CS342 Operating Systems

Homework 1

Ata Seren

21901575

Section-2

1. Before reading the book, I went over the presentations to remember the lectures and refresh my knowledge of the concepts. Then I started to read the book but I also opened the presentations to make sure of the topics. The first thing I realize that the book has lots of details about the topics and concept we discussed in the lectures. Even these details are sometimes hard to understand and remember, I made sure of that I learned the concepts very well.

2. First of all, I needed to decide whether I download and setup Ubuntu directly on my hard drive or create a virtual machine. I chose the latter option since I have the experience of setting up a virtual machine from my internship and it is safer and easier than setting up on directly my hard drive. For virtual machine, I downloaded Oracle VM VirtualBox which I know to use. Then, I downloaded the Ubuntu version mentioned in the homework document, from its official website.

   Before proceeding, I found a tutorial on YouTube about setting up the Ubuntu on a virtual machine with VirtualBox. I tried to follow the same process for the setup to prevent any errors caused by myself. I have a laptop with 16 GB of RAM and 1 TB of HDD. I used 8 GB of the RAM and 80 GB of HDD for the virtual machine. I perfectly set the virtual machine and ran it. Additionally, I downloaded extensions that allow the window to fit my monitor which is a property that isn't default.

   I had few Linux experiences in some CS classes since we use Dijkstra machine for projects and homework. However, I also used Linux for my extracurricular trainings to improve myself on cyber security. These are the 10 of the Linux commands I learned:

   - cd: used to navigate through the files and directories
   - pwd: displays the full path of the current working directory
   - ls: displays the contents of the current working directory
   - mkdir: creates a new directory with given parameters such as name and location of the directory
   - touch: used to create a blank new file with a desired file type
   - cp: used to copy a file to a desired location
   - sudo: used to perform tasks that require administrative or root permissions
   - chmod: used to change the read, write, and execute permissions of files and directories
   - uname: displays detailed information about Linux system like the machine name, operating system, kernel, etc.
   - tar: used to archive multiple files into a tarball which is a common Linux file format similar to zip format

3. The kernel executable is named as "vmlinuz" and located under "/boot".

   Version of the running kernel is "5.13.0-28-generic" which I got using "uname -r" command.

4. From kernel.org, I downloaded 5.15.19 version of the kernel which is the closest available version to my running kernel's version. In the kernel file, in /linux-5.15.19/ location specifically, I found following directories:

    - arch
    - block
    - certs
    - crypto
    - Documentation
    - drivers
    - fs
    - include
    - init
    - ipc
    - kernel
    - lib
    - LICENSES
    - mm
    - net
    - samples
    - scripts
    - security
    - sound
    - tools
    - usr
    - virt

5. The system call definition of linux-5.15.19 is at the location:
   /linux-5.15.19/arch/x86/entry/syscalls/syscall_64.tbl
   It is named as "syscall_64.tbl" and there is also 32-bit version of the same file named as "syscall_32.tbl".

   System call names corresponding to system call numbers in the homework document are:
    - 3: close
    - 35: nanosleep
    - 110: getppid
    - 210: io_cancel

6. The "strace" command is used to capture, monitor, and troubleshoot the programs in the system. It records and intercepts the system calls, which is the property I used for this homework.

I used “strace ls” command and received the following output:

```
execve("/usr/bin/ls", ["ls"], 0x7fff6dcd3b60 /* 58 vars
*/) = 0
brk(NULL)                                = 0x55bd4673f000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffdbfc68d50) = -1
EINVAL (Invalid argument)
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=74018, ...}) = 0
mmap(NULL, 74018, PROT_READ, MAP_PRIVATE, 3, 0) =
0x7eff5860c000
close(3)                                 = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libselinux.so.1",
O_RDONLY|O_CLOEXEC) = 3
read(3,
"\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0@p\0\0\0\0\
0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=163200, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7eff5860a000
mmap(NULL, 174600, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE,
3, 0) = 0x7eff585df000
mprotect(0x7eff585e5000, 135168, PROT_NONE) = 0
mmap(0x7eff585e5000, 102400, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) =
0x7eff585e5000
mmap(0x7eff585fe000, 28672, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1f000) =
0x7eff585fe000
mmap(0x7eff58606000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) =
0x7eff58606000
mmap(0x7eff58608000, 6664, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7eff58608000
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
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) = 0x7eff583ed000
mprotect(0x7eff58412000, 1847296, PROT_NONE) = 0
mmap(0x7eff58412000, 1540096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) =
0x7eff58412000
mmap(0x7eff5858a000, 303104, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) =
0x7eff5858a000
mmap(0x7eff585d5000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) =
0x7eff585d5000
mmap(0x7eff585db000, 13528, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7eff585db000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpcre2-8.so.0",
O_RDONLY|O_CLOEXEC) = 3
read(3,
"\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\340\"\0\0\
0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=584392, ...}) = 0
mmap(NULL, 586536, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE,
3, 0) = 0x7eff5835d000
mmap(0x7eff5835f000, 409600, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x7eff5835f000
mmap(0x7eff583c3000, 163840, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x66000) =
0x7eff583c3000
mmap(0x7eff583eb000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8d000) =
0x7eff583eb000
close(3)                                = 0
```

```
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libdl.so.2",
O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0
\22\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=18816, ...}) = 0
mmap(NULL, 20752, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7eff58357000
mmap(0x7eff58358000, 8192, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) =
0x7eff58358000
mmap(0x7eff5835a000, 4096, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) =
0x7eff5835a000
mmap(0x7eff5835b000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) =
0x7eff5835b000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpthread.so.0",
O_RDONLY|O_CLOEXEC) = 3
read(3,
"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\201\0\
0\0\0\0\0"..., 832) = 832
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301
V)Yf]\223\337"..., 68, 824) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=157224, ...}) = 0
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301
V)Yf]\223\337"..., 68, 824) = 68
mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE,
3, 0) = 0x7eff58334000
mmap(0x7eff5833b000, 69632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) =
0x7eff5833b000
mmap(0x7eff5834c000, 20480, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) =
0x7eff5834c000
mmap(0x7eff58351000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) =
0x7eff58351000
mmap(0x7eff58353000, 13432, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7eff58353000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7eff58332000
arch_prctl(ARCH_SET_FS, 0x7eff58333400) = 0
mprotect(0x7eff585d5000, 12288, PROT_READ) = 0
mprotect(0x7eff58351000, 4096, PROT_READ) = 0
mprotect(0x7eff5835b000, 4096, PROT_READ) = 0
mprotect(0x7eff583eb000, 4096, PROT_READ) = 0
```

```
mprotect(0x7eff58606000, 4096, PROT_READ) = 0
mprotect(0x55bd44c81000, 4096, PROT_READ) = 0
mprotect(0x7eff5864c000, 4096, PROT_READ) = 0
munmap(0x7eff5860c000, 74018)           = 0
set_tid_address(0x7eff583336d0)         = 3585
set_robust_list(0x7eff583336e0, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7eff5833bbf0,
sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO,
sa_restorer=0x7eff583493c0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7eff5833bc90,
sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO,
sa_restorer=0x7eff583493c0}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
statfs("/sys/fs/selinux", 0x7ffdbfc68ca0) = -1 ENOENT (No
such file or directory)
statfs("/selinux", 0x7ffdbfc68ca0)      = -1 ENOENT (No
such file or directory)
brk(NULL)                               = 0x55bd4673f000
brk(0x55bd46760000)                     = 0x55bd46760000
openat(AT_FDCWD, "/proc/filesystems", O_RDONLY|O_CLOEXEC)
= 3
fstat(3, {st_mode=S_IFREG|0444, st_size=0, ...}) = 0
read(3, "nodev\tsysfs\nnodev\ttmpfs\nnodev\tbd"..., 1024)
= 369
read(3, "", 1024)                       = 0
close(3)                                = 0
access("/etc/selinux/config", F_OK)     = -1 ENOENT (No
such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/locale-archive",
O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=17339232, ...}) =
0
mmap(NULL, 17339232, PROT_READ, MAP_PRIVATE, 3, 0) =
0x7eff572a8000
close(3)                                = 0
ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0
ioctl(1, TIOCGWINSZ, {ws_row=24, ws_col=80, ws_xpixel=0,
ws_ypixel=0}) = 0
openat(AT_FDCWD, ".",
O_RDONLY|O_NONBLOCK|O_CLOEXEC|O_DIRECTORY) = 3
fstat(3, {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
getdents64(3, /* 26 entries */, 32768)  = 904
getdents64(3, /* 0 entries */, 32768)   = 0
close(3)                                = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0),
...}) = 0
write(1, "Desktop  Documents  Downloads  M"..., 72Desktop
Documents  Downloads  Music  Pictures  Public
     Templates  Videos
```

```
) = 72
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

7. The “time” command is used to determine how long a given command takes to run. It provides timing statistics about the execution of a specified command. Following are the outputs and the definitions of this command:
    - real: The time from start to finish of the call of the command. It can also be defined as the time from the moment the user hit the Enter key until the moment the command is completed.
    - user: The amount of CPU time spent in the user mode.
    - sys: The amount of CPU time spent in the kernel mode.

    Following tables are the outputs of time command for different commands:

    Time strace ls

real	0m0,014s
user	0m0,009s
sys	0m0,000s

    time cp textfile.txt textcopy.txt

real	0m0,001s
user	0m0,002s
sys	0m0,000s

    time cd Desktop

real	0m0,000s
user	0m0,000s
sys	0m0,000s

    time ls

real	0m0,001s
user	0m0,001s
sys	0m0,000s

time touch texttouch.txt

real	0m0,001s
user	0m0,001s
sys	0m0,000s

8. I implemented a C code that creates a linked list with 10000 random integers. I measured the time passed during this insertion by using gettimeofday(). Following code and Makefile successfully run on VSCode on my host PC with Windows 10 OS and Linux terminal on my virtual machine with Ubuntu.

   Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/time.h>

struct Node {
    int data;
    struct Node* next;
};

void printList(struct Node* n)
{
    while (n != NULL) {
        printf("%d ", n->data);
        n = n->next;
    }
}

//Adds a node with given integer to a linked list
void addNewNode(struct Node** head, int number)
{
    //New node has been created
    struct Node* newNode = (struct Node*)
malloc(sizeof(struct Node));

    //New pointer has been created to traverse on the
linked list
    struct Node *pointToList = *head;

    //Data and pointer of the node has been set
    newNode->data = number;
    newNode->next = NULL;

    //In case if list is empty
    if (*head == NULL)
    {
```

```
        *head = newNode;
    }
    else
    {
        while (pointToList->next != NULL)
        {
            pointToList = pointToList->next;
        }

        //New node has been inserted to the linked list by
        //using the pointer of last node to point new node
        pointToList->next = newNode;
    }
}
int main()
{
    struct Node* head = NULL;

    srand(time(NULL));
    /*
    printf("%d ", rand());
    printf("%d ", rand());
    printf("%d ", rand());
    */

    //To save the time when insertion starts
    struct timeval beforeInsertion;
    gettimeofday(&beforeInsertion, NULL);

    //Insertion of 10000 nodes with random numbers
    for(int i = 0; i < 10000; i++)
    {
        addNewNode(&head, rand());
    }

    //To save the time when insertion ends
    struct timeval afterInsertion;
    gettimeofday(&afterInsertion, NULL);
    printf("Execution of the insertion of 10000 nodes with
random values took %ld seconds and %ld microseconds\n",
afterInsertion.tv_sec - beforeInsertion.tv_sec,
    afterInsertion.tv_sec - beforeInsertion.tv_sec+
afterInsertion.tv_usec - beforeInsertion.tv_usec);
    return 0;
}
```

Makefile:

```
all: list
list: list.c
     gcc -o list list.c
clean:
     rm -f list list.o *~
```