Bilkent University
Fall 2023-2024

Department of Computer Engineering

CS342
Operating Systems

Homework #1

Deniz Tuna Onguner
22001788

Section 01
İbrahim Körpeoğlu

24 September 2023

1. Choices, Experiences, and Issues Encountered regarding the Process of the Installation of Linux/Ubuntu

I embarked on my journey to install Ubuntu on my MacBook, initially installing it on a partition of my disk. Unfortunately, my initial endeavor came to a halt, leading me to pivot towards the utilization of a virtual machine. It became evident that Ubuntu does not seamlessly support MacBooks released post-2018 [6], and my own is a 2019 model. This compatibility hiccup manifested itself through the non-responsive nature of my keyboard and touchpad when I attempted to navigate on the Ubuntu Installer. This unforeseen challenge served as a catalyst, compelling me to reassess my strategy and ultimately transition to a virtual machine as my preferred solution.

Subsequently, I found myself in a situation where I had to format my MacBook and undergo the reinstallation of macOS. Regrettably, I was unable to rectify the unintended repercussions stemming from my previous attempts at installing Ubuntu. In my quest for a solution, I reached out to Apple Support, seeking their expertise and guidance. Their prescribed remedy was to initiate a comprehensive disk wipe, effectively erasing all data, and then proceed with a complete reinstallation of the macOS operating system.

Following the successful reinstallation of macOS, I was now prepared to embark on the journey of setting up a virtual machine to run Ubuntu. Drawing from the wisdom of friends who had previously navigated this path successfully, I decided to heed their advice and opted for *VirtualBox* [7] as my preferred virtualization software.

The subsequent steps in the process proved to be relatively easier. I began by installing the ISO file of the specified Ubuntu version and then diligently followed the instructions provided by VirtualBox. This sequence of actions culminated in the successful completion of the Ubuntu installation within the virtual environment. Notably, I encountered a minor hurdle along the way, which was the need to adjust the default language settings to access the terminal and proceed with the setup.

2. Ten Linux Commands Learned

Command	Description
ssh	The ssh command is to connect to an external machine on the network with the use of the ssh protocol [1].
cd	The cd command is to navigate through directories [1].
scp	The scp command allows the secure transferring of files between the local host and the remote host or between two remote hosts [2].
mkdir	The mkdir command is to create new directories from within the terminal [1].
sudo	The sudo command allows users to run commands with privileges that only the root user has [3].
rm	The rm command is used to delete/remove files and folders [1].
pwd	The pwd command is to print the current working directory onto the terminal [1].
touch	To create a new file, the touch command is used [1].
ls	The ls command lists all the files and directories under a specified directory [4].
clear	The clear command is for clearing the terminal screen [5].

3. Ubuntu 22.04 LTS Default Kernel

The location where the **kernel executable** resides in as the follows:

/boot/vmlinuz-{kernel-version}

/boot/vmlinuz-6.2.0-33-generic

So, the current kernel version is ***6.2.0-33-generic***.

4. New Kernel Download

The subdirectories in the freshly downloaded kernel *(version: 6.1.55)* are as the follows:

```
~/Downloads/linux-6.1.55$ ls -a
```

```
.               CREDITS                  include
..              crypto                   init
arch            Documentation            io_uring
block           drivers                  ipc
certs           fs                       Kbuild
.clang-format   .get_maintainer.ignore   Kconfig
.cocciconfig    .gitattributes           kernel
COPYING         .gitignore               lib

LICENCES        rustfmt.toml
.mailmap        samples
MAINTAINERS     scripts
Makefile        security
mm              sound
net             tools
README          usr
rust            virt
```

5. The System Call Table of the New Kernel

The system call table can be found at the following directory:

```
~/Downloads/linux-6.1.55/arch/x86/entry/syscalls/
```

The file of the table searched is named as *“syscall_64.tbl”*.

```
0       common    read          sys_read
1       common    write         sys_write
2       common    open          sys_open
3       common    close         sys_close
4       common    stat          sys_newstat
5       common    fstat         sys_newfstat
6       common    lstat         sys_newlstat
35      common    nanosleep     sys_ nanosleep
100     common    times         sys_times
160     common    setrlimit     sys_ setrlimit
```

6. Tracing the System Calls

Sample output for the `strace` command:

```
$ strace ls
```

```
execve("/usr/bin/ls", ["ls"], 0x7ffdecdd9a10 /* 55 vars */) = 0
brk(NULL)                               = 0x563e17a47000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffdfa57b640) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f69d1c6b000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=58663, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 58663, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f69d1c5c000
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\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=166280, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 177672, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f69d1c30000
mprotect(0x7f69d1c36000, 139264, PROT_NONE) = 0
mmap(0x7f69d1c36000, 106496, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7f69d1c36000
mmap(0x7f69d1c50000, 28672, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x20000) = 0x7f69d1c50000
mmap(0x7f69d1c58000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x27000) = 0x7f69d1c58000
mmap(0x7f69d1c5a000, 5640, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f69d1c5a000
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\0P\237\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\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\"\233}\305\t\5?\344\337^)\350b\231\21\360"..., 68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, AT_EMPTY_PATH) = 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\0"..., 784, 64) = 784
mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f69d1a00000
mmap(0x7f69d1a28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f69d1a28000
mmap(0x7f69d1bbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f69d1bbd000
mmap(0x7f69d1c15000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7f69d1c15000
mmap(0x7f69d1c1b000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f69d1c1b000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f69d1c2d000
arch_prctl(ARCH_SET_FS, 0x7f69d1c2d800) = 0
set_tid_address(0x7f69d1c2dad0)         = 3335
set_robust_list(0x7f69d1c2dae0, 24)     = 0
rseq(0x7f69d1c2e1a0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f69d1c15000, 16384, PROT_READ) = 0
mprotect(0x7f69d19fe000, 4096, PROT_READ) = 0
mprotect(0x7f69d1c58000, 4096, PROT_READ) = 0
mprotect(0x563e16f4d000, 4096, PROT_READ) = 0
mprotect(0x7f69d1ca5000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f69d1c5c000, 58663)           = 0
statfs("/sys/fs/selinux", 0x7ffdfa57b680) = -1 ENOENT (No such file or directory)
statfs("/selinux", 0x7ffdfa57b680)      = -1 ENOENT (No such file or directory)
getrandom("\xcc\x14\x04\x53\x5c\x74\x4d\x2a", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x563e17a47000
brk(0x563e17a68000)                     = 0x563e17a68000
openat(AT_FDCWD, "/proc/filesystems", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0444, st_size=0, ...}, AT_EMPTY_PATH) = 0
read(3, "nodev\tsysfs\nnodev\ttmpfs\nnodev\tbd"..., 1024) = 387
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=8852304, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 8852304, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f69d1000000
close(3)                                = 0
ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0
ioctl(1, TIOCGWINSZ, {ws_row=40, ws_col=132, ws_xpixel=0, ws_ypixel=0}) = 0
openat(AT_FDCWD, ".", O_RDONLY|O_NONBLOCK|O_CLOEXEC|O_DIRECTORY) = 3
newfstatat(3, "", {st_mode=S_IFDIR|0750, st_size=4096, ...}, AT_EMPTY_PATH) = 0
getdents64(3, 0x563e17a4eab0 /* 20 entries */, 32768) = 632
getdents64(3, 0x563e17a4eab0 /* 0 entries */, 32768) = 0
close(3)                                = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
write(1, "Desktop  Documents  Downloads  M"..., 78Desktop  Documents  Downloads  Music  Pictures       Public  snap  Templates  Vid
eos
) = 78
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

7. Execution Times of the System Calls

```
$ time ssh tuna.onguner@dijkstra.ug.bcc.bilkent.edu.tr
```

```
tuna.onguner@dijkstra.ug.bcc.bilkent.edu.tr's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Sep 23 17:41:57 2023 from 139.179.221.147
[tuna.onguner@dijkstra ~]$ exit
logout
Connection to dijkstra.ug.bcc.bilkent.edu.tr closed.

real    0m9,609s
user    0m0,017s
sys     0m0,009s
```

`real:` This is the actual wall-clock time that elapsed while the command was running. It includes not only the time the CPU spent executing the command but also any time the process was waiting, such as I/O operations or time spent waiting for other resources [8].

`user:` This represents the amount of CPU time spent executing the command in user mode. User mode is the part of the CPU's operation where it is executing code that belongs to the user's process, such as the specific command you ran. It does not include time spent in the kernel, handling system calls or other kernel-related tasks [8].

`sys:` This is the amount of CPU time spent by the command in system mode. The sys time measures the time spent in kernel-related activities on behalf of the command [8].

For performance analysis and profiling, it's common to focus on the **user** and **sys** times to understand how much CPU time a command consumes, as this provides insight into the computational workload of the command itself. However, the **real** time is often more relevant from a user's perspective because it reflects the total time it takes for a command to complete [8].

8. Doubly Linked List of Integers in C

8.1. Implementations of DLLNode and insert method [9]

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

// Node structure for a doubly linked list
typedef struct DLLNode {
    int num;
    struct DLLNode *prev;
    struct DLLNode *next;
} DLLNode;

/**
 * Inserts a new node into a sorted doubly linked list.
 * @param head The head of the list.
 * @param numToInsert The number to insert into the list.
 */
void insert(DLLNode **head, const int numToInsert) {

    // Create a new node
    DLLNode *newNode = (DLLNode *) malloc(sizeof(DLLNode));

    // Initialize the new node
    newNode->num = numToInsert;
    newNode->prev = NULL;
    newNode->next = NULL;

    // If the list is empty, insert the new node as the head
    if (*head == NULL) {
        *head = newNode;
        return;
    }

    // If the new node is greater than the head, insert it as the new head
    if (numToInsert < (*head)->num) {
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
        return;
    }

    // Find the node that the new node should be inserted after
    DLLNode *current = *head;
    while (current->next != NULL && current->next->num < numToInsert) {
        current = current->next;
    }

    // Insert the new node after the node that we found
    newNode->next = current->next;
    newNode->prev = current;
    if (current->next != NULL) {
        current->next->prev = newNode;
    }
    current->next = newNode;
}
```

8.2. Implementation of the main method [10]

```c
int main() {
    // Initialize the list as empty
    DLLNode *head = NULL;

    // Seed the random number generator
    unsigned const int seed_for_srand = (unsigned int) time(0);
    srand(seed_for_srand);

    // Measure time before insertion
    struct timespec exeStart, exeEnd;
    clock_gettime(CLOCK_REALTIME, &exeStart);

    // Insert 5000 random integers into the list
    for (int i = 0; i < 5000; i++) {
        const int randomInteger = rand();
        insert(&head, randomInteger);
    }

    // Measure time after insertion
    clock_gettime(CLOCK_REALTIME, &exeEnd);

    // Calculate the time taken for insertion
    const double exeRunTime = (double) ((exeEnd.tv_sec - exeStart.tv_sec) + (exeEnd.tv_nsec - exeStart.tv_nsec));

    printf("Time taken for all the insertions ≈ %1.f nanoseconds = %f seconds\n", exeRunTime, exeRunTime / 1e9);

    // Free the memory allocated for nodes
    while (head != NULL) {
        DLLNode *temp = head;
        head = head->next;
        free(temp);
    }

    return 0;
}

```

8.3. Makefile

```
CC = gcc
CFLAGS = -Wall -g


insert: insert.c
	$(CC) $(CFLAGS) -o $@ $<


clean:
	rm -f insert

```

References

[1] “Top 50+ linux commands you must know,” DigitalOcean, https://www.digitalocean.com/community/tutorials/linux-commands (accessed Sep. 22, 2023).

[2] “SCP command in linux with examples,” GeeksforGeeks, https://www.geeksforgeeks.org/scp-command-in-linux-with-examples/#:~:text=scp%20(secure%20copy)%20command%20in,or%20between%20two%20remote%20hosts. (accessed Sep. 22, 2023).

[3] “Sudo Command in linux with examples,” GeeksforGeeks, https://www.geeksforgeeks.org/sudo-command-in-linux-with-examples/#:~:text=Sudo%20is%20a%20command%20in,sometimes%20it%20can%20be%20risky. (accessed Sep. 22, 2023).

[4] M. Heller, “LS command in linux: Listing files & directories,” Linode Guides & Tutorials, https://www.linode.com/docs/guides/ls-command-in-linux/ (accessed Sep. 22, 2023).

[5] S. Zivanov, “How to clear terminal in linux {clear screen in linux}: Phoenixnap KB,” Knowledge Base by phoenixNAP, https://phoenixnap.com/kb/clear-terminal (accessed Sep. 22, 2023).

[6] “How can I install ubuntu on a separate partition on My Mac,” How can I install Ubuntu on a separate pa... - Apple Community, https://discussions.apple.com/thread/251380411?answerId=252695611022#252695611022 (accessed Sep. 23, 2023).

[7] “Welcome to Virtualbox.org!,” Oracle VM VirtualBox, https://www.virtualbox.org/ (accessed Sep. 23, 2023).

[8] IraimbilanjaIraimbilanja et al., “What do ‘real’, ‘user’ and ‘sys’ mean in the output of time(1)?,” Stack Overflow, https://stackoverflow.com/questions/556405/what-do-real-user-and-sys-mean-in-the-output-of-time1 (accessed Sep. 23, 2023).

[9] “Insertion in a doubly linked list,” GeeksforGeeks, https://www.geeksforgeeks.org/introduction-and-insertion-in-a-doubly-linked-list/ (accessed Sep. 23, 2023).

[10] “What is the proper way to use clock_gettime()?,” Stack Overflow, https://stackoverflow.com/questions/53708076/what-is-the-proper-way-to-use-clock-gettime (accessed Sep. 23, 2023).