**Kabuk (Shell) Nedir?**

Linux'un da içinde bulunduğu UNIX sistemlerinde komutları yorumlamak ve yönetmek için kullanılan programa **kabuk (shell)** denir. Kabuk, bilgisayarın yönetimini uygulamalar ele almadan önce bilgisayarları çalıştırmak için kullanılan komutlarnı yazıldığı bir tür paneldir ve kabuk bütün Linux sistemlerde bulunması zorunlu olan birimdir. Linux'ta farklı terminal benzetici (emulator) programları mevcuttur. Bunlardan bahsedilmesi gerekirse;

• **xterm:** X Window sistemi için genel bir terminal benzetici programıdır. Menüleri bulunmaz.

• **gnome** terminal: GNOME ile birlikte gelen varsayılan terminal benzetici programıdır. Xterm emülatörü ile ciddi benzerlikleri vardır. Renkli metin düzenleme ve pencerelerde fare komutlarının işlemesi bunlar arasında belli başlılarıdır. Bu benzerliğin yanında Gnome Terminal emülatörü Xterm'den daha fazla sistem kaynağı tüketir. Ayrıca Gnome Terminal emülatörü, metinlerin içine gizlenen URL'leri görünür haliyle kullanarak internet tarayıcısı ya da ftp sunucuna erişimi sağlar.

• **konsole**: KDE masaüstü ortamıyla gelen emülatör programıdır. 'konsole' ile çoklu dil metin kodlaması ve metni farklı renklerle görüntülenebilir.

Terminal penceresini açıldığında ekrana ilk gelen çıktı aşağıdaki gibidir.

kullanıcıadı@bilgisayaradı:~$

İlk olarak kullanıcı adı belirtilmiştir. @ (amphora)'dan sonra gelen kısım bilgisayar adını belirtir. ~ (tilde) işareti ise o anda bulunulan dizinin yerine kullanılmıştır. Örneğin; /usr adındaki dizinine geçilmiş olsaydı ~ yerine /usr yazılması gerekirdi.

root@bilgisayaradı:/usr #

En son karakter ise açılmış olan hesabın yetki seviyesini belirtir.

Varsayılan ayarlarla genel olarak $ (dolar) işareti tanımlı olarak gelecektir.

• $ (dolar) işareti normal bir kullanıcı için varsayılan işarettir. Bir diğer adıyla sınırlı kullanıcı oturumunda bulunduğunu gösterir.

• # (diyez), root kullanıcısı için varsayılan işarettir. Buna karşın istenilen herhangi bir karakteri görüntülemek için istem değiştirilebilir.

**Projede Kabuk (Shell)**

Projeye pipe ve named pipe hakkında araştıma yaparak başladım. Öncelikle çalışan bir named pipe yapmaya çalıştım. Named pipe iki farklı programın işletim sisteminin yarattığı bir dosya üzerinden haberleşmesiymiş bunu öğrendim. İlk çalışan named pipe uygulamamı yazdıktan sonra kalan ödevin en zor kısımlarından birisini halletmiş oldum. Sonrasında named pipe i kullanarak program.c den database.c ye veri gönderdim ve karşı databese.c de bu veriyi okuyup ekrana bastım.

Bu adımı gerçekleştirdikten sonra gerçek sorguyu program.c de kullancıdan fgets kullanarak aldım. Aldığım sorguyu named pipe aracılığıyla database.c ye gönderdim. Database.c de bu sorguyu parçalarına ayırdım. İstenen kısım için 3 farklı ihtimal vardı. Bunlar '\*', 'ad' ve 'number' verileri idi. Bu 3 ihtimale göre koşullarımı yazdım. Sonrasında gelen sorgu içerisinden hangi dosyada okuyacağımı bulmaya çalıştım. Sorguyu boşluklara göre parçalara bölerek dosya adını çıkarttım. Bu bilgiye göre uygun dosyayı okumaya başladım. Aradığım değeri gördüğümde dosyayı okumayı bıraktım ve tekrar named pipe adımına geldim. Bulduğum verininin kullacının istediği kısımlarını named pipe aracılığı ile program.c ye gönderdim. gelen verilerileri program.c de okuduktan sonra kullanıcıya gösterdim ve kaydetmeyi isteyip istemediğini sordum. Kullanıcı evet cevabını verirse ise fork kullanarak bir çocuk proses yaratıp bu proses ile exec komutunu kullanarak keydet isimli programı çalıştırdım. Kaydet isimli programa argüman olarak gönderdiğim sorgu sonucunu sonuc.txt dosyasına kaydettim ve prosesi terminate ettim. Bu sıradan Program çocuk prosesin terminate olmasını bekliyordu. Bu terminate olduktan sonra bekleme işlemini bitirip tekrar sorgu almak için kullancıdan girdi beklemeye başladı.

Anladığım kadarıyla projenin tamamını yapmaya çalıştım. Bu sırada databaseden yapılan sorguları simule etmiş oldum. Bu ödev programlar arası iletişimi anlamama çok yardımcı oldu.