Question 9.1
By instrumenting the binary code of the target program, AFL can track code coverage during execution. Furthermore, AFL uses the feedback obtained from the instrumented binary to mutate input data. This way inputs that lead to the exploration of new code paths are prioritised, which maximizes the effectiveness of the fuzzing process.
Thus, in order for AFL to efficiently and successfully find bugs and vulnerabilities, the binary needs to be instrumented.

```
[*] Validating target binary...

[-] Looks like the target binary is not instrumented! The fuzzer depends on
    compile-time instrumentation to isolate interesting test cases while
    mutating the input data. For more information, and for tips on how to
    instrument binaries, please see /usr/share/doc/afl/README.

    When source code is not available, you may be able to leverage QEMU
    mode support. Consult the README for tips on how to enable this.
    (It is also possible to use afl-fuzz as a traditional, "dumb" fuzzer.
    For that, you can use the -n option - but expect much worse results.)

[-] PROGRAM ABORT : No instrumentation detected
         Location : check_binary(), afl-fuzz.c:6843
```

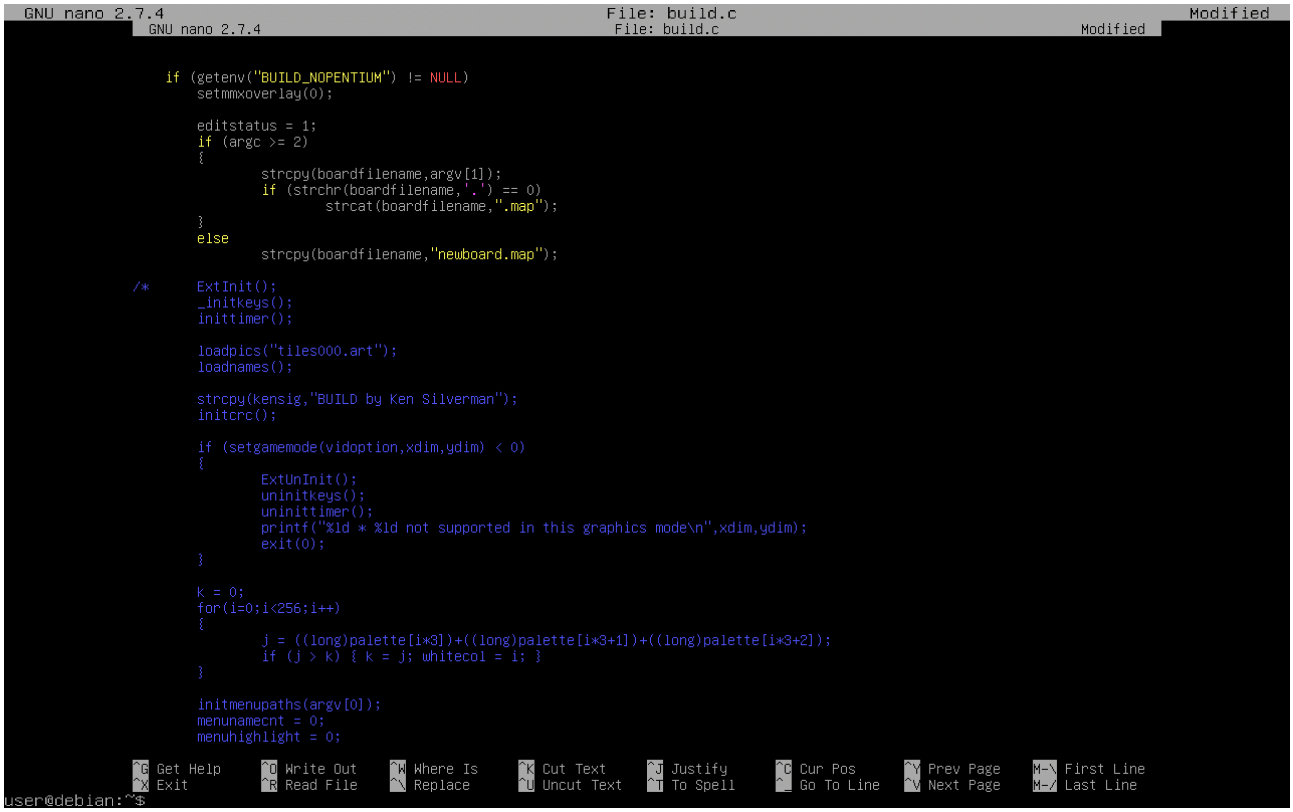
Question 9.2
AFL still “fails" because it does nothing. It fails to start fuzzing and find crashes.

```
[-]  SYSTEM ERROR : Unable to create '.build.afl/generated/board.map'
    Stop location : write_to_testcase(), afl-fuzz.c:2469
       OS message : No such file or directory
user@debian:~/buildengine$ _
```

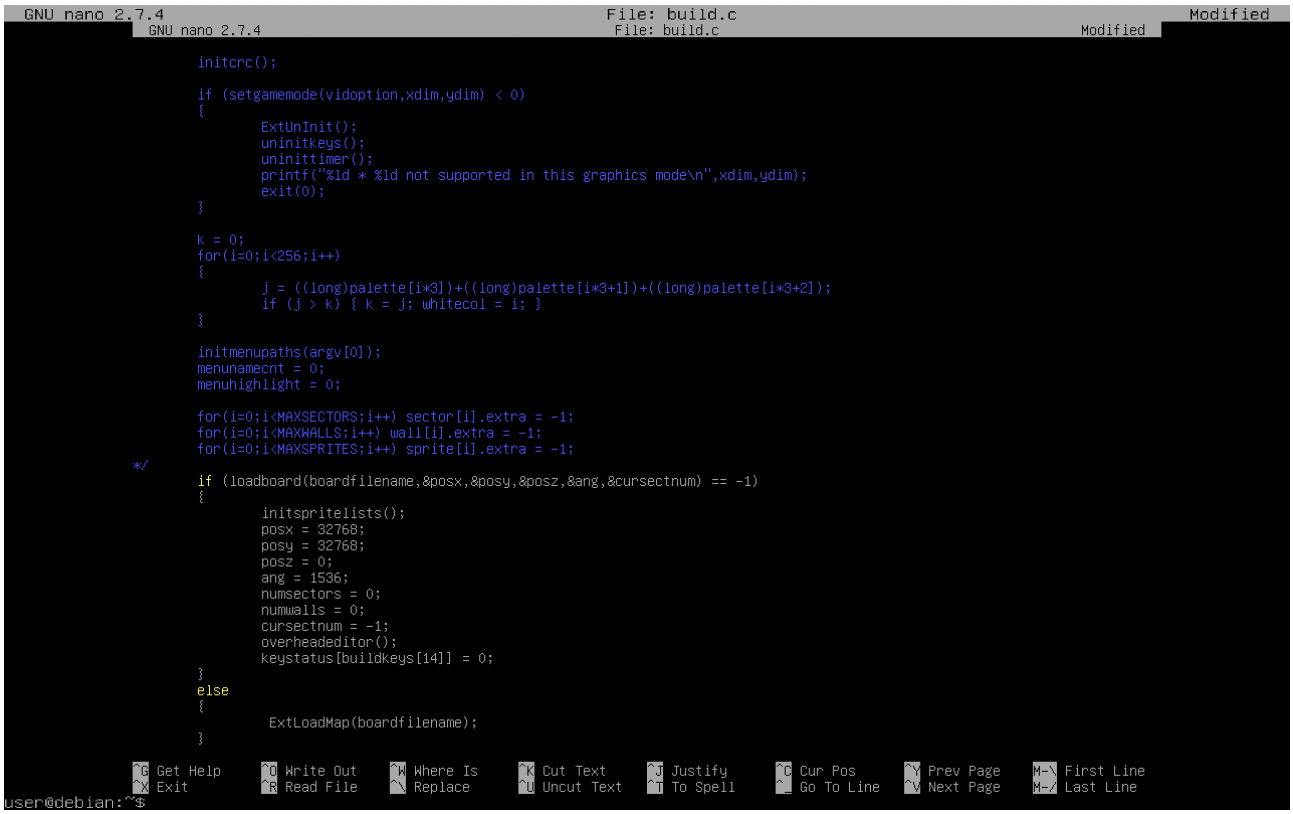
Question 9.3
We remove by commenting out the following:
_platform_init(argc, argv, "BUILD editor by Ken Silverman", “BUILD”);

```
 GNU nano 2.7.4                    File: build.c                         Modified
 GNU nano 2.7.4                    File: build.c                         Modified

                                        case 1: case 2: ExtEditSectorData(searchsector); break;
                                        case 0: case 4: ExtEditWallData(searchwall); break;
                                        case 3: ExtEditSpriteData(searchwall); break;
                                }
                                keystatus[0x41] = 0, keystatus[0x42] = 0;
                        }

                }
                if (keystatus[buildkeys[14]] > 0)  /* Enter */
                {
                        overheadeditor();
                        keystatus[buildkeys[14]] = 0;
                }
        }


        int main(int argc,char **argv)
        {
                char ch, quitflag;
                long i, j, k;

          _// _platform_init(argc, argv, "BUILD editor by Ken Silverman", "BUILD");

           if (getenv("BUILD_NOPENTIUM") != NULL)
                setmmxoverlay(0);

                editstatus = 1;
                if (argc >= 2)
                {
                        strcpy(boardfilename,argv[1]);
                        if (strchr(boardfilename,'.') == 0)
                                strcat(boardfilename,".map");
                }
                else
                        strcpy(boardfilename,"newboard.map");

        /*      ExtInit();
                _initkeys();
                inittimer();

                loadpics("tiles000.art");
                loadnames();


 ^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  ^Y Prev Page  M-\ First Line
 ^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line ^V Next Page  M-/ Last Line
user@debian:~$
```

As well as, we remove by commenting out everything shown in blue from the following screenshots:

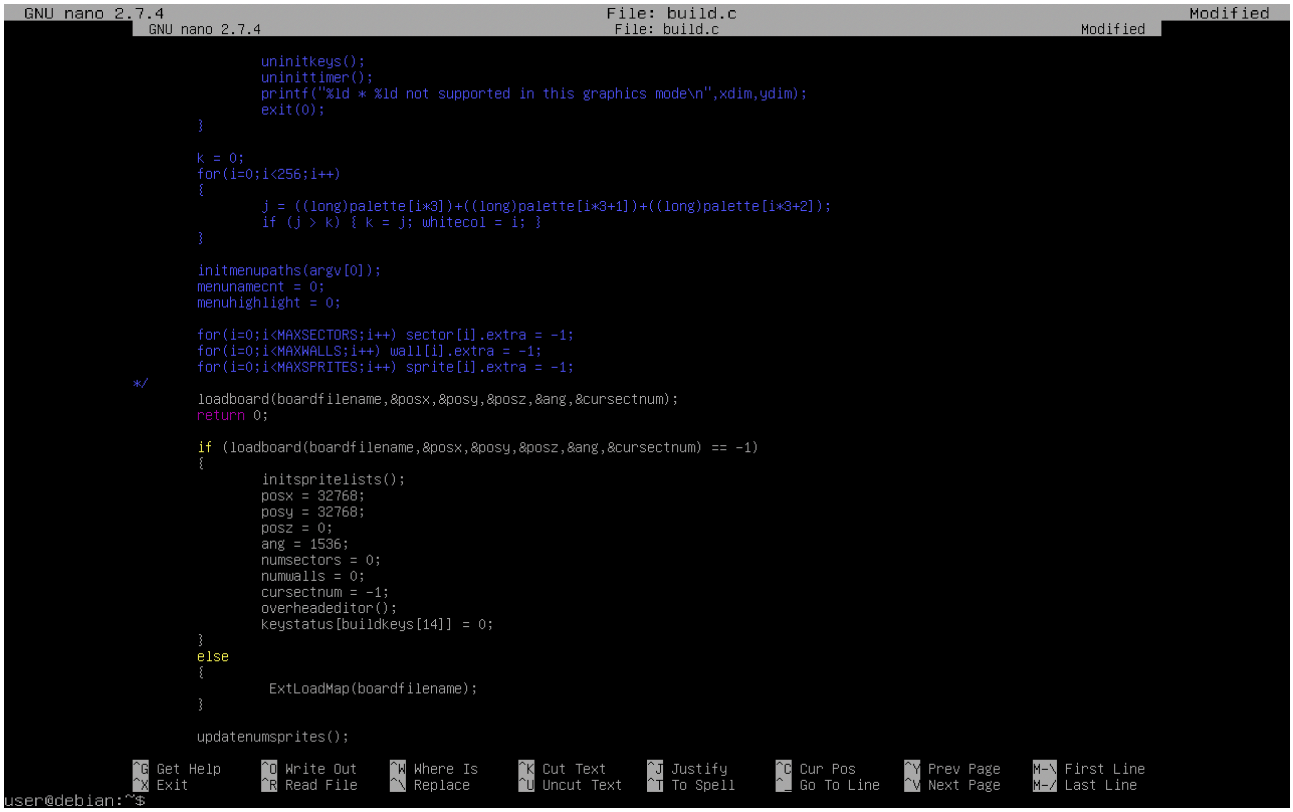


```
  GNU nano 2.7.4                          File: build.c                                 Modified
  GNU nano 2.7.4                          File: build.c                                 Modified


    if (getenv("BUILD_NOPENTIUM") != NULL)
        setmmxoverlay(0);

        editstatus = 1;
        if (argc >= 2)
        {
                strcpy(boardfilename,argv[1]);
                if (strchr(boardfilename,'.') == 0)
                        strcat(boardfilename,".map");
        }
        else
                strcpy(boardfilename,"newboard.map");

/*      ExtInit();
        _initkeys();
        inittimer();

        loadpics("tiles000.art");
        loadnames();

        strcpy(kensig,"BUILD by Ken Silverman");
        initcrc();

        if (setgamemode(vidoption,xdim,ydim) < 0)
        {
                ExtUnInit();
                uninitkeys();
                uninittimer();
                printf("%ld * %ld not supported in this graphics mode\n",xdim,ydim);
                exit(0);
        }

        k = 0;
        for(i=0;i<256;i++)
        {
                j = ((long)palette[i*3])+((long)palette[i*3+1])+((long)palette[i*3+2]);
                if (j > k) { k = j; whitecol = i; }
        }

        initmenupaths(argv[0]);
        menunamecnt = 0;
        menuhighlight = 0;

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  ^Y Prev Page  M-\ First Line
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell ^_ Go To Line ^V Next Page  M-/ Last Line
user@debian:~$
```

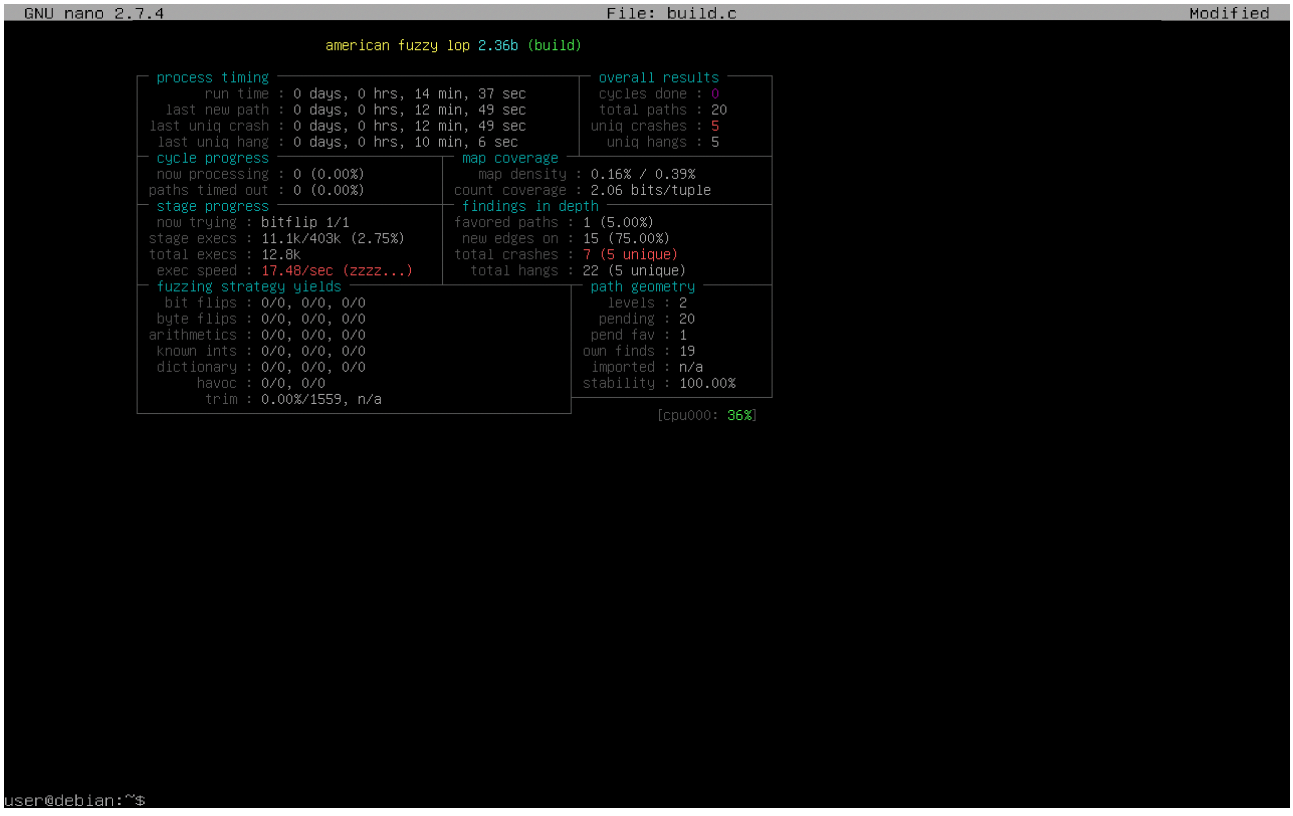


```
  GNU nano 2.7.4                          File: build.c                                 Modified
  GNU nano 2.7.4                          File: build.c                                 Modified

        initcrc();

        if (setgamemode(vidoption,xdim,ydim) < 0)
        {
                ExtUnInit();
                uninitkeys();
                uninittimer();
                printf("%ld * %ld not supported in this graphics mode\n",xdim,ydim);
                exit(0);
        }

        k = 0;
        for(i=0;i<256;i++)
        {
                j = ((long)palette[i*3])+((long)palette[i*3+1])+((long)palette[i*3+2]);
                if (j > k) { k = j; whitecol = i; }
        }

        initmenupaths(argv[0]);
        menunamecnt = 0;
        menuhighlight = 0;

        for(i=0;i<MAXSECTORS;i++) sector[i].extra = -1;
        for(i=0;i<MAXWALLS;i++) wall[i].extra = -1;
        for(i=0;i<MAXSPRITES;i++) sprite[i].extra = -1;
*/
        if (loadboard(boardfilename,&posx,&posy,&posz,&ang,&cursectnum) == -1)
        {
                initspritelists();
                posx = 32768;
                posy = 32768;
                posz = 0;
                ang = 1536;
                numsectors = 0;
                numwalls = 0;
                cursectnum = -1;
                overheadeditor();
                keystatus[buildkeys[14]] = 0;
        }
        else
        {
                 ExtLoadMap(boardfilename);
        }

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  ^Y Prev Page  M-\ First Line
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell ^_ Go To Line ^V Next Page  M-/ Last Line
user@debian:~$
```

Question 9.4
We update the code to quit the program once the code parsing the map file has finished by putting “return 0;” after the call to
loadboard(boardfilename,&posx,&posy,&posz,&ang,&cursectnum);



Question 9.5
We let AFL run for 14 minutes. It generates 5 unique crashes and 5 unique hangs:



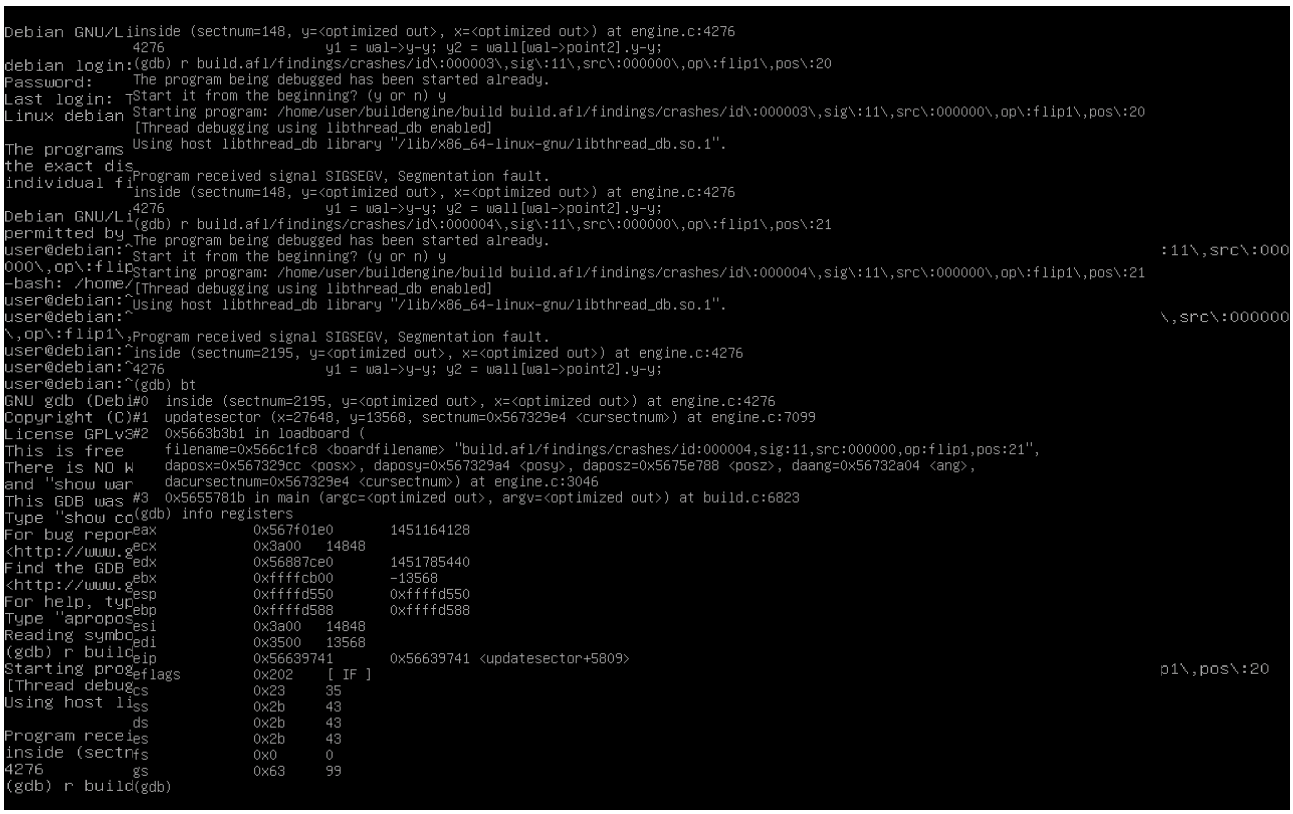
Question 9.6

```
 GNU nano 2.7.4                                File: build.c                                                  Modified
[1]+  Terminated              ./script.sh
user@debian:~/buildengine$ ls
a.asm          bstub.o        buildperl.c  dos_drvr.c       game.c         multi.c             README-dosbins.txt    unix_compat.c
a.c            build          buildperl.h  doxyfile         game.o         names.h             README-win32bins.txt  unix_compat.h
a_gnu.c        build2.txt     build.sh     engine.c         game.pl        nsnoal.map          script.sh             unix_compat.o
a.h            build.afl      build.txt    Engine.dsp       k.asm          nukeland.map        sdl_driver.c          utils
a_nasm.asm     build.c        cache1d.c    engine.h         kdmeng.c       platform.h          sdl_driver.o          ves2.h
a.o            buildgl.c      cache1d.h    engine.o         kensig.map     pragmas.c           setup.dat             win32_compat.h
ascboard.map   buildgl.h      cache1d.o    engine_protos.h  Makefile       pragmas_gnu.c       sreadme.txt
a_visualc.c    build.h        CHANGELOG    Engine.vcproj    Makefile.dos   pragmas.h           stuff.dat
boards.map     buildinf.txt   CONTRIB      evilal.map       Makefile.w32   pragmas.o           test.map
bstub.c        BUILDLIC.TXT   display.h    FILEID.DIZ       mmulti.c       pragmas_visualc.c   tests
bstub.h        build.o        doscmpat.h   game             mmulti.o       README              TODO
user@debian:~/buildengine$ cd
user@debian:~$ ls
automount.sh  buildengine  CFI  UFO
user@debian:~$ cd buildengine
user@debian:~/buildengine$ cd build.afl/findings/crashes
user@debian:~/buildengine/build.afl/findings/crashes$ ls
id:000000,sig:11,src:000000,op:flip1,pos:20  id:000003,sig:11,src:000000,op:flip1,pos:20
id:000001,sig:11,src:000000,op:flip1,pos:20  id:000004,sig:11,src:000000,op:flip1,pos:21
id:000002,sig:11,src:000000,op:flip1,pos:20  README.txt
user@debian:~/buildengine/build.afl/findings/crashes$
```

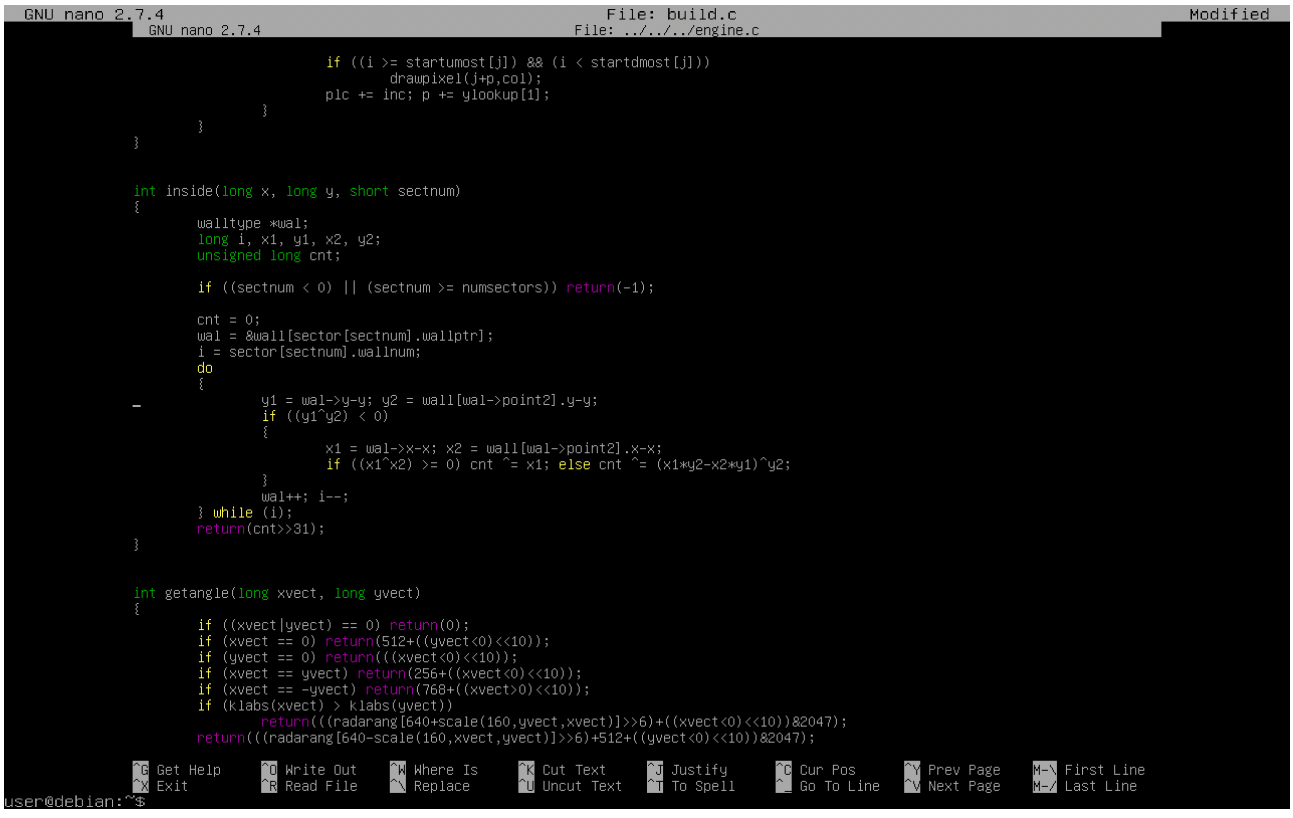
```
 GNU nano 2.7.4                                File: build.c                                                  Modified
Segmentation fault
user@debian:~/buildengine$ gdb -q ../../build^C
user@debian:~/buildengine$ cd build.afl/findings/crashes/
user@debian:~/buildengine/build.afl/findings/crashes$ gdb -q ../../../build
Reading symbols from ../../../build...done.
(gdb) r id:000000,sig:11,src:000000,op:flip1,pos:20
Starting program: /home/user/buildengine/build id:000000,sig:11,src:000000,op:flip1,pos:20
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[Inferior 1 (process 25389) exited normally]
(gdb) quit
user@debian:~/buildengine/build.afl/findings/crashes$ ../../../build  build.afl/findings/crashes/id\:000000\,sig\:11\,src\:00000
0\,op\:flip1\,pos\:20
user@debian:~/buildengine/build.afl/findings/crashes$ ../../../build  id\:000000\,sig\:11\,src\:000000\,op\:flip1\,pos\:20
user@debian:~/buildengine/build.afl/findings/crashes$ ../../../build  id\:000000\,sig\:11\,src\:000000\,op\:flip1\,pos\:20
user@debian:~/buildengine/build.afl/findings/crashes$ ../../../build  id\:000000\,sig\:11\,src\:000000\,op\:flip1\,pos\:20 ^C
user@debian:~/buildengine/build.afl/findings/crashes$ ../../../build
user@debian:~/buildengine/build.afl/findings/crashes$ ../../../build id\:000000\,sig\:11\,src\:000000\,op\:flip1\,pos\:20
user@debian:~/buildengine/build.afl/findings/crashes$ cd ..
user@debian:~/buildengine/build.afl/findings$ cd ..
user@debian:~/buildengine/build.afl$ cd ..
user@debian:~/buildengine$ ./build  build.afl/findings/crashes/id\:000000\,sig\:11\,src\:000000\,op\:flip1\,pos\:20
Segmentation fault
user@debian:~/buildengine$ gdb ./build
GNU gdb (Debian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 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 ./build...done.
(gdb) r build.afl/findings/crashes/id\:000000\,sig\:11\,src\:000000\,op\:flip1\,pos\:20
Starting program: /home/user/buildengine/build build.afl/findings/crashes/id\:000000\,sig\:11\,src\:000000\,op\:flip1\,pos\:20
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Program received signal SIGSEGV, Segmentation fault.
inside (sectnum=179, y=<optimized out>, x=<optimized out>) at engine.c:4276
4276                    y1 = wal->y-y; y2 = wall[wal->point2].y-y;
(gdb) _
user@debian:~$
```

```
Debian GNU/Li
debian login:
Password:
Last login: T
Linux debian

The programs
the exact dis
individual fi

Debian GNU/Li
permitted by
user@debian:~
000\,op\:flip
-bash: /home/
user@debian:~
user@debian:~
\,op\:flip1\,
user@debian:~
user@debian:~
GNU gdb (Debi
Copyright (C)
License GPLv3
This is free
There is NO w
and "show war
This GDB was
Type "show co
For bug repor
<http://www.g
Find the GDB
<http://www.g
For help, typ
Type "apropos
Reading symbo
(gdb) r build
Starting prog
[Thread debug
Using host li

Program recei
inside (sectn
4276
(gdb) r build
```

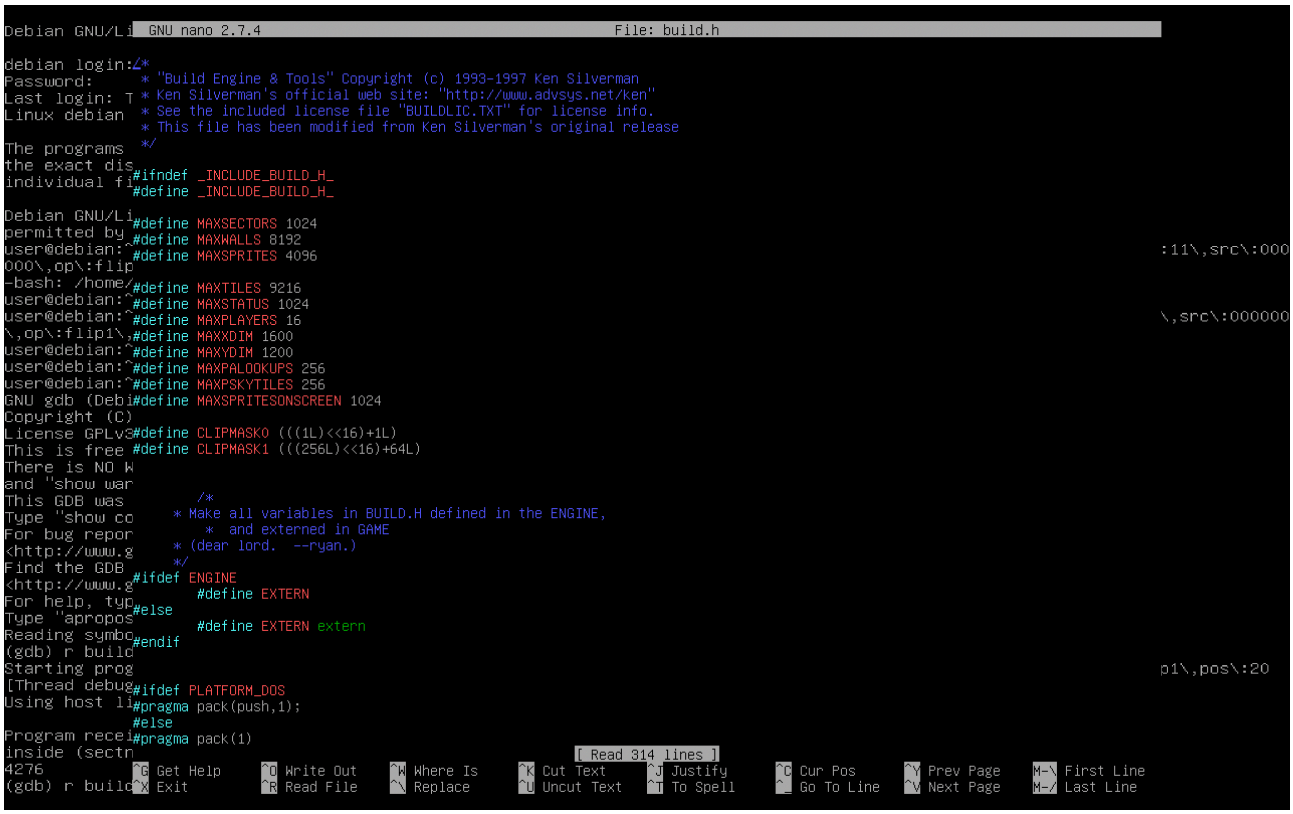
```
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/user/buildengine/build build.afl/findings/crashes/id\:000001\,sig\:11\,src\:000000\,op\:flip1\,pos\:20
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Program received signal SIGSEGV, Segmentation fault.
0x5663964c in inside (sectnum=155, y=<optimized out>, x=<optimized out>) at engine.c:4276
4276                    y1 = wal->y-y; y2 = wall[wal->point2].y-y;
(gdb) r build.afl/findings/crashes/id\:000002\,sig\:11\,src\:000000\,op\:flip1\,pos\:20
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/user/buildengine/build build.afl/findings/crashes/id\:000002\,sig\:11\,src\:000000\,op\:flip1\,pos\:20
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Program received signal SIGSEGV, Segmentation fault.
inside (sectnum=148, y=<optimized out>, x=<optimized out>) at engine.c:4276
4276                    y1 = wal->y-y; y2 = wall[wal->point2].y-y;
(gdb) r build.afl/findings/crashes/id\:000003\,sig\:11\,src\:000000\,op\:flip1\,pos\:20
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/user/buildengine/build build.afl/findings/crashes/id\:000003\,sig\:11\,src\:000000\,op\:flip1\,pos\:20
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Program received signal SIGSEGV, Segmentation fault.
inside (sectnum=148, y=<optimized out>, x=<optimized out>) at engine.c:4276
4276                    y1 = wal->y-y; y2 = wall[wal->point2].y-y;
(gdb) r build.afl/findings/crashes/id\:000004\,sig\:11\,src\:000000\,op\:flip1\,pos\:21
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/user/buildengine/build build.afl/findings/crashes/id\:000004\,sig\:11\,src\:000000\,op\:flip1\,pos\:21
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Program received signal SIGSEGV, Segmentation fault.
inside (sectnum=2195, y=<optimized out>, x=<optimized out>) at engine.c:4276
4276                    y1 = wal->y-y; y2 = wall[wal->point2].y-y;
(gdb) bt
#0  inside (sectnum=2195, y=<optimized out>, x=<optimized out>) at engine.c:4276
#1  updatesector (x=27648, y=13568, sectnum=0x567329e4 <cursectnum>) at engine.c:7099
#2  0x5663b3b1 in loadboard (
    filename=0x566c1fc8 <boardfilename> "build.afl/findings/crashes/id:000004,sig:11,src:000000,op:flip1,pos:21",
    daposx=0x567329cc <posx>, daposy=0x567329a4 <posy>, daposz=0x5675e788 <posz>, daang=0x56732a04 <ang>,
    dacursectnum=0x567329e4 <cursectnum>) at engine.c:3046
#3  0x5655781b in main (argc=<optimized out>, argv=<optimized out>) at build.c:6823
(gdb) info registers_
```

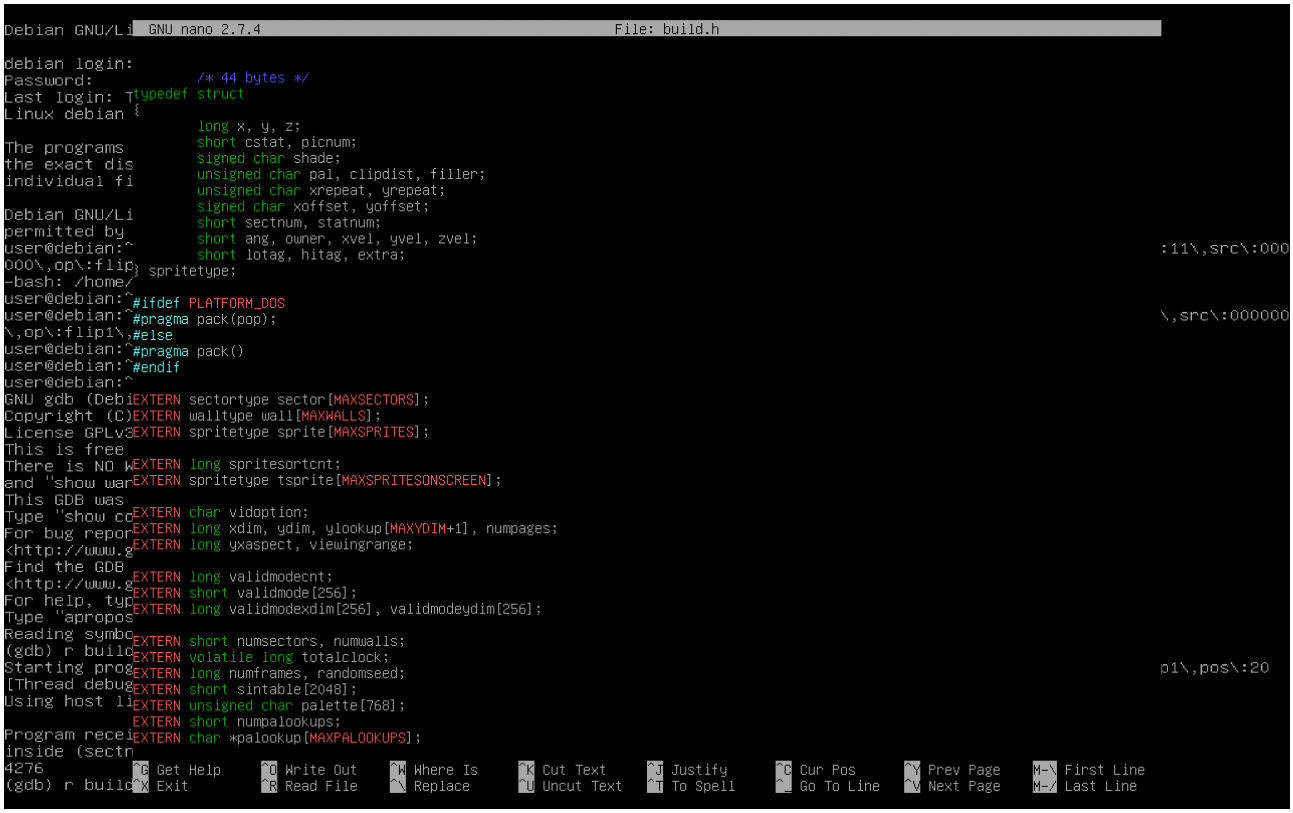


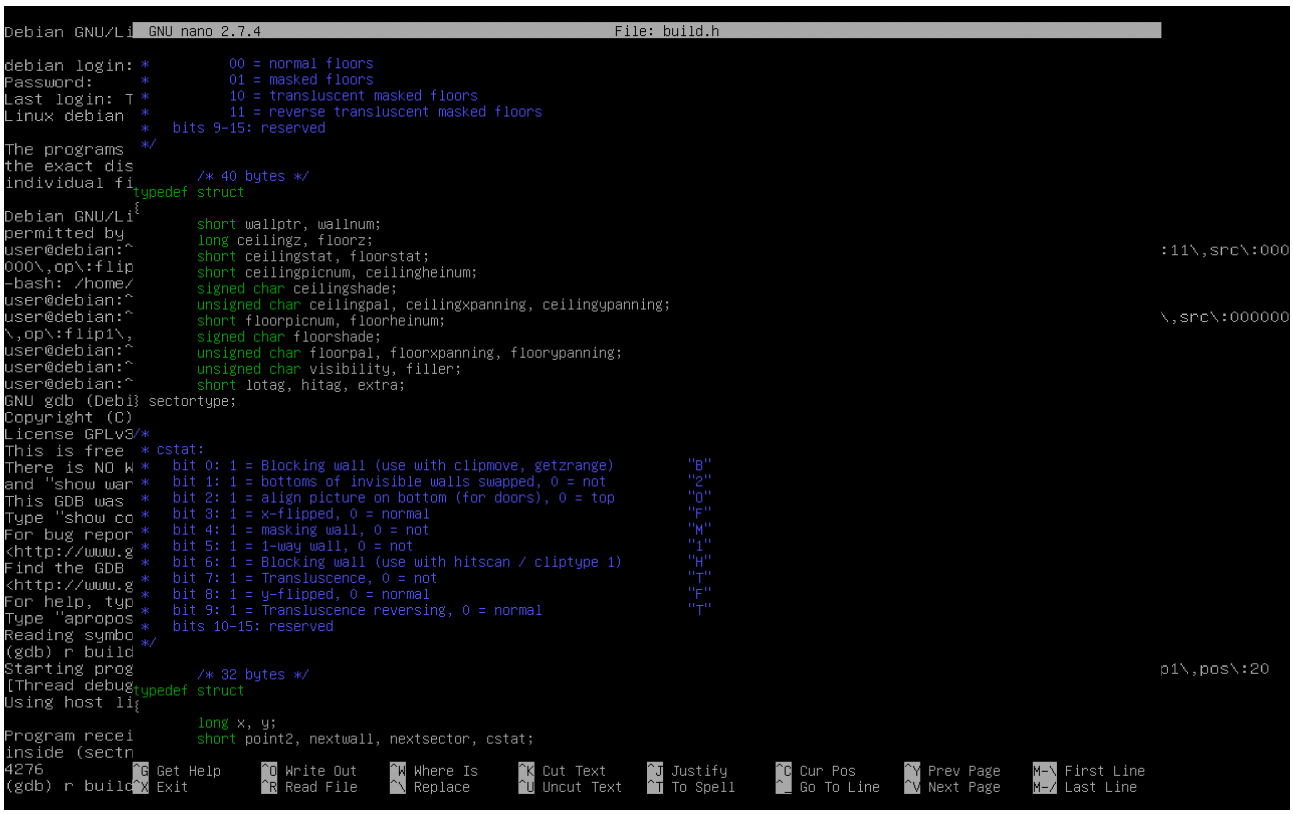
The crash happens at line 4276 of code engine.c, which is found in the following line:

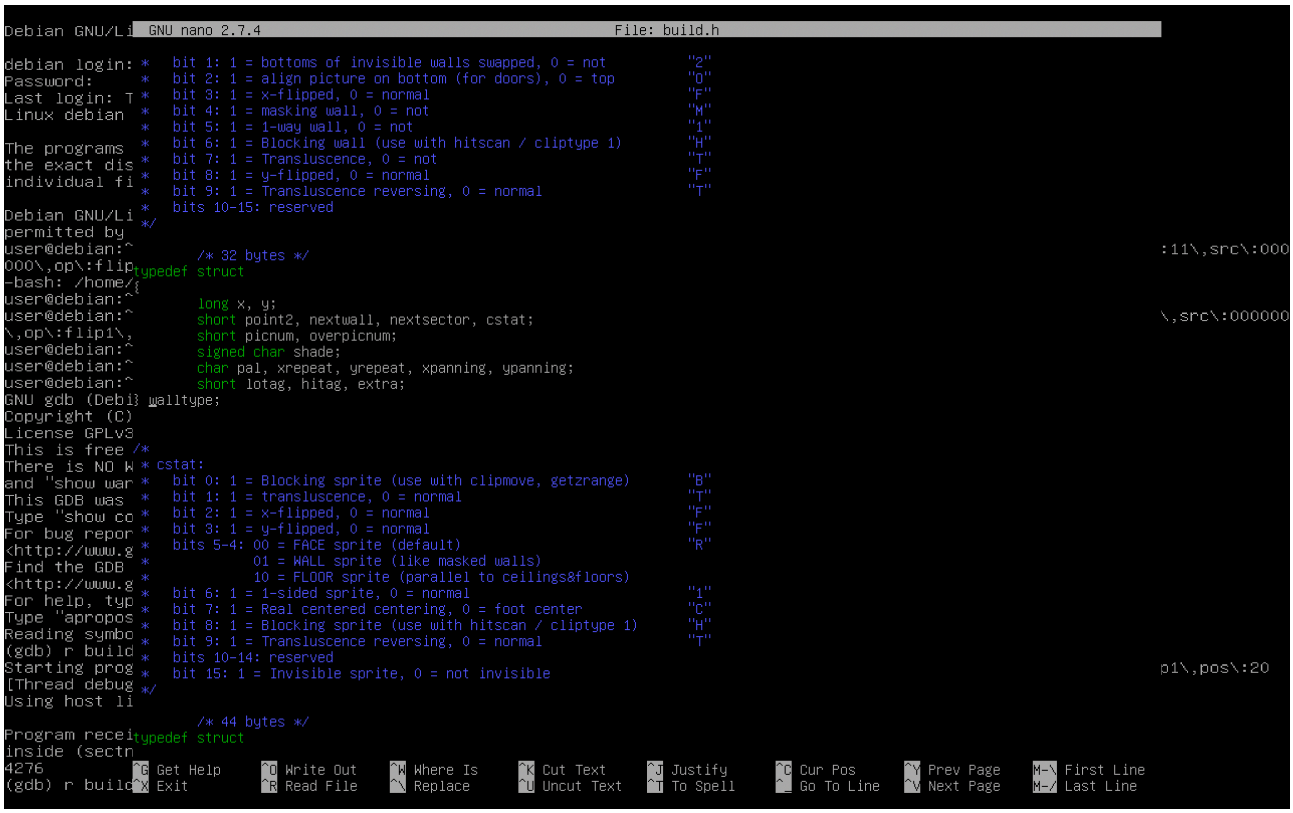


Let's examine the function *inside*. We see it uses variables *sectnum, wallnum, wall,* and *sector*. Let's examine those variables in *build.h*:



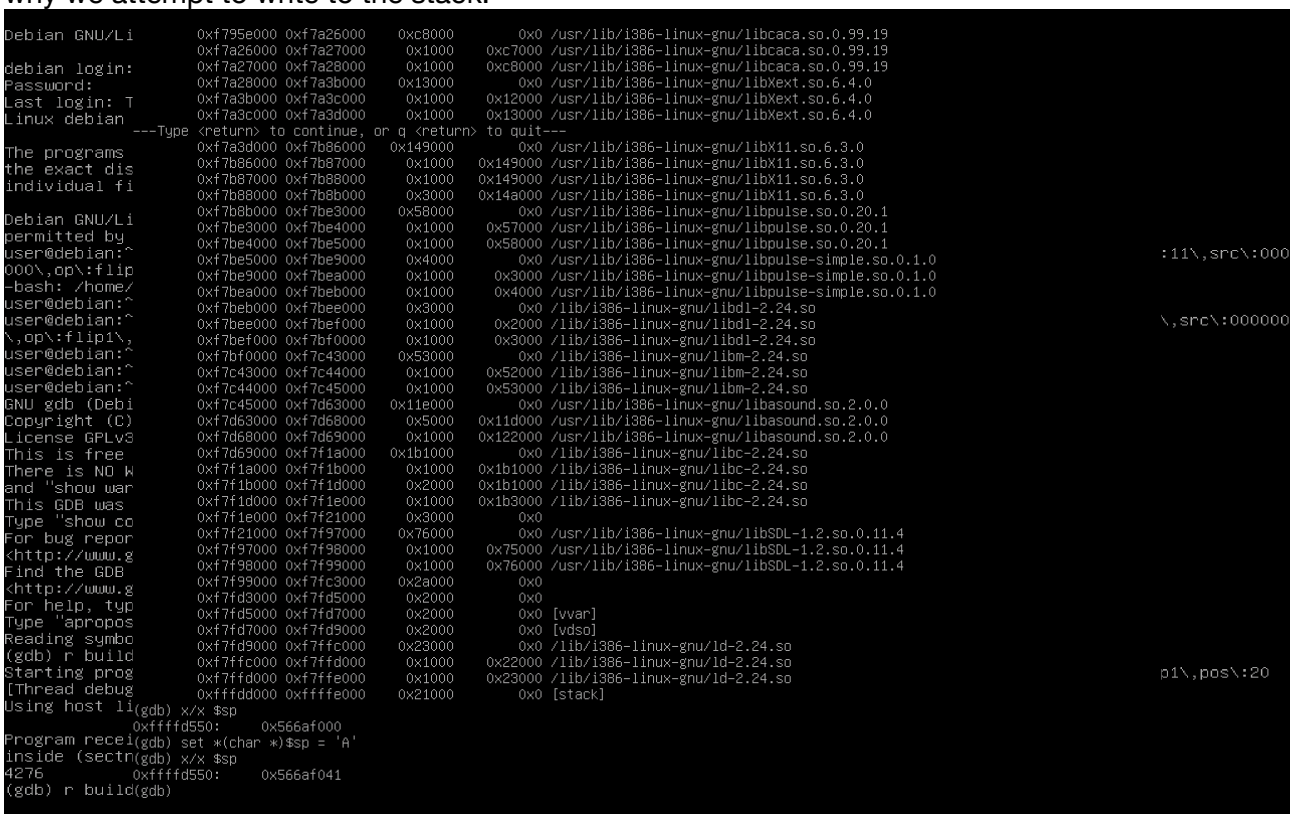






We can see that we have global buffers *sector* and *wall*. The *sector* variable can hold a maximum of 1024 *sector* structures, while a *sector* structure represents 40 bytes. The *wall* variable can hold a maximum of 8192 *wall* structures, while a *wall* structure represents 32 bytes. If an attacker can control the number of bytes to copy to the global buffers, there would be a buffer overflow vulnerability.

Furthermore, we use *(gdb) info proc mappings* to check the permissions of the stack. However, the permissions of the stack are not mentioned explicitly, as seen in the screenshot below. That's why we attempt to write to the stack:



We can see that the stack is writable. Therefore, an attacker can overwrite the stack to redirect the control flow to their own code. Thus, the crash is exploitable. The attacker can generate a map to trigger the overflow to overwrite the return address on the stack. This way, the attacker can execute arbitrary code.