Filesystems:

Filesystem dizinlerin hiyerarşik olarak düzenlenmesi demektir.

Unixte,kök dosya systemi "/" ile başlar.Bunun yanında kök başlangıcın kısımları olan diğer subfilesystemler de vardır.Makinenizdeki filesystemleri görmek için "df" komutunu girebilirsiniz.Bunun sonucunda şöyle bir şey göreceksiniz;

```
UNIX> df
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/sd0a              11167    5952    4099    59%    /
/dev/sd0g             140591   84016   42516    66%    /usr
/dev/sd0h             171959  141421   13343    91%    /blugreen/homes
-memory-               55720       0   55720     0%    /tmp
plank:(pid130)             4       4       0   100%    /amd
galoob:/export/sun4-sos4_1
                      335803  313356       0   100%
/a/galoob/export/sun4-sos4_1
cs:/var/spool/mail    221351  122607   76609    62%    /a/cs/var/spool/mail
alphard:/canary/homes
                      411687  293685   76833    79%
/a/alphard/canary/homes
UNIX>
```

Burda her satır sonundaki giriş farklı bir filesystemi gösterir.Farkedeceğiniz üzere bunlar "/" nin alt dizinleridir.Her bir filesystem farklı diskte veya disk partisyonunda(bölümünde) bulunmaktadır.

Filename: Bir dizinde görünen dosya adı.

Pathname: Eğik çizgi ile ayrılmış sıfır ya da daha fazla dosyadan oluşan yol adı.

"Is-a" komutunu girdiğinizde geçerli dizindeki tüm dosya adlarını listeler.

```
UNIX> cd ~huangj/cs360/notes/Rbtree-1
UNIX> ls -a
.               jh.c            makefile        mysorti.c       mysortu2.c
..              lecture         mysort.c        mysortu0.c      mysortu3.c
README          lecture.html    mysort2.c       mysortu1.c      randfile
UNIX>
```

("-a" yazmazsanız , "." İle başlayan dosya adlarını listelemez.) Not: Her dizinde iki dosya adı vardır: "." ve ".."

"." Geçerli dizindir ".." geçerli dizinin üstüdür.

Pwd komutu geçerli dizinin tam yolunu söyler. Cd komutu sizi dizinleri arasında taşır:

```
UNIX> cd ~huangj/cs360/notes/Rbtree-1
UNIX> pwd
/home/huangj/cs360/notes/Rbtree-1
```

```
UNIX> cd .
UNIX> pwd
/home/huangj/cs360/notes/Rbtree-1
UNIX> cd ..
UNIX> pwd
/home/huangj/cs360/notes
UNIX> cd ../../..
UNIX> pwd
/home
UNIX> cd .././././.
UNIX> pwd
/
UNIX> cd ..
UNIX> pwd
/
UNIX>
```

Not : "/" ana dizinin kendisidir.

Absolute Pathname: Eğik çizgi ile başlayan yol adı.

Relative Pathname: Eğik çizgi ile başlamayan yol adı.

Working Directory: Relative pathname ile ilişkili dizin.

Home Directory: Kullanıcının ilk girişindeki working directory**.**

I/O

I/O detaylarına daha sonra gireceğiz.

Programlar ve süreçleri:

Program: Ya doğrudan,yada tercüman yardımı ile derleyiciler ve/veya bağlayıcılar tarafından yürütülebilir bir dosya.

Process: Bir programın yürütülme örneği.

Process ID: İşletim sistemi tarafından bir prosese verilmiş numara.

Program örnekleri:

- /bin/ls- - bu doğrudan yürütülebilir bir programdır.
- /usr/ucb/vi- -buda doğrudan yürütülebilir bir programdır.
- ~huangi/cs360/notes/Rbtree-1/mysort.c- -bu çalıştırılabilmesi için derleyiciye ihtiyacı olan bir programdır.
- /blugreen/homes/plank/bin/calc- -bu bir kabuk programıdır- -bu program /bin/sh. Tarafından çalıştırılmaya ihtiyaç duyar.

Bir programı çalşıtırdığınızda, bu çalıştırılma aşamasında bir süreç olarak adlandırılır. Örneğin, bir pencereye “vi ~ huangi/cs360/notes/Rbtree-1/mysort.c” yazdığımızı varsayalım. Bu vi programı /usr/ucb/vi. Klasörü içinde bulunur ve çalıştırılır.bu yürütme örneği bir proses olarak adlanrılır.eğer başka bir pencereye giderseniz ve “ps x” yazarsanız bu şuanda çalışmakta olan bütün süreçleri /işlemleri listeler:

```
UNIX> ps x
 1394 p0 IW    0:00 xclock -geometry 100x100-0-0 -update 60
 1416 p0 S     0:01 twm
 1436 p1 IW    0:00 -sh (csh)
 1427 p2 IW    0:00 -sh (csh)
 1428 p3 IW    0:00 -sh (csh)
 1443 p5 S     0:01 -sh (csh)
 2328 p5 S     0:00 vi /home/huangj/cs360/notes/Rbtree-1/mysort.c
 1444 p6 IW    0:00 -sh (csh)
 2329 p6 R     0:00 ps x
UNIX>
```

Aynı anda birden fazla vi işlemini çalıştırabileceğinizi unutmayın.diğer bir pencereye gidin ve “vi/home/cs360/notes/Rbtree-1/mysort2.c” yazın.şimdi “ps x” yazdığınızda ikinci işlemleri göreceksiniz.

```
UNIX> ps x
 1394 p0 IW    0:00 xclock -geometry 100x100-0-0 -update 60
 1416 p0 S     0:01 twm
 1436 p1 IW    0:00 -sh (csh)
 1427 p2 S     0:00 -sh (csh)
 2330 p2 S     0:00 vi /home/huangj/cs360/notes/Rbtree-1/mysort2.c
 1428 p3 IW    0:00 -sh (csh)
 1443 p5 IW    0:01 -sh (csh)
 2328 p5 IW    0:00 vi /home/huangj/cs360/notes/Rbtree-1/mysort.c
 1444 p6 S     0:00 -sh (csh)
 2331 p6 R     0:00 ps x
UNIX>
```

Program ile process arasındaki farkı anlamanız gerekir. **Ps** komutunun birinci sütunundaki numara "process id’dir”.

Kullanıcı Kimliği

• **Kullanıcı ID**: *Sistem yöneticisi tarafından her kullanıcıya verilen numaradır.*

Sistem tarafından kullanıcıyı tanımlamak için 2 yol vardır: User Id ve group id. Group id hakkında fazla bahsetmeyeceğiz. Mevcut UserID’niz "**getuid()**" sistem çağrısı tarafından veriliyor– ana sayfayı oku.

Mevcut UID’iniz görüntülemek için **ch1b.c**’yi deneyin:

```
#include < stdio.h >

main()
```

```
{
  printf("%d\n", getuid());
}
```

Sinyaller

- **Sinyal**: *Bir program kesmesidir*.
- **Sinyal İşleyici**: *Programın sinyallerin incellikle üstesinden gelmek için kullanıldığı mekanizma*.

Sinyaller program içinde eş zamanlı olmayan olaylar için bir yöntem ya da gidişattır. Örneğin, **ch1c.c** programını derleyin ve çalıştırın:

```
#include < stdio.h >

main()
{
  int i;

  i = 0;
  while (1) {
    i++;
    printf("%d\n", i);
    sleep(5);
  }
}
```

Bu program kendini her 5 saniyede bir arttıran sayaç işletir. Programın birkaç saniye çalışmasına izin verin, daha sonra <CNTL-Z> yazın. Bu, programa “DUR” sinyalini gönderir ve programı durdurur. Şimdi tekrar kabukta olacaksınız. Eğer “ps” yazarsanız, aşağıdakine benzer bir yazı görürsünüz.

```
2483 p5 T 0:00 ch1c
```

Buradaki “T” programın işletilmesinin durduğunu gösterir. Başlatmak için, programa “BAŞLAT” sinyalini gönderecek olan “fg” komutunu yazabilirsiniz.

Şimdi program çalışır durumdayken <CNTL-C> yazarsanız programı kesen bir “INT” sinyali gönderip programı öldürürsünüz. Segmentasyon hataları da aynı zamanda birer sinyaldir. Bu sinyallerle başa çıkmak için programınızı istediğiniz şekilde yazabilirsiniz. Bu, dönem içinde daha sonra göreceğimiz ileri düzey bir programlama tekniğidir.

Sinyal, ANSI C standardının bir parçasıdır ve ANSI ’87 destekleyen her sistemde aynı şekilde çalışmalıdır. Asıl farklılıklar bazı ekstra sinyallerden kaynaklanır ve

bunlar asıl sinyal setlerinin ötesinde farklı sistemlerde de desteklenir. ANSI C tarafından belirtilen ana sinyal seti oldukça küçüktür.