Chapter 9: Signal (5/5)

김선영
sunyzero@gmail(dot)com
버 전: 2017-05-27

가메 출판사 http://www.kame.co.kr
저자홈페이지 http://sunyzero.tistory.com

Alternative signal stack

alternative stack

- SA_ONSTACK
  - sigstack from 4.2BSD
  - 표준화로 인해 ... sigaltstack()
- SIGSEGV (Segmentation violation) 처리
  - segmentation fault가 core dump를 남길 때 디버깅을 위한 제대로 된 정보를 덤프하는 목적

sigaltstack

```
int  sigaltstack(const stack_t *restrict ss, stack_t *restrict oss);
typedef  struct  sigaltstack {
    void     *ss_sp;     /* stack base or pointer. */
    size_t   ss_size;    /* stack size. */
    int      ss_flags;   /* flags */
} stack_t;
```

- ss_flags

SS_ONSTACK	대체 시그널 스택이 사용 중임을 알려준다. 사용중인 시그널 스택은 변경 할 수 없다. 이 플래그는 사용자가 사용할 수 없고, 이전 시그널 스택을 백업하는 경우에 보고 용도.
SS_DISABLE	현재 대체 시그널 스택을 비활성화 시킨다. 시그널 대체 스택은 유저 스택을 사용하는 방식으로 되돌아 간다.

segv_sigaltstack.c (1/4)

```
#define _XOPEN_SOURCE   700
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <signal.h>
#include <time.h>
#define SZ_BUFFER 1024*1024     /* 1MiB */

int exhaust_stack(int count);
void inst_sighandler();
void sa_handler_segv(int signum);
int flag_altstack;  /* 0:off, nonzero:on(execute sigaltstack) */

#define     SZ_SIGHANDLER_STACK     16384
stack_t     g_ss;       /* signal stack structure */
```

문제가 있는 코드이다.
예제 실행 후 수정해보자.

segv_sigaltstack.c (2/4)

```
int main(int argc, char *argv[])
{
    if (argc == 2 && argv[1][0] == '1') {
        printf("[enabled] alternate signal stack.\n");
        flag_altstack = 1;
    } else {
        printf("[disabled] alternate signal stack.\n");
    }
    printf("SIGSTKSZ(%d) MINSIGSTKSZ(%d)\n", SIGSTKSZ, MINSIGSTKSZ);
    inst_sighandler();
    exhaust_stack(100);
    return 0;
}
```

segv_sigaltstack.c (3/4)

```
int exhaust_stack(int count) {
    char buffer[SZ_BUFFER]; /* 1 MiB stack */
    if (count <= 0) {
        printf(">> stopping recursive func.\n");
        return 0;
    }
    printf("[%d] Current stack addr(%p)\n", count, buffer);
    exhaust_stack(count-1);
    return 0;
}
void sa_handler_segv(int signum) {
    time_t  t_now = time(0);
    struct tm   *tm_now = localtime(&t_now);
    snprintf(g_ss.ss_sp, g_ss.ss_size,
            "SEGV: SigNo(%d) Time(%02d:%02d:%02d)",
            signum, tm_now->tm_hour, tm_now->tm_min, tm_now->tm_sec);
    printf("%s\n", (char *)g_ss.ss_sp);
    fflush(stdout);
    abort(); // core 덤프를 위해... SA_RESETHAND 플래그 설정 후 삭제할 것!
}
```

이건 예제니까
signal handler에서
printf를 사용한 것이다.
원래는 하면 X!

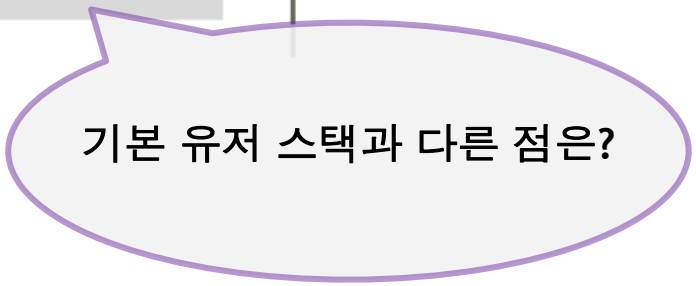
segv_sigaltstack.c (4/4)

```
void inst_sighandler() {
    struct sigaction sa_segv;
    memset(&sa_segv, 0, sizeof(struct sigaction));
    if (( g_ss.ss_sp = malloc(SZ_SIGHANDLER_STACK)) == NULL) {
        perror("malloc");
        exit(EXIT_FAILURE);
    }
    g_ss.ss_size = SZ_SIGHANDLER_STACK;
    g_ss.ss_flags = 0;
    if (flag_altstack) {
        if (sigaltstack(&g_ss, NULL) == -1) { // alt. stack 설정
            perror("sigaltstack");
            exit(EXIT_FAILURE);
        }
        sa_segv.sa_flags = SA_ONSTACK; // alt. stack을 사용하도록 설정
    } else {
        sa_segv.sa_flags = 0;
    }
    sa_segv.sa_handler = sa_handler_segv;
    sigaction(SIGSEGV, &sa_segv, 0); // SIGSEGV 시그널 핸들러 설치
}
```

segv_sigaltstack : example

```
$ ./segv_sigaltstack 1
[enabled] alternate signal stack.
SIGSTKSZ(8192) MINSIGSTKSZ(2048)
[100] Current stack addr(0x7fff1d7c2110)
[99] Current stack addr(0x7fff1d6c20f0)
[98] Current stack addr(0x7fff1d5c20d0)
[97] Current stack addr(0x7fff1d4c20b0)
[96] Current stack addr(0x7fff1d3c2090)
[95] Current stack addr(0x7fff1d2c2070)
[94] Current stack addr(0x7fff1d1c2050)
SEGV: SigNo(11) DateTime(0112-01-27/17:06:00)
중지됨 (core dumped)
```

```
$ ./segv_sigaltstack 0
[disabled] alternate signal stack.
SIGSTKSZ(8192) MINSIGSTKSZ(2048)
[100] Current stack addr(0x7fffe5de4f00)
[99] Current stack addr(0x7fffe5ce4ee0)
[98] Current stack addr(0x7fffe5be4ec0)
[97] Current stack addr(0x7fffe5ae4ea0)
[96] Current stack addr(0x7fffe59e4e80)
[95] Current stack addr(0x7fffe58e4e60)
[94] Current stack addr(0x7fffe57e4e40)
세그멘테이션 오류 (core dumped)
```



gdb : segv_sigaltstack (1)

```
$ gdb segv_sigaltstack core.69933
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-80.el7
Copyright (C) 2013 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/sunyzero/alsp_3rd/src/9.signal/segv_sigaltstack...done.

warning: core file may not match specified executable file.
[New LWP 69933]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Core was generated by `./segv_sigaltstack 1'.
Program terminated with signal 6, Aborted.
#0  0x00007f903136d1d7 in raise () from /lib64/libc.so.6
Missing separate debuginfos, use: debuginfo-install glibc-2.17-157.el7_3.1.x86_64
```

gdb : segv_sigaltstack (2)

```
(gdb) bt
#0  0x00007f903136d1d7 in raise () from /lib64/libc.so.6
#1  0x00007f903136e8c8 in abort () from /lib64/libc.so.6
#2  0x0000000000400b17 in sa_handler_segv (signum=11) at segv_sigaltstack.c:61
#3  <signal handler called>
#4  0x0000000000400a33 in exhaust_stack (count=<error reading variable: Cannot acce
ss memory at address 0x7ffd44489aac>) at segv_sigaltstack.c:40
#5  0x0000000000400a81 in exhaust_stack (count=94) at segv_sigaltstack.c:47
#6  0x0000000000400a81 in exhaust_stack (count=95) at segv_sigaltstack.c:47
#7  0x0000000000400a81 in exhaust_stack (count=96) at segv_sigaltstack.c:47
#8  0x0000000000400a81 in exhaust_stack (count=97) at segv_sigaltstack.c:47
#9  0x0000000000400a81 in exhaust_stack (count=98) at segv_sigaltstack.c:47
#10 0x0000000000400a81 in exhaust_stack (count=99) at segv_sigaltstack.c:47
#11 0x0000000000400a81 in exhaust_stack (count=100) at segv_sigaltstack.c:47
#12 0x0000000000400a21 in main (argc=2,argv=0x7ffd44c89c98) at segv_sigaltstack.c:35
```

gdb : segv_sigaltstack (3)

```
(gdb) p g_ss
$1 = {ss_sp = 0x1ac4010, ss_flags = 0, ss_size = 16384}

(gdb) p g_ss.ss_sp
$2 = (void *) 0x1ac4010

(gdb) p (char *)g_ss.ss_sp
$5 = 0x1ac4010 "SEGV: SigNo(11) Time(05:57:14)"
```

SIGSEGV 발생시 heap 영역에서
디버깅이나 오류 정정에 중요한 데이터를 저장할 필요가
있을 때 매우 유용하다.