T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

İŞLETİM SİSTEMLERİ PROJE RAPORU

2024-2025 GÜZ DÖNEMİ – LINUX KABUK(SHELL) UYGULAMASI

Ali KOÇ – G221210059

Bedirhan CAN – G221210053

Zeynep Dilara KURNAZ – G221210011

Burak Emre SARIKOÇ – G221210077

Necib TAVLAŞOĞLU – B221210023

Projenin Github Adresi: https://github.com/koc-ali03/grp16-ogr2-subB

PROJE TANIMI

Bu proje, Linux ortamında C programlama dili kullanılarak bir kabuk (shell) uygulaması geliştirilmesini amaçlamaktadır. Kabuk, kullanıcının komut satırına yazdığı komutları alır, bu komutları çalıştırır, gerektiğinde komutların çıktısını yönlendirir ve kullanıcının verdiği komutları anlamak için temel Linux sistem çağrılarını kullanır. Proje, komutların birbiriyle sıralanması, giriş ve çıkış yönlendirmelerinin yapılması, arka planda komut çalıştırma ve boru (pipe) işlemlerinin gerçekleştirilmesi gibi temel işletim sistemi işlevlerini kapsar. Bu uygulama, proses yönetimi, I/O yönlendirme ve Linux signal kullanımı konularını derinlemesine öğrenmeye olanak tanımaktadır.

UYGULANAN ADIMLAR:

1. **Prompt (Komut İstemi)** – *(Ali Koç)*

   - Kabuk uygulaması başlatıldığında ve her komutun tamamlanmasından sonra veya her arka plan komutunun ardından ">" komut istemi ekrana basılacaktır.

2. **Quit Komutu (Built-in Komut)** – *(Ali Koç)*

   - `quit` komutu eklendi: Kabuk bu komut alındığında sonlanacak şekilde yapılandırılmıştır.

   - Eğer komut verildiğinde arka planda çalışan bir işlem varsa, kabuk bu işlemleri tamamladıktan sonra sonlanacaktır. Bu sayede, tüm işlemlerin güvenli bir şekilde tamamlanması sağlanmaktadır.

3. **Tekli Komut İcrası** – *(Ali Koç)*

   - Kabuk, kullanıcıdan alınan tekli komutları çalıştırmak için bir alt proses oluşturur.

   - Örneğin, `ls -l` komutunu çalıştırmak için fork, exec, ve wait sistem çağrıları kullanılmıştır.

4. **Giriş Yönlendirme (Input Redirection)** – *(Bedirhan Can)*

   - Kullanıcıdan gelen komutlar, giriş yönlendirme için < operatörüne göre işlenmiştir.

   - Örneğin, `cat < file.txt` komutu, file.txt dosyasının içeriğini ekrana yazdıracaktır.

5. **Çıkış Yönlendirme (Output Redirection)** – *(Zeynep Dilara Kurnaz)*

   - Çıkış yönlendirme işlemi için > operatörü kullanılarak komutların çıktıları belirli bir dosyaya yönlendirilmiştir.

   - Örneğin, `cat file1 > file2` komutunun çıktısı, file2 dosyasına yazılacaktır.

6. **Arka Plan Çalışma** – *(Burak Emre Sarıkoç)*

   - Arka plan komutları, komutun sonuna & eklenerek çalıştırılabilir. Kabuk, arka planda çalışan komutların tamamlanmasını beklemeden diğer komutları alabilir.

   - Arka plan komutlarının tamamlandığında çıktı olarak pid ve dönüş değeri ekrana yazdırılacaktır.

7. **Boru (Pipe)** – *(Necib Tavlaşoğlu)*

   - Boru (pipe) işlemleri, bir komutun çıktısının diğer komutun girişine bağlanması için kullanılmıştır.

   - Örneğin, `find /etc | grep ssh | grep conf` komut zinciri, /etc içindeki ssh ile ilgili dosyaların listeleneceği ve conf içeren dosyaların filtreleneceği bir işlem zinciri oluşturmuştur.

SONUÇ

Bu proje, Linux ortamında C programlama dilini kullanarak bir kabuk uygulaması geliştirme sürecinde, işletim sistemlerinin temel işlevlerine dair derinlemesine bir anlayış kazandırmıştır. Geliştirilen kabuk uygulaması, sistem çağrılarını kullanarak proses yönetimi, giriş ve çıkış yönlendirme, arka planda işlem yürütme ve boru işlemleri gibi temel işletim sistemi işlevlerini deneyimleme fırsatı sunmuştur. Ayrıca, kullanıcı dostu bir komut satırı yorumlayıcı oluşturma becerisi geliştirilmiştir. Geliştirilen bu kabuk, işletim sistemlerinin çalışma prensiplerini daha iyi anlamamıza ve bu prensiplere uygun uygulamalar geliştirmemize yardımcı olmuştur. Proje, işletim sistemlerinin temel yapı taşlarını anlamada önemli bir adım olmuştur ve aynı zamanda pratik yaparak teorik bilgilere dayalı uygulama geliştirme becerisini pekiştirmiştir.