İŞLETİM SİSTEMLERİ PROJE ÖDEVİ RAPORU

GitHub Link: https://github.com/kesicioglufurkan/IsletimSistemleriProje

ÖDEV: Bu ödevin amacı, proses (süreç) yönetimi, I/O (Giriş/Çıkış) ve Linux signal kullanımının temellerini öğrenmek ve temel bir Kabuk uygulaması geliştirmek.

GRUP NO:27

GRUP ÜYELERİ:

-G221210383 Metin Aydın

-G221210071 Bilal avcı

-G221210089 Furkan Ay

-G221210047 Furkan Kesicioğlu

-G221210045 Mehmet Bosdancı

Proje Tanımı

Bu proje, Linux işletim sisteminin temel bileşenlerinden biri olan kabuk (shell) uygulamasını geliştirmeyi amaçlamaktadır. Kabuk, bir komut satırı yorumlayıcısı olarak kullanıcıdan gelen komutları alıp işleten ve çıktıları görüntüleyen bir yazılımdır. Bu projede geliştirilen kabuk uygulaması, temel komut yönetimi, giriş-çıkış yönlendirme, arka plan işletimi ve boru (pipe) gibi özellikleri desteklemektedir. Proje, hem teorik hem de pratik bilgi birikimini geliştirmeyi hedefleyen bir yapıya sahiptir.

Amaç ve Hedefler

Projenin temel amaçları:

- Kullanıcıdan gelen komutları analiz etmek ve uygun şekilde çalıştırmak.
- Proses (süreç) yönetimi ve Linux sistem çağrılarını etkin bir şekilde kullanma becerisini geliştirmek.
- Giriş ve çıkış yönlendirme işlemlerini gerçekleştirmek.
- Arka plan işletimi destekleyerek aynı anda birden fazla komutun yürütülmesini sağlamak.
- Boru kullanarak bir komutun çıktısını başka bir komutun girdisine yönlendirmek.
- Kullanıcıya esnek ve modern bir komut işleme deneyimi sunmak.

Proje Yapısı

Proje, modüler yapıda geliştirilmiş olup, her bir modül belirli bir işlevi gerçekleştirmek için tasarlanmıştır:

1. **Girdi Yönlendirme**: Komut girdilerinin dosyalardan okunmasını sağlar. Kullanıcı "<" operatörünü kullanarak bir dosyadaki veriyi komuta girdirebilir.
2. **Çıktı Yönlendirme**: Komut çıktısını bir dosyaya yazılmasını sağlar. ">" operatörü ile bir dosyaya veri yazmak mümkün olur.
3. **Arka Plan İşletimi**: Komutların çalışmasını beklemeden başka komutları çalıştırır. "C" sembolü kullanılarak bir komut arka planda çalıştırılabilir.
4. **Boru (Pipe)**: Bir komutun çıktısını, başka bir komutun girdisi olarak kullanır. "|" sembolü bu işlevi gerçekleştirir.
5. **Signal Yönetimi**: Arka planda çalışan işlemlerin durumunu izler ve tamamlanan işlemlerle ilgili bilgi verir. Özellikle SIGCHLD sinyali kullanılmıştır.

Kullanılan Yöntemler ve Araçlar

- **C Programlama Dili**: Projenin temel programlama dili olarak tercih edilmiştir. Performans ve sistem seviyesinde kontrol sağlayan yapısı bu seçimin temel nedenidir.
- **Linux Sistem Çağrıları**: Özellikle "fork", "exec", "wait" ve "dup2" gibi çağrılar kullanılmıştır. Bu çağrılar, proses oluşturma ve yönetme gibi önemli işlemleri gerçekleştirir.
- **Signal Kullanımı**: Arka plan işlemlerini yönetmek için SIGCHLD sinyali kullanılmıştır. Bu yöntem, sistem kaynaklarını verimli kullanma açısından çok faydalıdır.

Ana Fonksiyonlar

1. **Komut Ayrıştırma (parse_command)**:
    - Kullanıcı girdiğinden gelen komutları parçalar.
    - Argümanları belirleyerek komutun çalıştırılmasını sağlar.
    - Girilen komut yapısının çözülmesi ve argümanlara ayrılması bu fonksiyon ile mümkün olur.
2. **Komut Yürütme (execute_command)**:
    - Bağımsız bir komutun veya argümanların yürütülmesini sağlar.
    - Giriş-çıkış yönlendirmesi işlevini destekler.
    - Komutun başarıyla işletilip işletilmediği hakkında bilgi verir.
3. **Arka Plan İşlemleri Yönetimi**:
    - Arka plan komutların sayısını takip eder ve tamamlanan işlemler için geri bildirim verir.
    - Arka planda çalışan işlem bitiminde proses kimliği (PID) ve çıkış kodunu gösterir.
4. **Boru (pipe)**:
    - Birden fazla komutun ardışık olarak çalıştırılmasını sağlar.
    - Çıktıdan girdiye veri aktarımı yapar.
    - Kullanıcıdan gelen zincirleme komutları ayrıştırıp, her birini doğru sırada işler.

Derleme ve Çalıştırma Talimatları

Projenin derlenmesi için bir **Makefile** dosyası kullanılmıştır. Projeyi derlemek ve çalıştırmak için aşağıdaki adımlar izlenebilir:

1. Makefile kullanarak projeyi derleyin:

```
make
```

2. Kabuk uygulamasını çalıştırın:

```
./bin/shell
```

3. Hataları ve proses durumlarını izlemek için geliştirme sırasında debug modunu etkinleştirin.

Sonuç

Bu proje, Linux tabanlı sistemlerde süreç yönetimi, I/O yönlendirme ve kabuk uygulamalarının çalışma prensiplerini anlamak için etkili bir uygulama olmuştur. Bu proje istenilen bütün görevleri yerine getirmektedir herhangi bir problem yoktur.