SED

\*Stream editörün kısaltmasıdır.

\*Ed line editörden türetilmiştir.

\*Sed ‘3,4p’ foo.txt // İlgili verilen akış üzerinde verdiğimiz aralığın duplicate(tekrardan yazılması) etmeyi sağlıyor. (tek parametre ile çalışmıyor)

\*sed ‘4q’ foo.txt dosyanın başından itibaren verilen satıra kadar yazdırma işlemi yapar ve geri kalanı yazdırılmaz. Mesela 4. Satırda basmayı bırakacak bu kod.

\* sed -n ‘3,4p’ foo.txt sadece verilen aralığın gösterilmesini sağlar. Sadece ilgili kısmı getiriyor.

\*sed -n ‘$p’ foo.txt ilgili dosyanın sadece son satırını getirdi.

\*sed -n ‘3,$!p’ foo.txt verilen aralıktaki satırların dışındaki satırları getirir. 3. Satırdan sonrakileri getirir.

NOT: kısacası -n verirsek bastırıyor, -n yazmazsak duplicate yapıyor.

SED DÜZENLİ İFADELER İLE KULLANIMI

\*sed -n ‘/^ FROM: /p’ $HOME/mbox “$HOME/mbox” dosyası üzerinde “ From:” (<BOŞLUK> From:) ile başlayan satırları bana göster diyoruz.

Ls -l | sed -n ‘/^…..w/p’ Başlangıçtan itibaren 5 karakter ne olursa olabilir 6. karakter “w” olsun. Ls çıktısında 1. Değer dosya tipiydi, diğerleri 3er ugo. Yani aslında grubun yazma hakkın sahip dosyalarını listelemiş olduk.

Sed:Substition DEĞİŞTİRME İŞLEMİ

Sed ‘s/|/:/g’ data.txt burada slashlar arasında kalan karakterleri değiştirir. Yani her bulduğu “/” için “:” koyar. Buradaki “g” aramanın sürekli yapılacağı anlamına geliyor. Yani işlemi yaptıktan sonra bir sonraki işlem kaldığımız yerden devam eder. “g” yi vermezsek en baştan tekrar arar

AWK

Aho, Weinberger ve Kerninga oluşturan kişiler ve baş harflerinden oluşuyor.

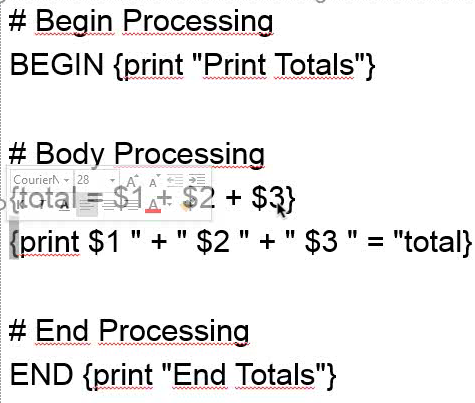
Awk -f awk.script foo.txt

Begin içerisine, belirtilen dosyayı okuma işlemine geçmeden önce yapmak istediğimiz işlemleri buraya yazıyoruz.

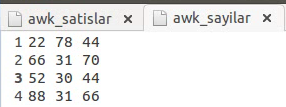
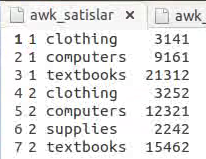
Ortadaki süslü parantezler içerisinde ise, alınan dosyanın HER BİR SATIRI için gerçekleşecek işlemleri yapıyoruz.

End içerisinde ise Okuma işlemi bittikten sonra gerçekleştirilecek işlemlerin belirtildiği yerdir.

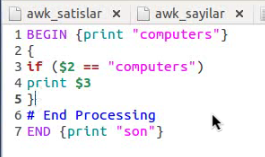
Örnek:



Yukarıdaki örnekte dolarlar ile belirtilen yerler sütunlara tekabül eder.

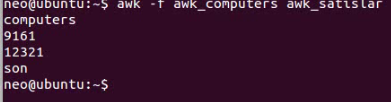


Yukarıdaki kod 1. 2. Ve 3. Sütunları toplayarak total e atıyor ve onu print ediyor.

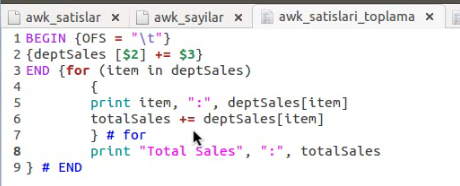
yandaki kod parçasında 2. Sütundaki her eleman computer mi diye kontrol ediyor, eğer öyle ise gidip fiyatını yazdırıyor. Tüm computerler bitince de son yazdırıyor.

Yukarıdaki kodu koşmak için;

Awk -f awk\_computers awk\_satislar yazıyoruz. Çıktı:



AWK Döngüleri

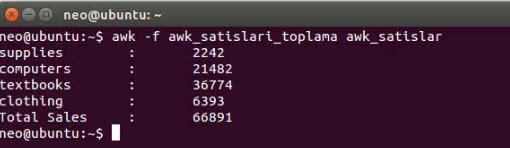


Yukarıdaki kod parçasında OFS bir çevre değişkenidir ve bu değerin default değeri tek bir boşluktur, eğer biz bunu değiştirirsek yazdırma anında elemanlar arasındaki karakteri değiştirmiş oluyoruz. Yani kısaca basma işleminde kullanılacak işlem boşlul değil, “TAB” olacağı için (\ t) tab bırakacak aralara. Sütunsal bazda hizalama yapmak için.

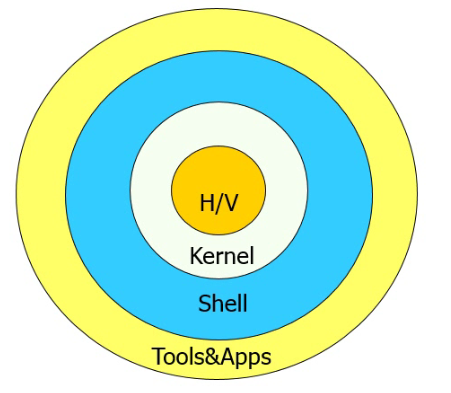
Her bir turda yani ilk satırda icra edilecek kısım deptSales[$2], gidip clothingi alıyor sonra değerini topluyor, 2. Adımda gidip computersi topluyor, yani aslında deptSales[computers] gibi bir indisin içerisine değer atıyor. Böyle bir durumda Computers için listedeki tüm computersleri toplayarak tek bir index içinde toplam değerlerini toplamış oluyor.

For döngüsü içerisinde itemleri alıyor, aldığı itemleri yazdırıyor, sonrasında itemin bulunduğu indexi diziye verip o birimin toplam satışını görmüş oluyoruz. Total sales içerisinde ise bütün birimlerin kendi değerlerini tekrar toplayarak totaldeki satışı oluşturuyor, for bitince de toplam yapılan bütün genel satış değerini yazdırıyor.

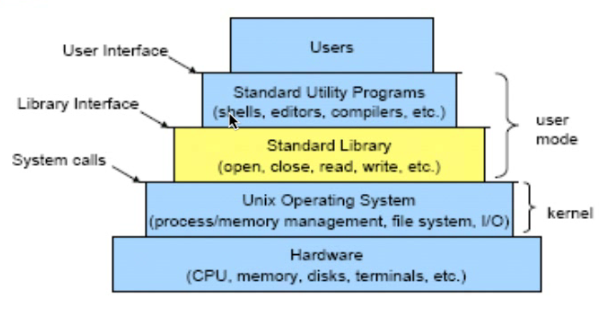
Çıktı:



FILE MANAGEMENT



Unix tabanlı sistemlerin katmanları:



UNIX SİSTEM PROGRAMLAMA

Programlar sistem çağrılarını kullanarak kütüphanelere erişirler.

Sistem çağrı tipleri:

\*File I/O dosya giriş çıkışları

\*Process Management process yönetimi

\*Inter Process Communication(IPC)

\*Signal Handling

C dili vs C++

C de string tipi diye bişey yok.

C için;

\*Strcpy() kopyalıyor

\*Strcmp() karşılaştırma yapıyor.

C++ de printf ve cout var

C de scanf ve fgets() var

BASIT DOSYA GİRİŞ ÇIKIŞLARI I/O

Dosyalar okuma veya yazma modunda açılabiliyor.

Bu sistem çağrılarını kullanabilmek için <stdio.h> (Standard input output kısaltması) incude etmemiz lazım

Her bir dosya bir dosya işaretçisi tarafından referanslanır.

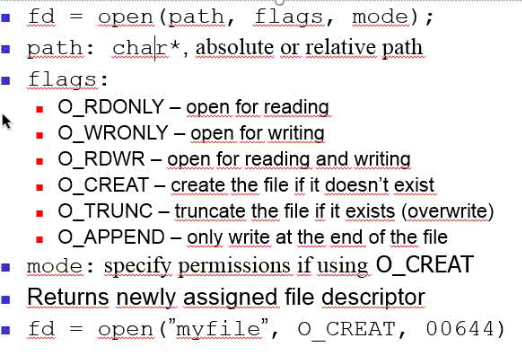
3 tane dosya otomatik olarak açılır:

FD 0 standart input

FD1 standart output

FD2 standart error

Open() FONKSİYONU



Open içindeki flag dediği şey dosy tipidir. Mode hangi modda açılacağı. Path ise dosyanın yolunu verir.

O\_RDONLY ylnızca okunur

O\_WRONLY sadece yaılabilir.

O\_RDWR okunabilir ve yazılabilir.

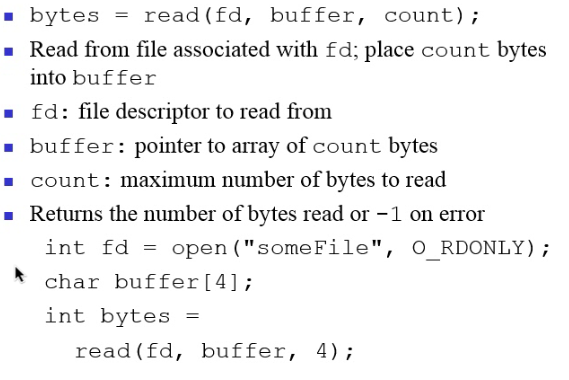
O\_CREAT öyle bir dosya yoksa oluşturur

O\_TRUNC dosya varsa içini boşaltır

O\_APPEND dosyanın sonuna ekler.

Fd= open(“myfile”, O\_CREAT, 00644) burada program koşunca myfile diye dosya oluşturacak sürekli.

Red() FONKSİYONU

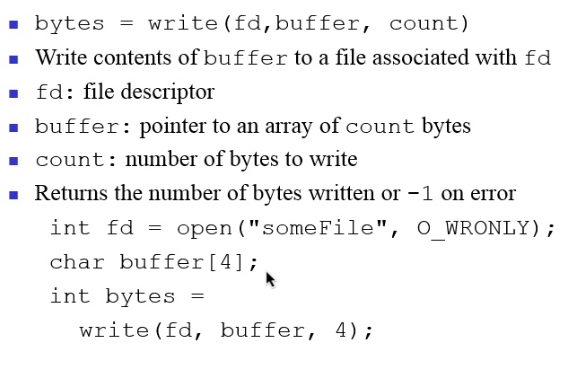


1.parametresi Okuma yapacağımız dosya tanımlayıcısı.

2.parametresi Okunacak verilerin aktarılacağı dizidir.(buffer)

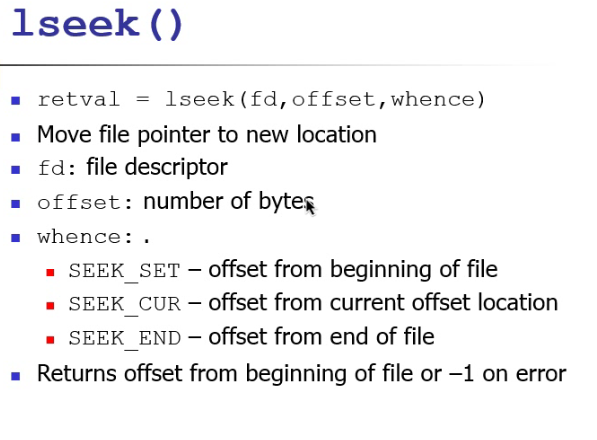
3.parametresi Dosyadan kaç tane veri okuyacağımızı gireriz. (genelde buffer kadar beklenir ama daha az da olabilir.)

Write() FONKSİYONU



Close() fonksiyonu ile işleminiz bitince kapatabilirsiniz dosyayı.

Lseek() fonksiyonu ile arama işlemleri yapılır.



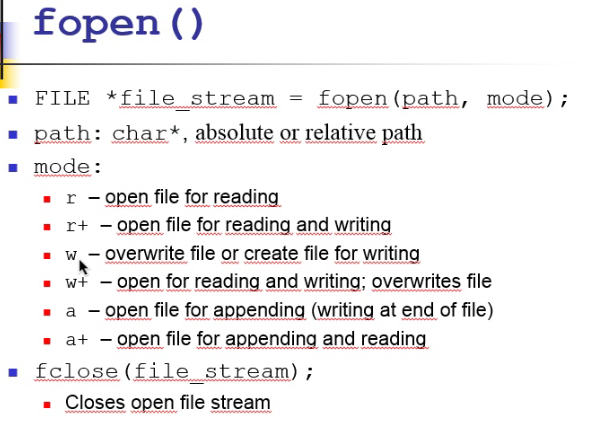
2. parametre Offset değeri dosya içinde nerede olduğumuz yer.

3. parametre offset değerinin baz alınacağı durum.

SEEK\_SET Dosyanın başını gösteriyor

SEEK\_CUR aradığımız veriyi bulunca kaldığımız yerden devam edebilmemiz için o konumda CUR kullanılıyor.(recursive bi mantıkla dosya sonuna kadar gidebilmek için)

SEEL\_END aynı şekilde CUR gibi ancak sondan arama yapıyor.



\*r dosyayı okuma modu

\*r+ okuma yazma modu

\*w dosyayı yazma modunda açıyoruz, dosya yoksada oluşturur

\*w+ modunda açarsak dosyayı hem okuma hem yazma modunda açıyoruz, eğer dosya varsa da içini boşaltıyoruz.

\*a dosyayı sondan eklemek için açıyor

\*a+ dosyayı hem sondan eklemek hemde okumak amacıyla açıyor.

Fclose ile içine verilen dosyayı kapatıyor kod içinde.