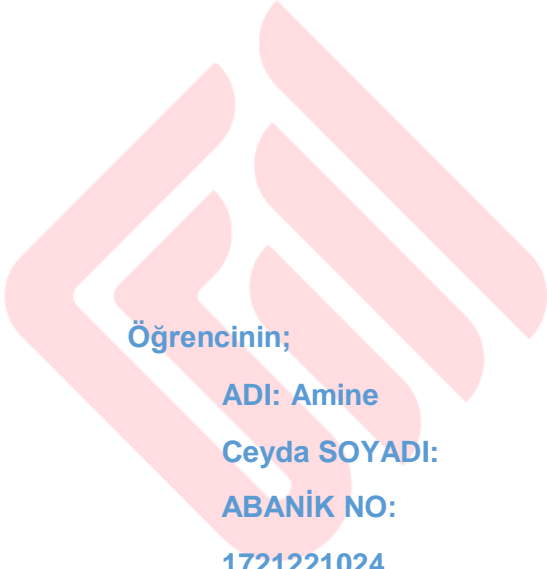




**FATİH  
SULTAN  
MEHMET**  
VAKIF ÜNİVERSİTESİ



**FATİH  
SULTAN  
MEHMET**  
VAKIF ÜNİVERSİTESİ

Öğrencinin;

ADI: Amine

Ceyda SOYADI:

ABANIK NO:

1721221024

BÖLÜM: Bilgisayar

Mühendisliği Projenin;

KONUSU: Linux Shell Yazımı

Dersin;

ADI: Operating System

EĞİTMEN: Prof.Dr. Ali Yılmaz

ÇAMURCU Arş.Gör.

Samet KAYA

Arş.Gör.Kadir ARAM

## İçindekiler

1-Proje Hakkında Bilinmesi Gerekenler: .....	3
➤ Shell (Kabuk) Nedir? .....	3
➤ Bash Nedir? .....	3
➤ Fork() Nedir? .....	3
2-Proje Konusu: .....	3
3-Proje Süresince Yapılanlar: .....	3
➤ myshell.c .....	3
➤ tekrar.c .....	4
➤ işlem.c .....	4
➤ topla.c .....	4
➤ cikar.c .....	4
➤ Makefile: .....	5
4-Kaynakça: .....	5

FATİH  
SULTAN  
MEHMET  
VAKIF ÜNİVERSİTESİ

## 1-Proje Hakkında Bilinmesi Gerekenler:

### ➤ Shell (Kabuk) Nedir?

Linux'un da içinde bulunduğu UNIX sistemlerinde komutları yorumlamak ve yönetmek için kullanılan programa kabuk (shell) denir. Komutların gerektirdiği kaynakları bulur ve kullanacak kaynaklara iletir. Bütün LINUX sistemlerde bulunması zorunlu olan birimdir.

### ➤ Bash Nedir?

Bash, çoğu LINUX sisteminde varsayılan kabuk olarak ayarlanmıştır. Her kabuğun kendine özgü programlama dili yapısı vardır. Bash kabuğu ise güçlü programlama özellikleriyle karmaşık programların rahatça yazılmasına izin verir. Mantıksal operatörler, döngüler, değişkenler ve modern programlama dillerinde bulunan pek çok özellik bash kabuğunda da vardır ve işleyiş tarzları da neredeyse diğerleriyle aynıdır. Bash çeşitli uyumluluk kiplerinde çalıştırılabilir, böylece farklı kabuklar gibi davranabilmektedir.

### ➤ Fork() Nedir?

Fork() komutu çalıştırıldığı anda çocuk (child), ve onu oluşturan process ise veli (parent) adını almaktadır. Çocuk yaratıldığı anda velinin hangi değişkeni var ise onun hepsinin kopyasını alır. Böylece çocuk velinin bulunduğu konumdan itibaren çalışır. fork() komutu çalıştırıldığı anda çocuk (child), ve onu oluşturan process ise veli (parent) adını almaktadır. Çocuk yaratıldığı anda velinin hangi değişkeni var ise onun hepsinin kopyasını alır ve bulunduğu konumdan itibaren çalışır.

## 2-Proje Konusu:

Projede temel bir linux shell programı yazıldı. Bu shell programının kendine özgü komutları, yanıtları, hata çıktıları tasarlandı. Proje kapsamında; toplama, çıkarma işlemleri, girilen değeri istediğimiz kadar ekrana yazdıran tekrar işlemi, proje dizininde bulunan dosyaların herhangi birinin içindekileri (kod / metin vs.) görüp inceleyebileceğimiz cat komutu, terminali tamamen silen clear komutu ve terminalden çıkılmasını sağlayan bir exit komutu implemente edilmesi hedeflenmiştir.

## 3-Proje Süresince Yapılanlar:

### ➤ myshell.c:

- while ile komut satırının "myshell>" ile başlaması, "exit" komutu girilene kadar programdan çıkılmaması ve komut alınabilmesi sağlandı.
- Tek komut için ve çift komut için parsellenip pipe (|) a göre hareket eden fonksiyonları main de çağırıldı.
- If'le komutlar kontrol edildi. Birinci parametre
- "exit" ise program kapandı ve ana komut sistemine geri dönüldü.myshell den çıkıldı.
- "tekrar" ise fork ile çocuk process oluşturuldu. Bu process exec ile tekrar programını çalıştırdı ve ana process bekledi. (Burada wait kullanmayın kuralı sonradan değişmişti.)
- "işlem" ise fork ile çocuk process oluşturuldu. Bu process exec ile işlem programını çalıştırdı ve ana process bekledi.

- "cat" ise fork ile çocuk process oluşturuldu. Bu process exec ile bin'in altındaki cat'i çalıştırdı ve ana process bekledi.

- "clear" ile ekran temizlendi.

#### ➤ tekrar.c:

Tekrar komutunda kullanıcıdan alınan değer kullanıcının istediği kadar ekrana yansıtılması hedeflendi. Alınan ilk argüman tekrar sınıfının çalışmasını ikincisi hangi girdinin basılacağı üçüncüsü içe kaç kez basılacağını belirtildi.

#### ➤ işlem.c:

Projenin işlem komutunun altında iki farklı sınıf çalışmaktadır. Bunlardan biri topla diğeri çıkar argümanıdır. İşlem argümanından sonra hangi işlemin yazıldığına göre çalışacak sınıfın seçimi gerçekleştirildi. Pipe / Uzun çizgi ( "|" ) sembolü ile birden çok komut sırasıyla çalıştırıldı.

#### ➤ topla.c:

Parametre ile verilen iki sayı toplanıp sonuç döndürüldü.

#### ➤ cıkar.c:

Parametre ile verilen iki sayıdan ikincisi birincisinden çıkarılıp sonuç döndürüldü.

```
ceyda@ceyda-VirtualBox:~/Desktop/Operating System Myshell$ ./myshell

myshell>>islem topla 2 5 | islem cıkar 5 7
2 + 5 = 7
5 - 7 = -2

myshell>>tekrar ceyda 5
ceyda
ceyda
ceyda
ceyda
ceyda

myshell>>cat topla.c
#include<stdio.h>
#include<stdlib.h>

int main(int argc, char* arguments[]){
// Toplama islemi icin gerekli kodlar
int x = atoi(arguments[0]);
int y = atoi(arguments[1]);
int result = x + y;
printf("%d + %d = %d \n",x,y,result);
return 0;
}
```

```
ceyda@ceyda-VirtualBox: ~/Desktop/Operating System Mys...

myshell>>islem topla 77 88
77 + 88 = 165

myshell>>islem cıkar.c 25 88
*HATA*: 'cıkar.c' bir komut değildir. (topla veya cıkar)

*** stack smashing detected ***: terminated

myshell>>islem cıkar 25 88
25 - 88 = -63

myshell>>clear
```

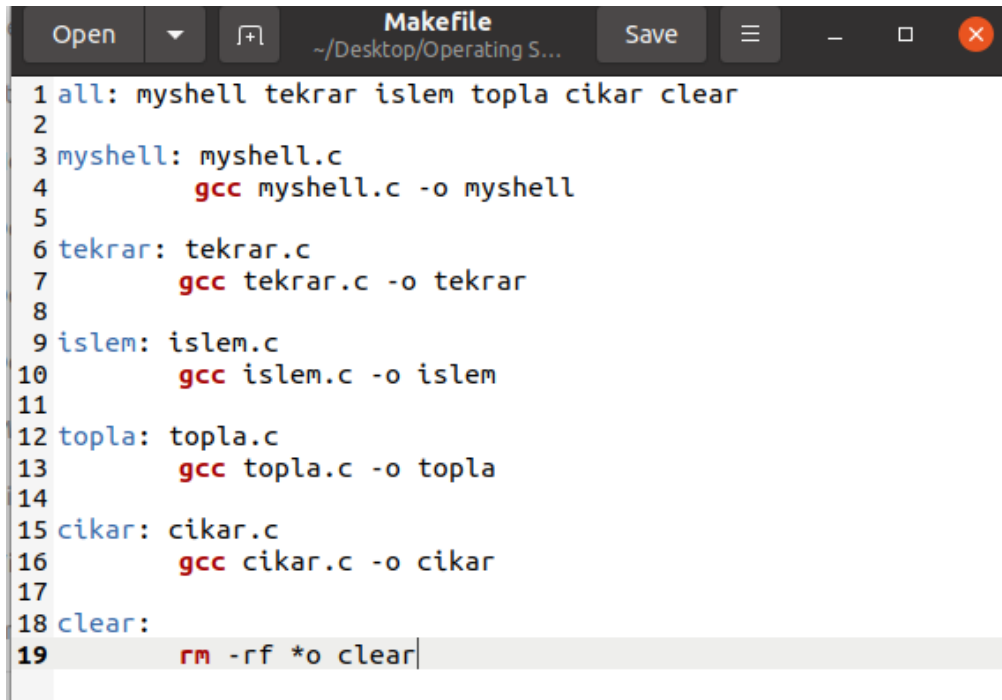
```
myshell>>exit
```

Program kapandı.

```
ceyda@ceyda-VirtualBox:~/Desktop/Operating System Myshell$
```

#### ➤ Makefile:

Makefile dosyası programın farklı işletim sistemlerinde basit bir şekilde derlenip doğru çalışmasını sağlar. Bu projede oluşturulan programların derlenmesi için yazılması gereken Linux terminal komutları makefile'a yazıldı.



```
1 all: myshell tekrar islem topla cikar clear
2
3 myshell: myshell.c
4     gcc myshell.c -o myshell
5
6 tekrar: tekrar.c
7     gcc tekrar.c -o tekrar
8
9 islem: islem.c
10    gcc islem.c -o islem
11
12 topla: topla.c
13    gcc topla.c -o topla
14
15 cikar: cikar.c
16    gcc cikar.c -o cikar
17
18 clear:
19    rm -rf *o clear
```

#### 4- Kaynakça:

- Operating System Lab Videoları
- <https://bidb.itu.edu.tr>