Rüzgar Ayan 21801984

2020-2021 SPRING CS342 HOMEWORK 1 Report

Task 1.

In order to install Ubuntu, I first downloaded the VirtualBox software and the .iso file of the given specific release of Ubuntu. With some help from an online guide on VirtualBox, I have created a virtual machine using the downloaded .iso file. I have allocated to the virtual machine half of my RAM, half of my CPU cores and 20 GBs of my disk space during the installation steps. After successfully starting the virtual machine, the Linux commands I've learned and used are as follows:

- touch: To create a .c file to write the C program.
- man: To read and learn the descriptions of the other commands.
- cat: To read the contents of the .c file that I have created and some other files.
- clear: To clear the terminal after long and ugly outputs.
- grep: To search for a specific text in a file.
- df: To display the disk usage of the system.
- diff: To compare two files.
- tar and zip: To compress the homework in the end.
- ping: To check the internet connection of the virtual machine by pinging a server.
- history: To see my previously used commands in the terminal.

Task 2.

I have found the kernel executable in the location /boot with the name vmlinuz-5.8.0-41-generic. The output of the "uname –r" command was accordingly 5.8.0-41-generic.

Task 3.

I have downloaded the kernel with the name linux-5.10.12.tar.xz since it was the closest one to my current kernel version. Its subdirectories were:

- arch
- block
- certs
- crypto

- Documentation
- drivers
- fs
- include
- init
- ipc
- kernel
- lib
- LICENCES
- mm
- net
- samples
- scripts
- security
- sound
- tools
- ysr
- virt

Task 4.

By searching syscall in the downloaded kernel, I was able to find the system call tables with the names **"syscall_32.tbl"** and **"syscall_64.tbl"**. They were in /linux-5.10.12/arch/x86/entry/syscalls/. For writing the system call names below, I have used the **"syscall_64.tbl"** one.

- System call number = 3 System call name = close
- System call number = 35 System call name = nanosleep
- System call number = 110 System call name = getppid
- System call number = 210 System call name = io_cancel

Task 5.

I learned that **strace** is used for tracing the system calls of the following command. I tried it with the "ls" command.

Sample output for "strace ls":

```
zzlawlzz@zzlawlzz-VirtualBox:~/Desktop$ strace ls
execve("/usr/bin/ls", ["ls"], 0x7ffe7846dde0 /* 61 vars */) = 0
brk(NULL)                               = 0x55e8659fd000
```

```
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffff7ffb870) = -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=73485, ...}) = 0
mmap(NULL, 73485, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f274c3ed000
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\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) =
0x7f274c3eb000
mmap(NULL, 174600, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f274c3c0000
mprotect(0x7f274c3c6000, 135168, PROT_NONE) = 0
mmap(0x7f274c3c6000, 102400, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x6000) = 0x7f274c3c6000
mmap(0x7f274c3df000, 28672, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1f000) = 0x7f274c3df000
mmap(0x7f274c3e7000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x26000) = 0x7f274c3e7000
mmap(0x7f274c3e9000, 6664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f274c3e9000
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) = 0x7f274c1ce000
mprotect(0x7f274c1f3000, 1847296, PROT_NONE) = 0
mmap(0x7f274c1f3000, 1540096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f274c1f3000
mmap(0x7f274c36b000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x19d000) = 0x7f274c36b000
mmap(0x7f274c3b6000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1e7000) = 0x7f274c3b6000
mmap(0x7f274c3bc000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f274c3bc000
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\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) = 0x7f274c13e000
mmap(0x7f274c140000, 409600, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x7f274c140000
mmap(0x7f274c1a4000, 163840, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x66000) = 0x7f274c1a4000
mmap(0x7f274c1cc000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x8d000) = 0x7f274c1cc000
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) = 0x7f274c138000
mmap(0x7f274c139000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1000) = 0x7f274c139000
mmap(0x7f274c13b000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000)
= 0x7f274c13b000
mmap(0x7f274c13c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x3000) = 0x7f274c13c000
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\301V)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\301V)Yf]\223\337"..., 68, 824)
= 68
mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f274c115000
mmap(0x7f274c11c000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x7000) = 0x7f274c11c000
mmap(0x7f274c12d000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x18000) = 0x7f274c12d000
mmap(0x7f274c132000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1c000) = 0x7f274c132000
mmap(0x7f274c134000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f274c134000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f274c113000
arch_prctl(ARCH_SET_FS, 0x7f274c114400) = 0
mprotect(0x7f274c3b6000, 12288, PROT_READ) = 0
mprotect(0x7f274c132000, 4096, PROT_READ) = 0
mprotect(0x7f274c13c000, 4096, PROT_READ) = 0
mprotect(0x7f274c1cc000, 4096, PROT_READ) = 0
mprotect(0x7f274c3e7000, 4096, PROT_READ) = 0
mprotect(0x55e86469e000, 4096, PROT_READ) = 0
mprotect(0x7f274c42c000, 4096, PROT_READ) = 0
munmap(0x7f274c3ed000, 73485)           = 0
set_tid_address(0x7f274c1146d0)         = 3808
set_robust_list(0x7f274c1146e0, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7f274c11cbf0, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f274c12a3c0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f274c11cc90, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f274c12a3c0}, 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", 0x7ffff7ffb7c0) = -1 ENOENT (No such file or directory)
statfs("/selinux", 0x7ffff7ffb7c0)      = -1 ENOENT (No such file or directory)
brk(NULL)                               = 0x55e8659fd000
brk(0x55e865a1e000)                     = 0x55e865a1e000
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) = 373
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=8291184, ...}) = 0
```

```
mmap(NULL, 8291184, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f274b92a000
close(3)                                = 0
ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0
ioctl(1, TIOCGWINSZ, {ws_row=54, ws_col=177, 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, /* 11 entries */, 32768)  = 320
getdents64(3, /* 0 entries */, 32768)   = 0
close(3)                                = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "index.html  index.html.1  list\tl"..., 75index.html  index.html.1  list
list.c  list.h  Makefile  prog  prog.c  program
) = 75
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Task 6.

By searching on the web, I learned that **real** shows the total time elapsed from the beginning of the command call to the termination. **user** and **sys** shows the **CPU time** used in the user and kernel modes, respectively. The table below shows the commands that I've tried to time.

Command	real	user	sys
time cp file1 copy (100MB file)	0m0.241s	0m0.005s	0m0.144s
time man ls	0m2.497s	0m0.046	0m0.039
time cd Desktop	0m0.000s	0m0.000s	0m0.000s
time wget google.com	0m0.318s	0m0.002s	0m0.003s

Task 7.

My implementation of linked list in **"list.c"** together with the main() function:

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

struct ListNode {
    int data;
    struct ListNode* next;
};
typedef struct ListNode* ListNode;

struct List {
    ListNode head;
    int size;
};
```

```
typedef struct List* List;

List initList()
{
    List l;
    l = malloc(sizeof(struct List));
    ListNode head = NULL;
    l->head = head;
    l->size = 0;

    return l;
}

void insertFirst(List l, int data)
{
    ListNode newNode = malloc(sizeof(struct ListNode));
    newNode->data = data;

    ListNode oldHead = l->head;
    l->head = newNode;

    l->size = l->size + 1;
    if (oldHead == NULL)
    {
        newNode->next = NULL;
        return;
    }
    newNode->next = oldHead;
}


int deleteFirst(List l)
{
    if (l->size == 0)
    {
        return -1;
    }

    int data = l->head->data;
    ListNode head = l->head;
    l->head = head->next;
    free(head);
    l->size = l->size - 1;

    return data;
}

void deleteList(List l)
{
    while (l->size > 0)
    {
        deleteFirst(l);
    }
    free(l);
}

int getSize(List l)
```

```
{
    return l->size;
}

int main()
{
    struct timeval start_time;
    gettimeofday(&start_time, NULL);
    int ADD_CNT = 10000;
    srand(time(NULL));

    List list = initList();

    int i;
    for (i = 0; i < ADD_CNT; i++)
    {
        insertFirst(list, rand());
    }

    deleteList(list);

    struct timeval end_time;
    gettimeofday(&end_time, NULL);
    long elapsedMicroSeconds = (end_time.tv_usec - start_time.tv_usec);
    printf("%lu\n", elapsedMicroSeconds);

    return 0;
}
```

For the Makefile, I have taken the template from the assignment and removed “-g“ and “-Wall” since they were causing errors and I couldn’t find out what they are for. So, the Makefile I used was:

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

After using the commands “make” and “./list”, the output I got was **432** which is the time it took for the program to insert 10000 numbers to the linked list in microseconds.