System Security - Attack and Defense for Binaries

CS 4390/5390, Spring *2026*

Instructor: MD Armanuzzaman (***Arman***)

Agenda

- Background knowledge
  - Compiler, linker, loader
  - x86 and x86-64 architectures and ISA
  - Linux fundamentals
    - Linux file permissions
    - Set-UID programs
    - Memory map of a Linux process
    - System calls
    - Environment and Shell variables
    - ELF files
    - Reverse engineering tools

Background Knowledge: Memory Map of a Linux Process

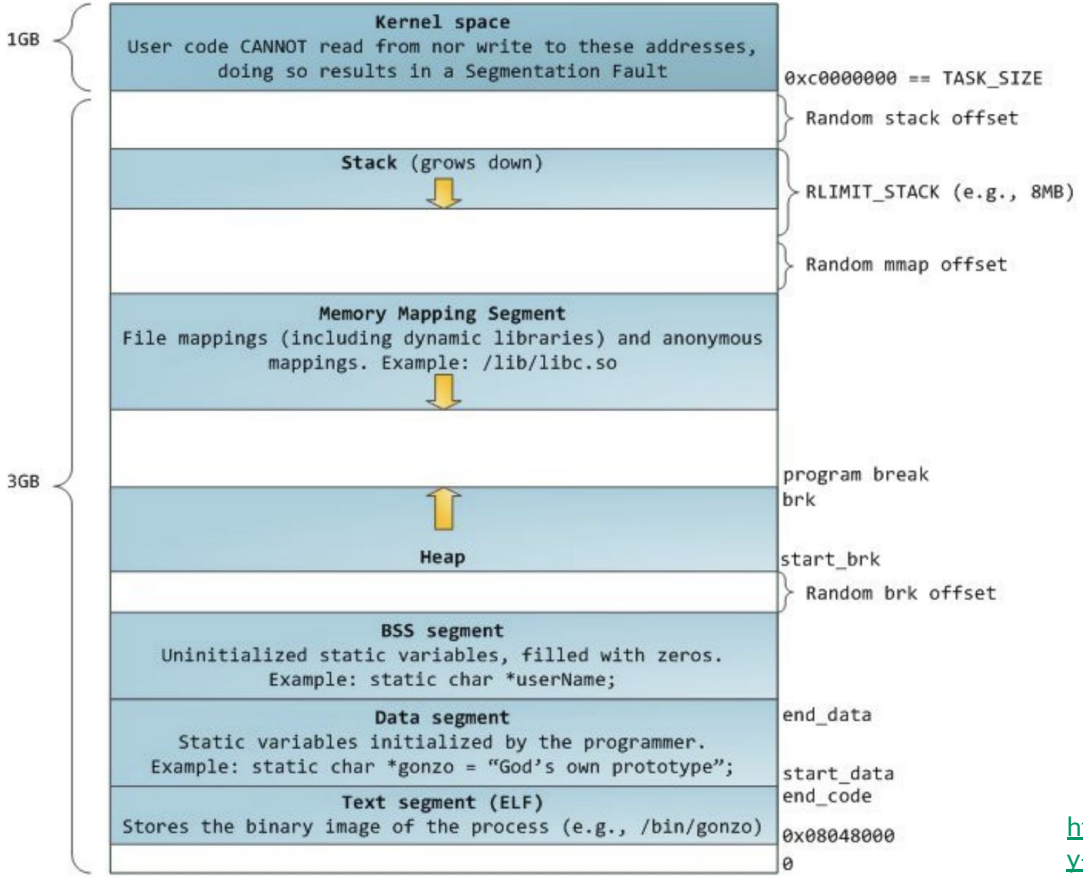
Memory Map of Linux Process (32 bit)

Each process in a multi-tasking OS runs in its own memory sandbox.

This sandbox is the **virtual address space**, which in 32-bit mode is **always a 4GB block of memory addresses**.

These virtual addresses are mapped to physical memory by **page tables**, which are maintained by the operating system kernel and consulted by the processor

Memory Map of Linux Process (32 bit)



https://manybutfinite.com/post/anatomy-of-a-program-in-memory/

NULL Pointer in C/C++

int * pInt = NULL;

In possible definitions of NULL in C/C++:

```
#define NULL ((char *)0)
#define NULL 0

//since C++11
#define NULL nullptr
```

/proc/pid_of_process/maps

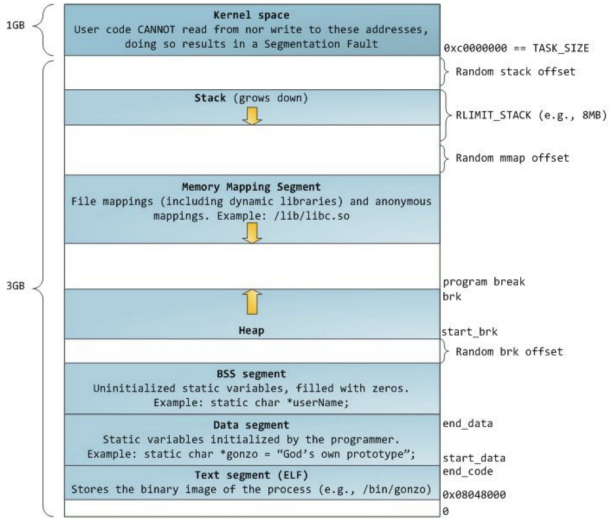
Example processmap.c

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  getchar();
  return 0;
}
```

cat /proc/pid/maps

pmap -X pid

pmap -X `pidof pm`




```
arman@aserver:~$ pmap -X 746491
746491:   ./processmap_32
  Address Perm   Offset Device   Inode Size  Rss  Pss Pss_Dirty Referenced Anonymous KSM LazyFree ShmemPmdMapped FilePmdMapped Shared_Hugetlb Private_Hugetlb Swap SwapPss Locked THPeligible ProtectionKey Mapping
5d978000 r--p 00000000  fc:00 3145956    4    4    4         0          4         0   0        0              0             0              0               0    0       0      0           0             0 processmap_32
5d979000 r-xp 00001000  fc:00 3145956    4    4    4         0          4         0   0        0              0             0              0               0    0       0      0           0             0 processmap_32
5d97a000 r--p 00002000  fc:00 3145956    4    0    0         0          0         0   0        0              0             0              0               0    0       0      0           0             0 processmap_32
5d97b000 r--p 00002000  fc:00 3145956    4    4    4         4          4         4   0        0              0             0              0               0    0       0      0           0             0 processmap_32
5d97c000 rw-p 00003000  fc:00 3145956    4    4    4         4          4         4   0        0              0             0              0               0    0       0      0           0             0 processmap_32
5e74c000 rw-p 00000000  00:00       0  136    4    4         4          4         4   0        0              0             0              0               0    0       0      0           0             0 [heap]
efde9000 r--p 00000000  fc:00 5544451  140  140  140         0        140         0   0        0              0             0              0               0    0       0      0           0             0 libc.so.6
efe0c000 r-xp 00023000  fc:00 5544451 1532  784  784         0        784         0   0        0              0             0              0               0    0       0      0           0             0 libc.so.6
eff8b000 r--p 001a2000  fc:00 5544451  532   64   64         0         64         0   0        0              0             0              0               0    0       0      0           0             0 libc.so.6
f0010000 r--p 00226000  fc:00 5544451    8    8    8         8          8         8   0        0              0             0              0               0    0       0      0           0             0 libc.so.6
f0012000 rw-p 00228000  fc:00 5544451    4    4    4         4          4         4   0        0              0             0              0               0    0       0      0           0             0 libc.so.6
f0013000 rw-p 00000000  00:00       0   40   16   16        16         16        16   0        0              0             0              0               0    0       0      0           0             0
f0024000 rw-p 00000000  00:00       0    8    8    8         8          8         8   0        0              0             0              0               0    0       0      0           0             0
f0026000 r--p 00000000  00:00       0   16    0    0         0          0         0   0        0              0             0              0               0    0       0      0           0             0 [vvar]
f002a000 r-xp 00000000  00:00       0    8    4    4         0          4         0   0        0              0             0              0               0    0       0      0           0             0 [vdso]
f002c000 r--p 00000000  fc:00 5544448    4    4    4         0          4         0   0        0              0             0              0               0    0       0      0           0             0 ld-linux.so.2
f002d000 r-xp 00001000  fc:00 5544448  140  140  140         0        140         0   0        0              0             0              0               0    0       0      0           0             0 ld-linux.so.2
f0050000 r--p 00024000  fc:00 5544448   56   56   56         0         56         0   0        0              0             0              0               0    0       0      0           0             0 ld-linux.so.2
f005e000 r--p 00031000  fc:00 5544448    8    8    8         8          8         8   0        0              0             0              0               0    0       0      0           0             0 ld-linux.so.2
f0060000 rw-p 00033000  fc:00 5544448    4    4    4         4          4         4   0        0              0             0              0               0    0       0      0           0             0 ld-linux.so.2
ff9bc000 rw-p 00000000  00:00       0  132   12   12        12         12        12   0        0              0             0              0               0    0       0      0           0             0 [stack]
                                     ==== ==== ==== ======== ========== ======== === ======== ============== ============= ============== ============== ==== ======= ====== =========== =============
                                     2788 1272 1272        72       1272        72   0        0              0             0              0               0    0       0      0           0             0 KB
```

Memory Map of Linux Process (64 bit system)

```
arman@aserver:~$ pmap -X 751600
751600:   ./processmap_64
         Address Perm   Offset Device   Inode Size  Rss Pss Pss_Dirty Referenced Anonymous KSM LazyFree ShmemPmdMapped FilePmdMapped Shared_Hugetlb Private_Hugetlb Swap SwapPss Locked THPeligible ProtectionKey Mapping
    56e4fc8ed000 r--p 00000000  fc:00 3146755    4    4   4         0          4         0   0        0              0             0              0               0    0       0      0           0             0 processmap_64
    56e4fc8ee000 r-xp 00001000  fc:00 3146755    4    4   4         0          4         0   0        0              0             0              0               0    0       0      0           0             0 processmap_64
    56e4fc8ef000 r--p 00002000  fc:00 3146755    4    0   0         0          0         0   0        0              0             0              0               0    0       0      0           0             0 processmap_64
    56e4fc8f0000 r--p 00002000  fc:00 3146755    4    4   4         4          4         4   0        0              0             0              0               0    0       0      0           0             0 processmap_64
    56e4fc8f1000 rw-p 00003000  fc:00 3146755    4    4   4         4          4         4   0        0              0             0              0               0    0       0      0           0             0 processmap_64
    56e51f43c000 rw-p 00000000  00:00       0  132    4   4         4          4         4   0        0              0             0              0               0    0       0      0           0             0 [heap]
    7d0ef6400000 r--p 00000000  fc:00 5540086  160  160   3         0        160         0   0        0              0             0              0               0    0       0      0           0             0 libc.so.6
    7d0ef6428000 r-xp 00028000  fc:00 5540086 1568  864  19         0        864         0   0        0              0             0              0               0    0       0      0           0             0 libc.so.6
    7d0ef65b0000 r--p 001b0000  fc:00 5540086  316    0   0         0          0         0   0        0              0             0              0               0    0       0      0           0             0 libc.so.6
    7d0ef65ff000 r--p 001fe000  fc:00 5540086   16   16  16        16         16        16   0        0              0             0              0               0    0       0      0           0             0 libc.so.6
    7d0ef6603000 rw-p 00202000  fc:00 5540086    8    8   8         8          8         8   0        0              0             0              0               0    0       0      0           0             0 libc.so.6
    7d0ef6605000 rw-p 00000000  00:00       0   52   20  20        20         20        20   0        0              0             0              0               0    0       0      0           0             0
    7d0ef67fa000 rw-p 00000000  00:00       0   12    8   8         8          8         8   0        0              0             0              0               0    0       0      0           0             0
    7d0ef6804000 rw-p 00000000  00:00       0    8    4   4         4          4         4   0        0              0             0              0               0    0       0      0           0             0
    7d0ef6806000 r--p 00000000  fc:00 5540083    4    4   0         0          4         0   0        0              0             0              0               0    0       0      0           0             0 ld-linux-x86-64.so.2
    7d0ef6807000 r-xp 00001000  fc:00 5540083  172  172   3         0        172         0   0        0              0             0              0               0    0       0      0           0             0 ld-linux-x86-64.so.2
    7d0ef6832000 r--p 0002c000  fc:00 5540083   40   40   0         0         40         0   0        0              0             0              0               0    0       0      0           0             0 ld-linux-x86-64.so.2
    7d0ef683c000 r--p 00036000  fc:00 5540083    8    8   8         8          8         8   0        0              0             0              0               0    0       0      0           0             0 ld-linux-x86-64.so.2
    7d0ef683e000 rw-p 00038000  fc:00 5540083    8    8   8         8          8         8   0        0              0             0              0               0    0       0      0           0             0 ld-linux-x86-64.so.2
    7ffe0aeac000 rw-p 00000000  00:00       0  132   12  12        12         12        12   0        0              0             0              0               0    0       0      0           0             0 [stack]
    7ffe0afb9000 r--p 00000000  00:00       0   16    0   0         0          0         0   0        0              0             0              0               0    0       0      0           0             0 [vvar]
    7ffe0afbd000 r-xp 00000000  00:00       0    8    4   0         0          4         0   0        0              0             0              0               0    0       0      0           0             0 [vdso]
ffffffffff600000 --xp 00000000  00:00       0    4    0   0         0          0         0   0        0              0             0              0               0    0       0      0           0             0 [vsyscall]
                                              ==== ==== === ======== ========== ========= === ======== ============== ============= ============== ============== ==== ======= ====== ========== =============
                                              2684 1348 129        96       1348        96   0        0              0             0              0               0    0       0      0           0             0 KB
```

Background Knowledge: System Calls

What is System Call?

When a process needs to invoke a kernel service, it invokes a procedure call in the operating system interface using special instructions (not a **call** instruction in x86). Such a procedure is called a system call.

The system call enters the kernel; the kernel performs the service and returns. Thus a process alternates between executing in user space and kernel space.

System calls are generally not invoked directly by a program, but rather via wrapper functions in glibc (or perhaps some other library).

Popular System Call

On Unix, Unix-like and other POSIX-compliant operating systems, popular system calls are open, read, write, close, wait, exec, fork, exit, and kill.

Many modern operating systems have hundreds of system calls. For example, Linux and OpenBSD each have over 300 different calls, FreeBSD has over 500, Windows 7 has close to 700.

Glibc interfaces

Often, but not always, the name of the wrapper function is the same as the name of the system call that it invokes

For example, glibc contains a function **chdir()** which invokes the underlying "chdir" system call.

Tools: strace & ltrace

```
ctf@misc_firstflag_64:/$ strace ./misc_firstflag_64
execve("./misc_firstflag_64", ["./misc_firstflag_64"], 0x7ffc81da94a0 /* 13 vars */) = 0
access("/etc/suid-debug", F_OK)         = -1 ENOENT (No such file or directory)
brk(NULL)                               = 0x5ca81e6ff000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffec0373bf0) = -1 EINVAL (Invalid argument)
fcntl(0, F_GETFD)                       = 0
fcntl(1, F_GETFD)                       = 0
fcntl(2, F_GETFD)                       = 0
access("/etc/suid-debug", F_OK)         = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=47355, ...}) = 0
mmap(NULL, 47355, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7a8c6a314000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7a8c6a312000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7a8c6a120000
mprotect(0x7a8c6a145000, 1847296, PROT_NONE) = 0
mmap(0x7a8c6a145000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7a8c6a145000
mmap(0x7a8c6a2bd000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0x7a8c6a2bd000
mmap(0x7a8c6a308000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7a8c6a308000
mmap(0x7a8c6a30e000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7a8c6a30e000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7a8c6a313540) = 0
mprotect(0x7a8c6a308000, 12288, PROT_READ) = 0
mprotect(0x5ca7f9cd0000, 4096, PROT_READ) = 0
mprotect(0x7a8c6a34d000, 4096, PROT_READ) = 0
munmap(0x7a8c6a314000, 47355)           = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
brk(NULL)                               = 0x5ca81e6ff000
brk(0x5ca81e720000)                     = 0x5ca81e720000
write(1, "Congratulations on getting your "..., 45Congratulations on getting your first flag!!
) = 45
openat(AT_FDCWD, "/flag", O_RDONLY)     = -1 EACCES (Permission denied)
write(1, "Error: Cannot open the flag file"..., 36Error: Cannot open the flag file!!!
) = 36
exit_group(0)                           = ?
+++ exited with 0 +++
```

misc/firstflag

main.c

```
int main(int argc, char *argv[])
{
  printf("Congratulations on getting your first flag!!\n");
  print_flag();
}
```

flag.h

```
int print_flag()
{
        FILE *fp = NULL;
        char buff[MAX_FLAG_SIZE] = {0};
        fp = fopen("/flag", "r");
        if (fp == NULL)
{
         printf("Error: Cannot open the flag file!!!\n");
        return 1;
 }
        fread(buff, MAX_FLAG_SIZE - 2, 1, fp);
        printf("The flag is: %s\n", buff);
        fclose(fp);
        return 0;
}
```

Tools: strace & ltrace

```
ctf@misc_firstflag_64:/$ strace ./misc_firstflag_64
execve("./misc_firstflag_64", ["./misc_firstflag_64"], 0x7ffc81da94a0 /* 13 vars */) = 0
access("/etc/suid-debug", F_OK)         = -1 ENOENT (No such file or directory)
brk(NULL)                               = 0x5ca81e6ff000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffec0373bf0) = -1 EINVAL (Invalid argument)
fcntl(0, F_GETFD)                       = 0
fcntl(1, F_GETFD)                       = 0
fcntl(2, F_GETFD)                       = 0
access("/etc/suid-debug", F_OK)         = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=47355, ...}) = 0
mmap(NULL, 47355, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7a8c6a314000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0"..., 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7a8c6a312000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0"..., 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7a8c6a120000
mprotect(0x7a8c6a145000, 1847296, PROT_NONE) = 0
mmap(0x7a8c6a145000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7a8c6a145000
mmap(0x7a8c6a2bd000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0x7a8c6a2bd000
mmap(0x7a8c6a308000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7a8c6a308000
mmap(0x7a8c6a30e000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7a8c6a30e000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7a8c6a313540) = 0
mprotect(0x7a8c6a308000, 12288, PROT_READ) = 0
mprotect(0x5ca7f9cd0000, 4096, PROT_READ) = 0
mprotect(0x7a8c6a34d000, 4096, PROT_READ) = 0
munmap(0x7a8c6a314000, 47355)           = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
brk(NULL)                               = 0x5ca81e6ff000
brk(0x5ca81e720000)                     = 0x5ca81e720000
write(1, "Congratulations on getting your "..., 45Congratulations on getting your first flag!!
) = 45
openat(AT_FDCWD, "/flag", O_RDONLY)     = -1 EACCES (Permission denied)
write(1, "Error: Cannot open the flag file"..., 36Error: Cannot open the flag file!!!
) = 36
exit_group(0)                           = ?
+++ exited with 0 +++
```

Execve - first system call
Access - check file permission
Brk - check data segment/heap
Arch_prctl - set architecture-specific thread state
Fcntl - manipulate file descriptor
Openat - similar to open
Fstat - get file status
Mmap - map files or devices into memory
Close
Read
Pread64 - similar to read
Mprotect - set protection on a region of memory
Munmap - map files or devices into memory
Write
Exit_group

Use “man 2 syscall_name” to check out its usage

Making a System Call in x86/64 Assembly

On x86/x86-64, most system calls rely on the software interrupt.

A software interrupt is caused either by an exceptional condition in the processor itself, or a special instruction (the **int 0x80** instruction or **syscall** instruction)

For example: a divide-by-zero exception will be thrown if the processor's arithmetic logic unit is commanded to divide a number by zero as this instruction is in error and impossible.

Making a System Call in x86 Assembly (INT 0x80)

x86 (32-bit)

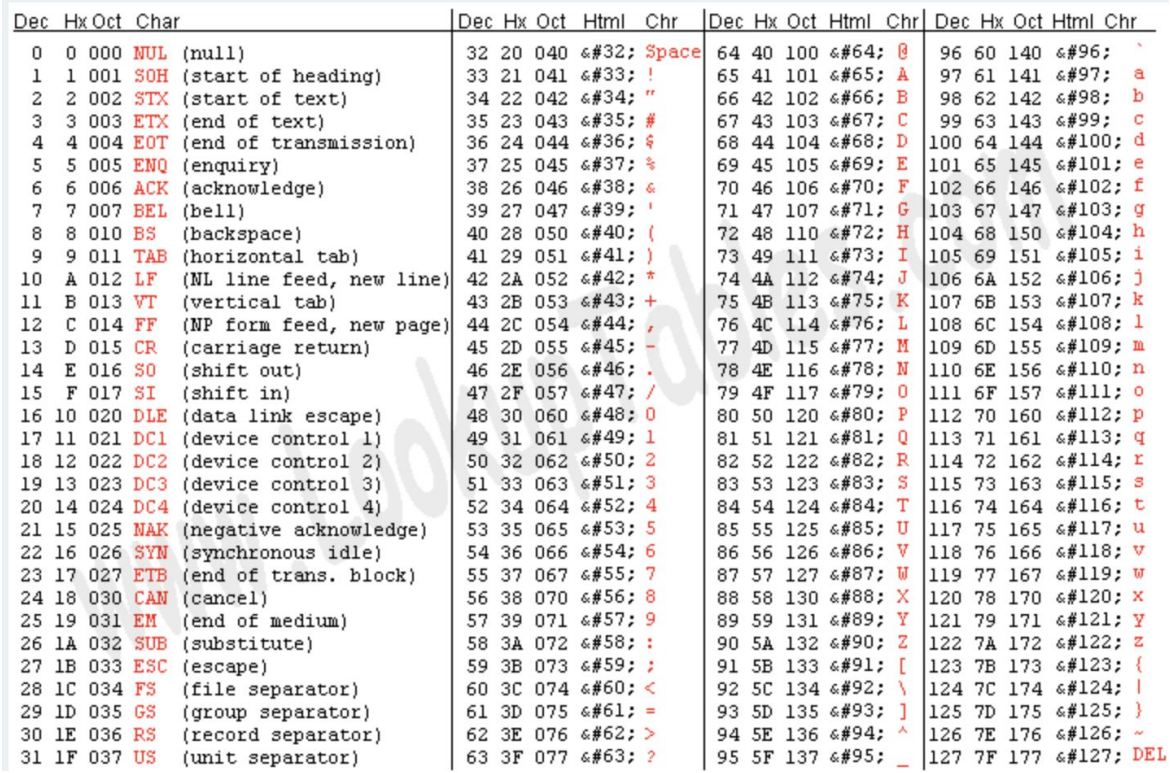
Compiled from Linux 4.14.0 headers.

NR	syscall name	references	%eax	arg0 (%ebx)	arg1 (%ecx)	arg2 (%edx)	arg3 (%esi)	arg4 (%edi)	arg5 (%ebp)
0	restart_syscall	man/ cs/	0x00	-	-	-	-	-	-
1	exit	man/ cs/	0x01	int error_code	-	-	-	-	-
2	fork	man/ cs/	0x02	-	-	-	-	-	-
3	read	man/ cs/	0x03	unsigned int fd	char *buf	size_t count	-	-	-
4	write	man/ cs/	0x04	unsigned int fd	const char *buf	size_t count	-	-	-
5	open	man/ cs/	0x05	const char *filename	int flags	umode_t mode	-	-	-
6	close	man/ cs/	0x06	unsigned int fd	-	-	-	-	-
7	waitpid	man/ cs/	0x07	pid_t pid	int *stat_addr	int options	-	-	-
8	creat	man/ cs/	0x08	const char *pathname	umode_t mode	-	-	-	-
9	link	man/ cs/	0x09	const char *oldname	const char *newname	-	-	-	-
10	unlink	man/ cs/	0x0a	const char *pathname	-	-	-	-	-
11	execve	man/ cs/	0x0b	const char *filename	const char *const *argv	const char *const *envp	-	-	-
12	chdir	man/ cs/	0x0c	const char *filename	-	-	-	-	-
13	time	man/ cs/	0x0d	time_t *tloc	-	-	-	-	-
14	mknod	man/ cs/	0x0e	const char *filename	umode_t mode	unsigned dev	-	-	-
15	chmod	man/ cs/	0x0f	const char *filename	umode_t mode	-	-	-	-
16	lchown	man/ cs/	0x10	const char *filename	uid_t user	gid_t group	-	-	-
17	break	man/ cs/	0x11	?	?	?	?	?	?

https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md#x86-32_bit

Making a System Call in x86 Assembly

```
xor eax,eax
push eax
push 0x68732f2f
push 0x6e69622f
mov ebx,esp
push eax
push ebx
mov ecx,esp
mov al,0xb
int 0x80
```

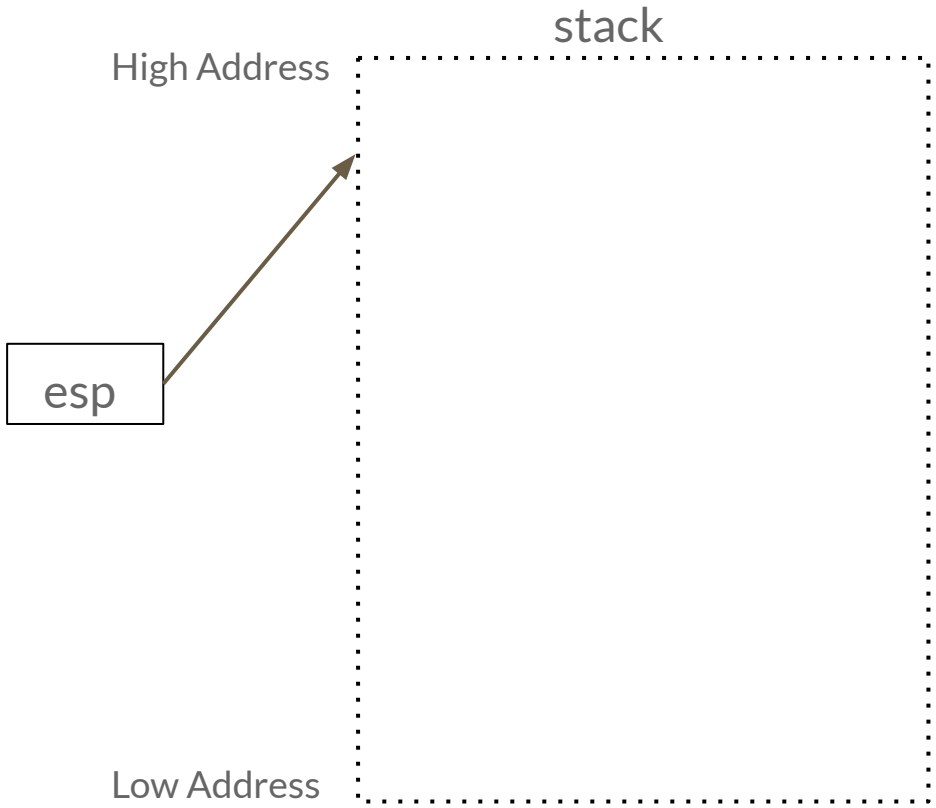


Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

https://shell-storm.org/shellcode/files/shellcode-827.html

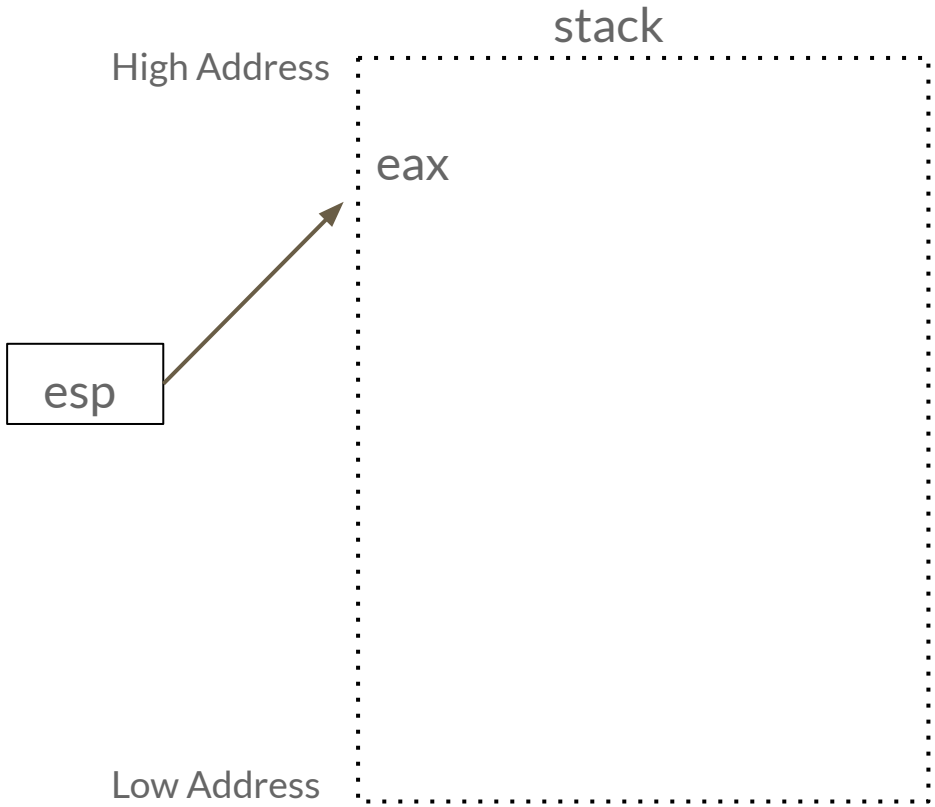
Making a System Call in x86 Assembly

```
xor eax,eax
push eax
push 0x68732f2f
push 0x6e69622f
mov ebx,esp
push eax
push ebx
mov ecx,esp
mov al,0xb
int 0x80
```



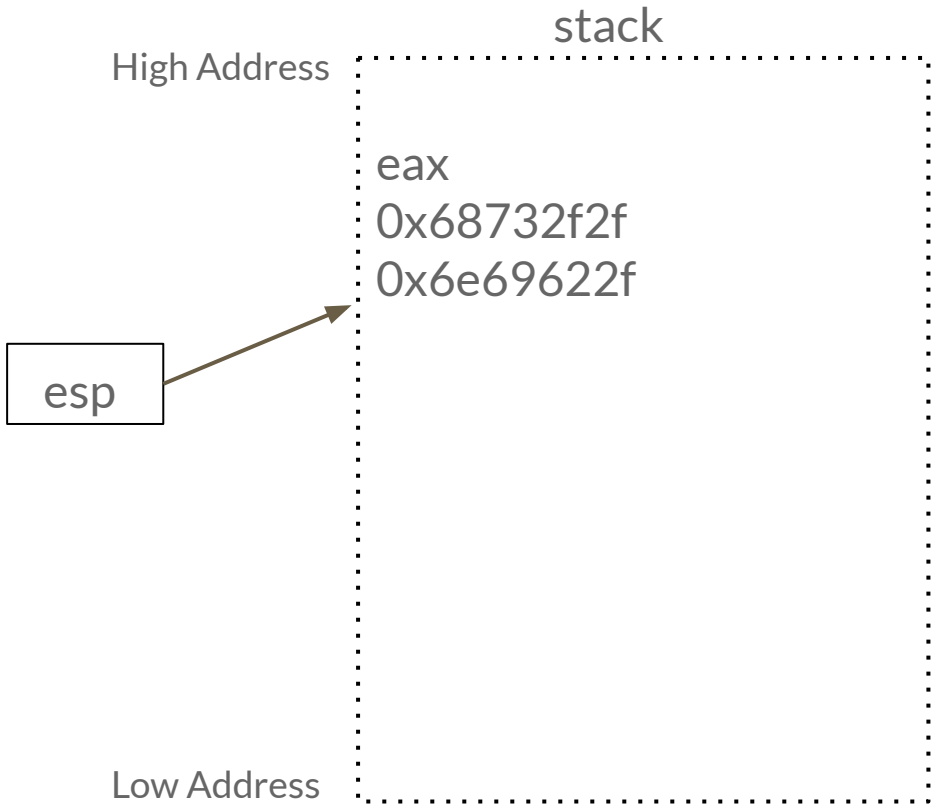
Making a System Call in x86 Assembly

```
xor eax,eax
push eax
push 0x68732f2f
push 0x6e69622f
mov ebx,esp
push eax
push ebx
mov ecx,esp
mov al,0xb
int 0x80
```



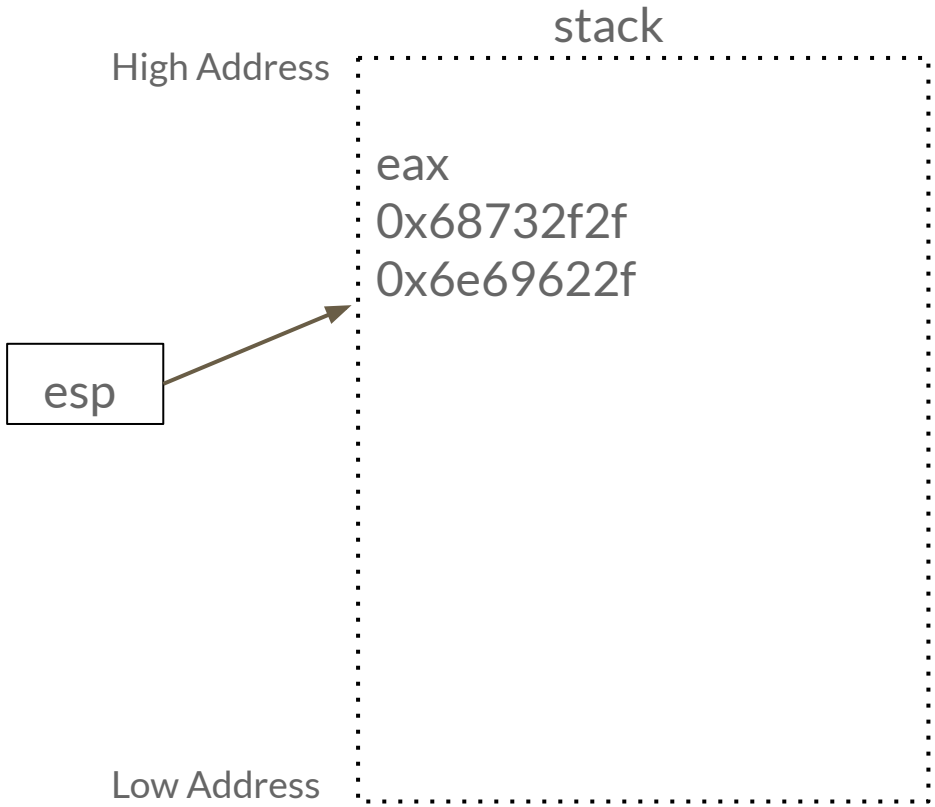
Making a System Call in x86 Assembly

```
xor eax,eax
push eax
push 0x68732f2f
push 0x6e69622f
mov ebx,esp
push eax
push ebx
mov ecx,esp
mov al,0xb
int 0x80
```



Making a System Call in x86 Assembly

```
xor eax,eax
push eax
push 0x68732f2f
push 0x6e69622f
mov ebx,esp
push eax
push ebx
mov ecx,esp
mov al,0xb
int 0x80
```



Making a System Call in x86 Assembly

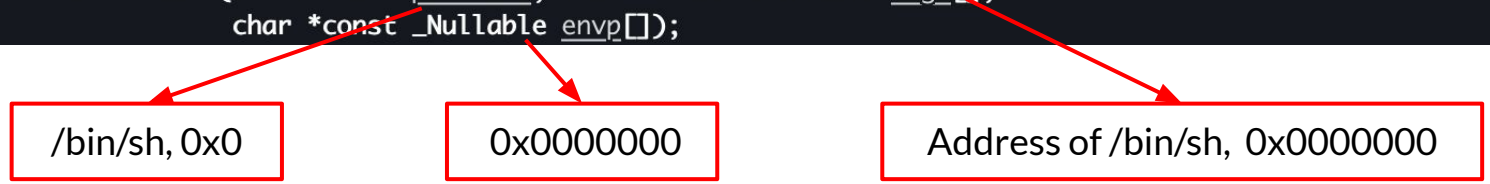
```
execve(2)                                                    System Calls Manual

NAME
       execve - execute program

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <unistd.h>

       int execve(const char *pathname, char *const _Nullable argv[],
                  char *const _Nullable envp[]);
```



execve(“/bin/sh”, address of string “/bin/sh”, 0)

Making a System Call in x86_64 (64-bit) Assembly

x86_64 (64-bit)

Compiled from Linux 4.14.0 headers.

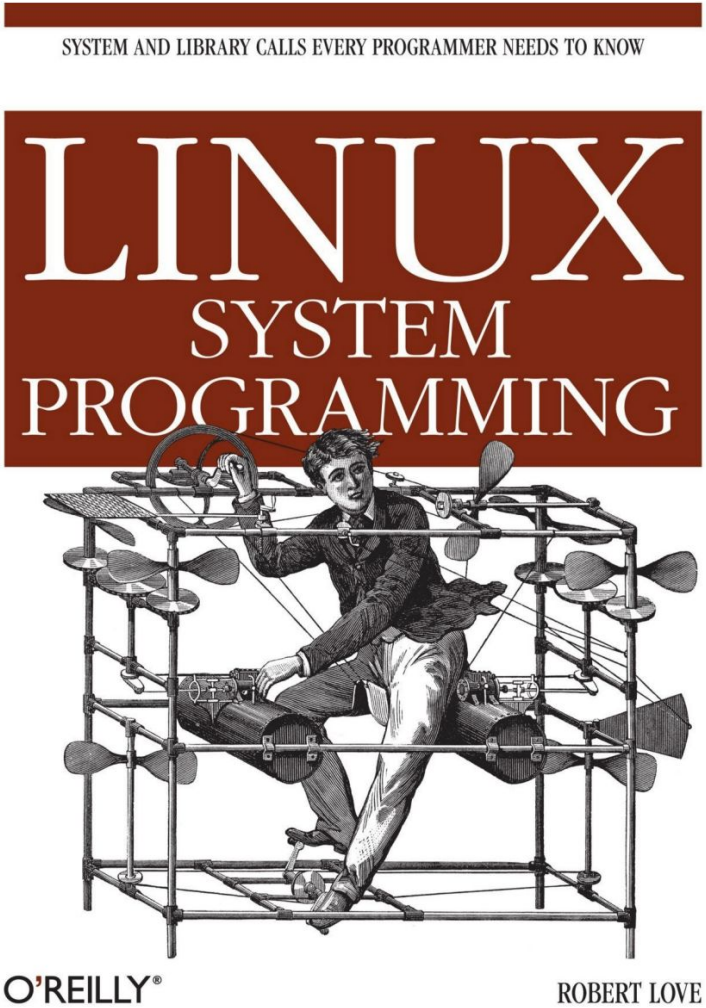
NR	syscall name	references	%rax	arg0 (%rdi)	arg1 (%rsi)	arg2 (%rdx)	arg3 (%r10)	arg4 (%r8)	arg5 (%r9)
0	read	man/ cs/	0x00	unsigned int fd	char *buf	size_t count	-	-	-
1	write	man/ cs/	0x01	unsigned int fd	const char *buf	size_t count	-	-	-
2	open	man/ cs/	0x02	const char *filename	int flags	umode_t mode	-	-	-
3	close	man/ cs/	0x03	unsigned int fd	-	-	-	-	-
4	stat	man/ cs/	0x04	const char *filename	struct __old_kernel_stat *statbuf	-	-	-	-
5	fstat	man/ cs/	0x05	unsigned int fd	struct __old_kernel_stat *statbuf	-	-	-	-
6	lstat	man/ cs/	0x06	const char *filename	struct __old_kernel_stat *statbuf	-	-	-	-
7	poll	man/ cs/	0x07	struct pollfd *ufds	unsigned int nfds	int timeout	-	-	-
8	lseek	man/ cs/	0x08	unsigned int fd	off_t offset	unsigned int whence	-	-	-
9	mmap	man/ cs/	0x09	?	?	?	?	?	?
10	mprotect	man/ cs/	0x0a	unsigned long start	size_t len	unsigned long prot	-	-	-
11	munmap	man/ cs/	0x0b	unsigned long addr	size_t len	-	-	-	-
12	brk	man/ cs/	0x0c	unsigned long brk	-	-	-	-	-
13	rt_sigaction	man/ cs/	0x0d	int	const struct sigaction *	struct sigaction *	size_t	-	-
14	rt_sigprocmask	man/ cs/	0x0e	int how	sigset_t *set	sigset_t *oset	size_t sigsetsize	-	-
15	rt_sigreturn	man/ cs/	0x0f	?	?	?	?	?	?
16	ioctl	man/ cs/	0x10	unsigned int fd	unsigned int cmd	unsigned long arg	-	-	-

https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md#x86-32_bit

Making a System Call in x86_64 (64-bit) Assembly

NR	syscall name	references	%rax	arg0 (%rdi)	arg1 (%rsi)	arg2 (%rdx)	arg3 (%r10)	arg4 (%r8)	arg5 (%r9)
59	execve	man/ cs/	0x3b	const char *filename	const char *const *argv	const char *const *envp	-	-	-

```
push rax
xor rdx, rdx
xor rsi, rsi
mov rbx,'/bin//sh'
push rbx
push rsp
pop rdi
mov al, 59
syscall
```


SYSTEM AND LIBRARY CALLS EVERY PROGRAMMER NEEDS TO KNOW
LINUX
SYSTEM
PROGRAMMING
O'REILLY®
ROBERT LOVE

Background Knowledge: System Calls

Channels of Communication for Linux Process

Every process in Linux has three initial, standard channels of communication:

- Standard Input (stdin, fd=0) is the channel through which the process takes input. For example, your shell uses Standard Input to read the commands that you input.
- Standard Output (stdout, fd=1) is the channel through which processes output normal data, such as the flag when it is printed to you in previous challenges or the output of utilities such as ls.
- Standard Error (stderr, fd=2) is the channel through which processes output error details. For example, if you mistype a command, the shell will output, over standard error, that this command does not exist.

Examples

Redirecting output > or 1>

echo hi > asdf echo hi 1> asdf

Appending output >>

echo hi >> asdf

Redirecting errors 2>

/challenge/run 2> errors.log

Redirecting input <

rev < messagefile

Channels of Communication for Linux Process

Process can also take input from command line arguments

ls -al c

at /flag

cat 1.txt 2.txt 3.txt

Pipe

The | (pipe) operator. Standard output from the command to the left of the pipe will be connected to (piped into) the standard input of the command to the right of the pipe.

echo hello-world | wc -c

Background Knowledge: Environment and Shell Variables

Environment and Shell Variables

Environment and Shell variables are a set of dynamic **named values,** stored within the system that are used by applications launched in shells.

KEY=value

KEY="Some other value"

KEY=value1:value2

The names of the variables are case-sensitive (UPPER CASE). Multiple values must be separated by the colon : character. There is no space around the equals = symbol.

Environment and Shell Variables

Environment variables are variables that are available system-wide and are inherited by all spawned child processes and shells.

Shell variables are variables that apply only to the current shell instance. Each shell such as zsh and bash, has its own set of internal shell variables.

Common Environment Variables

USER - The current logged in user.

HOME - The home directory of the current user.

EDITOR - The default file editor to be used. This is the editor that will be used when you type edit in your terminal.

SHELL - The path of the current user's shell, such as bash or zsh.

LOGNAME - The name of the current user.

PATH - A list of directories to be searched when executing commands.

LANG - The current locales settings.

TERM - The current terminal emulation.

MAIL - Location of where the current user's mail is stored.

Commands

env – The command allows you to run another program in a custom environment without modifying the current one. When used without an argument it will print a list of the current environment variables.

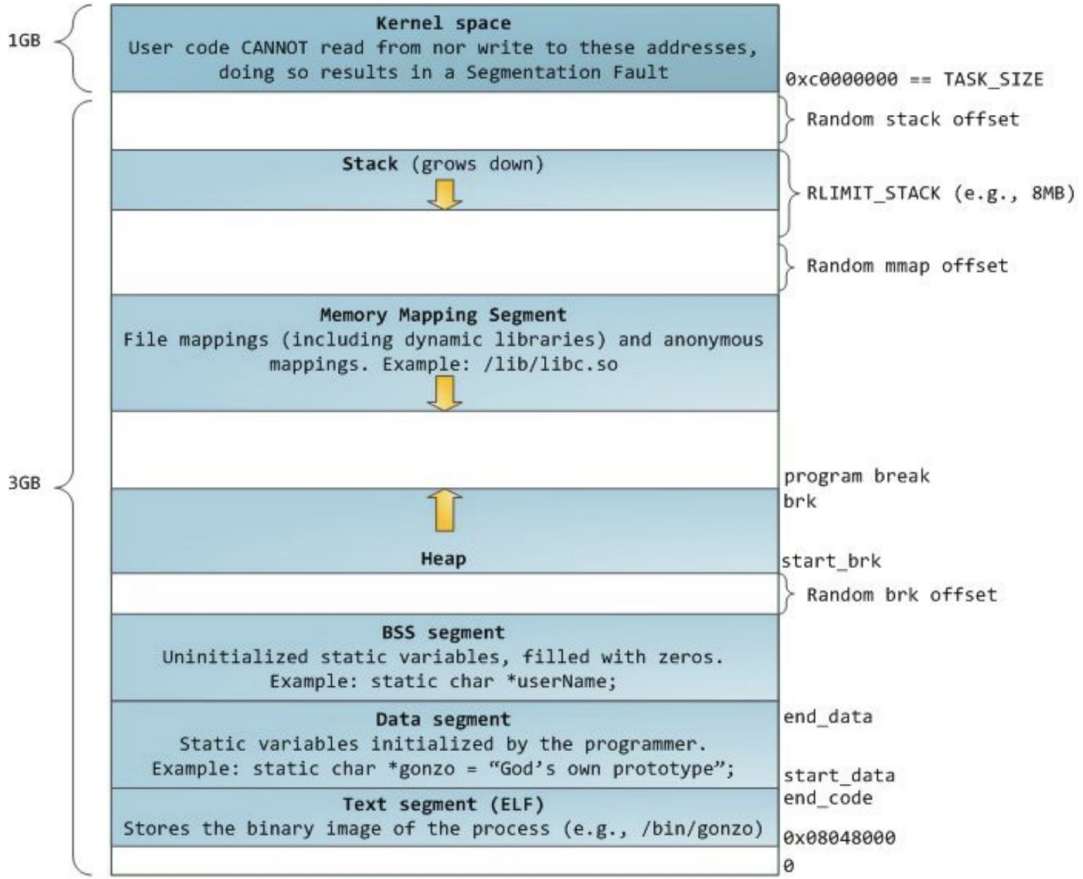
printenv – The command prints all or the specified environment variables.

set – The command sets or unsets shell variables. When used without an argument it will print a list of all variables including environment and shell variables, and shell functions.

unset – The command deletes shell and environment variables.

export – The command sets environment variables

The environment variables live towards the top of the stack, together with command line arguments



Background Knowledge: Manual Binary Analysis Tools

Tools for this class

file

readelf

strings

nm

objdump

GDB

[optional] IDA Pro

[optional] ghidra

[optional] Binary Ninja

GDB Cheat Sheet

Start gdb using:

```
gdb <binary>
```

Pass initial commands for gdb through a file

```
gdb <binary> –x <initfile>
```

To start the program and breakpoint at main()

```
start <argv>
```

To start the program and breakpoint at _start

```
start <argv>
```

To run the program without breakpoint

```
r <argv>
```

Use another progrom’s output as stdin in GDB:

```
r <<< $(python2 -c "print '\x12\x34'*5")
```

GDB Cheat Sheet

Set breakpoint at address:

```
b *0x80000000
```

Set breakpoint at beginning of a function:

```
b main
```

```
....
```

```
b <filename:line number>
```

```
b <line number>
```

Disassemble 10 instructions from an address:

```
x/10i 0x80000000
```

Exam 15 dword (w) from an address; show hex (x):

```
x/15wx 0x80000000
```

Exam 3 qword (g) from an address; show hex (x):

```
x/3gx 0x80000000
```

GDB Cheat Sheet

To show breakpoints

```
info b
```

To remove breakpoints

```
clear <function name>
clear *<instruction address>
clear <filename:line number>
clear <line number>
```

GDB Cheat Sheet

Use “examine” or “x” command

x/32xw <memory location> to see memory contents at memory location, showing 32 hexadecimal words

x/5s <memory location> to show 5 strings (null terminated) at a particular memory location

x/10i <memory location> to show 10 instructions at particular memory location

See registers

info reg

Step an instruction

si

GDB Script

Use “examine” or “x” command

x/32xw <memory location> to see memory contents at memory location, showing 32 hexadecimal words

x/5s <memory location> to show 5 strings (null terminated) at a particular memory location

x/10i <memory location> to show 10 instructions at particular memory location

See registers

info reg

Step an instruction

si

Shell Cheat Sheet

Run a program and use another program's output as a parameter

```
program $(python2 -c "print '\x12\x34'*5")
```

Reading

1. https://ig.thc.org/how-does-linux-start-a-process