İŞLETİM SİSTEMLERİ PROJE ÖDEVİ 2024 GÜZ

SAKARYA
ÜNİVERSİTESİ

Grup 3 üyeleri :

Hüseyin Göbekli (G211210041) 2B

Okan Başol (G211210083) 2B

Şimal Ece Kazdal (G221210068) 2B

Muhammed İrfan Bakar (G221210596) 2B

Betül Kurt (G221210054) - 2C

Proje Tanımı ve Amacı :

Bu proje, temel bir Linux kabuk (shell) uygulamasını geliştirerek işletim sistemi kavramlarını pratikte anlamayı ve uygulamayı amaçlamaktadır. Linux kabuğu, kullanıcıdan alınan komutları çalıştırarak çıktısını sağlayan bir komut satırı arayüzüdür. Proje kapsamında, süreç yönetimi, giriş ve çıkış yönlendirme (I/O yönlendirme), boru kullanımı (pipe) ve Linux sinyal işleme gibi temel işletim sistemi kavramları detaylı bir şekilde incelenmiştir.

Bu çalışmada, Linux kabuğunun temel işlevselliklerini sağlayan bir sistem geliştirilmiş, komut çalıştırma, yönlendirme, boru işlemleri, arka plan süreçleri ve komut geçmişi gibi özellikler detaylı bir şekilde uygulanmıştır. Projenin sonucunda ortaya çıkan yazılım, kullanıcıların Linux kabuğundaki temel işlemleri rahatlıkla gerçekleştirmesine olanak sağlamaktadır.

Proje Süreci

Projenin Planlanması:

Projenin başlangıcında, Linux kabuğunun genel çalışma prensipleri detaylı bir şekilde analiz edildi. Kabuk uygulamasının, kullanıcıdan aldığı girişleri nasıl işleyeceği, yönlendirme işlemlerini nasıl gerçekleştireceği ve boru hattı (pipe) bağlantılarını nasıl kuracağı üzerine detaylı bir plan hazırlandı. Bu süreçte, Linux sistem çağrıları ve sinyal işleme mekanizmalarının projeye entegrasyonu planlandı. Ayrıca, projenin kapsadığı tüm işlevlerin birbirleriyle uyumlu çalışması ve kullanıcı deneyimini kolaylaştıracak şekilde tasarlanması hedeflendi.

Kabuk İsteminin Tasarımı ve Temel Yapının Oluşturulması

İlk adımda, kabuk çalıştırıldığında kullanıcıdan giriş almasını sağlayan bir istem (prompt) tasarlandı. Kullanıcının komut girebileceği bu arayüz, her komutun çalıştırılmasının ardından yeniden görüntülenerek kullanıcıyı bir sonraki komutu girmeye teşvik etti. Ayrıca, kullanıcıdan alınan girişlerin doğru bir şekilde ayrıştırılması ve işlemlere hazır hale getirilmesi için gerekli fonksiyonlar yazıldı. Bu aşama, kabuğun temel yapısının oluşturulması için önemli bir başlangıç oldu.

Komut Çalıştırma Mekanizması

Kabuk, kullanıcıdan aldığı komutları bir alt süreç (child process) oluşturup bu süreçte çalıştıracak şekilde tasarlandı. Alt süreçlerin oluşturulması için fork çağrısı, komutların çalıştırılması için ise execvp fonksiyonu kullanıldı. Bu mekanizma sayesinde, ana süreç (kabuk) alt süreçlerin tamamlanmasını bekleyip ardından kullanıcıdan yeni komut alabilecek hale getirildi. Komutların eksiksiz bir şekilde çalıştırılması ve ardından çıkan sonuçların kullanıcıya sunulması bu aşamada başarıyla uygulandı.

Giriş ve Çıkış Yönlendirme

Kullanıcılar, < operatörü ile bir dosyadan giriş alabilir ve > operatörü ile çıktıyı bir dosyaya yönlendirebilir. Bu özellik, dup2 sistem çağrısı kullanılarak alt süreçlerin standart giriş ve çıkışlarının yeniden yönlendirilmesiyle gerçekleştirildi. Örneğin, bir dosyadaki veriler ekrana yazdırılabilir veya bir komutun çıktısı başka bir dosyaya aktarılabilir. Giriş ve çıkış yönlendirme işlemleri, kabuk uygulamasına dinamik bir yapı kazandırmıştır.

Boru (Pipe) Kullanımı

Proje kapsamında, boru hattı (pipe) işlemleri de detaylı bir şekilde uygulanmıştır. Borular, bir komutun çıktısını bir sonraki komutun girdisi olarak kullanmasını sağlar. Örneğin, bir dosya listesinden belirli bir uzantıya sahip dosyaları filtrelemek için ls | grep .txt gibi bir komut çalıştırılabilir. Bu işlemler, pipe ve dup2 sistem çağrılarıyla gerçekleştirildi. Bu özellik, özellikle veri akışının zincirleme şekilde işlenmesini gerektiren durumlarda büyük bir kolaylık sağlamıştır.

Arka Plan Süreç Yönetimi

Kabuk, kullanıcıların komutları arka planda çalıştırmasına da olanak tanır. Bu özellik sayesinde, komutun sonuna & eklenerek komut arka planda yürütülür ve kullanıcı yeni komutlar girmeye devam edebilir. Örneğin, sleep 5 & komutu arka planda çalıştırıldığında, kullanıcı diğer işlemlerine devam edebilir. Ayrıca, arka plan süreci tamamlandığında, kabuk otomatik olarak sürecin kimlik bilgisini (PID) ve çıkış durumunu kullanıcıya bildirir. Bu özellik, çoklu işlemlerin yönetilmesini kolaylaştırmıştır.

Komut Geçmişi ve Yeniden Kullanımı

Kabuk, kullanıcı tarafından girilen tüm komutları bir listeye kaydeder ve history komutuyla bu geçmişin görüntülenmesini sağlar. Komut geçmişi, kullanıcıların daha önce çalıştırdığı komutları görmesini ve gerektiğinde tekrar çalıştırmasını kolaylaştırır. Bu özellik, kullanıcı deneyimini zenginleştiren önemli bir işlevdir.

Çoklu Komut İşleme

Kullanıcı, ; operatörüyle ayrılmış birden fazla komutu aynı satırda çalıştırabilir. Kabuk, bu komutları sırasıyla çalıştırır ve her bir komutun sonuçlarını ayrı ayrı görüntüler. Örneğin, echo 12; sleep 2; echo 13 komutu çalıştırıldığında, önce 12 ekrana yazdırılır, ardından 2 saniye beklenir ve sonunda 13 yazdırılır. Bu özellik, zincirleme işlemlerin kolayca yürütülmesini sağlar.

Sinyal İşleme

Linux sinyallerinin işlenmesi, kabuk uygulamasına esneklik kazandırmıştır. Örneğin, kullanıcı Ctrl+C tuş kombinasyonunu kullandığında kabuk bu sinyali yakalar ve çalışmaya devam eder. Benzer şekilde, Ctrl+Z sinyali de uygun şekilde işlenmiştir. Bu özellik, kabuğun kesintisiz bir kullanıcı deneyimi sunmasını sağlar.

Sonuç

Proje, temel bir Linux kabuk uygulamasının geliştirilmesini hedefleyen kapsamlı bir çalışmadır. Kabuk uygulaması, süreç yönetimi, giriş/çıkış yönlendirme, boru işlemleri, arka plan süreçleri ve sinyal işleme gibi özelliklerle donatılmıştır. Geliştirilen kabuk, hem kullanıcı dostu bir arayüz sunmuş hem de işletim sistemi kavramlarını pratikte uygulama fırsatı sağlamıştır.

Bu proje, işletim sistemi kavramlarının daha iyi anlaşılmasını sağlamış ve Linux tabanlı sistemlerde çalışmanın temellerini güçlendirmiştir. Ortaya çıkan uygulama, temel Linux işlemlerini başarıyla gerçekleştirebilen bir araç olarak tamamlanmıştır.