T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

İŞLETİM SİSTEMLERİ DERSİ

PROJE ÖDEVİ RAPORU

B221210073 - Berkay GÖÇER

B221210078 - Burak ODABAŞ

B211210021 - Buse Nur ÖĞÜNŞEN

B221210040 - Mert BAYIR

G221210073 - Elif GÜNAYDIN

ÖDEV LİNKİ: https://github.com/Burakodbs/GRUP35

GİRİŞ

İşletim sistemleri, modern bilgisayar sistemlerinin çekirdeğini oluşturan karmaşık yazılım yapılarından biridir. Bu sistemler, bilgisayar kaynaklarını verimli bir şekilde yönetmek ve kullanıcı işlemlerini etkin bir şekilde yürütmek amacıyla tasarlanmıştır. Görevlendirme algoritmaları, proses yönetimi, girdi/çıktı kontrolü ve kaynak yönetimi gibi birçok bileşen bu sistemlerin temel işlevleri arasında yer alır.

Kabuk (shell), işletim sistemlerinin kritik bir parçası olarak kullanıcılarla sistem arasında bir köprü işlevi görür. Kullanıcıların komutlarını yorumlayarak uygun sistem çağrılarına dönüştüren kabuklar hem sistem kaynaklarını yönetir hem de bu kaynakların paylaşımını optimize eder. İşletim sistemleriyle etkileşimde bulunan bu yapı, girdi/çıktı yönlendirme, borulama (pipe) ve arka plan işlemleri gibi çeşitli mekanizmaları destekleyerek kullanıcı deneyimini geliştirir ve işletim sisteminin performansını iyileştirir.

PROJENİN AMACI VE KAPSAMI

Projenin amacı, bir kabuk (shell) uygulaması geliştirilerek proses (süreç) yönetimi, I/O (Giriş/Çıkış) ve Linux signal kullanımının temellerini öğrenmektir. Kabuk bir komut satırı yorumlayıcısıdır: Kullanıcıdan alınacak komutlar için sürekli olarak standart girişi okur, karşılık gelen programları (varsa) çalıştırır ve kullanıcı tarafından sonlanana kadar çıktılarını görüntüler. Ana bir prosesten (kabuk prosesi) yeni prosesler oluşturabilir, komutları yürütebilir ve I/O yönlendirmesi (redirection) yapabilir.

PROJE ÖNEM VE GÜNCELLİK

Kabuk uygulamaları, modern bilişim altyapısının temel taşlarından biridir. Özellikle sunucuların, gömülü sistemlerin ve bulut bilişim ortamlarının çoğu Linux tabanlı sistemler kullanmaktadır. Projede ele alınan özellikler, sektörde sık karşılaşılan senaryoları anlamaya ve bu senaryolar için çözümler üretmeye yöneliktir. Proje, sektördeki çoklu görev yürütme, eşzamanlılık ve paralellik gibi temel problemlere yönelik çözüm yollarını anlamak ve geliştirmek için önemli bir deneyim kazandırmıştır.

GELİŞTİRİLEN YAZILIM

Linux Kabuk (Shell): Komut satırından kullanıcı ile işletim sistemi arasında bir arayüz sağlar. Kabuk, komut istemini (prompt) görüntüleyerek kullanıcı tarafından girilen komutları okur. Kullanıcının girdiği metin, tek bir komut veya bir dizi komut içerebilir. Komutlar, kabuk tarafından tanımlanmış yerleşik (built in) komutlar ya da yürütülebilir dosyaların tam yolu ve ilgili argümanlarından oluşabilir. Yürütülebilir dosya isimleri ve argümanlar, karakter uzunluğu veya türü gibi sınırlamalara tabidir. Bu nedenle, çalıştırılabilir adlar ve bağımsız değişkenlerin boşluk içermeyen, doğru formatta dizeler olması gerekir.

Promt (Komut İstemi): Kullanıcıdan komut alınması için bir arayüz sağlar. Prompt, kullanıcı komutları tamamlandıktan sonra tekrar görünür. Kabuk, başladıktan ve her komutun tamamlanmasından sonra veya her arka plan komutunun hemen ardından ">" basar. Print buffer'ı boşaltmak için gerekli kod kullanımı sağlanmıştır.

Quit: Kabuk, quit komutu ile sonlandırılmaktadır. Önceki komutun arka planda çalıştığı zamanda program, yeni komutlara yanıt vermeyi durdurur ve tüm arka plan işlemlerinin bitmesini bekler. İlişkili bilgileri yazdıktan hemen sonra sonlandırılır.

Tekli Komutlar: Sisteme ait bir komutun veya programın yürütülmesini sağlar. Kabuk bağımsız değişkenlerle veya bağımsız değişkenlerle tek bir komutu okuduğunda komutu çalıştırır, komut tamamlanana kadar engellenir ve ardından istemine geri döner. Kabuk bir alt

proses oluşturur ve onu komutla belirtilen programa dönüştürür. Program işlemin tamamlanmasının ardından komut promt'unu görüntüleyerek yeni bir komut bekler. **Fork, exec, wait, waitpit** gibi ilgili LINUX sistem çağrıları kullanılmıştır.

Proses (Süreç) Yönetimi: İşletim sistemi tarafından yeni süreçlerin oluşturulması, yürütülmesi ve izlenmesi süreçlerini içerir. Çoklu proses sistemlerinde öncelik sıralaması ve kaynak tahsisi gibi yönetimsel işlemler bu kapsamdadır.

I/O (Giriş/Çıkış) Yönlendirme: Kabuk, her komutun standart çıktı ve girdisini yeniden yönlendirebilir. Komutların giriş ve çıkışlarının dosyalara, diğer komutlara veya diğer cihazlara yönlendirilmesini sağlar. Bu mekanizma, sistemdeki veri akışının kontrol edilmesini ve kaynakların verimli kullanımını sağlar.

Giriş Yönlendirme: Girişe yeniden yönlendirme (redirection) yapan bir komut girildiğinde program verilen komutu çalıştırmadan önce alt prosesin standart girdisini verilen giriş dosyasına yönlendirir. Hiçbir girdi dosyası yoksa "Giriş dosyası bulunamadı" yazdırır. İlgili LINUX sistem çağrısı **dup2** kullanılmıştır.

Çıkış Yönlendirme: Çıktısı yeniden yönlendirme (redirection) yapan bir komut girildiğinde program alt prosesin standart çıktısını çıkış dosyasına yönlendirir. İlgili LINUX sistem çağrısı **dup2** kullanılmıştır.

Boru (Pipe) Mekanizması: Komutlar arası veri akışını sağlamak için pipe sistemini kullanır. Bu yapı karmaşık iş akışlarını kolaylaştırır.

Birden fazla komut birbiriyle borular aracılığıyla bağlandığında i. komutunun çıkışı (i + 1) inci komutunun girişine beslenir. Son komutun çıkışı, dosyaya yönlendirilmediği sürece standart çıktıya bağlanır. İlk komutun girişi, işlemin giriş gerektirmesi durumunda standart girdiden başka bir dosyadan da yönlendirilebilir. Komut promptu görüntülenmeden önce pipe içindeki tüm komutların sonlanması gerekir. İlgili LINUX sistem çağrıları **pipe** ve **dup2** kullanılmıştır. Proseslerin stdin ve stdout dosya descriptörleri kullanılmıştır.

Arka Plan Prosesler: Proseslerin arka planda çalıştırılarak kullanıcıyla etkileşim devam ederken farklı görevlerin eşzamanlı olarak yürütülmesini sağlar. Arka planda çalışan süreçlerin durumlarının takip edilmesi ve sonuçlarının kullanıcıya iletilmesine olanak tanır.

Bir komut '&' ile bittiğinde arka planda çalıştırılır. Komutun tamamlanması için kabuk engellenmemekte ve hemen komut prompt'u görüntülenmektedir. Daha sonra kullanıcı yeni komutlar girebilir. Arka plan işlemi sona erdiğinde kabuk, çıkış durumunu proses kimliği ile [pid] retval: <exitcode> bilgilerini kullanıcıya bildirir.

İkinci prompt kabuk tarafından derhal yazdırılır ancak pid ve dönüş değeri arka plan işleminin sonlandırılmasından sonra yazdırılır.

Kabuk, arka planda çalışan bir proses varken bir quit komutu alırsa prompt'u görüntüler ve yeni komut girişini durdurur, tüm arka plan işlemleri sona erene kadar bekler. Daha sonra ilgili bilgileri bastırır ve derhal çıkar. İlgili LINUX sistem çağrısıları olan **sigaction** ve **WEXITSTATUS** kullanılmıştır.

Program ödevde istenenlere uygun formatta kaynak dosyaları (*.c ve *.h dosyaları), Makefile ve Readme dosyalarını geri döndürmektedir.