DUE: OCTOBER 8TH

LAB 3: BINARY BOMB LAB

CONTENTS

OBJECTIVE

- Defuse a binary bomb!
- 6 stages to defuse + 1 Hidden stage
- No source code - Use a debugger to track the right input

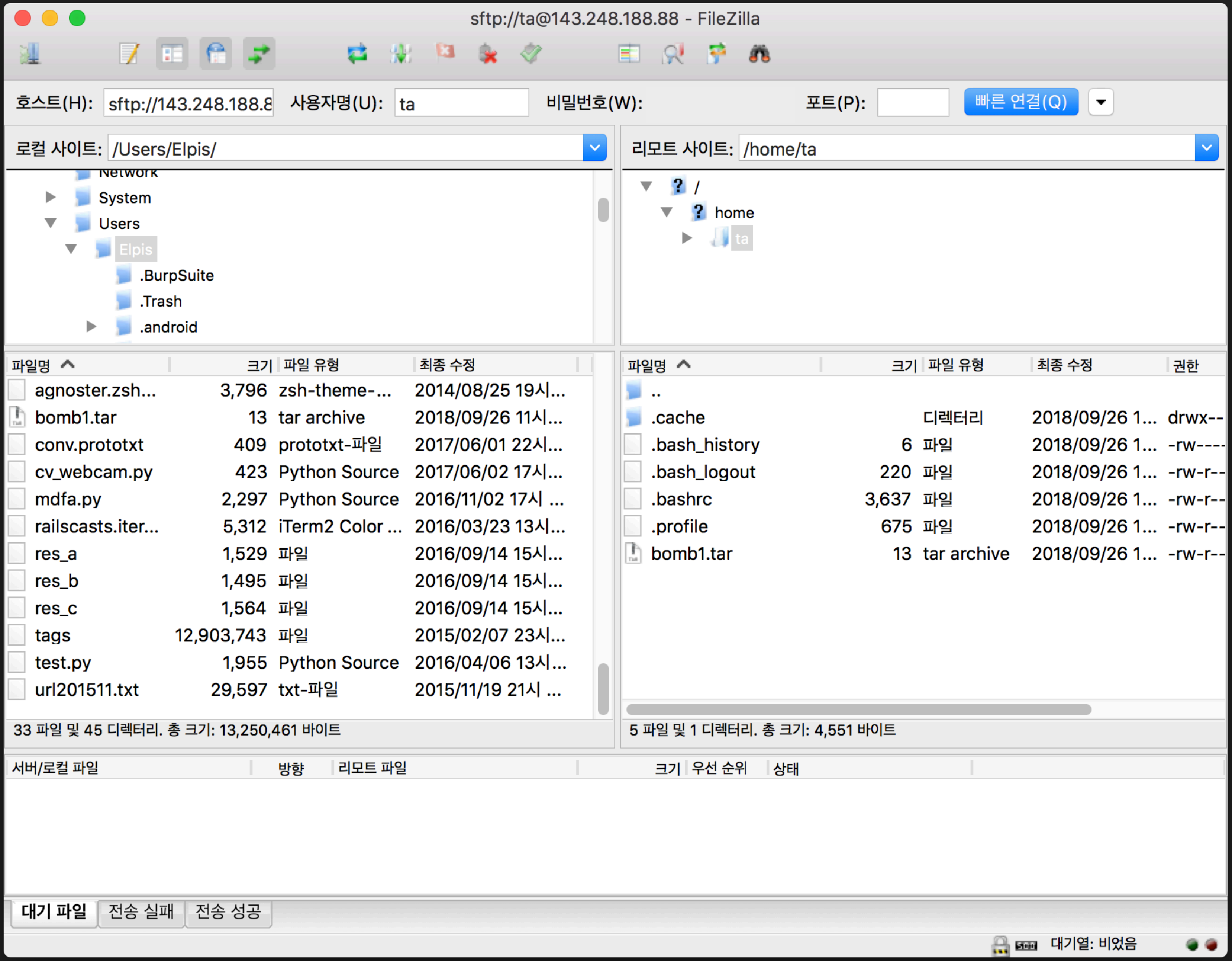
TOOLS - SCP

- Copy a file from a machine to another
- Windows: Use scp clients (WinSCP, FIleZilla, …)
- Mac / Linux: Use scp command built in the shell

TOOLS - SCP

- Copy a file from a machine to another
- Windows: Use scp clients (WinSCP, FIleZilla, …)
  - FileZilla - Convenient SCP Client (Available for Windows / macOS / Linux)
- Example: Move ‘`bomb1.tar`’ from local machine to ‘home folder (~)’ of account ‘`ta`’ in ‘`143.248.188.88`’

TOOLS - SCP



sftp://ta@143.248.188.88 - FileZilla

호스트(H): sftp://143.248.188.8 사용자명(U): ta 비밀번호(W): 포트(P): 빠른 연결(Q)

로컬 사이트: /Users/Elpis/

- Network
- System
- Users
  - Elpis
    - .BurpSuite
    - .Trash
    - .android

리모트 사이트: /home/ta

- /
  - home
    - ta

파일명	크기	파일 유형	최종 수정
agnoster.zsh...	3,796	zsh-theme-...	2014/08/25 19시...
bomb1.tar	13	tar archive	2018/09/26 11시...
conv.prototxt	409	prototxt-파일	2017/06/01 22시...
cv_webcam.py	423	Python Source	2017/06/02 17시...
mdfa.py	2,297	Python Source	2016/11/02 17시 ...
railscasts.iter...	5,312	iTerm2 Color ...	2016/03/23 13시...
res_a	1,529	파일	2016/09/14 15시...
res_b	1,495	파일	2016/09/14 15시...
res_c	1,564	파일	2016/09/14 15시...
tags	12,903,743	파일	2015/02/07 23시...
test.py	1,955	Python Source	2016/04/06 13시...
url201511.txt	29,597	txt-파일	2015/11/19 21시 ...

33 파일 및 45 디렉터리. 총 크기: 13,250,461 바이트

파일명	크기	파일 유형	최종 수정	권한
..				
.cache		디렉터리	2018/09/26 1...	drwx--
.bash_history	6	파일	2018/09/26 1...	-rw----
.bash_logout	220	파일	2018/09/26 1...	-rw-r--
.bashrc	3,637	파일	2018/09/26 1...	-rw-r--
.profile	675	파일	2018/09/26 1...	-rw-r--
bomb1.tar	13	tar archive	2018/09/26 1...	-rw-r--

5 파일 및 1 디렉터리. 총 크기: 4,551 바이트

서버/로컬 파일	방향	리모트 파일	크기	우선 순위	상태

대기 파일 | 전송 실패 | 전송 성공

대기열: 비었음

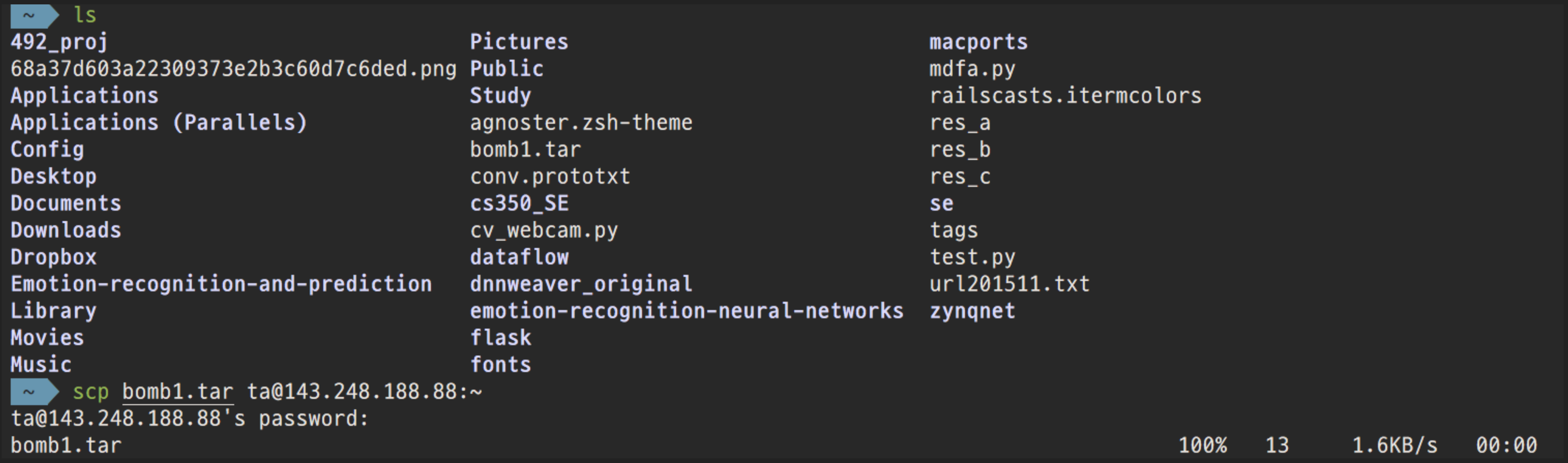
TOOLS - SCP

- Copy a file from a machine to another
- Mac / Linux: Use scp command built in the shell
  - From shell:
  - `scp [Bomb location] [Username]@[Server IP]:[Destination]`
- Example: Move '`bomb1.tar`' from local machine to 'home folder (~)' of account '`ta`' in '`143.248.188.88`'

TOOLS - SCP

- Copy a file from a machine to another
- Mac / Linux: Use scp command built in the shell

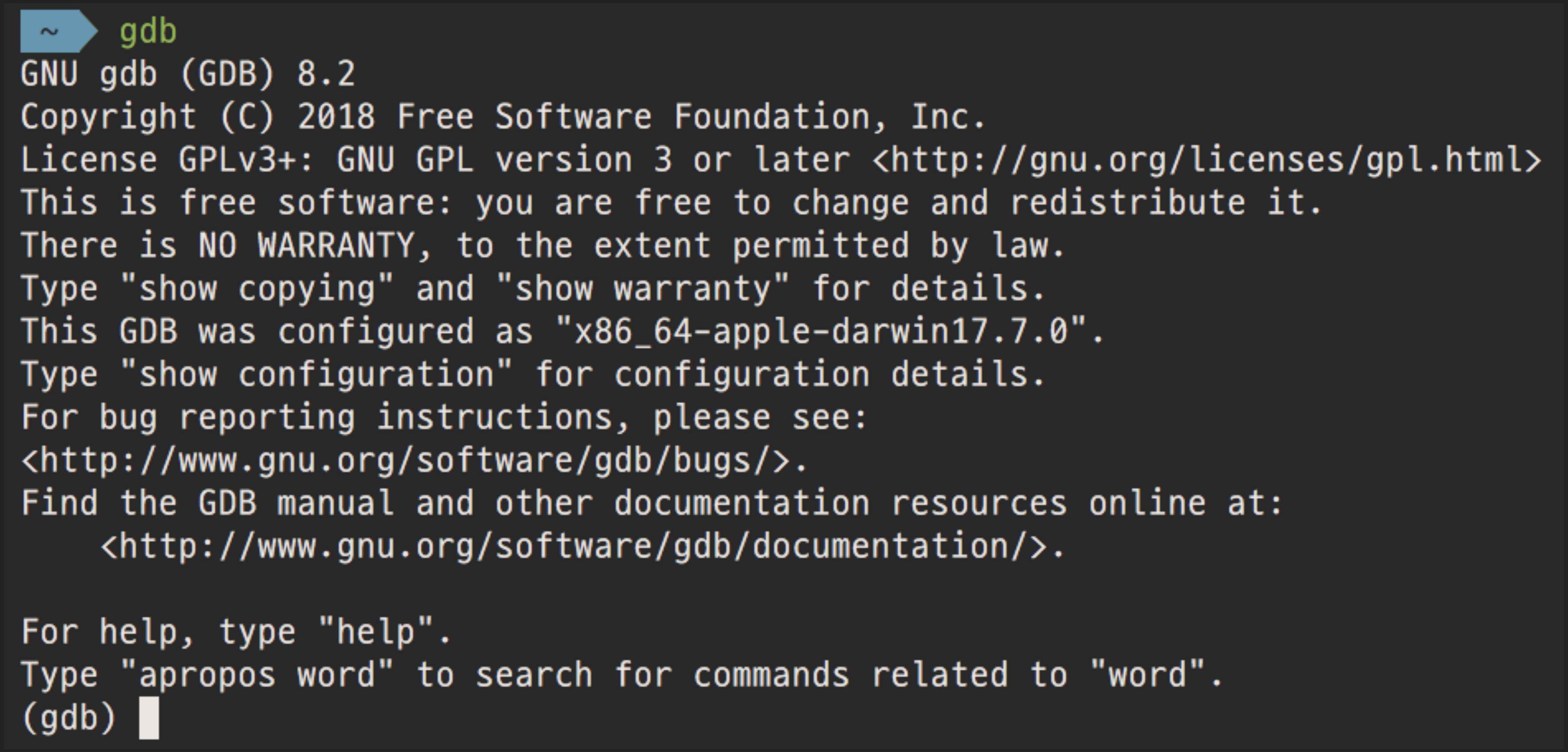
```
scp bomb1.tar ta@143.248.188.88:~
```



```
 ~  ls
492_proj                            Pictures                            macports
68a37d603a22309373e2b3c60d7c6ded.png Public                              mdfa.py
Applications                        Study                               railscasts.itermcolors
Applications (Parallels)            agnoster.zsh-theme                  res_a
Config                              bomb1.tar                           res_b
Desktop                             conv.prototxt                       res_c
Documents                           cs350_SE                            se
Downloads                           cv_webcam.py                        tags
Dropbox                             dataflow                            test.py
Emotion-recognition-and-prediction  dnnweaver_original                  url201511.txt
Library                             emotion-recognition-neural-networks zynqnet
Movies                              flask
Music                               fonts
 ~  scp bomb1.tar ta@143.248.188.88:~
ta@143.248.188.88's password:
bomb1.tar                                                                        100%   13     1.6KB/s   00:00
```

TOOLS - GDB

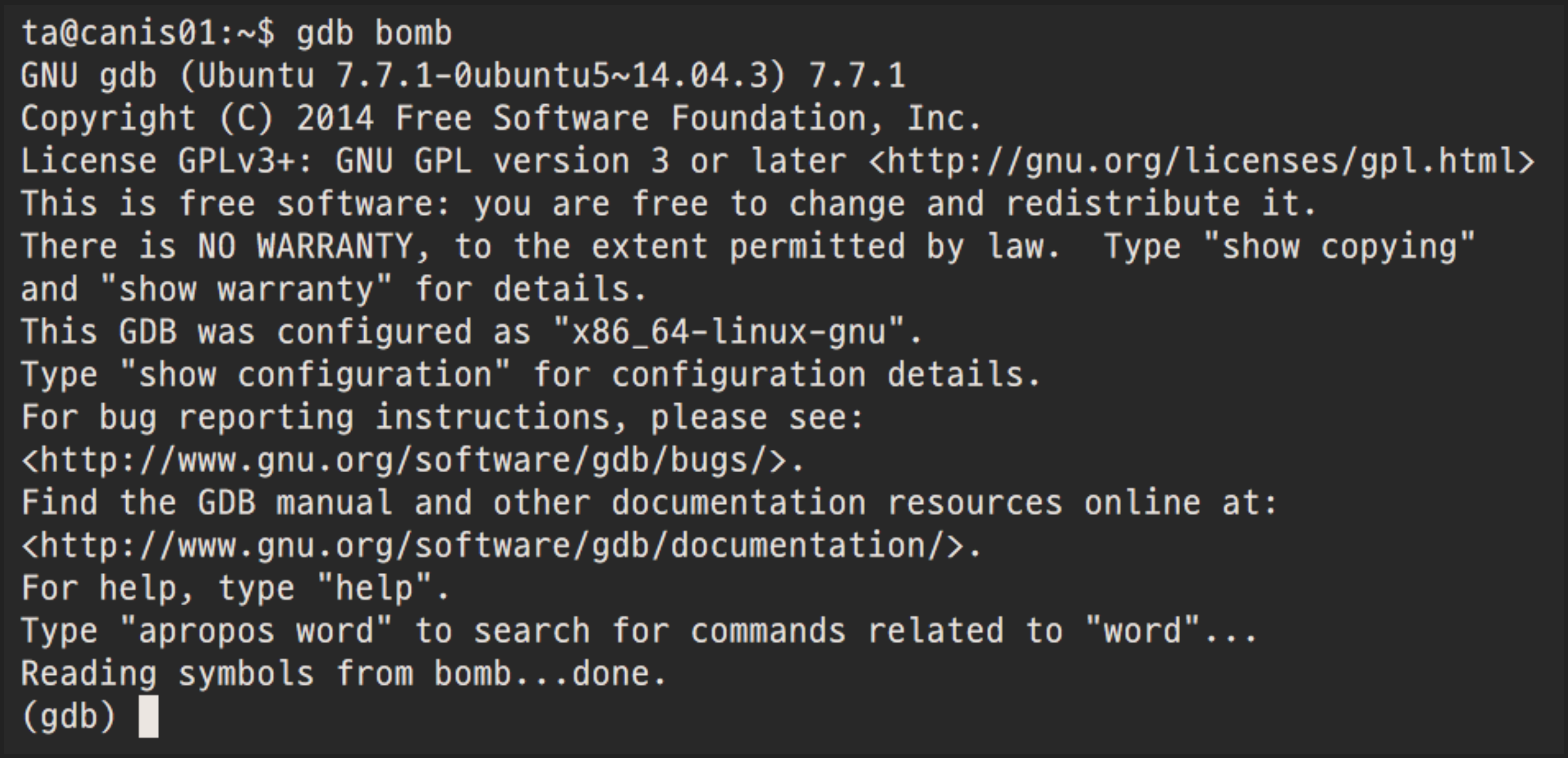
- GNU Debugger
- Shows machine state in real time



```
 ~  gdb
GNU gdb (GDB) 8.2
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-apple-darwin17.7.0".
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".
(gdb) 
```

TOOLS - GDB

- Start GDB
  - `gdb [options] [executable name]`
- Ex) Start GDB with file 'bomb'



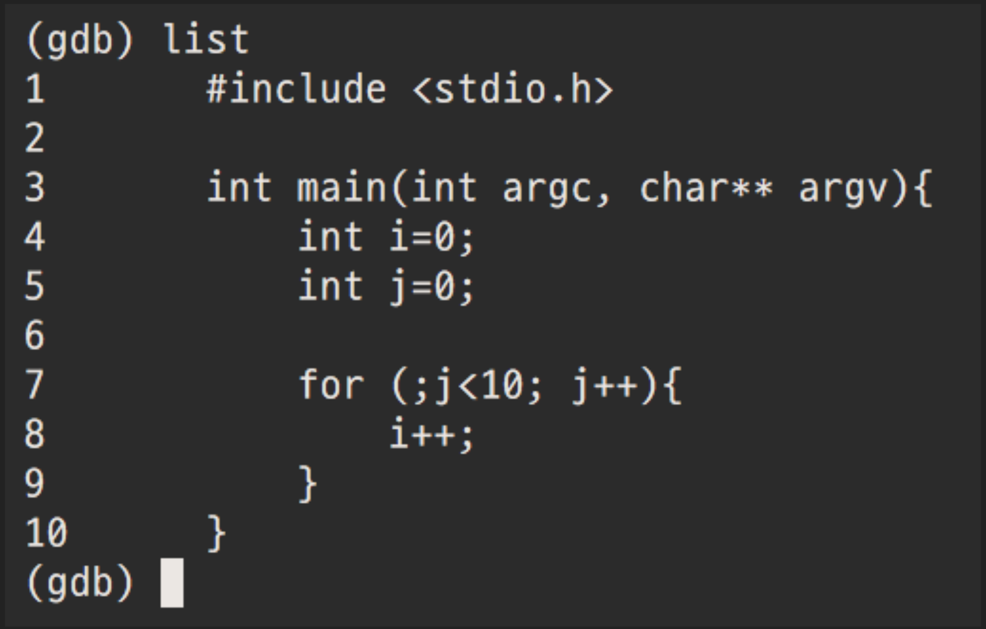
```
ta@canis01:~$ gdb bomb
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.3) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent 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 bomb...done.
(gdb) 
```

TOOLS - GDB

- Show source code
  - `list`
- Only available when binary is compiled with -g flag
- Bomb lab binaries are not compiled with -g flag enabled

TOOLS - GDB

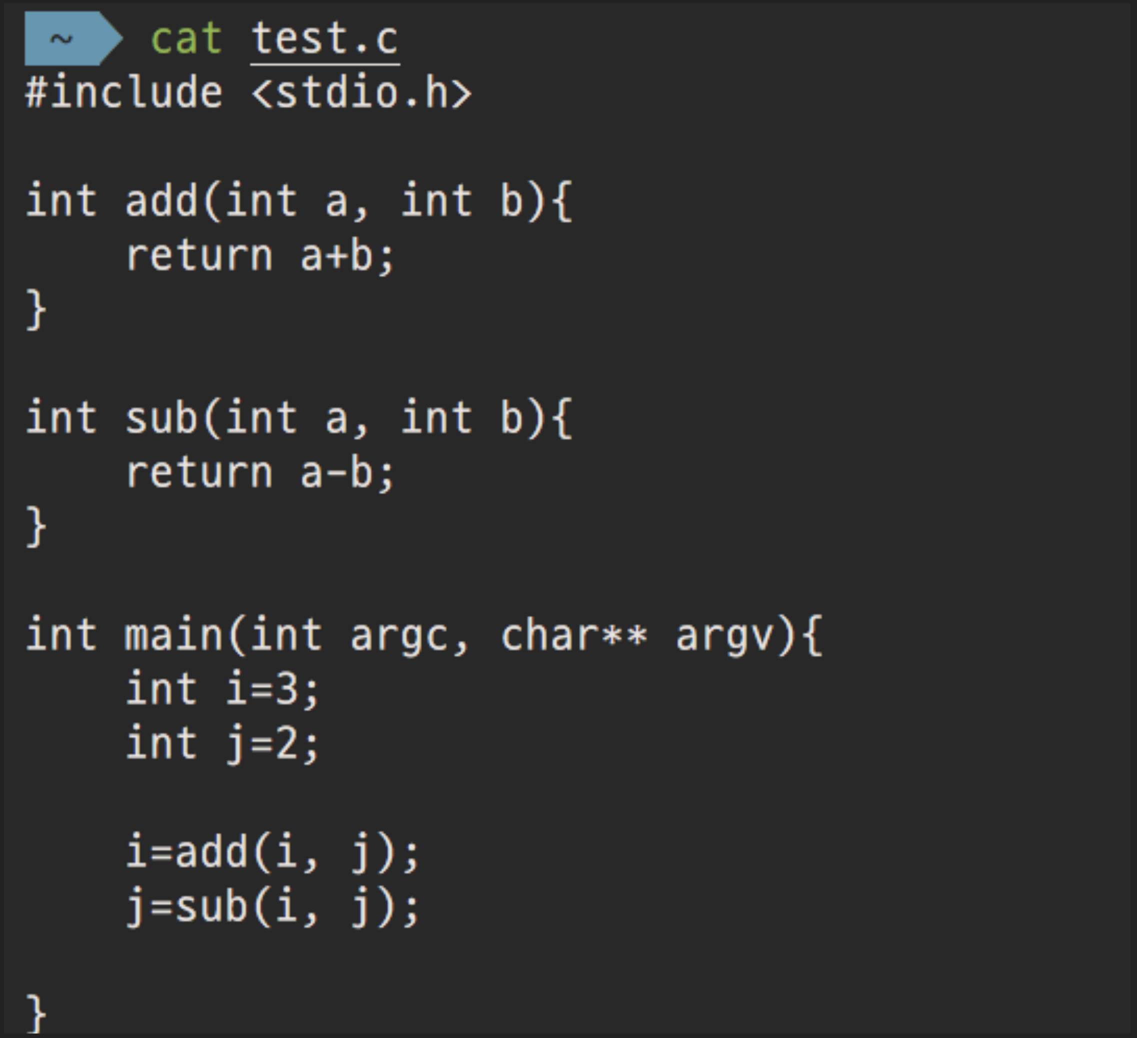
- Show source code
  - list



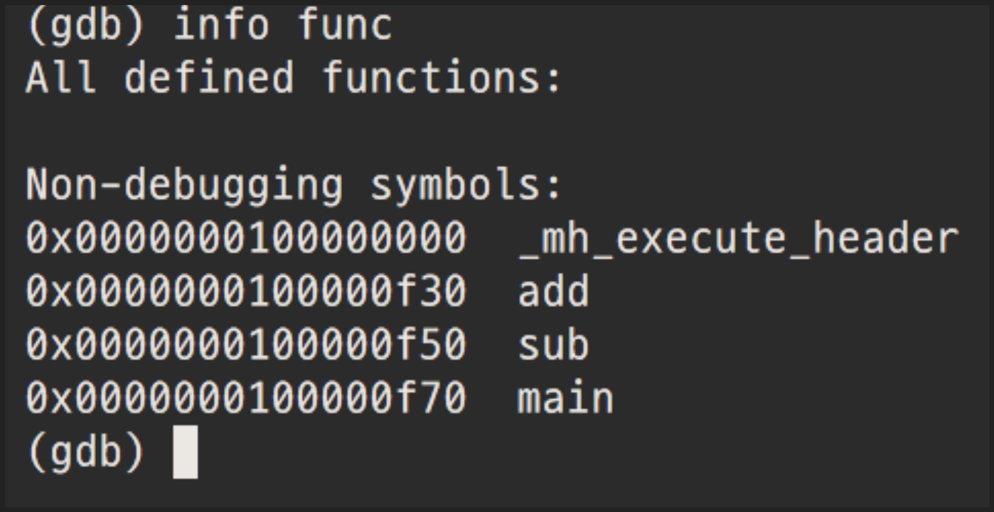
```
(gdb) list
1       #include <stdio.h>
2
3       int main(int argc, char** argv){
4           int i=0;
5           int j=0;
6
7           for (;j<10; j++){
8               i++;
9           }
10      }
(gdb) 
```

TOOLS - GDB

- Show functions defined in binary
  - `i(nfo) f(unc)`



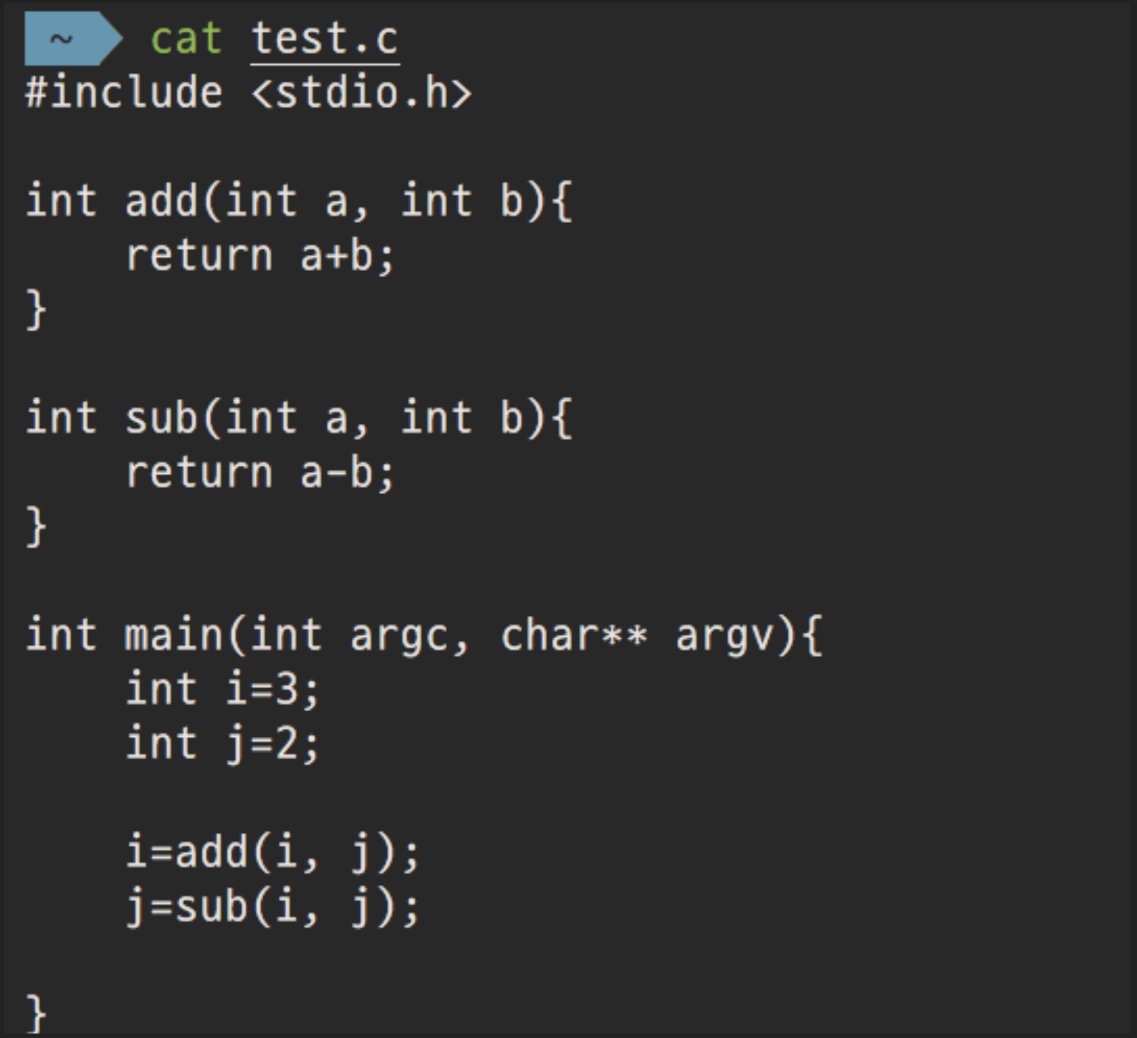
```
 ~  cat test.c
#include <stdio.h>

int add(int a, int b){
    return a+b;
}

int sub(int a, int b){
    return a-b;
}

int main(int argc, char** argv){
    int i=3;
    int j=2;

    i=add(i, j);
    j=sub(i, j);

}
```



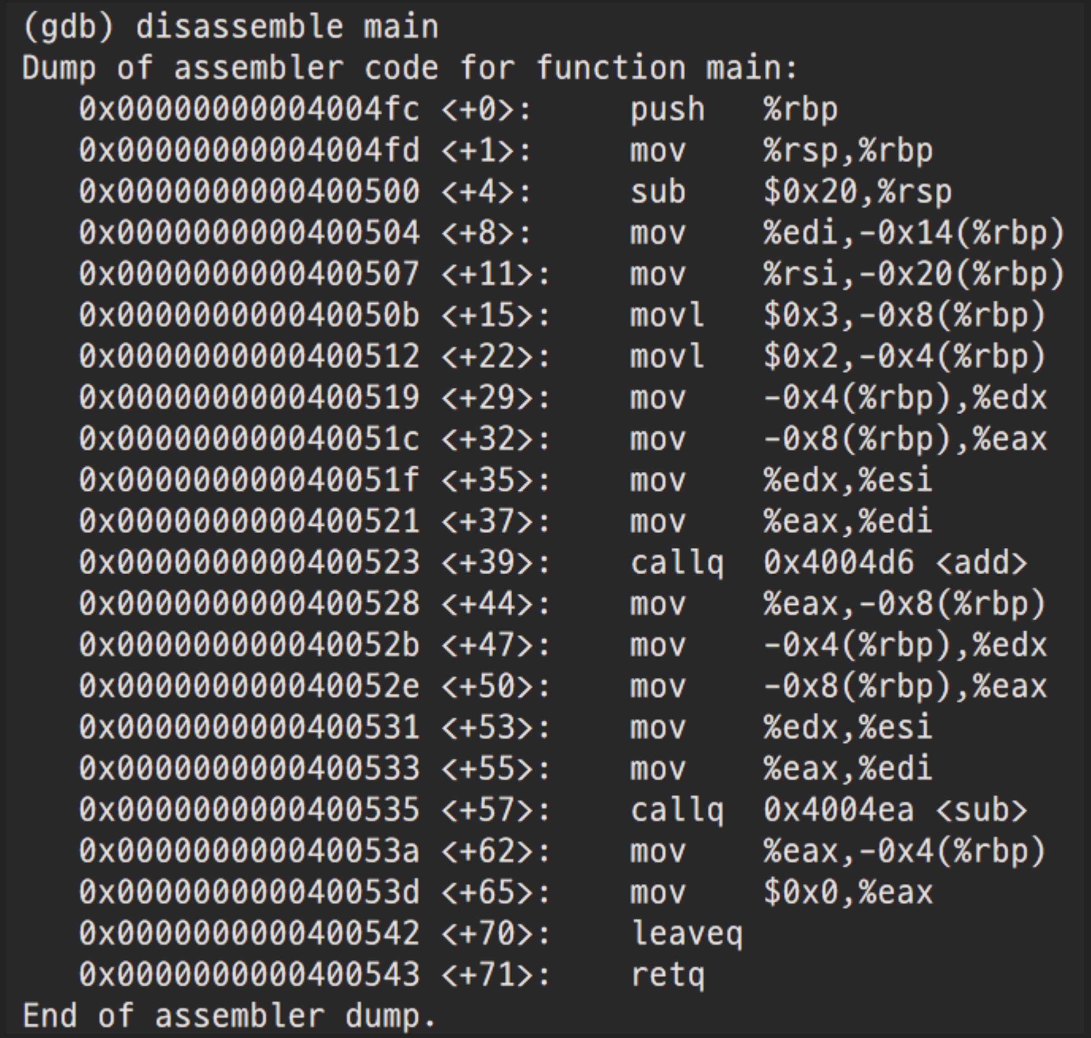
```
(gdb) info func
All defined functions:

Non-debugging symbols:
0x0000000100000000  _mh_execute_header
0x0000000100000f30  add
0x0000000100000f50  sub
0x0000000100000f70  main
(gdb) 
```

TOOLS - GDB

- View assembly of a function
  - disas(semble) [function name]



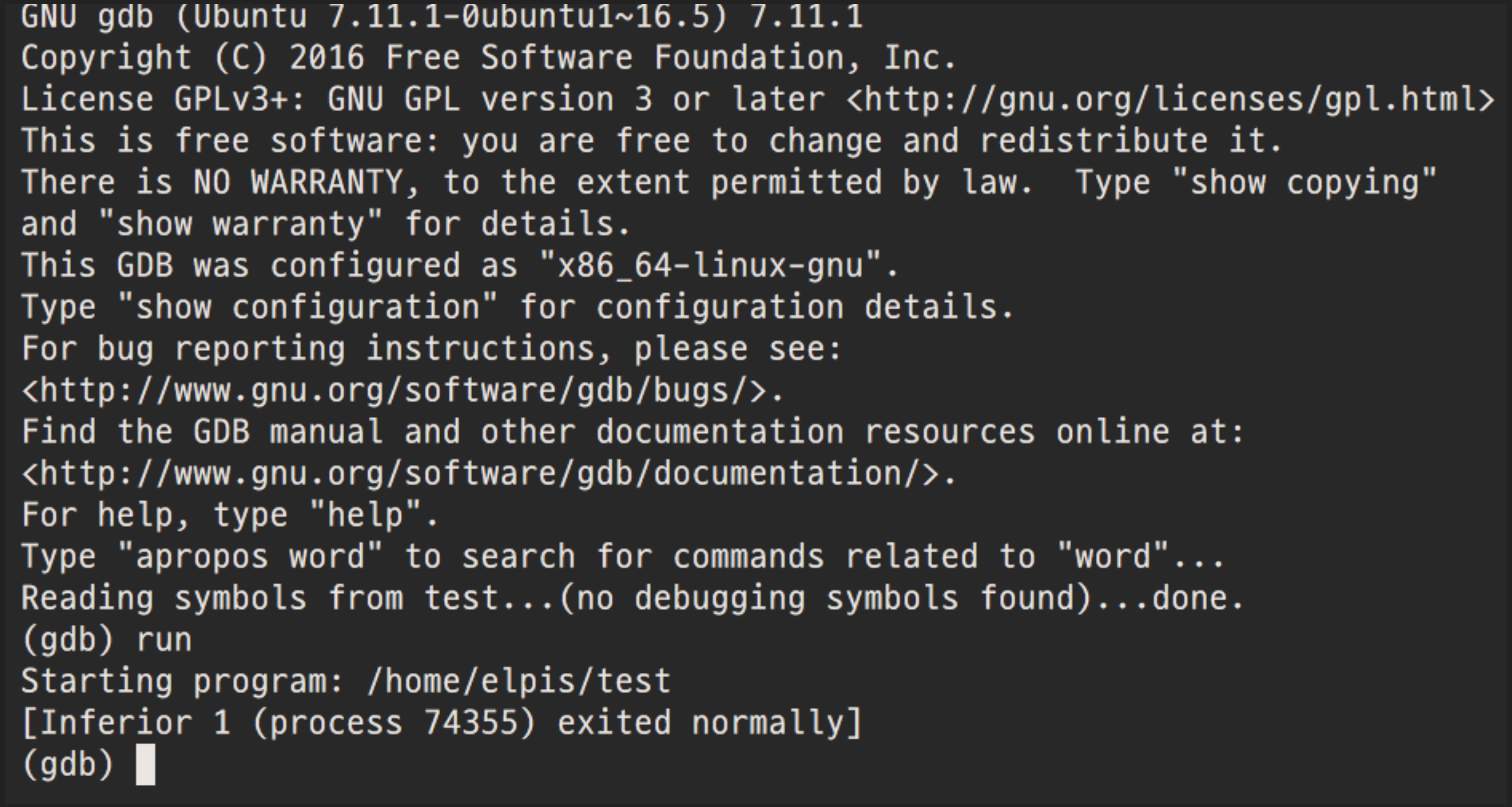
```
 ~  cat test.c
#include <stdio.h>

int add(int a, int b){
    return a+b;
}

int sub(int a, int b){
    return a-b;
}

int main(int argc, char** argv){
    int i=3;
    int j=2;

    i=add(i, j);
    j=sub(i, j);

}
```



```
(gdb) disassemble main
Dump of assembler code for function main:
   0x00000000004004fc <+0>:     push   %rbp
   0x00000000004004fd <+1>:     mov    %rsp,%rbp
   0x0000000000400500 <+4>:     sub    $0x20,%rsp
   0x0000000000400504 <+8>:     mov    %edi,-0x14(%rbp)
   0x0000000000400507 <+11>:    mov    %rsi,-0x20(%rbp)
   0x000000000040050b <+15>:    movl   $0x3,-0x8(%rbp)
   0x0000000000400512 <+22>:    movl   $0x2,-0x4(%rbp)
   0x0000000000400519 <+29>:    mov    -0x4(%rbp),%edx
   0x000000000040051c <+32>:    mov    -0x8(%rbp),%eax
   0x000000000040051f <+35>:    mov    %edx,%esi
   0x0000000000400521 <+37>:    mov    %eax,%edi
   0x0000000000400523 <+39>:    callq  0x4004d6 <add>
   0x0000000000400528 <+44>:    mov    %eax,-0x8(%rbp)
   0x000000000040052b <+47>:    mov    -0x4(%rbp),%edx
   0x000000000040052e <+50>:    mov    -0x8(%rbp),%eax
   0x0000000000400531 <+53>:    mov    %edx,%esi
   0x0000000000400533 <+55>:    mov    %eax,%edi
   0x0000000000400535 <+57>:    callq  0x4004ea <sub>
   0x000000000040053a <+62>:    mov    %eax,-0x4(%rbp)
   0x000000000040053d <+65>:    mov    $0x0,%eax
   0x0000000000400542 <+70>:    leaveq
   0x0000000000400543 <+71>:    retq
End of assembler dump.
```

TOOLS - GDB

- Run program
  - run



```
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
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 test...(no debugging symbols found)...done.
(gdb) run
Starting program: /home/elpis/test
[Inferior 1 (process 74355) exited normally]
(gdb) 
```

TOOLS - GDB

- Breakpoints
  - Without breakpoints, execution continues until it reaches the end of binary file
  - Stops executing when GDB meets the point
  - Need to stop execution in order to analyze variables on runtime

TOOLS - GDB

- Breakpoints

`b 5`
`break 5`
	Break at line 5 of current file

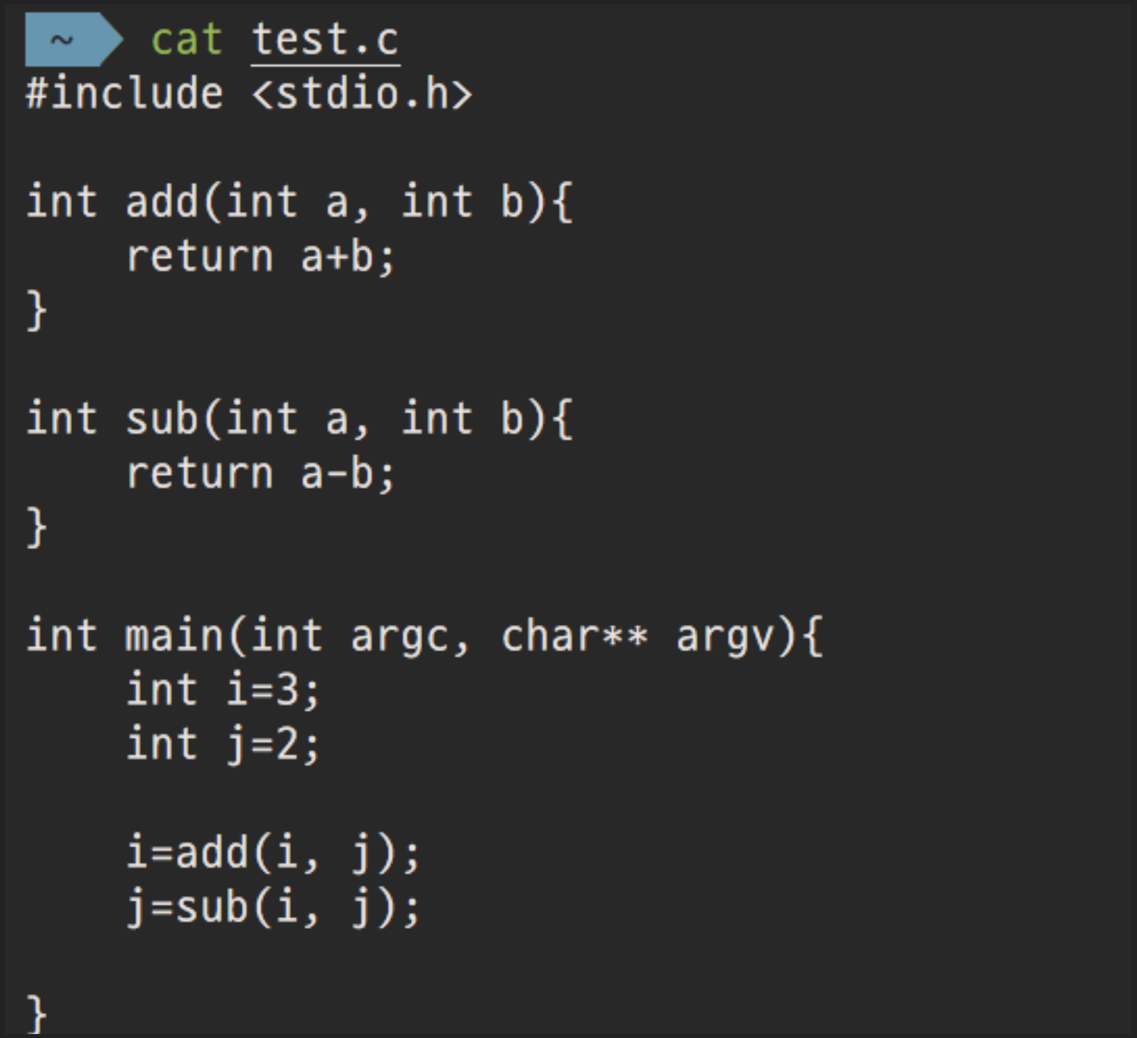
`b main`
`break main`
	Break at the beginning of main() function

`b hello.c:5`
`break hello.c:5`
	Break at line 5 of hello.c

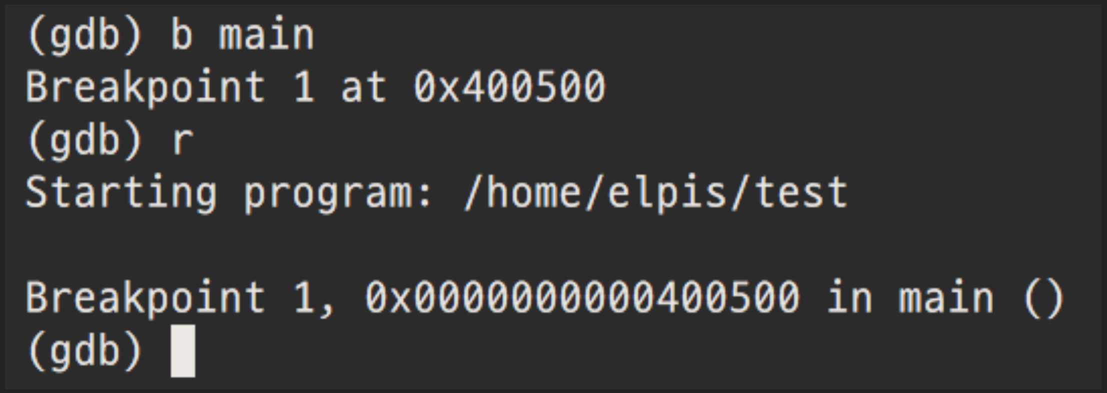
`b* 0x1234`
`break* 0x1234`
	Break at address 0x1234

TOOLS - GDB

- Breakpoints



```
 ~  cat test.c
#include <stdio.h>

int add(int a, int b){
    return a+b;
}

int sub(int a, int b){
    return a-b;
}

int main(int argc, char** argv){
    int i=3;
    int j=2;

    i=add(i, j);
    j=sub(i, j);

}
```

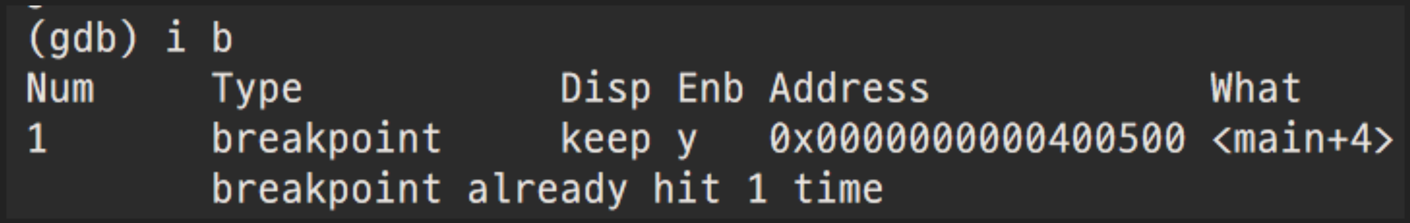


```
(gdb) b main
Breakpoint 1 at 0x400500
(gdb) r
Starting program: /home/elpis/test

Breakpoint 1, 0x0000000000400500 in main ()
(gdb) 
```

Execution stops at main

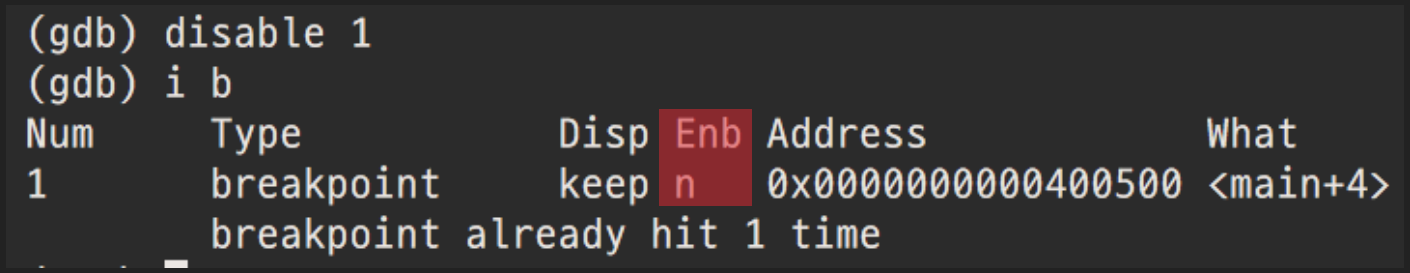
TOOLS - GDB

- List current breakpoints
  - `i(nfo) b(reakpoint)`



```
(gdb) i b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000400500 <main+4>
        breakpoint already hit 1 time
```

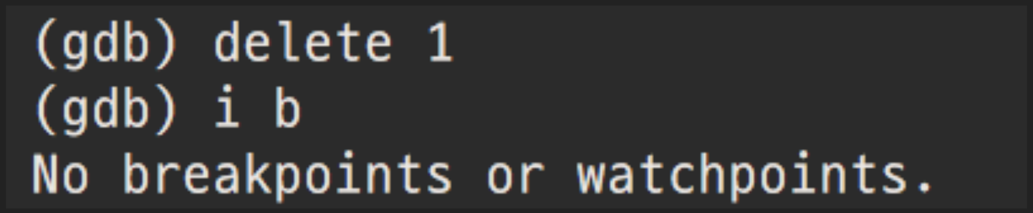
- Enable / Disable a breakpoint
  - `disable [Breakpoint number]`
  - Disable by breakpoint number



```
(gdb) disable 1
(gdb) i b
Num     Type           Disp Enb Address            What
1       breakpoint     keep n   0x0000000000400500 <main+4>
        breakpoint already hit 1 time
```

TOOLS - GDB

- Delete a breakpoint
    - `delete [Breakpoint number] or clear [Line number]`
    - Delete by breakpoint number / line number where breakpoint is set



```
(gdb) delete 1
(gdb) i b
No breakpoints or watchpoints.
```

TOOLS - GDB

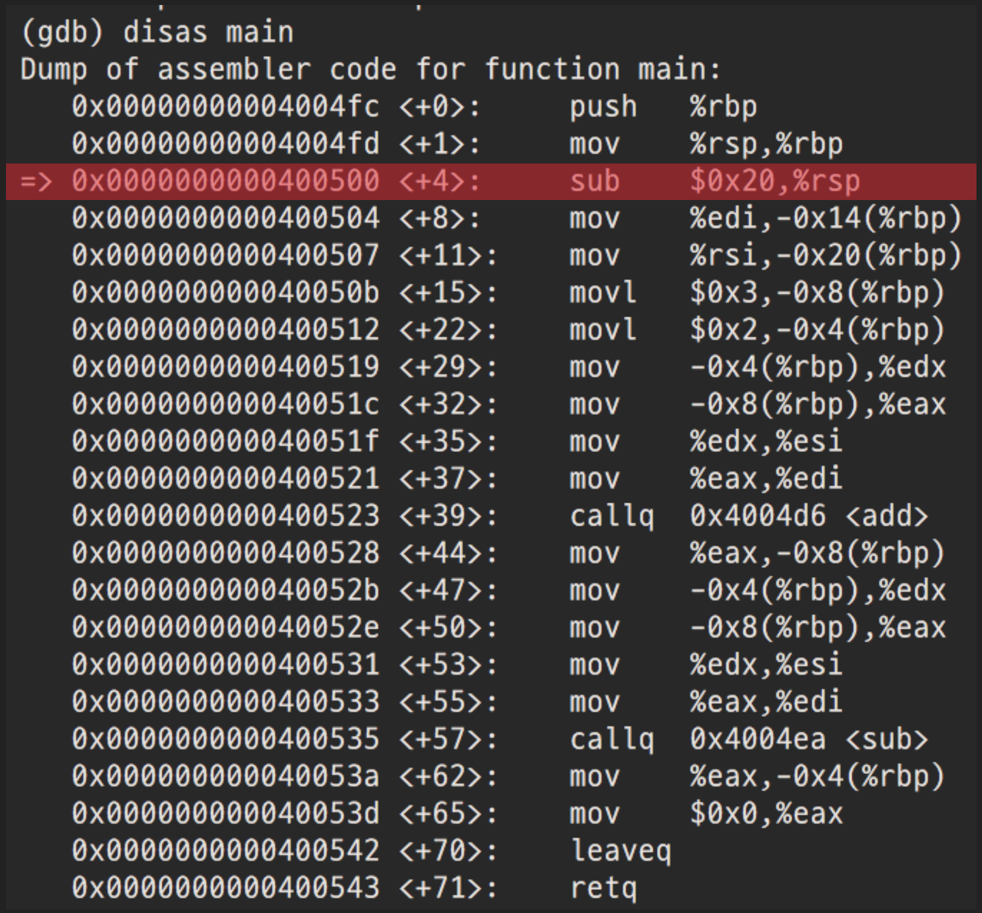
- Continuing after a breakpoint: `c(ontinue)`
- Do we need breakpoints at every line?
  - `n(ext) [number]` or `s(tep) [number]`

    -> Execute `[number]` of lines

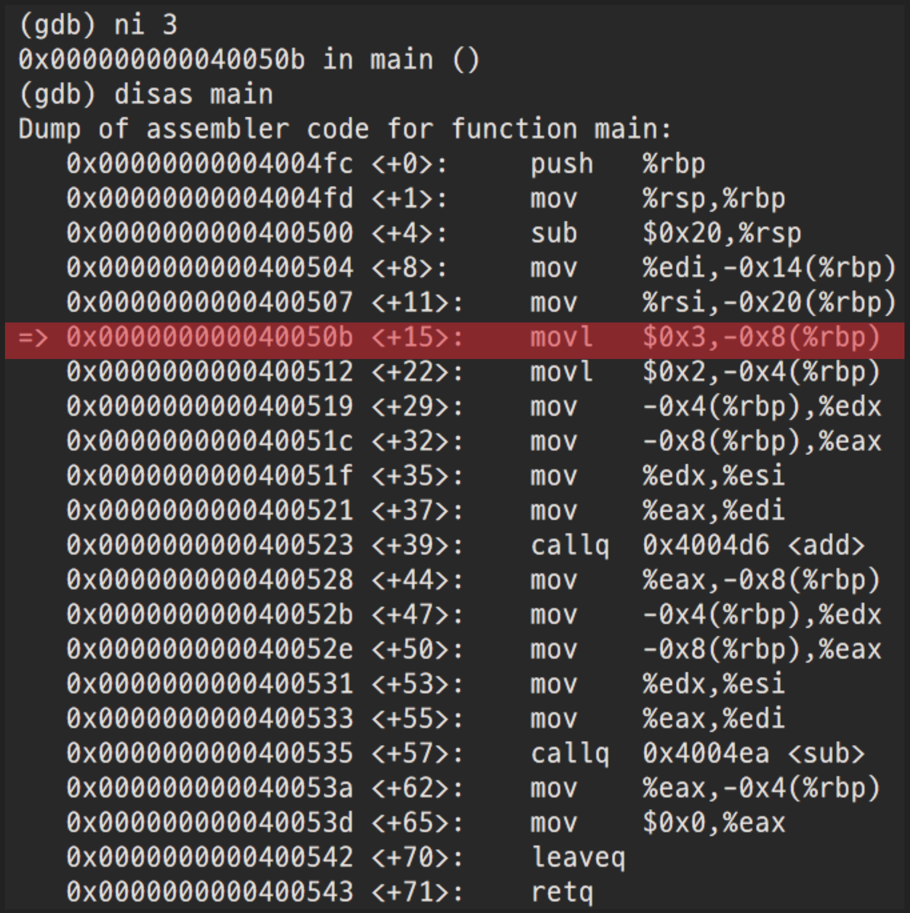
TOOLS - GDB

- ni [number]

  -> Execute [number] of machine instructions



```
(gdb) disas main
Dump of assembler code for function main:
   0x00000000004004fc <+0>:     push   %rbp
   0x00000000004004fd <+1>:     mov    %rsp,%rbp
=> 0x0000000000400500 <+4>:     sub    $0x20,%rsp
   0x0000000000400504 <+8>:     mov    %edi,-0x14(%rbp)
   0x0000000000400507 <+11>:    mov    %rsi,-0x20(%rbp)
   0x000000000040050b <+15>:    movl   $0x3,-0x8(%rbp)
   0x0000000000400512 <+22>:    movl   $0x2,-0x4(%rbp)
   0x0000000000400519 <+29>:    mov    -0x4(%rbp),%edx
   0x000000000040051c <+32>:    mov    -0x8(%rbp),%eax
   0x000000000040051f <+35>:    mov    %edx,%esi
   0x0000000000400521 <+37>:    mov    %eax,%edi
   0x0000000000400523 <+39>:    callq  0x4004d6 <add>
   0x0000000000400528 <+44>:    mov    %eax,-0x8(%rbp)
   0x000000000040052b <+47>:    mov    -0x4(%rbp),%edx
   0x000000000040052e <+50>:    mov    -0x8(%rbp),%eax
   0x0000000000400531 <+53>:    mov    %edx,%esi
   0x0000000000400533 <+55>:    mov    %eax,%edi
   0x0000000000400535 <+57>:    callq  0x4004ea <sub>
   0x000000000040053a <+62>:    mov    %eax,-0x4(%rbp)
   0x000000000040053d <+65>:    mov    $0x0,%eax
   0x0000000000400542 <+70>:    leaveq
   0x0000000000400543 <+71>:    retq
```



```
(gdb) ni 3
0x000000000040050b in main ()
(gdb) disas main
Dump of assembler code for function main:
   0x00000000004004fc <+0>:     push   %rbp
   0x00000000004004fd <+1>:     mov    %rsp,%rbp
   0x0000000000400500 <+4>:     sub    $0x20,%rsp
   0x0000000000400504 <+8>:     mov    %edi,-0x14(%rbp)
   0x0000000000400507 <+11>:    mov    %rsi,-0x20(%rbp)
=> 0x000000000040050b <+15>:    movl   $0x3,-0x8(%rbp)
   0x0000000000400512 <+22>:    movl   $0x2,-0x4(%rbp)
   0x0000000000400519 <+29>:    mov    -0x4(%rbp),%edx
   0x000000000040051c <+32>:    mov    -0x8(%rbp),%eax
   0x000000000040051f <+35>:    mov    %edx,%esi
   0x0000000000400521 <+37>:    mov    %eax,%edi
   0x0000000000400523 <+39>:    callq  0x4004d6 <add>
   0x0000000000400528 <+44>:    mov    %eax,-0x8(%rbp)
   0x000000000040052b <+47>:    mov    -0x4(%rbp),%edx
   0x000000000040052e <+50>:    mov    -0x8(%rbp),%eax
   0x0000000000400531 <+53>:    mov    %edx,%esi
   0x0000000000400533 <+55>:    mov    %eax,%edi
   0x0000000000400535 <+57>:    callq  0x4004ea <sub>
   0x000000000040053a <+62>:    mov    %eax,-0x4(%rbp)
   0x000000000040053d <+65>:    mov    $0x0,%eax
   0x0000000000400542 <+70>:    leaveq
   0x0000000000400543 <+71>:    retq
```

TOOLS - GDB

- Perform next until specific location
- `u(ntil) [Line number]`

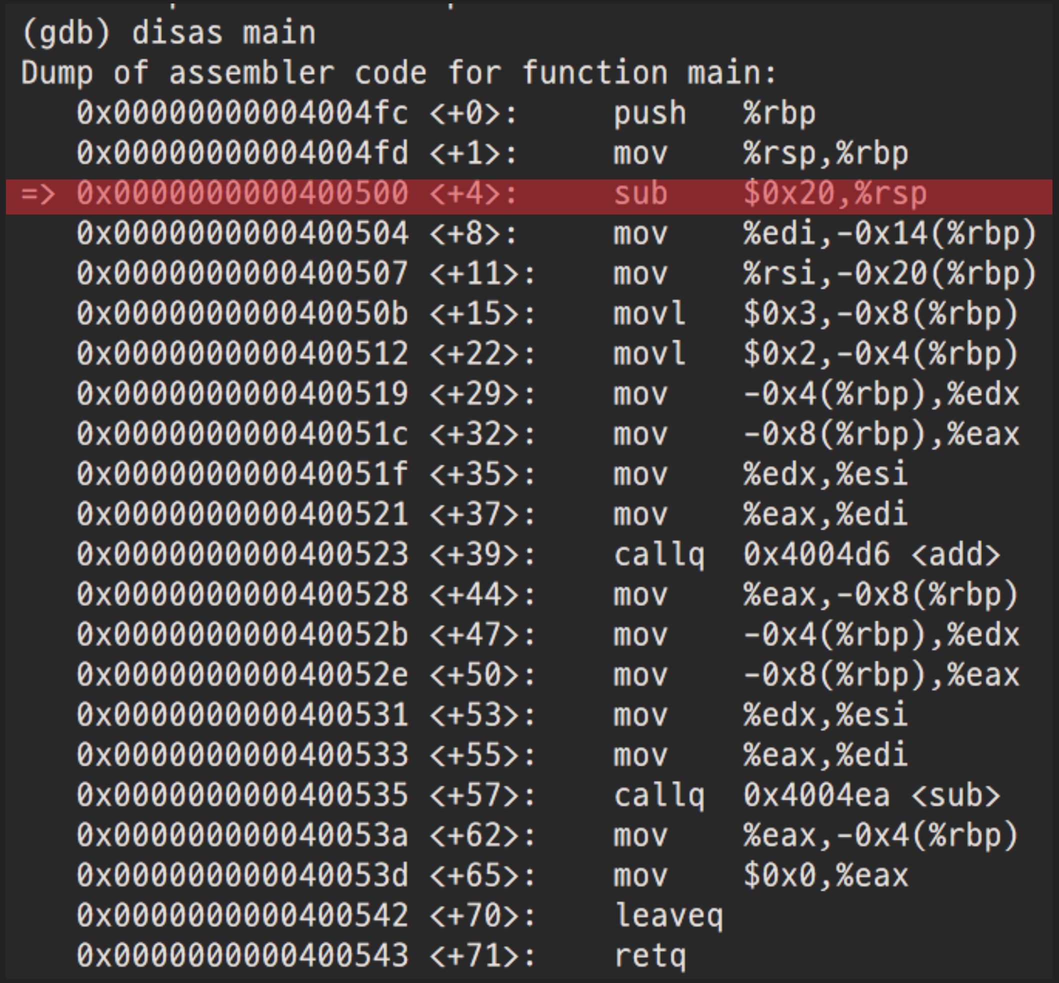
  -> Keep executing until `[Line number]` is met
- `u(ntil) *[Memory address]`

  -> Keep executing until `[Memory address]` is met

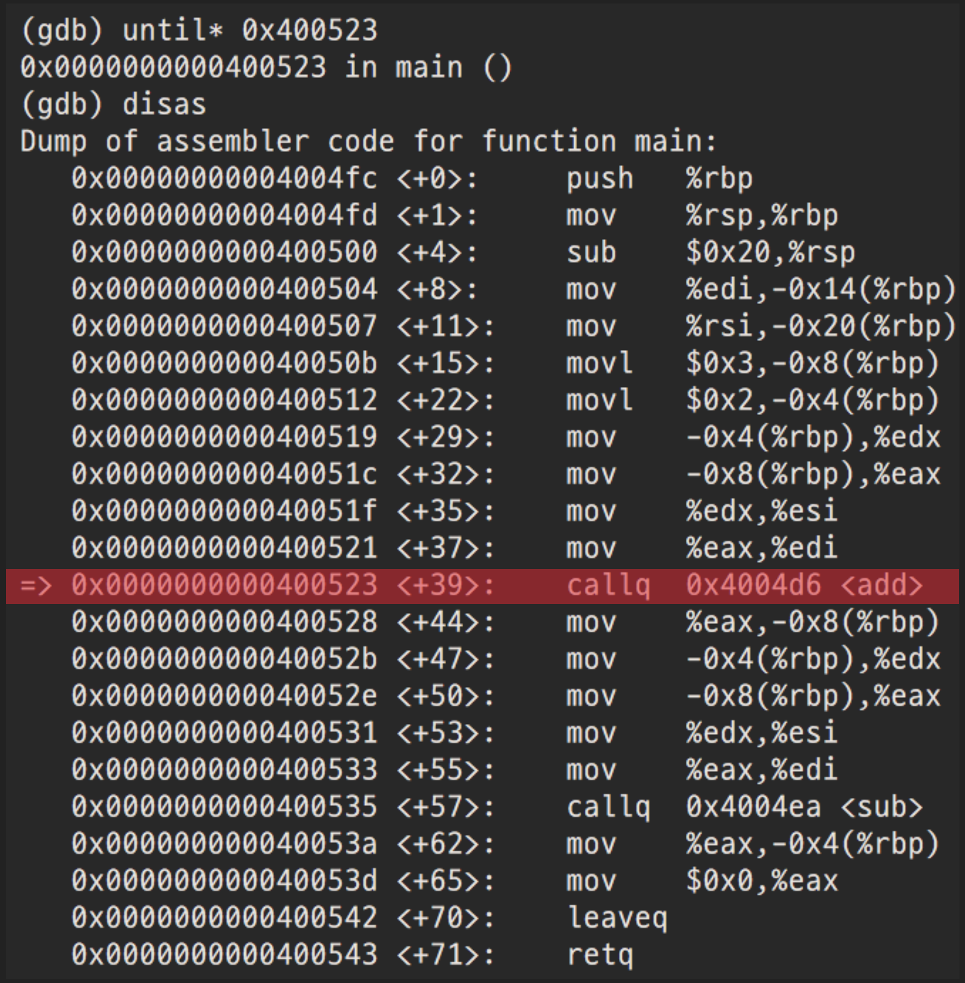
TOOLS - GDB

- u(ntil) *[Memory address]

  Ex) until* 0x400523



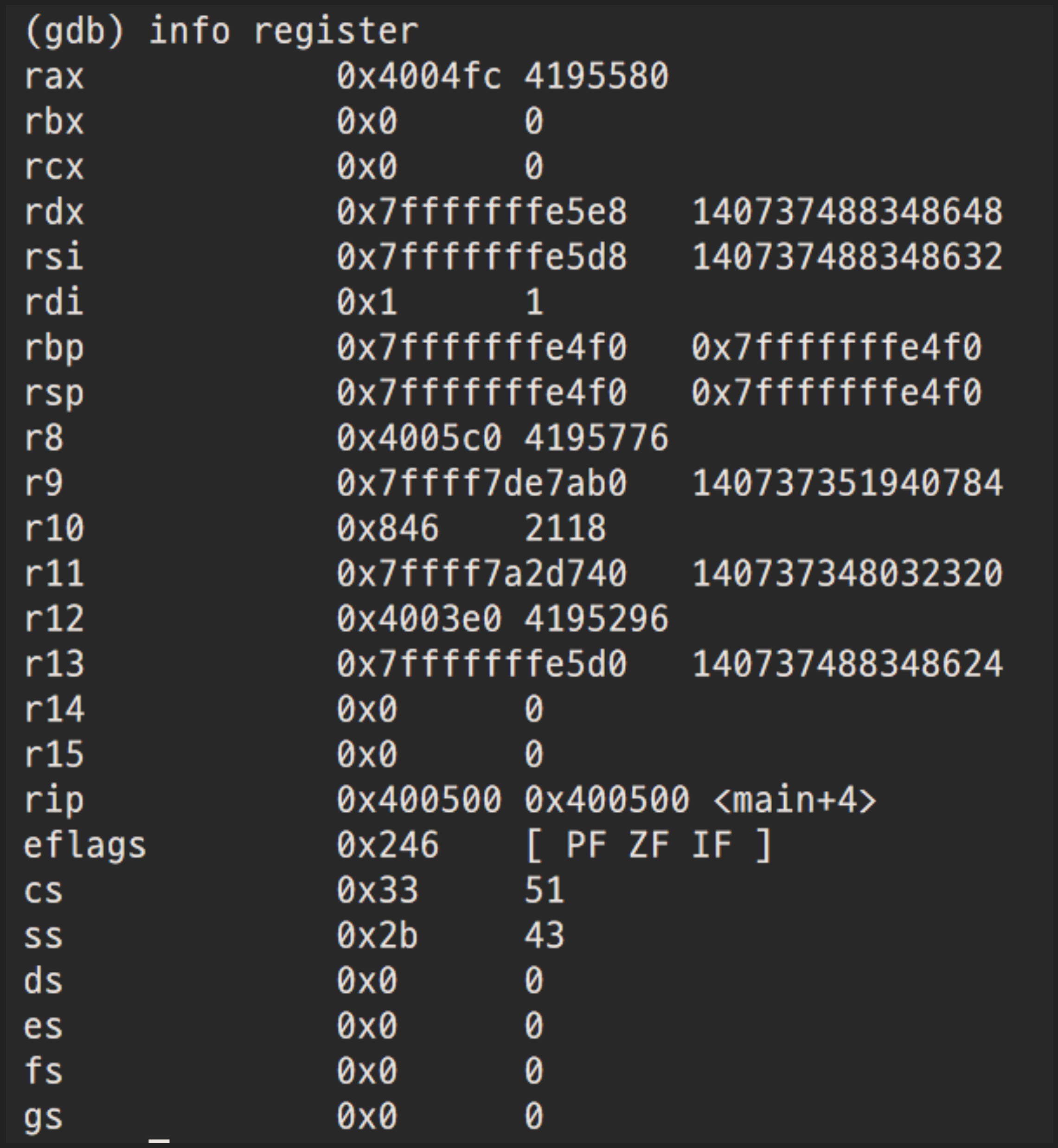
```
(gdb) disas main
Dump of assembler code for function main:
   0x00000000004004fc <+0>:     push   %rbp
   0x00000000004004fd <+1>:     mov    %rsp,%rbp
=> 0x0000000000400500 <+4>:     sub    $0x20,%rsp
   0x0000000000400504 <+8>:     mov    %edi,-0x14(%rbp)
   0x0000000000400507 <+11>:    mov    %rsi,-0x20(%rbp)
   0x000000000040050b <+15>:    movl   $0x3,-0x8(%rbp)
   0x0000000000400512 <+22>:    movl   $0x2,-0x4(%rbp)
   0x0000000000400519 <+29>:    mov    -0x4(%rbp),%edx
   0x000000000040051c <+32>:    mov    -0x8(%rbp),%eax
   0x000000000040051f <+35>:    mov    %edx,%esi
   0x0000000000400521 <+37>:    mov    %eax,%edi
   0x0000000000400523 <+39>:    callq  0x4004d6 <add>
   0x0000000000400528 <+44>:    mov    %eax,-0x8(%rbp)
   0x000000000040052b <+47>:    mov    -0x4(%rbp),%edx
   0x000000000040052e <+50>:    mov    -0x8(%rbp),%eax
   0x0000000000400531 <+53>:    mov    %edx,%esi
   0x0000000000400533 <+55>:    mov    %eax,%edi
   0x0000000000400535 <+57>:    callq  0x4004ea <sub>
   0x000000000040053a <+62>:    mov    %eax,-0x4(%rbp)
   0x000000000040053d <+65>:    mov    $0x0,%eax
   0x0000000000400542 <+70>:    leaveq
   0x0000000000400543 <+71>:    retq
```



```
(gdb) until* 0x400523
0x0000000000400523 in main ()
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004fc <+0>:     push   %rbp
   0x00000000004004fd <+1>:     mov    %rsp,%rbp
   0x0000000000400500 <+4>:     sub    $0x20,%rsp
   0x0000000000400504 <+8>:     mov    %edi,-0x14(%rbp)
   0x0000000000400507 <+11>:    mov    %rsi,-0x20(%rbp)
   0x000000000040050b <+15>:    movl   $0x3,-0x8(%rbp)
   0x0000000000400512 <+22>:    movl   $0x2,-0x4(%rbp)
   0x0000000000400519 <+29>:    mov    -0x4(%rbp),%edx
   0x000000000040051c <+32>:    mov    -0x8(%rbp),%eax
   0x000000000040051f <+35>:    mov    %edx,%esi
   0x0000000000400521 <+37>:    mov    %eax,%edi
=> 0x0000000000400523 <+39>:    callq  0x4004d6 <add>
   0x0000000000400528 <+44>:    mov    %eax,-0x8(%rbp)
   0x000000000040052b <+47>:    mov    -0x4(%rbp),%edx
   0x000000000040052e <+50>:    mov    -0x8(%rbp),%eax
   0x0000000000400531 <+53>:    mov    %edx,%esi
   0x0000000000400533 <+55>:    mov    %eax,%edi
   0x0000000000400535 <+57>:    callq  0x4004ea <sub>
   0x000000000040053a <+62>:    mov    %eax,-0x4(%rbp)
   0x000000000040053d <+65>:    mov    $0x0,%eax
   0x0000000000400542 <+70>:    leaveq
   0x0000000000400543 <+71>:    retq
```

TOOLS - GDB

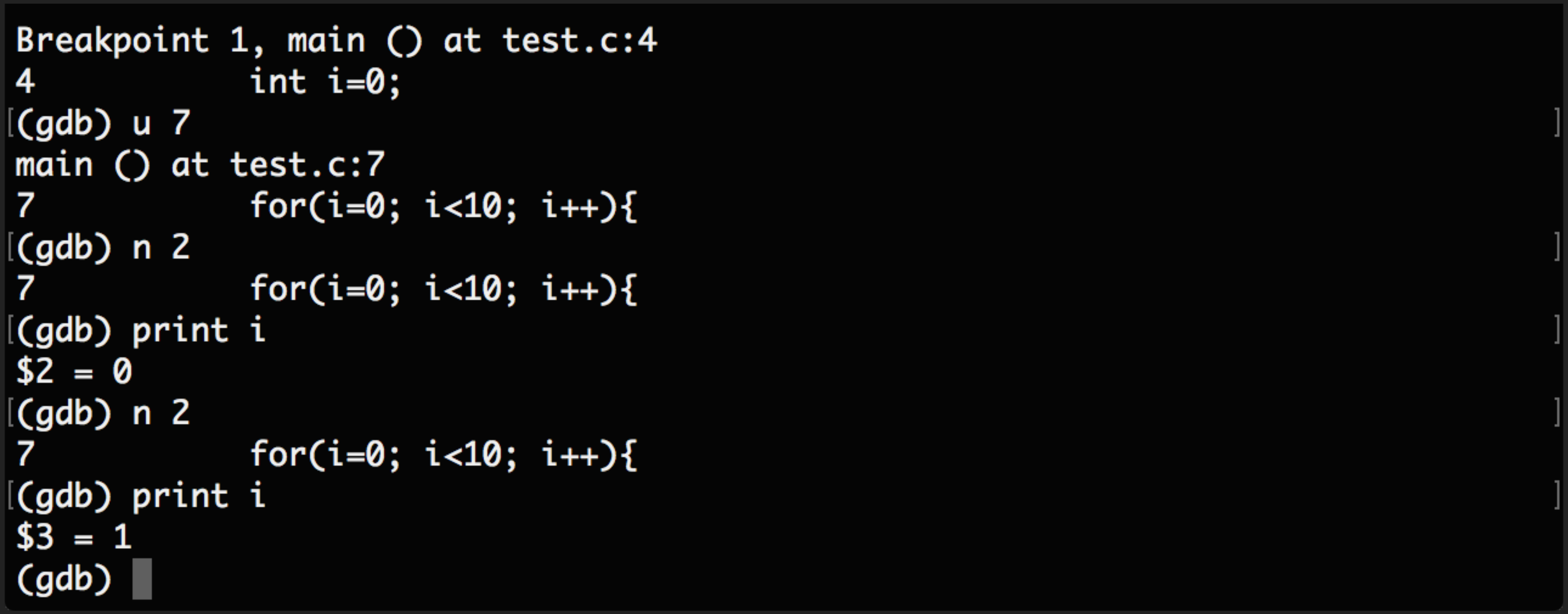
- View values stored in register
  - Registers can be viewed when breakpoint is set
  - i(nfo) r(egister)



```
(gdb) info register
rax            0x4004fc 4195580
rbx            0x0      0
rcx            0x0      0
rdx            0x7fffffffe5e8   140737488348648
rsi            0x7fffffffe5d8   140737488348632
rdi            0x1      1
rbp            0x7fffffffe4f0   0x7fffffffe4f0
rsp            0x7fffffffe4f0   0x7fffffffe4f0
r8             0x4005c0 4195776
r9             0x7ffff7de7ab0   140737351940784
r10            0x846    2118
r11            0x7ffff7a2d740   140737348032320
r12            0x4003e0 4195296
r13            0x7fffffffe5d0   140737488348624
r14            0x0      0
r15            0x0      0
rip            0x400500 0x400500 <main+4>
eflags         0x246    [ PF ZF IF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
```

TOOLS - GDB

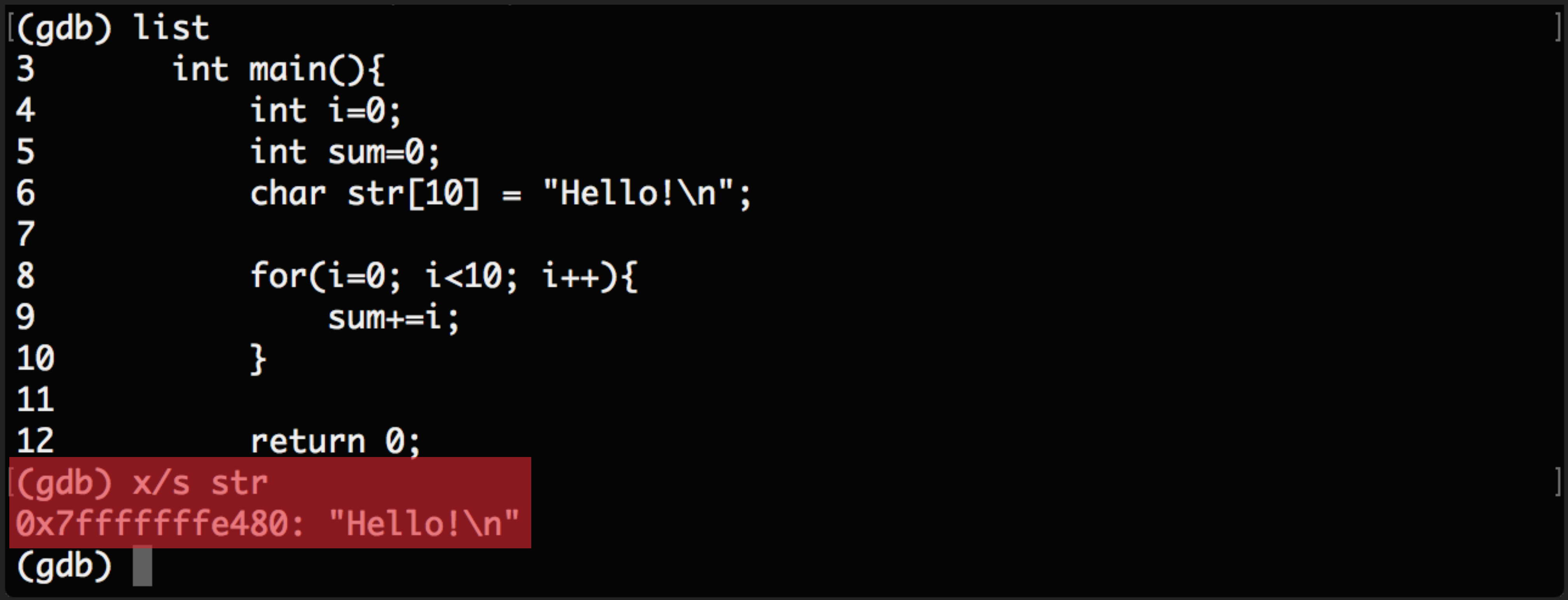
- Examine a variable
  - `print [Variable name]`



```
Breakpoint 1, main () at test.c:4
4             int i=0;
(gdb) u 7
main () at test.c:7
7             for(i=0; i<10; i++){
(gdb) n 2
7             for(i=0; i<10; i++){
(gdb) print i
$2 = 0
(gdb) n 2
7             for(i=0; i<10; i++){
(gdb) print i
$3 = 1
(gdb) 
```

TOOLS - GDB

- Examine a value in an address
  - x [Address]
- Useful in analyzing strings



```
(gdb) list
3       int main(){
4           int i=0;
5           int sum=0;
6           char str[10] = "Hello!\n";
7
8           for(i=0; i<10; i++){
9               sum+=i;
10          }
11
12          return 0;
(gdb) x/s str
0x7fffffffe480: "Hello!\n"
(gdb) 
```

TOOLS - GDB

- Examine a value in an address

x/x	Print value as hexadecimal	int i = 0xff x/x &i = 0x000000ff
x/t	Print value as binary	x/t &i = 00…0011111111
x/b	Print by byte	x/x &i = 0x000000ff x/xb &i = 0xff
x/w	Print by word	x/x &i == x/wx &i
x/s	Print string until \0 is met	char s[10]="hello\n" x/s s = "hello\n"
x/[Number]	Print [Number] of variables	x/xb &i = 0xff x/3xb &i = 0xff 0x00 0x00

TOOLS - GDB

- Examine a value in an address
  - Options can be combined
- Ex) x/4wx 0xbfff2a0: Read 4 words from address 0xbffff2a0 as hexadecimal

TOOLS - GDB

- Examine type of variable
  - `whatis [Variable name]`

FAQ

- Permission denied!
  - Linux files have file permissions, but Windows doesn't
  - Permissions are removed when given `.tar` file is unarchived on Windows

FAQ

- Permission denied!
  - Check permission of the file with ls (link) and fix file permission accordingly (link)
  - Cannot execute: `chmod +x [File name]`
  - Cannot read: `chmod +r [File name]`
  - Cannot wrote: `chmod +w [File name]`

FAQ

- Invalid host!
  - Bombs are made to work only on the provided servers (canis01~04, 06, 07)
  - Move the binary file to the servers provided to you