ECE357, Computer Operating Systems – Problem Set #7

Nikola Janjušević

December 15, 2018

assembly process: as_show.sh

```
#!/bin/sh
if [ -z $1 ]; then
    echo "usage: ./as_show.sh assembly_code.s"
    exit 1
fi
cat $1
as --64 $1 -o a.o && ld -m elf_x86_64 a.o -o a.out
./a.out
strace ./a.out
echo $?
```

Problem 2 – pure assembly

$./as_show.sh p2.s

```
.data
    str: .ascii "COOL\n"
.text
.global _start
_start:
    movq $1,   %rax  # 1 -> sys_write
    movq $1,   %rdi  # 1 -> stdout_fileno
    movq $str, %rsi
    movq $5,   %rdx  # 5 -> num chars
    syscall

COOL
Segmentation fault (core dumped)
execve("./a.out", ["./a.out"], [/* 68 vars */]) = 0
write(1, "COOL\n", 5COOL
)                     = 5
--- SIGSEGV {si_signo=SIGSEGV, si_code=SEGV_MAPERR, si_addr=0x5} ---
+++ killed by SIGSEGV (core dumped) +++
Segmentation fault (core dumped)
139
```

Problem 3 – exit code

The previous assembly code, `p2.s`, did not make a call to the `_exit` system call and was terminated by a segmentation fault as the processor attempted to continue to execute instructions beyond the defined text region, eventually attempting to access memory that was not currently mapped in (seen in `si_code=SEGV_MAPERR`). The following code, `p3.s`, uses the `_exit` system call.

```
$./as_show.sh p3.s
```

```
.data
    str: .ascii "COOL\n"
.text
.global _start
_start:
    movq $1,   %rax  # 1 -> sys_write
    movq $1,   %rdi  # 1 -> stdout_fileno
    movq $str, %rsi
    movq $5,   %rdx  # 5 -> num chars
    syscall

    movq $60,  %rax  # 60-> sys_exit
    movq $2,   %rdi  # 2 -> exit with code 2
    syscall
COOL
execve("./a.out", ["./a.out"], [/* 68 vars */]) = 0
write(1, "COOL\n", 5COOL
)                      = 5
exit(2)                                 = ?
+++ exited with 2 +++
2
```

Note that both `strace` and `$?` verify that the program exited with the specified value, 2.

Problem 4 – system call validation

The following program calls `syscall` with an invalid system call number.

```
$./as_show.sh p4.s
```

```
.data
    str: .ascii "COOL\n"
.text
.global _start
_start:
    movq $999, %rax  # 999-> invalid syscall number
    movq $1,   %rdi  # 1 -> stdout_fileno
    movq $str, %rsi
    movq $5,   %rdx  # 5 -> num chars
    syscall

    movq $60,  %rax  # 60-> sys_exit
    movq $2,   %rdi  # 2 -> exit with code 2
    syscall
execve("./a.out", ["./a.out"], [/* 68 vars */]) = 0
syscall_999(0x1, 0x6000de, 0x5, 0, 0, 0) = -1 (errno 38)
exit(2)                                 = ?
+++ exited with 2 +++
2
```

`strace` shows that the system call fails with `errno=38`, which corresponds to failure due to an invalid system call number. The program still exits as per the specified `_exit` system call.