

گزارش تمرین اول سیستم عامل پیشرفته

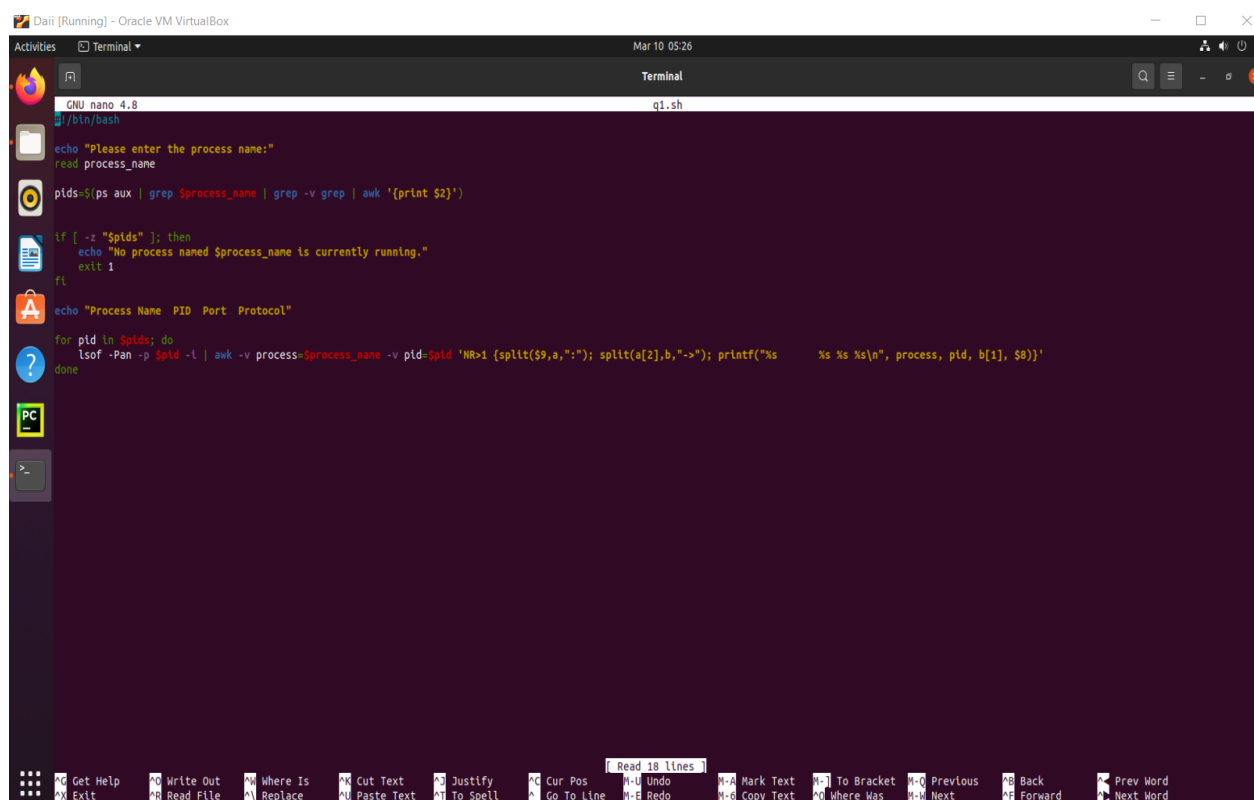
محمدباربدامیرمزلقانی-۸۱۰۱۰۲۳۴۸

نسخه ubuntu:

```
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.2 LTS
Release:        20.04
Codename:       focal
daii-VirtualBox :) >
```

سوال ۱:

ابتدا ساختار کلی اسکریپت را توضیح میدهم سپس به سراغ هر قسمت به صورت جزئی میرویم.



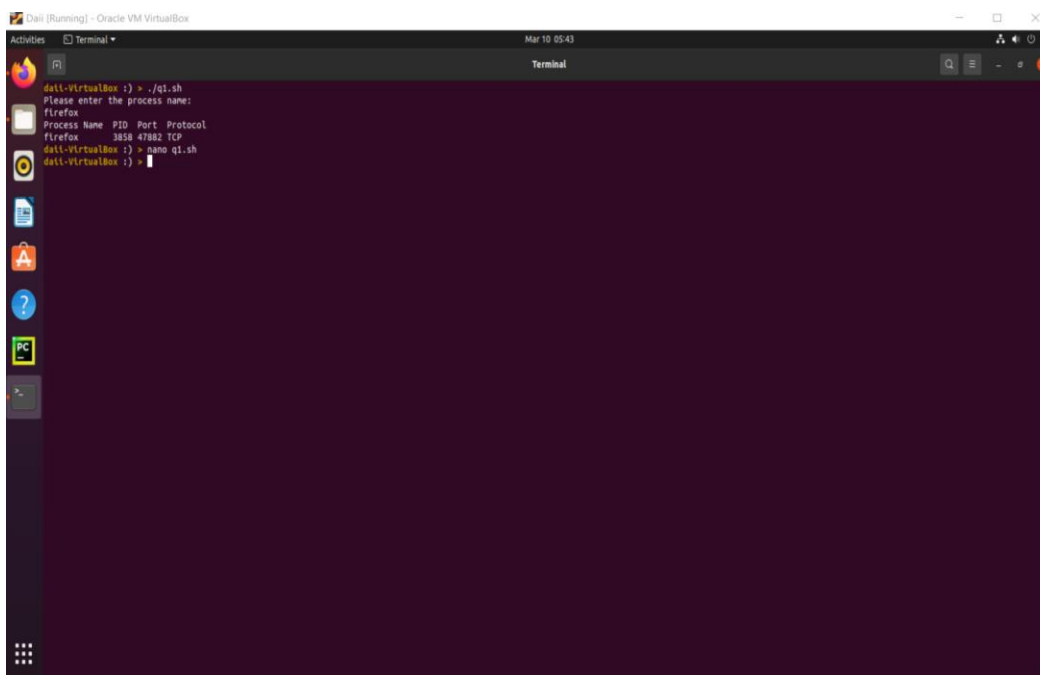
اولین چالشی که با آن مواجه شدیم این بود که از چه طریقی میتوان لیست پراسس های در حال اجرا را بدست آورد با اندکی جست و جو توانستیم دستور ps را پیدا کنیم ولی مشکلی که داشتیم این بود که مثلا firefox در آن دیده نمیشد که با جست و جوی بیشتر فهمیدیم پارامترهایی دارد که میتوان با تغییر آنها به پراسس های دیگر و سیستمی هم دست یافت.

که البته به همین راحتی بدست نمیامد لازم بود تا نام پراسس را در لیست جست و جو کنیم و همچنین خود grep را حذف کنیم چون در لیست هست در نهایت PID ها را در خروجی ذخیره کردیم.

در گام بعدی یک شرط میگذاریم که اگر همچنین پراسسی در حال حاضر در حال اجرا نبود این پیام نشان داده شود که پراسس خواسته شده در حال اجرا شدن نمیباشد.

گام بعدی و نهایی بدست آوردن پورت های پراسس و همچنین پروتکل آن میباشد با اندکی جست و جو فهمیدیم که دستور lsof اطلاعات خوبی در اختیارمان میگذارد، که شامل تمامی فایل های باز شده مرتبط با پراسس هست که شامل سوکت های شبکه هم میشود. پارامترهای مورد نیاز را اضافه میکنیم مثلا -P پورت ها را نمایش میدهد همچنین -n به جای نام هاست ها، آدرس های شبکه را میدهد در نهایت -a این دو مورد را AND منطقی میکند.

در ادامه با داشتن تمام اطلاعات تنها نحوه نمایش را تغییر میدهم بدین صورت که سطر اول که شامل هدرهاست حذف میشود و از ستون ها تنها NODE,NAME استفاده میشود که خود NAME هم آدرس کامل است اما ما تنها بخش پورت را میخواهیم پس از دستور split برای جدا کردن بخش مورد نظر استفاده میکنیم و خروجی در نهایت مطابق شکل زیر شد :



```
dali-VirtualBox:~$ ./q1.sh
Please enter the process name:
firefox
Process Name  PID  Port  Protocol
firefox      3850 47882 TCP
dali-VirtualBox:~$ nano q1.sh
dali-VirtualBox:~$
```

سوال ۲:

این سوال از نظر من (شاید راه سختی را برای سوال یک انتخاب کردم) ساده‌تر از سوال یک بود ابتدا مانند سوال یک ساختار کلی کد را می‌بینیم و در ادامه وارد هر کدام به صورت جزئی می‌شویم.



```
GNU nano 4.8
#!/bin/bash

src_files=(*.c)

for src_file in "${src_files[@]}; do
    obj_file="${src_file%.c}.o"
    if [ "$src_file" -nt "$obj_file" ] || [ ! -e "$obj_file" ]; then
        gcc -c "$src_file"
    fi
done

obj_files=(*.o)

gcc -o main "${obj_files[@]}"

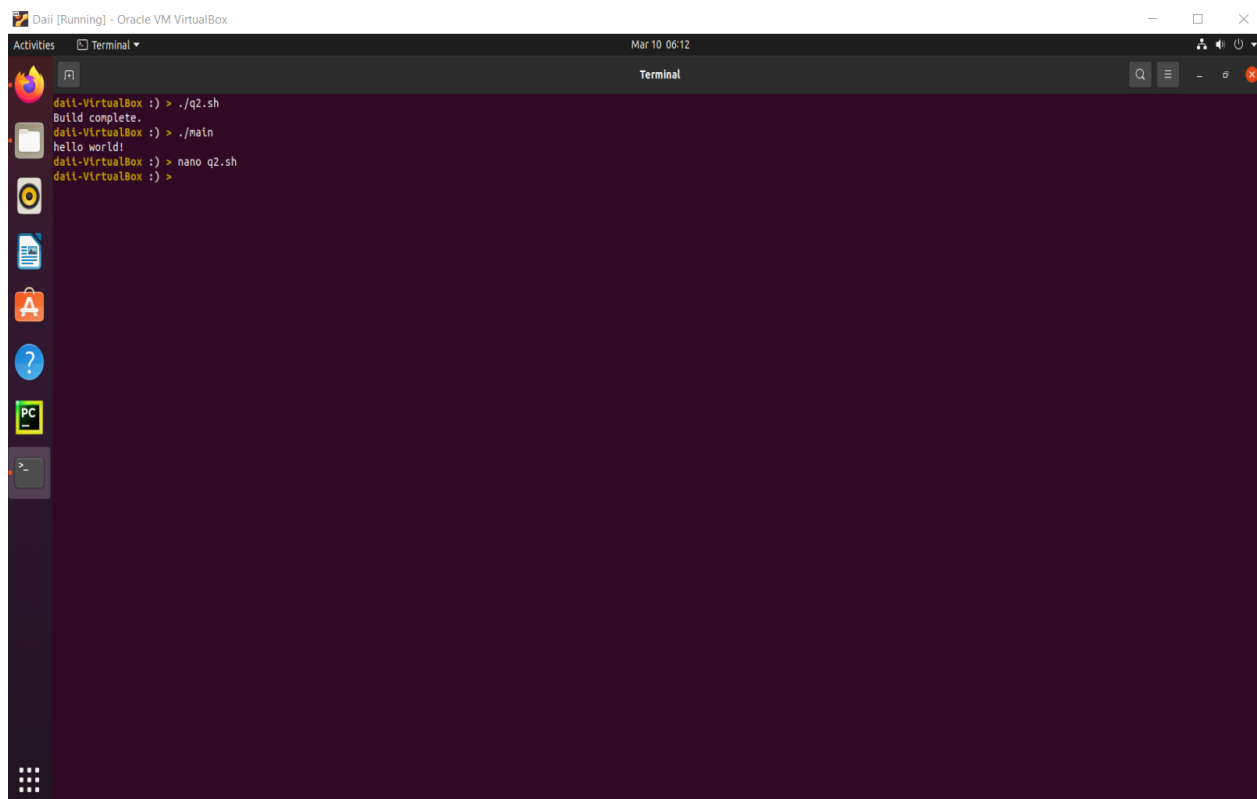
echo "Build complete."
```

ابتدا لازم است که فایل‌های C به برنامه شناسانده شوند ابتدا این کار را به صورت دستی می‌کردم ولی در ادامه تصمیم گرفتم برای جلوگیری از هاردکد کل فایل‌های موجود در دایرکتوری (که طبیعتاً مربوط به یک پروژه هستند) را بگیرم و در متغیری به نام `src_files` ریختم.

سپس لازم بود تا این فرآیند کامپایل که معمولاً وقتگیر هست را به صورت کد پیاده سازی کنیم، پس ابتدا لازم است تا فایل `Object` هر کدام را بدست بیاوریم برای این مورد، ابتدا یک متغیر آبجکت فایل بر مبنای اسم سورس تعریف می‌کنیم سپس یک شرط می‌گذاریم که اگر فایل سورس جدیدتر از آبجکت فایل بود یا کلاً آبجکت فایلی وجود نداشت دستور `gcc -c` برای آن سورس انجام شود در نهایت آبجکت‌های بدست آمده را به کمک

دستور `gcc -o لینک میکنیم و یک main نهایی میدهمیم همچنین برای تجربه بهتر کاربر Build complete` هم پرینت میشود.

خروجی در زیر دیده میشود :

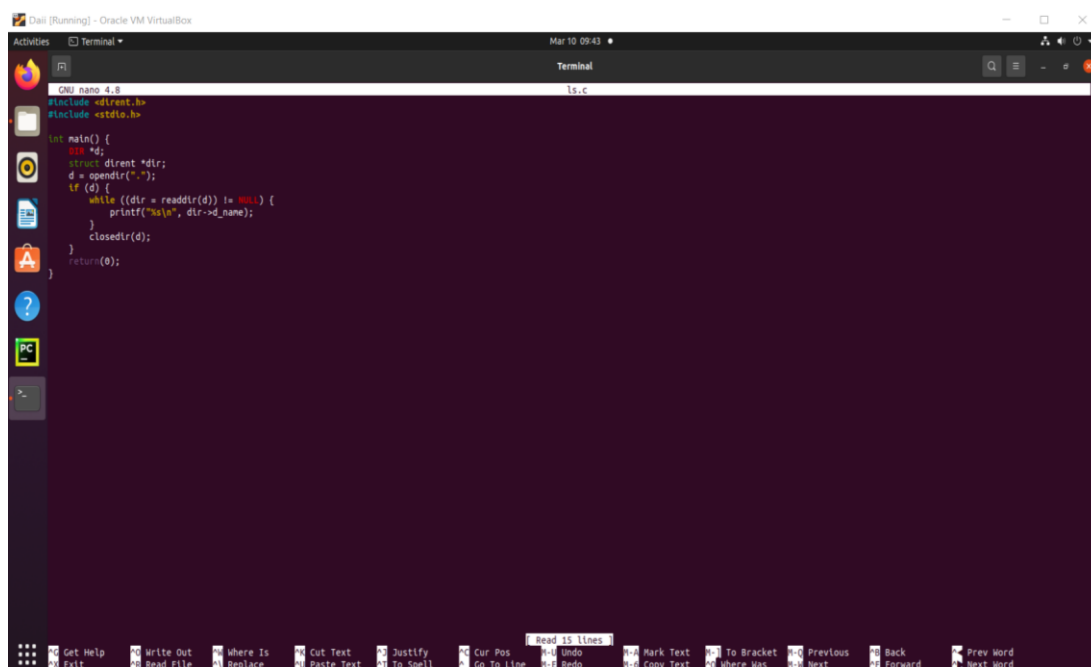


```
dali-VirtualBox :~$ ./q2.sh
Build complete.
dali-VirtualBox :~$ ./main
hello world!
dali-VirtualBox :~$ nano q2.sh
dali-VirtualBox :~$
```

سوال ۳:

در این سوال نیز بخش مربوط به کدنویسی bash پیچیدگی خاصی نداشت ولی در بخش مربوط به کدزنی C نیاز به اندکی جست و جو برای پیدا کردن دستورهای مربوط به دایرکتوری و یا خواندن فایل ها داشتیم مطابق با قسمت های قبل ابتدا ساختار کلی کدها را میبینیم و پس از آن به سراغ جزئیات پیاده سازی هر بخش میرویم .

کدهای زبان C:



The screenshot shows a terminal window titled 'Terminal' with a dark background. The code is being edited in nano 4.8. The code is as follows:

```
ls.c
#include <dirent.h>
#include <stdio.h>

int main() {
    DIR *d;
    struct dirent *dir;
    d = opendir(".");
    if (d) {
        while ((dir = readdir(d)) != NULL) {
            printf("%s\n", dir->d_name);
        }
        closedir(d);
    }
    return(0);
}
```

The terminal window has a status bar at the bottom with various keyboard shortcuts like 'Get Help', 'Write Out', 'Where Is', 'Cut Text', 'Justify', 'Cur Pos', 'H-U Undo', 'Mark Text', 'To Bracket', 'Previous', 'Back', 'Prev Word', 'Exit', 'Read File', 'Replace', 'Paste Text', 'To Spell', 'Go To Line', 'H-D Redo', 'Copy Text', 'Where Was', 'Next', 'Forward', and 'Next Word'. A tooltip 'Read 15 lines' is visible over the status bar.

Dail [Running] - Oracle VM VirtualBox

Mar 10 09:43

Terminal

```
GNU nano 4.8 wc.c
#include <stdio.h>

int main() {
    int c, wc = 0, ln_word = 0;

    while ((c = getchar()) != EOF) {
        if (c == '\n' || c == '\n' || c == '\t') {
            ln_word = 0;
        } else if (ln_word == 0) {
            ln_word = 1;
            ++wc;
        }
    }

    printf("%d\n", wc);
    return 0;
}
```

Get Help Exit Write Out Read File Replace Where Is Replace Cut Text Paste Text Justify To Spell Cur Pos Go To Line Undo Redo Mark Text Copy Text To Bracket Where Was Previous Next Back Forward Prev Word Next Word

Dail [Running] - Oracle VM VirtualBox

Mar 10 09:43

Terminal

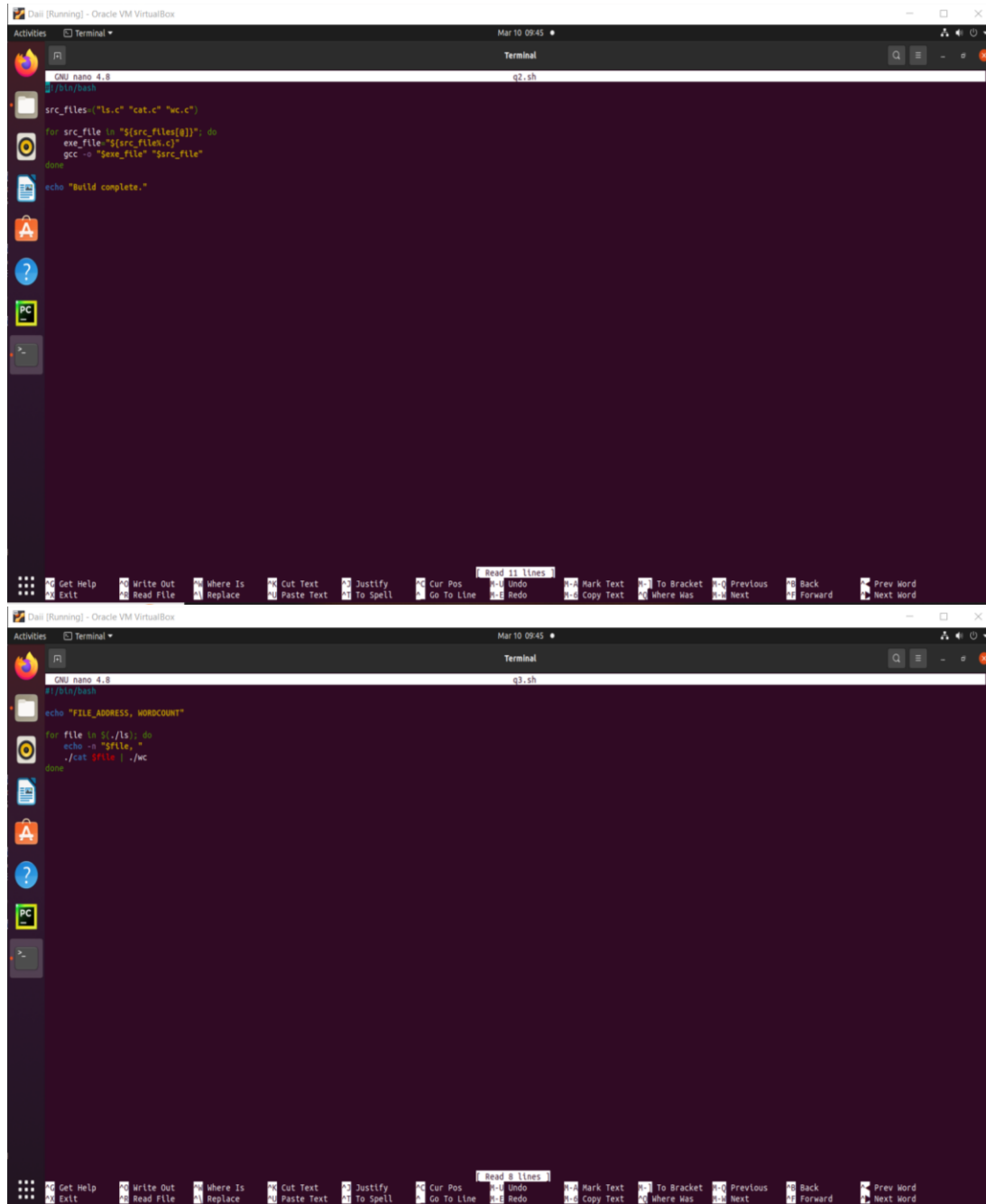
```
GNU nano 4.8 cat.c
#include <stdio.h>

int main(int argc, char *argv[]) {
    FILE *file;
    int c;

    file = fopen(argv[1], "r");
    if (file) {
        while ((c = getc(file)) != EOF) {
            putchar(c);
        }
        fclose(file);
    }
    return 0;
}
```

Get Help Exit Write Out Read File Replace Where Is Replace Cut Text Paste Text Justify To Spell Cur Pos Go To Line Undo Redo Mark Text Copy Text To Bracket Where Was Previous Next Back Forward Prev Word Next Word

اسکرپت های SHELL:



The image displays two screenshots of a Linux terminal window, likely running on Oracle VM VirtualBox. The terminal window has a title bar that reads "Dali [Running] - Oracle VM VirtualBox" and a status bar showing "Mar 10 09:45". The terminal is running a shell script named "q2.sh".

The first screenshot shows the following code in the terminal:

```
#!/bin/bash

src_files=("ls.c" "cat.c" "wc.c")

for src_file in "${src_files[@]}; do
    exe_file="${src_file}.o"
    gcc -o "$exe_file" "$src_file"
done

echo "Build complete."
```

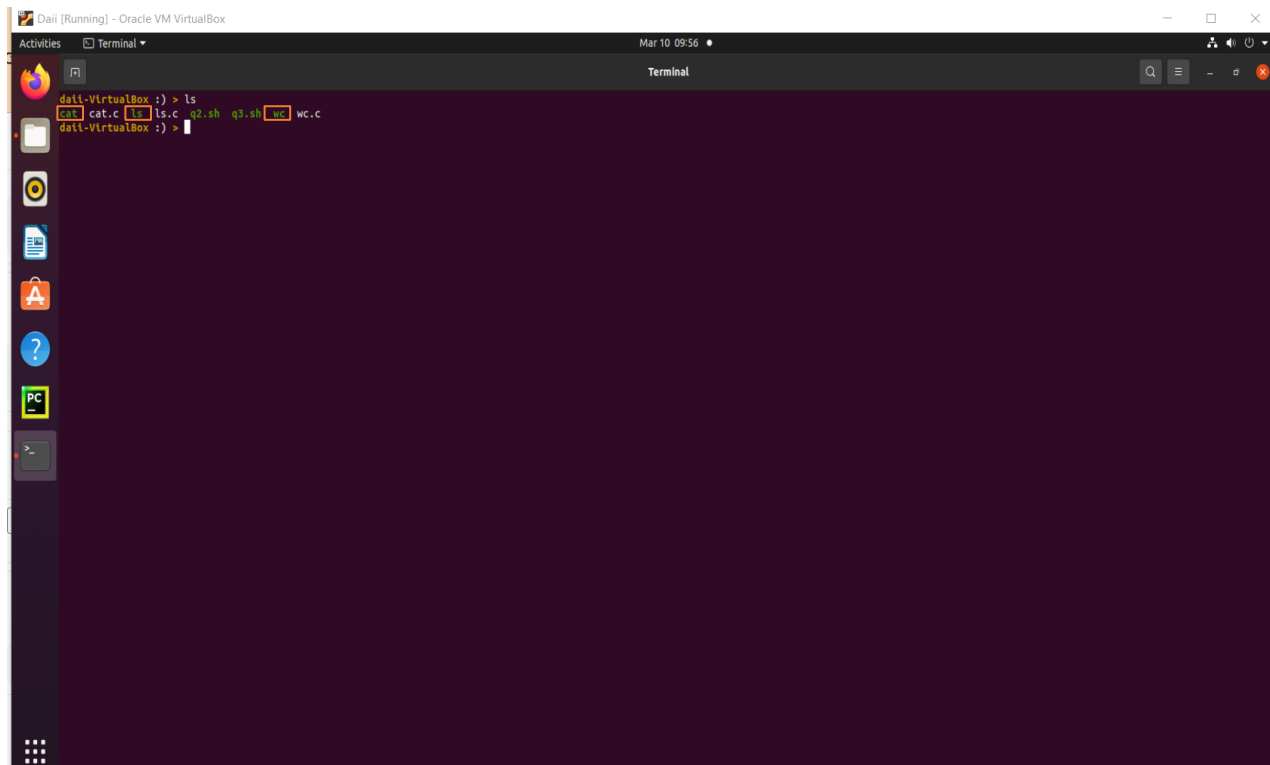
The second screenshot shows the following code in the terminal:

```
#!/bin/bash

echo "FILE_ADDRESS, WORDCOUNT"

for file in $(ls); do
    echo -n "$file, "
    ./cat $file | ./wc
done
```

قبل از این به سراغ کدهای قسمت سوم برویم لازم اصلاحاتی که در کد قسمت دو دیدم را کمی باز کنم که با توجه به اینکه تعدادی برنامه متفاوت هستند و لینک کردن بین آنها نیاز نیست هر کدام را در اسکریپت به صورت جدا کامپایل میکنیم که نتیجه آن، برنامه های قابل اجرا این سه دستور هست همانطور که در شکل زیر دیده میشود:



```
dali-VirtualBox:~$ ls
dali-VirtualBox:~$ cat cat.c | gcc ls.c -o ls -c q3.sh -o q3 -c wc.c
dali-VirtualBox:~$
```

حال به سراغ کدهای C میرویم از ls شروع میکنیم، در کد این قسمت از لایبرری dirent استفاده کردیم که یک استراکت از جنس DIR به ما میدهد، سپس پس از باز کردن آن خط به خط محتویاتی که از دایرکتوری میگیریم که در این مثال تنها شامل d_name هست را چاپ میکنیم تا زمانی که به NULL(انتها) برسیم.

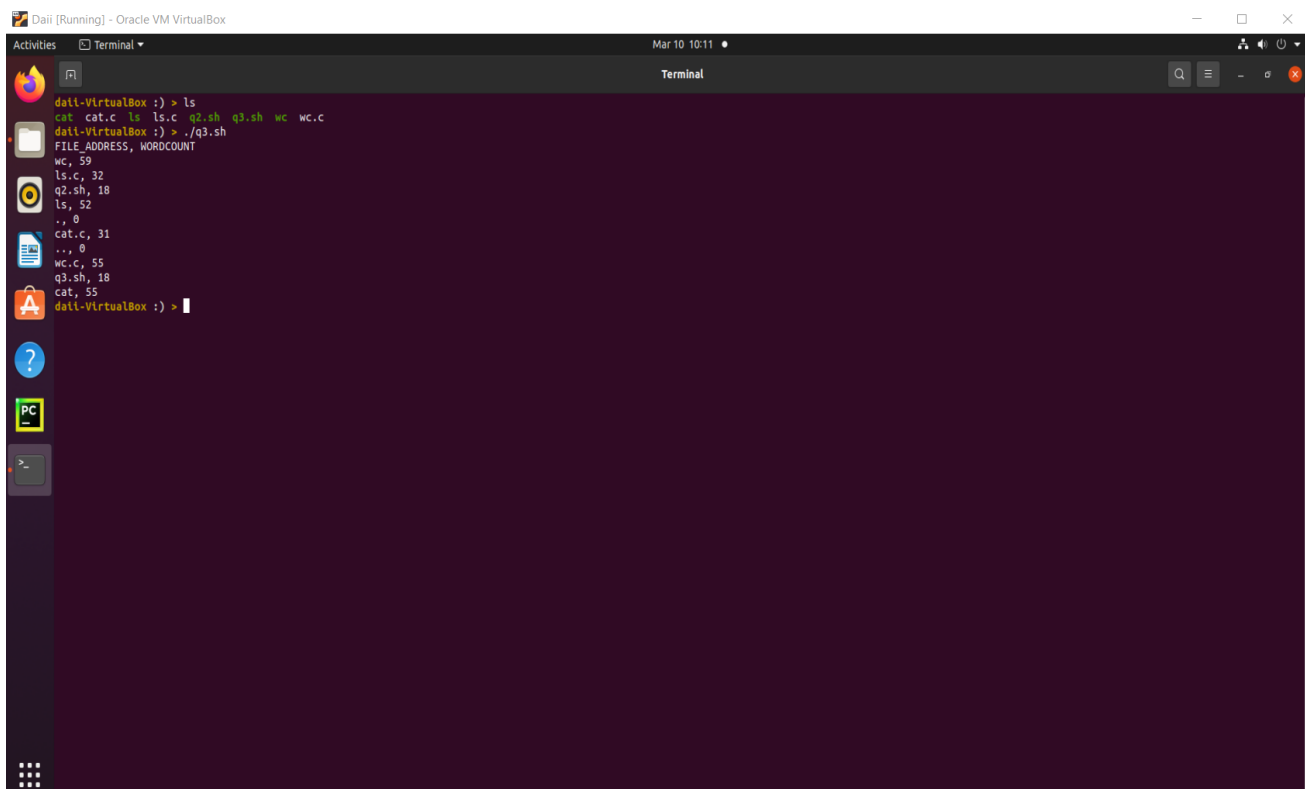
سپس به سراغ کد قسمت cat میرویم که نسبتا ساده است، از fopen برای باز کردن فایل ها استفاده میکنیم و مد کار کردن هم برابر با r برای خواندن میگذاریم، اگر بدون مشکل فایل را باز کردیم به صورت کاراکتر به کاراکتر از فایل میخوانیم و در stdout به کمک putchar میگذاریم و زمانی این کار را تمام میکنیم که به کاراکتر نشاندهنده اتمام فایل برسیم(EOF).

در نهایت برای wc هم کار ساده است، کاراکتر ها را از stdin میگیریم اگر اسپیس، تب یا خط داشتیم یعنی در کلمه نیستیم پس in_word را برابر صفر قرار میدهیم، اگر برابر چیز دیگری بود باید چک کنیم که قبل از آن

در کلمه نباشد تا نشان دهنده کلمه جدید باشد و به اندازه یک واحد **WC** را افزایش میدهیم که عملکرد ساده ای دارد.

در نهایت برای چک کردن کارکرد این فایلها نیاز به یک اسکریپت **shell** جدید داریم، در این شل از برنامه های خودمان استفاده میکنیم ابتدا یک **for** روی **ls** خودمان میزنیم سپس آدرس را از همین **ls** میگیریم و برای پیدا کردن **WC** ابتدا از **cat** خودمان کمک میگیریم و خروجی آن را به عنوان ورودی به برنامه **WC** مان میدهیم.

خروجی و صحت عملکرد را در ادامه میبینیم:



```
dall-VirtualBox :) > ls
cat cat.c ls ls.c q2.sh q3.sh wc wc.c
dall-VirtualBox :) > ./q3.sh
FILE_ADDRESS, WORDCOUNT
wc, 59
ls.c, 32
q2.sh, 10
ls, 52
., 0
cat.c, 31
., 0
wc.c, 55
q3.sh, 18
cat, 55
dall-VirtualBox :) >
```