting to display three halfwords(h) of memory,formatted as unsigned decimal integers('u') starting address at 0x722.

```
(gdb) break main
Breakpoint 1 at 0x722: file gcd.c, line 6.
(gdb) x/3uh 0x722
0x722 <main+8>: 18532    1163    10277
(gdb) x/4dh
0x728 <main+14>:      0      18432   17801   12792
(gdb) █

0x728 <main+14>:      0      18432   17801   12792
(gdb) x/5i
0x730 <main+22>:      rorb    $0x3d,-0x73(%rax)
0x734 <main+26>:      push    %rax
0x735 <main+27>:      add     %eax, (%rax)
0x737 <main+29>:      add     %bh, 0x0(%rax)
0x73d <main+35>:      callq  0x5e0 <printf@plt>
(gdb)
```

Strace for program and finding the system calls:

Code:

```
#include<stdio.h>
```

```
#include<fcntl.h>
```

```
#include<errno.h>
```

```
#include<stdlib.h>
```

```
extern int errno;
```

```
int main()
```

```
{
```

```

int fd = open("/home/linuxmint/yy.c", O_RDONLY);

printf("fd = %d\n", fd);

if (fd == -1)

{

    printf("Error Number % d\n", errno);

    perror("Program");

}

int ret;

char buff="tony"

ret = write(fd,buff, sizeof(buff));

if (sz == -1 && errno != EINTR) {

    perror("Write to output file");

    exit(EXIT_FAILURE);

}

return 0;

}

```

Output:

```

linuxmint@linuxmint-VirtualBox:~$ strace ./test
execve("./test", [ "./test" ], 0x7ffe2ad5f2d0 /* 49 vars */) = 0
brk(NULL)                               = 0x55c784cc1000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)     = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=107352, ...}) = 0
mmap(NULL, 107352, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f5353a97000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0\0\0\1\0\0\0\260\34\2\0\0\0\0....", 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f5353a95000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f535349a000
mprotect(0x7f5353681000, 2097152, PROT_NONE) = 0
mmap(0x7f5353881000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f5353881000
mmap(0x7f5353887000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f5353887000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f5353a964c0) = 0
mprotect(0x7f5353881000, 16384, PROT_READ) = 0
mprotect(0x55c7836e7000, 4096, PROT_READ) = 0
mprotect(0x7f5353ab2000, 4096, PROT_READ) = 0
munmap(0x7f5353a97000, 107352)          = 0
openat(AT_FDCWD, "/home/linuxmint/test2.c", O_RDONLY|O_CREAT, 001) = -1 EACCES (Permission denied)
write(-1, 0x7c, 1)                     = -1 EBADF (Bad file descriptor)
exit_group(0)                           = ?
+++ exited with 0 +++
linuxmint@linuxmint-VirtualBox:~$

```

```

linuxmint@linuxmint-VirtualBox:~$ gedit ii.c
linuxmint@linuxmint-VirtualBox:~$ gcc -g ii.c -o test
ii.c: In function 'main':
ii.c:23:11: warning: initialization makes integer from pointer without a cast [-Wint-conversion]
    char buff="tony";
          ^~~~~~
ii.c:24:11: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-function-declaration]
    ret = write(fd,buff,sizeof(buff));
          ^~~~~~
          fwrite
linuxmint@linuxmint-VirtualBox:~$ strace -e trace=write ./test
write(3, 0x3d, 1) = -1 EBADF (Bad file descriptor)
write(4, "Write to output file: Bad file d"... , 42Write to output file: Bad file descriptor
) = 42
write(1, "fd = 3/n", 8fd = 3/n) = 8
+++ exited with 1 +++
linuxmint@linuxmint-VirtualBox:~$

```

```

linuxmint@linuxmint-VirtualBox:~$ strace -e trace=open,read ls
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20b\0\0\0\0\0"... , 832) = 832
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\260\34\2\0\0\0\0\0"... , 832) = 832
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 \25\0\0\0\0\0\0"... , 832) = 832
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\16\0\0\0\0\0\0"... , 832) = 832
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\3\0>\0\1\0\0\000b\0\0\0\0\0\0"... , 832) = 832
read(3, "nodev\tsysfs\nodev\trootfs\nodev\tr"... , 1024) = 390
read(3, "", 1024) = 0
@#0.txt      gg      log.txt      out      test2.c      userlist
a.out        gg.c      maggi      Pictures  test4        uu.c
cc.c         hello.sh  maggi.txt  pp.c      testing      uu.sh
debug.sh     he.sh    magi       Public    testing1     val.c
Desktop      hh2.c    magii.txt  rr.c      test.txt     vamsi
Documents    hh.c     megha.txt  rr.sh     try1.c       Videos
Downloads    ii.c     meghu     sahi.sh   try1.txt     ww.c
ee.c         ii.sh    meghu1    she.sh    try2.txt     yy
ex2         lin.c    meghu.txt shreya    try3.txt     yy.c
example      linux.c  mm.c      ss.sh     try.txt      yyy
file1.txt    linux.sh Music     Templates  tt
file.c       lirt.txt myip      tes       tt.c
filename.txt lisst.txt new.c     test      tt.sh
file.txt     list     num.txt   test1     tt.txt
gcd.c        list.txt oo.c      test2     '.txt'
+++ exited with 0 +++

```

```

linuxmint@linuxmint-VirtualBox:~$ gedit hh.c
linuxmint@linuxmint-VirtualBox:~$ gcc -g hh.c -o test
hh.c:4:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^~~~
hh.c: In function 'main':
hh.c:9:11: warning: initialization makes integer from pointer without a cast [-Wint-conversion]
    char buff="I love myself";
          ^~~~~~
hh.c:10:5: warning: implicit declaration of function 'write' [-Wimplicit-function-declaration]
    ret=write(fd,buff,sizeof(buff));
        ^~~~~~
linuxmint@linuxmint-VirtualBox:~$ strace -e trace=write ./test
write(-1, 0x7c, 1) = -1 EBADF (Bad file descriptor)
+++ exited with 0 +++

```

Valgrind:

It is a memory mismanagement detector. It shows you memory leaks, deallocation errors, etc. Actually, Valgrind is a wrapper around a collection of tools that do many other things (e.g., cache profiling); however, here we focus on the default tool, memcheck. Memcheck can detect:

- Use of uninitialised memory
- Reading/writing memory after it has been free'd
- Reading/writing off the end of malloc'd blocks
- Reading/writing inappropriate areas on the stack
- Memory leaks -- where pointers to malloc'd blocks are lost forever
- Mismatched use of malloc/new/new [] vs free/delete/delete []
- Overlapping src and dst pointers in memcpy() and related functions

Example:

```

#include <stdio.h>
#include <stdlib.h>

int main(void)
{
char *p = malloc(1);
    *p = 'a';

char c = *p;

printf("\n [%c]\n",c);

free(p);
    c = *p;
return 0;
}

```

Output:

```

linuxmint@linuxmint-VirtualBox:~$ gedit val.c
linuxmint@linuxmint-VirtualBox:~$ gcc -g val.c -o test
linuxmint@linuxmint-VirtualBox:~$ valgrind --tool=memcheck --leak-check=yes ./test
==2365== Memcheck, a memory error detector
==2365== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2365== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==2365== Command: ./test
==2365==

[a]
==2365== Invalid read of size 1
==2365==    at 0x108728: main (val.c:14)
==2365==    Address 0x522d040 is 0 bytes inside a block of size 1 free'd
==2365==    at 0x4C30D3B: free (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==2365==    by 0x108723: main (val.c:13)
==2365==    Block was alloc'd at
==2365==    at 0x4C2FB0F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==2365==    by 0x1086EB: main (val.c:6)
==2365==
==2365==
==2365== HEAP SUMMARY:
==2365==    in use at exit: 0 bytes in 0 blocks
==2365==    total heap usage: 2 allocs, 2 frees, 1,025 bytes allocated
==2365==
==2365== All heap blocks were freed -- no leaks are possible
==2365==
==2365== For counts of detected and suppressed errors, rerun with: -v
==2365== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
linuxmint@linuxmint-VirtualBox:~$ █

```

The error summary is provided with 1 error because of calling the pointer after free function of that pointer.

Corrected code:

```

#include <stdio.h>

#include <stdlib.h>

int main(void)

```

```

{
char *p = malloc(1);

    *p = 'a';

char c = *p;

printf("\n [%c]\n",c);

free(p);

return 0;

}

```

Output:

```

linuxmint@linuxmint-VirtualBox:~$ gedit val.c
^C
linuxmint@linuxmint-VirtualBox:~$ gcc -g val.c -o test
linuxmint@linuxmint-VirtualBox:~$ valgrind --tool=memcheck --leak-check=yes ./test
==2385== Memcheck, a memory error detector
==2385== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2385== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==2385== Command: ./test
==2385==

[a]
==2385==
==2385== HEAP SUMMARY:
==2385==    in use at exit: 0 bytes in 0 blocks
==2385==   total heap usage: 2 allocs, 2 frees, 1,025 bytes allocated
==2385==
==2385== All heap blocks were freed -- no leaks are possible
==2385==
==2385== For counts of detected and suppressed errors, rerun with: -v
==2385== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
linuxmint@linuxmint-VirtualBox:~$

```

Example:2

Code:

```

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

```



```

char *p;

char c = *p;

printf("\n [%c]\n",c);

return 0;

}

```

Output:

```

linuxmint@linuxmint-VirtualBox:~$ gedit val.c
linuxmint@linuxmint-VirtualBox:~$ gcc -g val.c -o test
linuxmint@linuxmint-VirtualBox:~$ valgrind --tool=memcheck --leak-check=yes ./test
==2399== Memcheck, a memory error detector
==2399== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2399== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==2399== Command: ./test
==2399==
==2399== Use of uninitialised value of size 8
==2399==    at 0x108656: main (val.c:8)
==2399==
==2399== Invalid read of size 1
==2399==    at 0x108656: main (val.c:8)
==2399== Address 0x0 is not stack'd, malloc'd or (recently) free'd
==2399==
==2399==
==2399== Process terminating with default action of signal 11 (SIGSEGV)
==2399== Access not within mapped region at address 0x0
==2399==    at 0x108656: main (val.c:8)
==2399== If you believe this happened as a result of a stack
==2399== overflow in your program's main thread (unlikely but
==2399== possible), you can try to increase the size of the
==2399== main thread stack using the --main-stacksize= flag.
==2399== The main thread stack size used in this run was 8388608.
==2399==
==2399== HEAP SUMMARY:
==2399==    in use at exit: 0 bytes in 0 blocks
==2399==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==2399==
==2399== All heap blocks were freed -- no leaks are possible
==2399==
==2399== For counts of detected and suppressed errors, rerun with: -v
==2399== Use --track-origins=yes to see where uninitialised values come from
==2399== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
Segmentation fault (core dumped)
linuxmint@linuxmint-VirtualBox:~$

```

Here the error is the malloc allocation is not done for the pointer and calling through the character.

