İşletim Sistemleri Ödevi (14. Grup)

Grup Üyeleri:
- Eren Çoban - G221210090
- Onur Eğrikılıç - G221210066
- Ozan Malcı - B211210039
- Semih Öztürk - G221210003
- Yiğit Talha Adagülü – G221210093

Desteklenen Özellikler:
- Prompt
- Built-in komutlar (quit ve increment)
- Tekli komut icrası
- Giriş yönlendirme
- Çıkış yönlendirme
- Arkaplan çalışma (process bitince yazdırmıyor)
- Pipe

Proje Amacı

Bu proje, işletim sistemleri dersinde, Linux ortamında temel bir shell (kabuk) uygulaması geliştirmek üzere tasarlanmıştır. Amaç, kullanıcıdan komut almak, bu komutları işletim sistemi seviyesinde çalıştırmak, süreç ve giriş/çıkış yönetimini sağlamaktır.

Sistem Çalışma Prensibi

1. **Kullanıcıdan Girdi Alma:**
   - read_line fonksiyonu kullanılarak kullanıcıdan satır olarak giriş alınır.
   - Bu girdi, split_commands fonksiyonu ile komutlar arasında noktalı virgüle (;) göre ayrıştırılır.
   - Ayrıştırılmış her komut, boşluk karakterine göre split_whitespace ile parçalanarak çalıştırılmaya hazır hale getirilir.
2. **Komut Yürütme:**
   - Tekli komutlar, calistir fonksiyonu üzerinden çalıştırılır.
   - Eğer komutlar boru (|) veya giriş/çıkış yönlendirmeleri (<, >) içeriyorsa bu yapılar uygun şekilde ayrıştırılır ve yönetilir.
   - fork ve execvp sistem çağrıları kullanılarak süreçler oluşturulur ve komutlar işletim sistemi seviyesinde yürütülür.
3. **Özel Komutlar:**
   - increment: Girdi olarak aldığı bir tamsayıyı bir artırır ve çıktısını verir.
   - quit: Shell uygulamasını güvenli bir şekilde sonlandırır.
4. **Giriş/Çıkış Yönlendirme:**
   - < sembolü ile belirtilen dosya standart giriş olarak ayarlanır.
   - > sembolü ile belirtilen dosya standart çıkış olarak kullanılır.
   - Bu işlemler dup2 sistem çağrısı ile gerçekleştirilir.
5. **Boru Yapısı:**
   - Komutların birbirine borular (|) ile bağlandığı durumlarda, her komutun çıktısı bir sonraki komutun girdisi olarak ayarlanır.
   - Bu yapı pipe ve dup2 çağrılarıyla kontrol edilir.
6. **Arka Plan Çalıştırma:**
   - & sembolü ile işaretlenen komutlar arka planda çalıştırılır. Bu süreçler waitpid ile kontrol edilir ve tamamlandıktan sonra kullanıcıya bilgi verilir.

Kullanılan Fonksiyonlar

- **dongu:** Shell'in sürekli çalışmasını sağlayan ana döngü. Kullanıcıdan komutları alır, ayrıştırır ve ilgili fonksiyonları çağırır.
- **calistir:** Komutların yürütülmesinden sorumlu ana fonksiyon. Giriş/çıkış yönlendirmesi ve borular gibi detayları yönetir.
- **_increment:** Sayı artırma işlevi. Komut veya dosya üzerinden gelen girdiyi bir artırır ve çıktıyı ekrana verir.
- **read_line:** Kullanıcıdan standart giriş üzerinden satır okur. Girdi dinamik olarak genişletilebilen bir tamponda saklanır.
- **split_commands ve split_whitespace:** Komut satırını noktalı virgül ve boşluklara göre parçalar.

Teknik Detaylar

- **Sistem Çağrıları:**
  - **fork:** Yeni süreç oluşturmak için.
  - **execvp:** Alt süreçte bir programı çalıştırmak için.
  - **pipe:** Borular ile süreçler arası iletişim kurmak için.
  - **dup2:** Dosya tanımlayıcılarını yeniden yönlendirmek için.
  - **waitpid:** Arka planda çalışan süreçlerin durumunu kontrol etmek için.
- **Bellek Yönetimi:**
  - Kullanıcı girdileri dinamik olarak tahsis edilen tamponlarda saklanır ve ihtiyaç duyuldukça genişletilir.
  - Program sonunda tüm tahsis edilmiş bellek alanları serbest bırakılır.

Sonuç

Bu proje, temel bir Linux kabuğunun çalışma prensiplerini ve işletim sistemiyle etkileşim kuran süreç yönetimi, giriş/çıkış yönlendirme ve boru kullanımı gibi kavramları öğrenmek için başarılı bir uygulama sunar. Proje, hem kullanıcı dostu bir terminal deneyimi sağlar hem de teknik olarak sağlam bir altyapıya sahiptir.