**Pınar Yücel**
**21802188**

# CS342 Operating Systems - Spring 2021
# Homework #1

---

1. I installed Linux in a virtual machine to be able to use both Windows and Linux simultaneously on my computer and to avoid getting affected significantly by a possible system error in the future. I chose VirtualBox as virtualization software because it is maintained by Oracle. I followed a video step by step for the installation and I did not encounter any problems or errors during the process. I learned the following Linux commands:

    - cd : Changes between directories.
    - ls: Lists directory contents.
    - clear: Clears the terminal screen.
    - mkdir: Makes directories.
    - pwd: Prints the name of the working directory.
    - cp: Copies files and directories.
    - mv: Moves or renames files.
    - rm: Removes files or directories.
    - rmdir: Removes empty directories.
    - touch: Creates new empty files.

2. The kernel executable is named vmlinuz and located in the /boot directory of the root. Version number: 5.8.0-41-generic

3. I downloaded the 5.10.13 version. The name of the subdirectories:

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

- tools
- usr
- virt

4. Pathname: /linux-5.10.13/arch/x86/entry/syscalls/syscall_64.tbl

    - 3: close
    - 35: nanosleep
    - 110: getppid
    - 210: io_cancel

5. Strace traces the system calls and signals. Sample output for strace ls:

```
1.  ubuntu@ubuntu-VirtualBox:~$ strace ls
2.  execve("/usr/bin/ls", ["ls"], 0x7ffd4c31f580 /* 60 vars */) = 0
3.  brk(NULL)                           = 0x565146f4e000
4.  arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd543f0530) = -1 EINVAL (Invalid argument)
5.  access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
6.  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
7.  fstat(3, {st_mode=S_IFREG|0644, st_size=66821, ...}) = 0
8.  mmap(NULL, 66821, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9f987bd000
9.  close(3)                            = 0
10. openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3
11. 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
12. fstat(3, {st_mode=S_IFREG|0644, st_size=163200, ...}) = 0
13. mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
    0x7f9f987bb000
14. mmap(NULL, 174600, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9f98790000
15. mprotect(0x7f9f98796000, 135168, PROT_NONE) = 0
16. mmap(0x7f9f98796000, 102400, PROT_READ|PROT_EXEC,
    MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7f9f98796000
17. mmap(0x7f9f987af000, 28672, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
    0x1f000) = 0x7f9f987af000
18. mmap(0x7f9f987b7000, 8192, PROT_READ|PROT_WRITE,
    MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7f9f987b7000
19. mmap(0x7f9f987b9000, 6664, PROT_READ|PROT_WRITE,
    MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f9f987b9000
20. close(3)                            = 0
21. openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
22. 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
23. 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\0\0"..., 784, 64) = 784
24. 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
25. 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
26. fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
27. 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\0\0"..., 784, 64) = 784
28. 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
29. 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
30. mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9f9859e000
31. mprotect(0x7f9f985c3000, 1847296, PROT_NONE) = 0
32. mmap(0x7f9f985c3000, 1540096, PROT_READ|PROT_EXEC,
    MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f9f985c3000
33. mmap(0x7f9f9873b000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
    0x19d000) = 0x7f9f9873b000
34. mmap(0x7f9f98786000, 24576, PROT_READ|PROT_WRITE,
    MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f9f98786000
35. mmap(0x7f9f9878c000, 13528, PROT_READ|PROT_WRITE,
    MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f9f9878c000
36. close(3)                            = 0
37. openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpcre2-8.so.0", O_RDONLY|O_CLOEXEC) = 3
```

```
38. 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
39. fstat(3, {st_mode=S_IFREG|0644, st_size=584392, ...}) = 0
40. mmap(NULL, 586536, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9f9850e000
41. mmap(0x7f9f98510000, 409600, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f9f98510000
42. mmap(0x7f9f98574000, 163840, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x66000) = 0x7f9f98574000
43. mmap(0x7f9f9859c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8d000) = 0x7f9f9859c000
44. close(3) = 0
45. openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libdl.so.2", O_RDONLY|O_CLOEXEC) = 3
46. 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
47. fstat(3, {st_mode=S_IFREG|0644, st_size=18816, ...}) = 0
48. mmap(NULL, 20752, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9f98508000
49. mmap(0x7f9f98509000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f9f98509000
50. mmap(0x7f9f9850b000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f9f9850b000
51. mmap(0x7f9f9850c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f9f9850c000
52. close(3) = 0
53. openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3
54. 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
55. 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
56. fstat(3, {st_mode=S_IFREG|0755, st_size=157224, ...}) = 0
57. 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
58. mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9f984e5000
59. mmap(0x7f9f984ec000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f9f984ec000
60. mmap(0x7f9f984fd000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) = 0x7f9f984fd000
61. mmap(0x7f9f98502000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7f9f98502000
62. mmap(0x7f9f98504000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f9f98504000
63. close(3) = 0
64. mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f9f984e3000
65. arch_prctl(ARCH_SET_FS, 0x7f9f984e4400) = 0
66. mprotect(0x7f9f98786000, 12288, PROT_READ) = 0
67. mprotect(0x7f9f98502000, 4096, PROT_READ) = 0
68. mprotect(0x7f9f9850c000, 4096, PROT_READ) = 0
69. mprotect(0x7f9f9859c000, 4096, PROT_READ) = 0
70. mprotect(0x7f9f987b7000, 4096, PROT_READ) = 0
71. mprotect(0x5651468ae000, 4096, PROT_READ) = 0
72. mprotect(0x7f9f987fb000, 4096, PROT_READ) = 0
73. munmap(0x7f9f987bd000, 66821) = 0
74. set_tid_address(0x7f9f984e46d0) = 4318
75. set_robust_list(0x7f9f984e46e0, 24) = 0
76. rt_sigaction(SIGRTMIN, {sa_handler=0x7f9f984ecbf0, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f9f984fa3c0}, NULL, 8) = 0
77. rt_sigaction(SIGRT_1, {sa_handler=0x7f9f984ecc90, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f9f984fa3c0}, NULL, 8) = 0
78. rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
79. prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
80. statfs("/sys/fs/selinux", 0x7ffd543f0480) = -1 ENOENT (No such file or directory)
81. statfs("/selinux", 0x7ffd543f0480) = -1 ENOENT (No such file or directory)
82. brk(NULL) = 0x565146f4e000
83. brk(0x565146f6f000) = 0x565146f6f000
84. openat(AT_FDCWD, "/proc/filesystems", O_RDONLY|O_CLOEXEC) = 3
85. fstat(3, {st_mode=S_IFREG|0444, st_size=0, ...}) = 0
86. read(3, "nodev\tsysfs\nnodev\ttmpfs\nnodev\tbd"..., 1024) = 369
87. read(3, "", 1024) = 0
88. close(3) = 0
89. access("/etc/selinux/config", F_OK) = -1 ENOENT (No such file or directory)
90. openat(AT_FDCWD, "/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
```

```
fstat(3, {st_mode=S_IFREG|0644, st_size=8500896, ...}) = 0
mmap(NULL, 8500896, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9f97cc7000
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, /* 25 entries */, 32768)  = 872
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 +++
```

6. Time command runs programs and summarizes system resource usage.

    - real: The total time from start to finish of the call. It is the time from the moment you press Enter until the moment the command returns.
    - user: The amount of CPU time that is spent in the user mode.
    - sys: The amount of CPU time that is spent in the kernel mode.

| command (including arguments) | real | user | sys |
|---|---|---|---|
| time strace cp | 0.021s | 0.010s | 0.000s |
| time strace ls | 0.017s | 0.003s | 0.005s |
| time ls | 0.001s | 0.000s | 0.001s |
| time cp a aCopy | 0.002s | 0.000s | 0.001s |
| time cd Desktop/ | 0.000s | 0.000s | 0.000s |

Table 1: Time statistics for different program executions

7. Source code of list.c:

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

typedef struct node
{
    int data;
    struct node *next;
} node, *np, **npp;
```

```
11.
12.  // Creates a dummy head and returns it
13.  np createList()
14.  {
15.      // Allocate memory for the head node
16.      np hp = (struct node *)malloc(sizeof(struct node));
17.
18.      // Assign data -1 as a flag to indicate root
19.      hp->data = -1;
20.
21.      hp->next = NULL;
22.      return hp;
23.  }
24.
25.  void push(np head, int newElement)
26.  {
27.      // Allocate memory for the new node
28.      np nnp = (struct node *)malloc(sizeof(struct node));
29.
30.      nnp->data = newElement;
31.      nnp->next = NULL;
32.
33.      // Get the tail node pointer
34.      while (head->next != NULL)
35.      {
36.          head = head->next;
37.      }
38.
39.      // New node becomes the tail
40.      head->next = nnp;
41.  }
42.
43.  // Returns the current time in microseconds
44.  unsigned long getCurrentTime()
45.  {
46.      struct timeval timeValue;
47.      gettimeofday(&timeValue, NULL);
48.      unsigned long currentTime = timeValue.tv_usec;
```

```
49.         currentTime = currentTime + timeValue.tv_sec * 1e6;
50.         return currentTime;
51.     }
52.
53.     // Pushes indicated number of elements to the linked list
    and prints the time taken for pushing in micro seconds
54.     void initializeListRandomly(np head, int size)
55.     {
56.         // Safeguard
57.         if (head == NULL)
58.         {
59.             return;
60.         }
61.
62.         unsigned long start;
63.         unsigned long end;
64.         int i = 0;
65.         srand(time(NULL));
66.
67.         start = getCurrentTime();
68.
69.         for (i = 0; i < size; i++)
70.         {
71.             push(head, rand());
72.         }
73.
74.         end = getCurrentTime();
75.
76.           printf("Time taken to push %d random elements to the
    linked list is %ld micro seconds\n", size, end - start);
77.     }
78.
79.     // Deletes all of the nodes of a linked list
80.     void deleteList(npp hpp)
81.     {
82.         np current = *hpp;
83.         np next;
84.
```

```
85.         while (current != NULL)
86.         {
87.             next = current->next;
88.             free(current);
89.             current = next;
90.         }
91.
92.         *hpp = NULL;
93.     }
94.
95.     // Returns the size of a linked list excluding the dummy
    head
96.     int listSize(np head)
97.     {
98.         if (head == NULL)
99.         {
100.            return 0;
101.        }
102.
103.        int size = 0;
104.
105.        for (int i = 1; head->next != NULL; i++)
106.        {
107.            head = head->next;
108.            size = i;
109.        }
110.
111.        return size;
112.    }
113.
114.    // Prints the elements of a linked list excluding the dummy
    head
115.    void printList(np head)
116.    {
117.        if (head != NULL)
118.        {
119.            if (head->data != -1)
120.            {
```

```
            printf("%d\n", head->data);
        }

        printList(head->next);
    }
}

int main()
{
    // Create the dummy head
    np hp = createList();

    printf("List size: %d\n", listSize(hp));
    initializeListRandomly(hp, 10000);
    printf("List size: %d\n", listSize(hp));

    // Deallocate memory
    deleteList(&hp);
}
```