İŞLETİM SİSTEMLERİ ÖDEVİ

39.GRUP

B221210028 Beyazıt Han Bayraktar

B221210034 Berat Yılmaz

B221210373 Ali Aydın

B221210308 Mustafa Erdoğan

G211210002 Gökberk Atasoy

PROJE GİTHUB LİNKİ:

https://github.com/mustaffaerdogan/isletimsistemlerigrup39

Ödev Linux shell yapısının basit bir versiyonunu 8 fonksiyon içerecek şekilde C dili ile yapmamızı istiyor. Aşağıda ödevde yaptığımız main harici 7 fonksiyonun işlevi hakkında bilgi verdik.

1. prompt fonksiyonu: Kullanıcıdan komut bekleme
   Bu fonksiyon, kullanıcının terminalde komut girmesi için bir işaret (prompt) oluşturur. Ekrana > karakterini yazdırır ve tampon belleği temizler, böylece kullanıcı girdisi anında alınır. Fonksiyon oldukça basit bir görev yapar, ancak kullanıcı arayüzü açısından önemlidir. Bu, kabuğun ana döngüsü içinde sürekli çalıştırılır ve kullanıcı her seferinde yeni bir komut girebilir. Amacı, kullanıcıyı yönlendirmek ve beklenen bir komut olduğunu görsel olarak ifade etmektir.

2. quit fonksiyonu: Programdan güvenli çıkış
   Bu fonksiyon, terminalde quit komutu yazıldığında çalıştırılır. Çıkmadan önce, arka planda çalışan tüm işlemleri kontrol eder ve bu işlemlerin bitmesini bekler. İşlemler tamamlandığında, her birinin işlem kimliği (PID) ve dönüş değeri terminale yazdırılır. Bu, kullanıcıya hangi işlemlerin çalıştığını ve başarıyla tamamlanıp tamamlanmadığını anlaması için bilgi sağlar. Son olarak, exit(0) komutuyla program tamamen sonlandırılır.

3. execute_command fonksiyonu: Tek bir komut çalıştırma
   Bu fonksiyon, kullanıcının yazdığı basit bir komutu (örneğin ls, pwd) çalıştırır. Komut, yeni bir alt süreçte (child process) çalıştırılır ve bu süreç tamamlanana kadar ana süreç bekler. Komutun argümanları, boşluklarla ayrılarak bir diziye dönüştürülür ve ardından execvp kullanılarak çalıştırılır. Eğer komut çalıştırılamazsa bir hata mesajı yazdırılır. Bu

fonksiyon, kabuğun temel işlevlerinden biri olarak, kullanıcı komutlarının yürütülmesini sağlar.

4. redirect_input fonksiyonu: Giriş yönlendirme
   Bu fonksiyon, bir komutun girişini bir dosyadan okumak için kullanılır. Eğer komutta < işareti varsa, bu işaretin solundaki komut ve sağındaki dosya adı ayrıştırılır. Alt süreçte dosya açılır, standart giriş (STDIN) bu dosyaya yönlendirilir ve ardından komut çalıştırılır. Giriş dosyası bulunamazsa veya açılamazsa bir hata mesajı gösterilir. Bu fonksiyon, komutların dışardan veri almasını sağlamak için kullanılır.

5. redirect_output fonksiyonu: Çıkış yönlendirme
   Bir komutun çıktısını bir dosyaya yazmak için bu fonksiyon çalıştırılır. Komutta > işareti bulunursa, bu işaretin solundaki komut ve sağındaki dosya adı ayrıştırılır. Yeni bir alt süreç başlatılır ve dosya açılarak standart çıkış (STDOUT) bu dosyaya yönlendirilir. Eğer dosya açılamazsa bir hata mesajı verilir. Bu fonksiyon, komutların çıktısını saklamak veya başka bir işlem için kullanılabilir hale getirmek için gereklidir.

6. background_process fonksiyonu: Arka planda komut çalıştırma
   Bu fonksiyon, kullanıcı bir komutun sonuna & eklediğinde çalışır ve komutu arka planda çalıştırır. Komutun sonunda bulunan & işareti kaldırılır ve kalan komut bir alt süreçte yürütülür. Kullanıcıya, arka planda çalışan işlemin PID değeri gösterilir, böylece bu işlem gerektiğinde izlenebilir. Arka plan işlemleri tamamlandığında, bir sinyal işleyici (SIGCHLD) yardımıyla işlem durumu terminale yazdırılır. Bu özellik, kullanıcıların bir komutu çalıştırırken terminalde başka işlere devam etmesini sağlar.

7. pipe_commands fonksiyonu: İki komut arasında veri akışı sağlama
   Bu fonksiyon, bir komutun çıktısını başka bir komutun girdisi olarak yönlendirmek için kullanılır. Komutlar | sembolü ile ayrıştırılır ve her bir komut bir alt süreçte yürütülür. İlgili süreçler arasında veri akışı (pipe) kurulur, böylece ilk komutun çıktısı doğrudan ikinci komutun girdisi

haline gelir. Tüm pipe'lar kapatıldıktan sonra çocuk süreçler tamamlanır ve ana süreç bitişlerini bekler. Bu özellik, karmaşık komut zincirlerinin oluşturulmasını sağlar.