SAKARYA
ÜNİVERSİTESİ

2024-2025 Güz Dönemi

İşletim Sistemi Ödev Raporu

B211210033 Eren Sancar

B221210087 Fatma Selma Akpınar

B221210045 Habibe Bayram

B221210582 Ömer Elmas

B221210033 Şule Yılmaz

Ödevimizde, bir kabuk (shell) programının geliştirilmesi amaçlanmaktadır. Kabuk programı, kullanıcıdan gelen komutları alır, işler ve uygun çıktıyı sağlar. Bu tür bir program, komut satırı aracılığıyla çeşitli işlemleri yönetmeyi mümkün kılar. Raporda, verilen kodun işleyişi, kullanılan fonksiyonlar ve uygulanan yöntemler açıklanmaktadır.

Komutlar, giriş yönlendirmesi, çıkış yönlendirmesi, boru hattı (pipe) ve arka planda çalıştırma gibi özelliklere sahip olabilir. Program, şu şekilde çalışır. Kullanıcıdan komut alınır. Alınan komut, işlenir ve bir komut yapısına dönüştürülür. Komutlar, gereken işlemleri gerçekleştirecek şekilde çalıştırılır. Çıktılar ya ekrana yazdırılır ya da dosyaya yönlendirilir.

Programın ana fonksiyonu, kullanıcıdan sürekli komut almak ve bu komutları işlemekle sorumludur. İşlem sırasıyla:

Display_prompt() fonksiyonu ile kullanıcıya komut girmesi için bir işaret (>) gösterilir. Kullanıcıdan alınan komut fgets() fonksiyonu ile okunur. Eğer kullanıcı "quit" komutunu girerse, program sonlanır. Kullanıcı girdisi, parse_command() fonksiyonu ile bir Command yapısına dönüştürülür. Bu yapı, komut adı, argümanları, yönlendirmeler (input/output), boru hatları (pipe) gibi bilgileri içerir. Komut, execute_command() veya execute_pipe() fonksiyonları ile yürütülür. Komut işlendiğinde, kullanılan bellek free_command() fonksiyonu ile serbest bırakılır. Parse_command() fonksiyonu, kullanıcının girdiği komutları çözümleyerek bir Command yapısına dönüştürür.

Execute_command(): Tek bir komut çalıştırılmak istendiğinde kullanılır. Bu fonksiyon, komutun giriş ve çıkış yönlendirmelerini ayarlar ve gerekirse komut arka planda çalışacak şekilde başlatılır. Komutun çalıştırılması için fork() kullanılır, ardından execvp() ile komut yürütülür.

Execute_pipe(): Boru hattı (pipe) kullanarak birden fazla komut çalıştırılacaksa, bu fonksiyon devreye girer. Her komut, bir boru ile bir sonraki komuta bağlanır. Her komut için ayrı bir fork() işlemi yapılır ve komutlar paralel olarak çalıştırılır.

Yönlendirme, komutun giriş ve çıkış akışlarını dosyalara yönlendirmeyi sağlar. Bu özellik, dup2() fonksiyonu kullanılarak gerçekleştirilir:

Giriş yönlendirme (<): Komutun giriş verisi bir dosyadan alınır.
Çıkış yönlendirme (>): Komutun çıktısı bir dosyaya yazılır.
Bu durum, dosya tanımlayıcılarının değiştirilmesiyle sağlanır. Komut işlenirken, doğru dosya tanımlayıcıları atanarak komutun doğru şekilde çalışması sağlanır.
Komut sonunda & karakteri varsa, bu komutun arka planda çalıştırılması gerektiği anlaşılır. Bu, komutun fork() fonksiyonu ile bir alt süreçte çalıştırılmasını ve ana süreçten bağımsız olarak çalışmasını sağlar. Bu özellik, kullanıcının diğer komutları girmesine olanak tanırken, önceki komutların arka planda çalışmasını sağlar.

Programda kullanılan dinamik bellek, her işlem sonrası serbest bırakılır. Free_command() fonksiyonu, her bir komut için ayrılan belleği temizler. Bu, komut adı, argümanlar, dosya isimleri ve boru hattı komutları gibi belleğe alınan verileri içerir. Bellek yönetimi, sistem kaynaklarının verimli kullanılmasını sağlar ve bellek sızıntılarının oluşmasını önler.

Kodda her adımda hata kontrolü yapılmaktadır:
Fork() hataları kontrol edilir ve bir alt süreç oluşturulamazsa hata mesajı gösterilir.
Execvp() fonksiyonu ile komut çalıştırılamazsa, hata mesajı kullanıcıya iletilir.
Dosya açma (open()) işlemleri sırasında dosya bulunamadığında veya açılamadığında hata kontrolü yapılır.

Bu hata kontrol mekanizmaları, programın düzgün çalışmasını sağlar ve olabilecek sorunları kullanıcıya bildirir.

Özetle bu ödevde, temel işletim sistemi kavramları ve sistem çağrılarını kullanarak bir kabuk (shell) uygulaması geliştirmeyi amaçlamaktadır. Kullanıcı komutlarını işleyebilmek için yönlendirme, boru hattı (pipe) ve arka plan işlemleri gibi özellikler eklenmiştir. Program, sistem çağrıları, süreç yönetimi ve bellek yönetimi gibi temel işletim sistemi özelliklerini kullanarak çalışır.

Sonuç olarak, bu kabuk programı, kullanıcıların işletim sistemi ile etkileşimde bulunmasını sağlar ve farklı komutları başarılı bir şekilde işlemesini sağlar. Bu tür bir program, komut satırı aracılığıyla çeşitli görevleri yerine getirme yeteneğine sahip bir araçtır.