



**T.C.**  
**SAKARYA ÜNİVERSİTESİ**

**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**2024 GÜZ**

**İŞLETİM SİSTEMLERİ PROJE ÖDEVİ RAPORU**

**Kabuk (Shell) Uygulaması**

**Ali Osman Öcalır - B221210404 – 1. Öğretim B Grubu**  
**Zeynep İrem Tekin - B191210047 - 1. Öğretim B Grubu**  
**Tunahan Şahin - G201210023 - 2. Öğretim B Grubu**  
**Dursun Özer - B231210301 - 1. Öğretim A Grubu**  
**Büşra Kaya – B201210045 – 1. Öğretim C Grubu**

**Github Adresi : <https://github.com/busra-kaya/GRUP13-OS>**

## SAKARYA

Aralık, 2024

İşletim Sistemleri Dersi

## Kabuk (Shell) Uygulaması

## Özet

Kullanıcının temel kabuk işlemlerini Linux ortamında gerçekleştirmesine olanak tanıyan bir uygulama.

## 1. Giriş

1. Bu proje, temel kabuk (shell) işlemlerini uygulamak üzere tasarlanmıştır. Kullanıcı, komut istemcisi aracılığıyla komutlar çalıştırabilir, giriş/çıkış yönlendirmesi yapabilir, pipe (|) kullanarak komutları zincirleyebilir ve arka planda komut çalıştırabilir. Bu proje, aynı zamanda temel süreç yönetimi ve sinyal işleme mekanizmalarını da kapsamaktadır.

## 2. Proje Bileşenleri

## 2.1 Ana Yapı

- Prompt Görüntüleme:** Kullanıcıdan komut alınması ve girişin döngülü olarak işlenmesi.
- Komut Ayrıştırma:** Komutların ve argümanların ayıklandığı modüler.

## 2.2 Giriş/Çıkış Yönlendirmesi

- Giriş (<) ve Çıkış (>):** Kullanıcı komutlarını dosyalarla ilişkilendirme.
- Ekleme (>>) Modu:** Verilerin mevcut bir dosyaya eklenmesi.

## 2.3 Pipe (|) Mekanizması

- Pipe kullanarak bir komutun çıkışını, diğer komutun girdisi olarak kullanma.
- Komutların sıralı bir şekilde işlenmesi.

## 2.4 Arka Plan İşletimi

- Arka planda çalışan komutları takip etme ve tamamlanan süreçleri raporlama.
- Arka plan işlemleri tamamlanmadan kabuğun kapatılmasını engelleme.

## 2.5 Sinyal Yönetimi

- **SIGINT (Ctrl+C):** Uygulamanın kesintiye uğramadan çalışmasını sağlama.
  - **SIGCHLD:** Çocuk süreçlerin durumunu takip etme ve tamamlanan süreçleri temizleme.
- 

## 3. Teknik Detaylar

### 3.1 Komut İşleme Akışı

1. Kullanıcıdan bir komut alınır.
2. Komut pipe (|) veya noktalı virgül (;) ile ayrılmışsa alt komutlara ayrılır.
3. Her bir alt komut, sırasıyla veya pipe mekanizması ile çalıştırılır.
4. Giriş/çıkış yönlendirmeleri kontrol edilir ve uygulanır.
5. Yerleşik komutlar (örneğin quit) algılandığında ilgili işlevler devreye girer.

### 3.2 Modüller

- **main.c:** Ana program döngüsü ve temel kontrol mekanizmaları.
  - **parser.c:** Komut ve argümanları ayrıştırma.
  - **executor.c:** Komutların çalıştırılması.
  - **io\_redirect.c:** Giriş ve çıkış yönlendirme mekanizmaları.
  - **pipe.c:** Pipe mekanizmasını yöneten işlevler.
  - **signals.c:** Sinyal işlemleri ve arka plan süreç yönetimi.
  - **builtin.c:** Yerleşik komutların uygulanması.
  - **pipeparser.c:** Pipe ayrıştırma fonksiyonları.
- 

## 4. Kullanıcı Etkileşimi

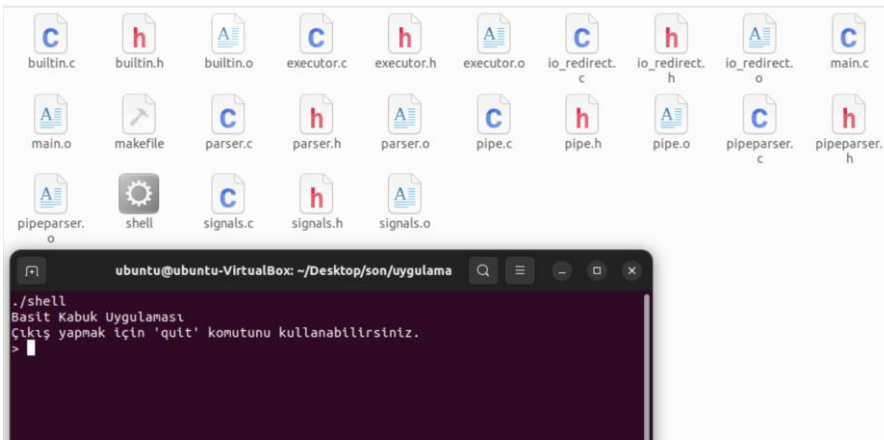
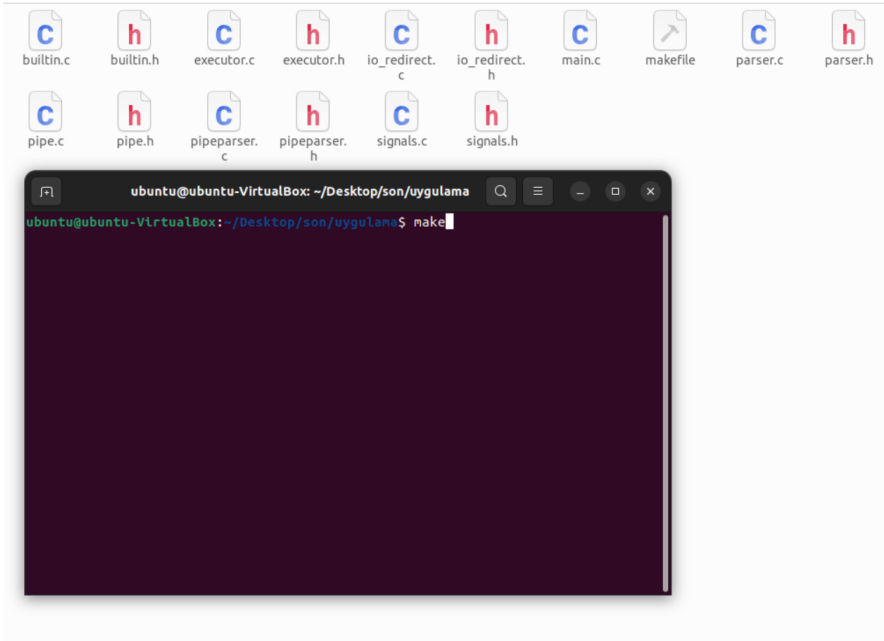
1. Kullanıcı bir komut girer.
2. Sistem komutun formatını ve geçerliliğini kontrol eder.
3. Çıkış dosyasına veya girdi dosyasından veri işlenebilir.

4. Komut pipe ile bağlanmışsa, zincirleme işlemler sırasıyla uygulanır.
5. Arka planda çalışan komutlar tamamlandığında bilgilendirme yapılır.

## 5. Puanlama ve İstenenler

2. Bu proje kapsamında aşağıdaki özelliklerin geliştirilmesi hedeflenmiştir:

**Prompt:** Kullanıcının komut girmesi için bir istem görüntülenir. (5 Puan)



**Built-in Komut:** Kabukta "quit" gibi yerleşik komutlar işlenir. (5 Puan)

```

ubuntu@ubuntu-VirtualBox: ~/Desktop/son/uygulama
./shell
Basit Kabuk Uygulaması
Çıkış yapmak için 'quit' komutunu kullanabilirsiniz.
> sleep 10 &
> quit
Quit komutu alındı. Arka plan işlemleri bekleniyor...
Yeni komut kabul edilmiyor.

[PID 11998] retval: 0
> Tüm arka plan işlemleri tamamlandı. Kabuk kapatılıyor...
ubuntu@ubuntu-VirtualBox:~/Desktop/son/uygulama$

```

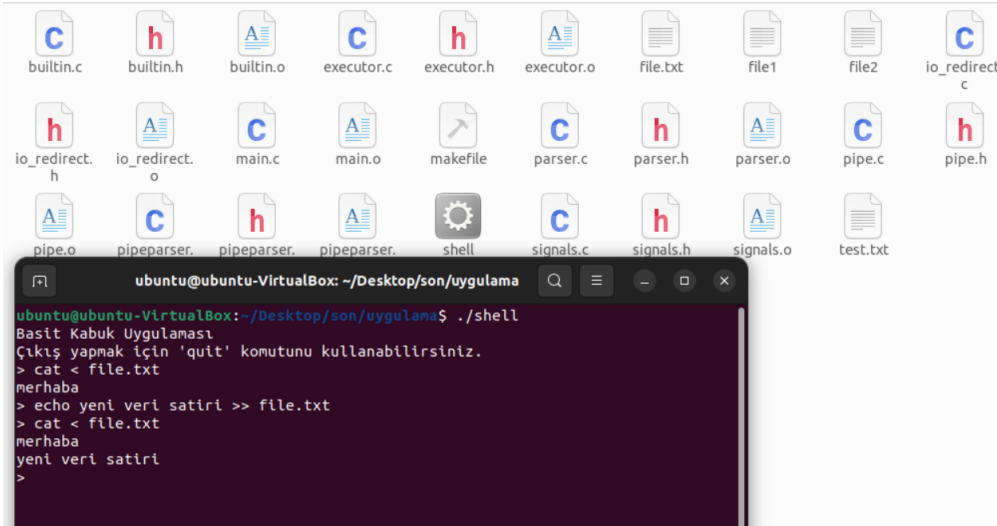
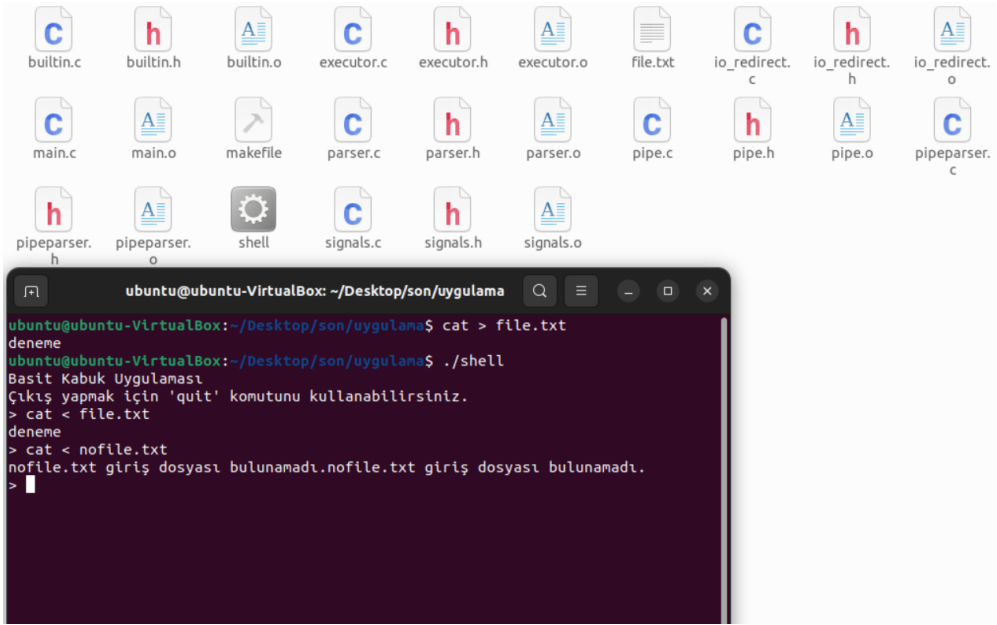
**Tekli Komut İcrası:** Bir komutun argümanlarıyla birlikte yürütülmesi sağlanır. (10 Puan)

```

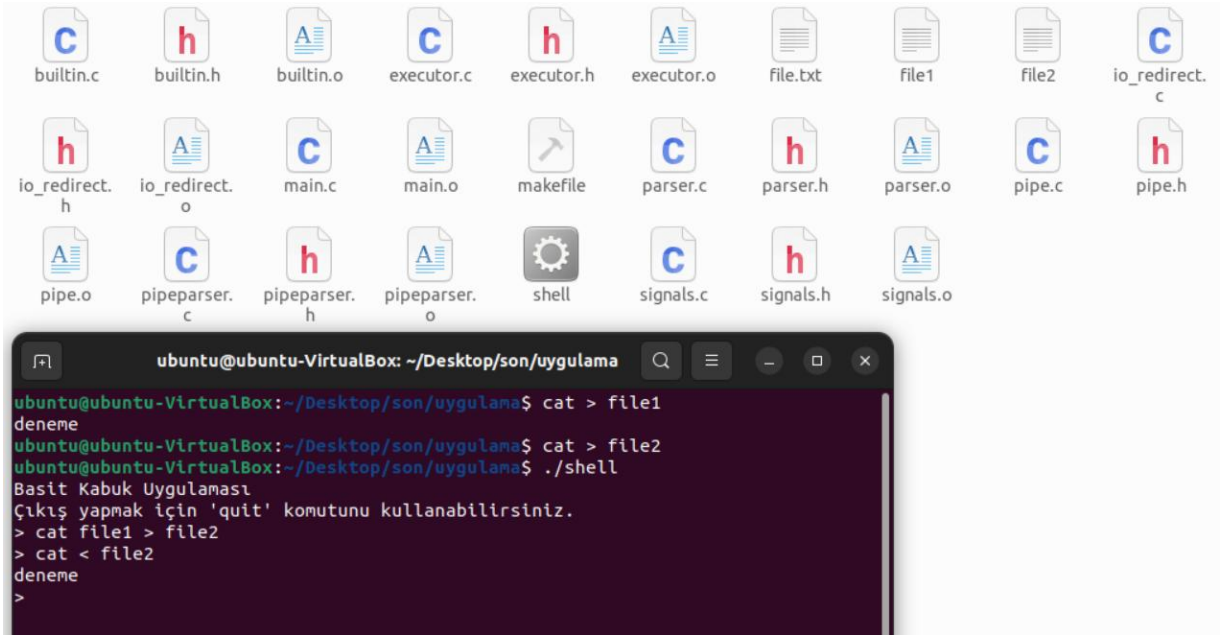
ubuntu@ubuntu-VirtualBox: ~/Desktop/son/uygulama
ubuntu@ubuntu-VirtualBox:~/Desktop/son/uygulama$ ./shell
Basit Kabuk Uygulaması
Çıkış yapmak için 'quit' komutunu kullanabilirsiniz.
> ls -l
total 188
-rw-rw-r-- 1 ubuntu ubuntu 1041 Ara 11 01:22 builtin.c
-rw-rw-r-- 1 ubuntu ubuntu 660 Ara 11 01:24 builtin.h
-rw-rw-r-- 1 ubuntu ubuntu 5360 Ara 11 04:07 builtin.o
-rw-rw-r-- 1 ubuntu ubuntu 1456 Ara 11 01:25 executor.c
-rw-rw-r-- 1 ubuntu ubuntu 327 Ara 11 01:26 executor.h
-rw-rw-r-- 1 ubuntu ubuntu 8616 Ara 11 04:07 executor.o
-rw-rw-r-- 1 ubuntu ubuntu 2079 Ara 11 01:27 io_redirect.c
-rw-rw-r-- 1 ubuntu ubuntu 736 Ara 11 01:28 io_redirect.h
-rw-rw-r-- 1 ubuntu ubuntu 6144 Ara 11 04:07 io_redirect.o
-rw-rw-r-- 1 ubuntu ubuntu 5122 Ara 11 01:39 main.c
-rw-rw-r-- 1 ubuntu ubuntu 13600 Ara 11 04:07 main.o
-rw-rw-r-- 1 ubuntu ubuntu 297 Ara 11 01:46 makefile
-rw-rw-r-- 1 ubuntu ubuntu 1545 Ara 11 01:29 parser.c
-rw-rw-r-- 1 ubuntu ubuntu 762 Ara 11 01:30 parser.h
-rw-rw-r-- 1 ubuntu ubuntu 6424 Ara 11 04:07 parser.o
-rw-rw-r-- 1 ubuntu ubuntu 2638 Ara 11 01:31 pipe.c
-rw-rw-r-- 1 ubuntu ubuntu 396 Ara 11 01:32 pipe.h
-rw-rw-r-- 1 ubuntu ubuntu 7168 Ara 11 04:07 pipe.o
-rw-rw-r-- 1 ubuntu ubuntu 734 Ara 11 01:33 pipeparser.c

```

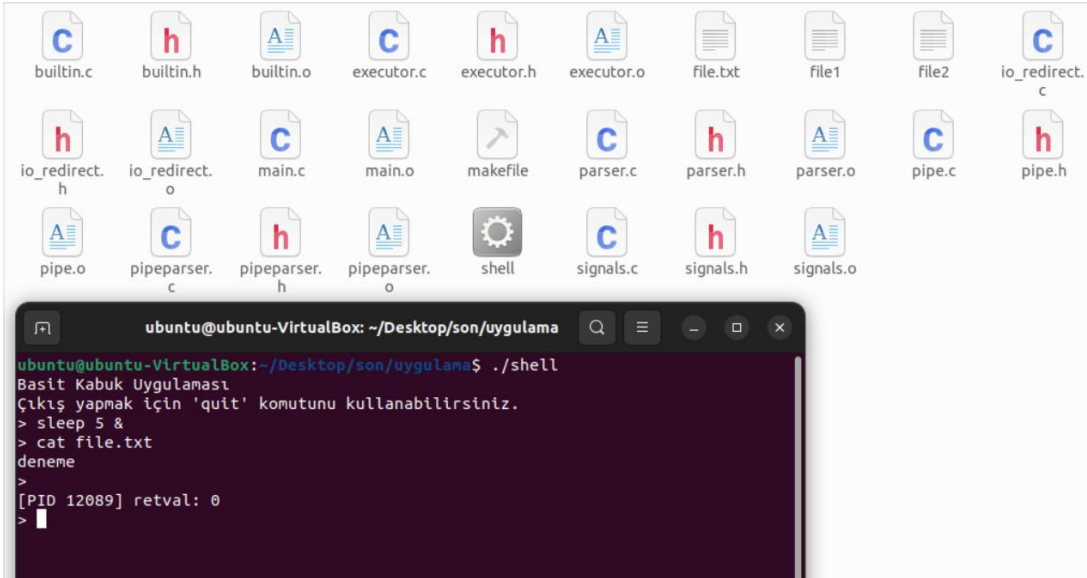
**Giriş Yönlendirme (<):** Bir dosyadan girdi almak için giriş yönlendirme işlemleri uygulanır. (20 Puan)



**Çıkış Yönlendirme (>):** Komut çıktısını bir dosyaya yazmak için çıkış yönlendirme işlemleri uygulanır. (20 Puan)



**Arka Plan Çalışma (&):** Komutun arka planda çalıştırılması ve süreçlerin takip edilmesi sağlanır. (20 Puan)



**Boru (|):** Bir komutun çıktısının, diğer bir komutun girdisi olarak kullanılması sağlanır.  
(20 Puan)

