

Stack Debugging

Young W. Lim

2017-07-15 Fri

1 Introduction

- References
- Compiling to IA32 Assembly
- Checking `/proc/<pid>/maps` file

"Self-service Linux: Mastering the Art of Problem Determination", Mark Wilding
"Computer Architecture: A Programmer's Perspective", Bryant & O'Hallaron

I, the copyright holder of this work, hereby publish it under the following licenses: GNU head Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

CC BY SA This file is licensed under the Creative Commons Attribution ShareAlike 3.0 Unported License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license compatible with this one.

preparation : for Linux Mint

- install packages
 - lib32gcc1
 - lib32gcc-dbg
 - gcc-multilib
- gcc -m32 t.c

stack frame address range

- stack frame starts from 0xc0000000
- stack bottom : 0xc0000000
- stack address range
 - 0xbfffe000 (low address)
 - 0xc0000000 (high address)
- stack grows toward lower address
 - from high address to low address
 - from 0xc0000000 to 0xbfffe000

stack frame in the Linux Mint

- stack address range
 - 0xbfffe000 (low address)
 - 0xc0000000 (high address)
 - 0x00002000
- stack address range in the Linux Mint
 - 0xfffa8000 (low address)
 - 0xfffc9000 (high address)
 - 0x00021000

code for displaying /proc/<pid>/maps

- local variables (stack variables) : stack_var
- local variable address : 0xffff77e8

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>

int main(void) {
    int stack_var = 5;
    char cmd[64];

    printf("addr(stack_var) = 0x%p \n\n", &stack_var);

    sprintf(cmd, "cat /proc/%d/maps", getpid());
    system(cmd);

    return 0;
}
```

result for displaying /proc/<pid>/maps (1)

```
addr(stack_var) = 0xffffc77e8
```

```
08048000-08049000 r-xp 00000000 08:51 424080 /home/young/a.out
08049000-0804a000 r--p 00000000 08:51 424080 /home/young/a.out
0804a000-0804b000 rw-p 00001000 08:51 424080 /home/young/a.out
0876d000-0878e000 rw-p 00000000 00:00 0 [heap]
f756a000-f756b000 rw-p 00000000 00:00 0
f756b000-f7718000 r-xp 00000000 08:51 1475199 /lib32/libc-2.23.so
f7718000-f7719000 ---p 001ad000 08:51 1475199 /lib32/libc-2.23.so
f7719000-f771b000 r--p 001ad000 08:51 1475199 /lib32/libc-2.23.so
f771b000-f771c000 rw-p 001af000 08:51 1475199 /lib32/libc-2.23.so
f771c000-f7720000 rw-p 00000000 00:00 0
f773d000-f773f000 r--p 00000000 00:00 0 [vvar]
f773f000-f7741000 r-xp 00000000 00:00 0 [vdso]
f7741000-f7763000 r-xp 00000000 08:51 1475178 /lib32/ld-2.23.so
f7763000-f7764000 rw-p 00000000 00:00 0
f7764000-f7765000 r--p 00022000 08:51 1475178 /lib32/ld-2.23.so
f7765000-f7766000 rw-p 00023000 08:51 1475178 /lib32/ld-2.23.so
ffffa8000-ffffc9000 rw-p 00000000 00:00 0 [stack]
```


result for displaying /proc/<pid>/maps (2)

```
addr(stack_var) = 0xffff77e8
```

```
ffffa8000-ffffc9000 rw-p 00000000 00:00 0          [stack]
```

```
from fffa8000
```

```
    fffc77e8  stack_var
```

```
to   fffc9000
```

checking /proc/<pid>/maps file

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>

extern char **environ;

int main(int argc, char *argv[]) {
    int stack_var = 5;
    char cmd[64];

    printf("&stack_var = %p\n", &stack_var);
    printf("&argc      = %p\n", &argc      );
    printf(" argv       = %p\n", argv       );
    printf(" environ    = %p\n", environ    );
    printf(" argv[0]    = %p\n", argv[0]    );
    printf("*environ   = %p\n", *environ   );

    sprintf(cmd, "cat /proc/%d/maps", getpid());
    system(cmd);
    return 0;
}
```

checking /proc/<pid>/maps results

```
&argc      = 0xffabaf80
argv       = 0xffabb014
environ    = 0xffabb01c
argv[0]    = 0xffabc382
*environ   = 0xffabc38a
08048000-08049000 r-xp 00000000 08:51 424080 /home/young/a.out
08049000-0804a000 r--p 00000000 08:51 424080 /home/young/a.out
0804a000-0804b000 rw-p 00001000 08:51 424080 /home/young/a.out
095b9000-095da000 rw-p 00000000 00:00 0 [heap]
f7589000-f758a000 rw-p 00000000 00:00 0
f758a000-f7737000 r-xp 00000000 08:51 1475199 /lib32/libc-2.23.so
f7737000-f7738000 ---p 001ad000 08:51 1475199 /lib32/libc-2.23.so
f7738000-f773a000 r--p 001ad000 08:51 1475199 /lib32/libc-2.23.so
f773a000-f773b000 rw-p 001af000 08:51 1475199 /lib32/libc-2.23.so
f773b000-f773f000 rw-p 00000000 00:00 0
f775c000-f775e000 r--p 00000000 00:00 0 [vvar]
f775e000-f7760000 r-xp 00000000 00:00 0 [vdso]
f7760000-f7782000 r-xp 00000000 08:51 1475178 /lib32/ld-2.23.so
f7782000-f7783000 rw-p 00000000 00:00 0
f7783000-f7784000 r--p 00022000 08:51 1475178 /lib32/ld-2.23.so
f7784000-f7785000 rw-p 00023000 08:51 1475178 /lib32/ld-2.23.so
ffa9c000-ffabd000 rw-p 00000000 00:00 0 [stack]
```

checking /proc/<pid>/maps file (3)

```
[from]      0xbfffe000

&stack_var 0xbffff2dc --- stack top      (low address)
&argc      0xbffff2f0
  argv     0xbffff334
  environ  0xbffff33c
  argv[0]  0xbffff4d5
*environ   0xbffff4de

[to]        0xc0000000 --- stack bottom (high address)
```

checking /proc/<pid>/maps file (4)

- [to] 0xc0000000 — stack bottom (high address)
- path name specified in exec()
- process environment variables
- argv strings
- argc
- aux vectors
- [from] 0xbfffe000