## Objectives

- Ideas and Skills
  - User's view of the Unix file system tree
  - Internal structure of Unix file system: inodes and data blocks
  - How directories are connected
  - Hard links, symbolic links: ideas and system calls
  - How pwd works
  - Mounting file systems
- System calls and functions
  - mkdir, rmdir, chdir
  - link, unlink, rename, symlink
- pwd

## Directories and files

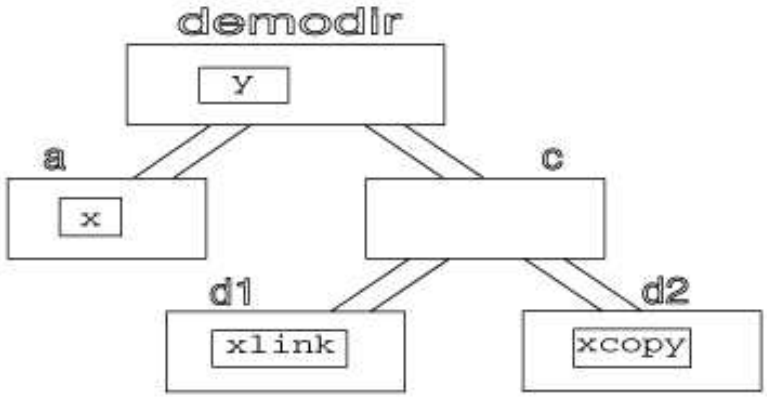



Figure 1: A tree of directories

## Unix Commands for directories and files

```
$mkdir demodir
$cd demodir
$mkdir demodir/a
$mkdir demodir/c
$cd c
mkdir d1 d2
```

```
$cd demodir
$cp /etc/group x
$cat x
root:x:0:root
bin:x:1:root,bin,daemon
disk:x:6:roo
...
$cp x copy.of.x
$rm copy.of.x
```

## Internal structure of the Unix file system

- Abstraction Zero: From platters to partitions, usually partions stop at the boundary of cylinders/tracks.
- Abstraction One: From partition to array of blocks.

  Index the sector for each partition in some order by a numbering system.

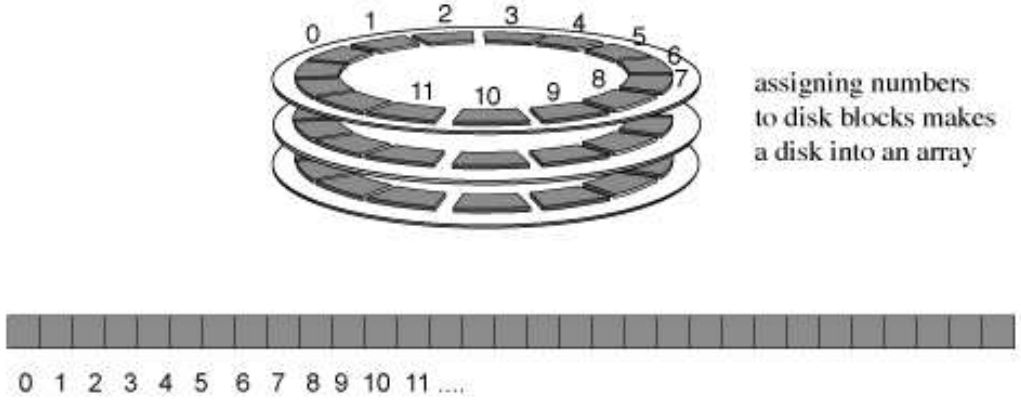



Figure 2: A tree of directories

- Abstraction Two: From an array of blocks to three regions:

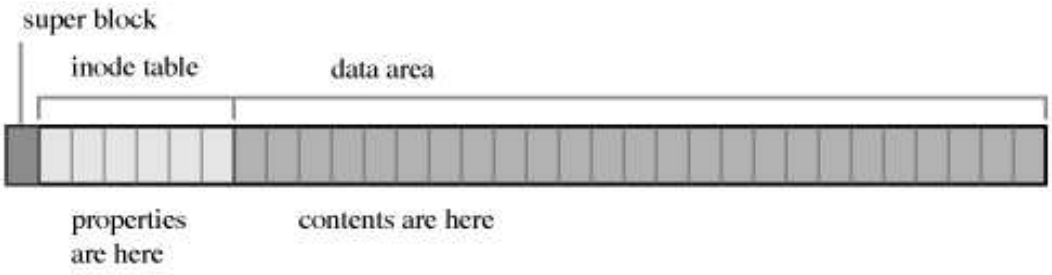



Figure 3: A tree of directories

- the **superblock**: the first block in the file system. It contains information about the organization of the file system itself.
- the **inode table**:inodes store the information of files such as size, user ID of the owner, time of last modification.
- the **data area**: the content of a file is in this area. Some files may take more than one block.

## Internal structure of the Unix file system

There are four steps in creating a file:

- Store properties: The kernel finds an available inode and write some file properties to the inode.
- Store data: The kernel finds a number of unoccupied/free blocks to save the content of the file.
- Record allocation: write the sequence of blocks taken by the file into the disk allocation section of the inode.
- Add filename into directory: The kernel adds an entry (inode number, filename) to some directory.

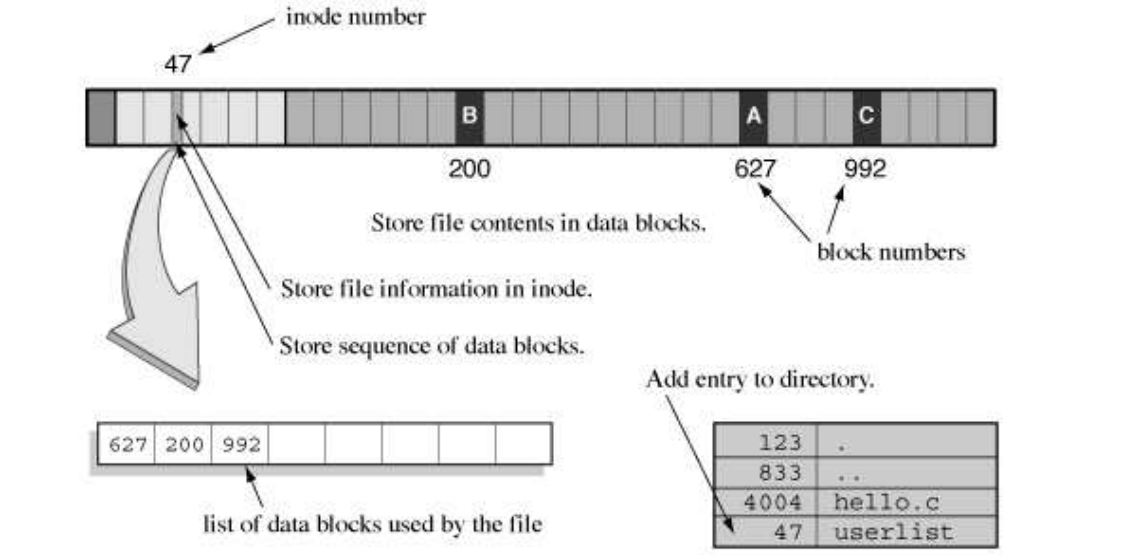



Figure 4: Internal structure of a file

## Internal structure of the Unix file system

A directory is a special kind of files that contains a list of names of files and their inode numbers.

```
[wch@localhost temp]$ ls -ia
1210078 .   1210797 glut-3.6          1211631 svl-1.5
  32705 ..  3679212 glut-3.6.tar.gz   3679211 svl-1.5.tar.gz
```

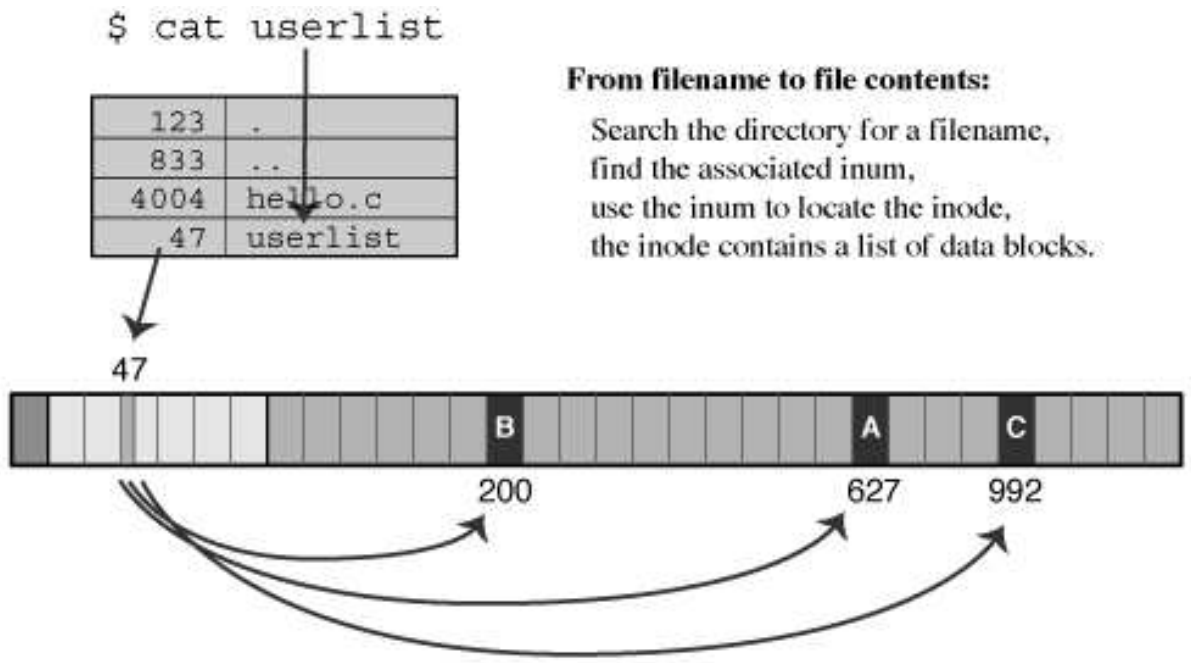
Mulitple files may point to the same inode.

## How cat works

$cat userlist

There are several steps in executing the above command

- Search in the directory for the filename and get the inode number
- Locate and read inode in the inode table of the file system.
- Read the data blocks




**From filename to file contents:**

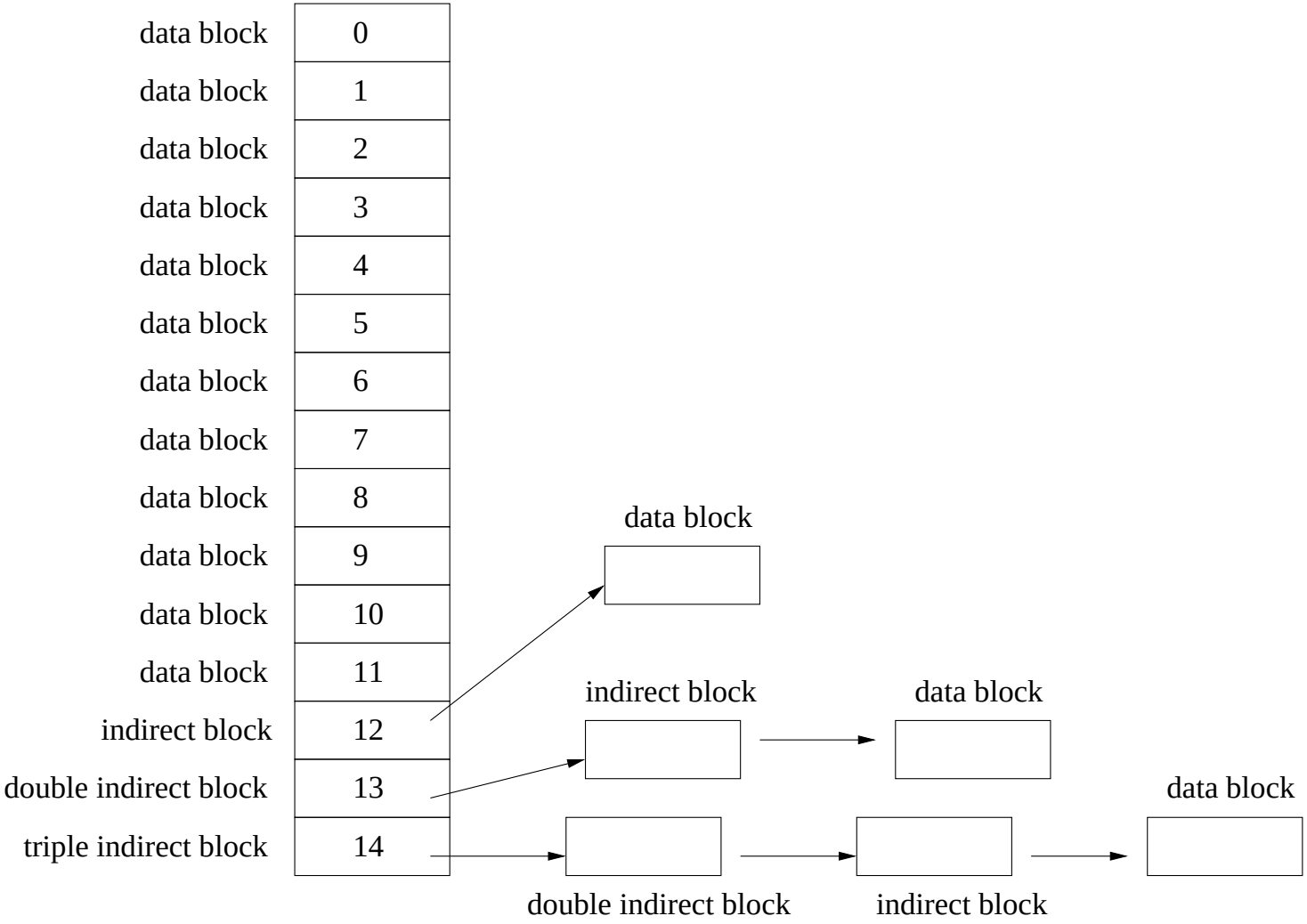
Search the directory for a filename,
find the associated inum,
use the inum to locate the inode,
the inode contains a list of data blocks.

## Structure of i-node

i-node in Unix file systems contains the following information

- owners
- timestamps
- size
- count:
- pointers to data blocks

# block allocation of the Unix file sytem

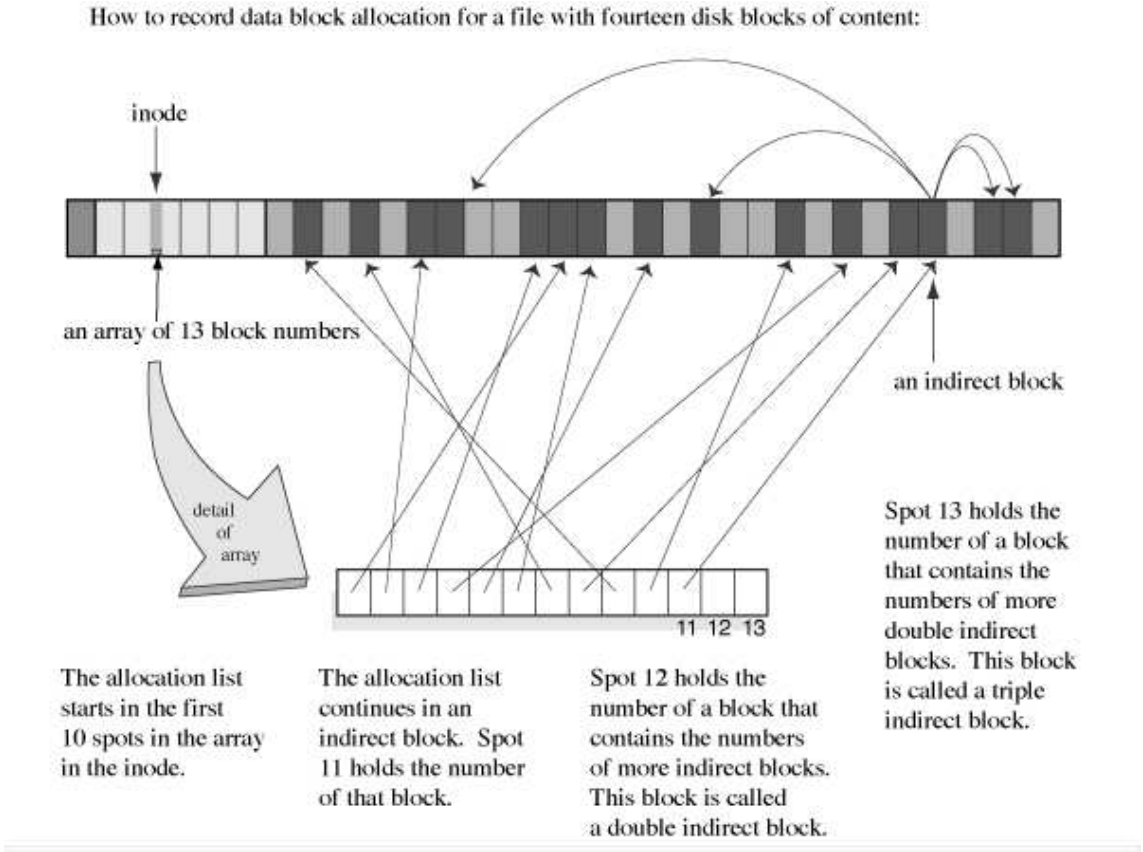


## block allocation of the Unix file sytem

If the block number is 4 bytes long, and each block is 4k bytes.

- datablock
  12*4k = 48k
- Single Indirect Blocks
  1k x 4kb = 4MB total space
- Double Indirect Blocks
  1 x 1k x 1k x 4kb = 4GB total space
- Triple Indirect Blocks
  1 x 1k x 1k x 1k x 4kb = 4TB of total storage space

# block allocation of the Unix file sytem

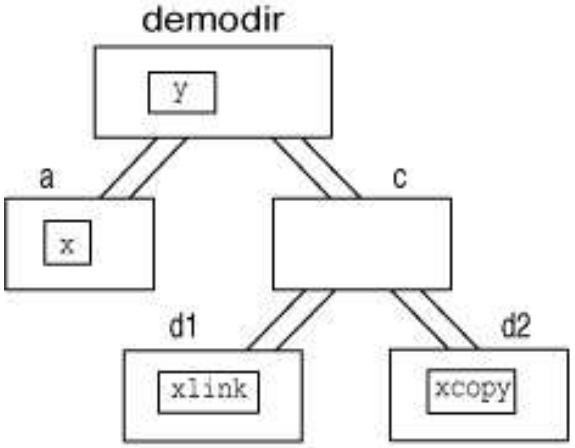



To list the inode numbers for all files under a directory, use

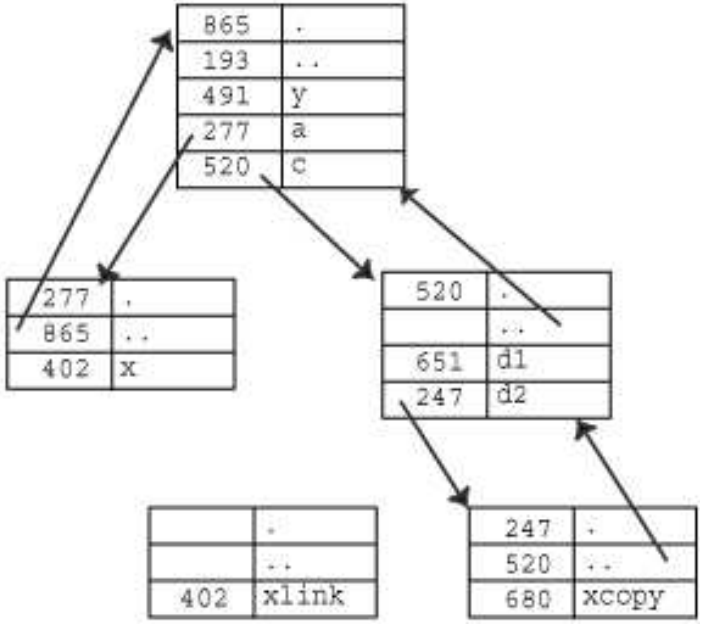
```
ls -iaR directoryName
```

# block allocation of the Unix file sytem

user view




system view



## System calls for directories

mkdir, rmdir, unlink, link, rename,chdir

```
$man 2 mkdir

MKDIR(2)                    Linux Programmers Manual                    MKDIR(2)

NAME
       mkdir - create a directory

SYNOPSIS
       #include <sys/stat.h>
       #include <sys/types.h>

       int mkdir(const char *pathname, mode_t mode);
RETURN VALUE
       mkdir returns zero on success, or -1 if an error occurred (in which case,
       errno  is  set appropriately).
```

# System calls for directories

```
$man 2 rmdir

RMDIR(2)                    Linux Programmers Manual                    RMDIR(2)

NAME
       rmdir - delete a directory

SYNOPSIS
       #include <unistd.h>

       int rmdir(const char *pathname);

RETURN VALUE
       On  success, zero is returned.  On error, -1 is returned, and
       errno is set appropriately.
```

# System calls for directories

```
$man 2 unlink

UNLINK(2)                  Linux Programmers Manual                  UNLINK(2)

NAME
       unlink - delete a name and possibly the file it refers to

SYNOPSIS
       #include <unistd.h>

       int unlink(const char *pathname);
RETURN VALUE
       On  success, zero is returned.  On error, -1 is returned, and errno
       is set appropriately.
```

# System calls for directories

```
$man 2 link

LINK(2)                    Linux Programmers Manual                    LINK(2)

NAME
       link - make a new name for a file

SYNOPSIS
       #include <unistd.h>

       int link(const char *oldpath, const char *newpath);
RETURN VALUE
       On  success, zero is returned.  On error, -1 is returned, and errno
       is set appropriately.
```

# System calls for directories

```
$man 2 rename

RENAME(2)                  Linux Programmers Manual                  RENAME(2)

NAME
       rename - change the name or location of a file

SYNOPSIS
       #include <stdio.h>

       int rename(const char *oldpath, const char *newpath);
```

# System calls for directories

```
$man 2 chdir

CHDIR(2)                    Linux Programmers Manual                    CHDIR(2)

NAME
       chdir, fchdir - change working directory

SYNOPSIS
       #include <unistd.h>

       int chdir(const char *path);
       int fchdir(int fd);
RETURN VALUE
       On success, zero is returned.  On error, -1 is returned, and errno
       is set  appropriately.
```

## How pwd works

- Step 1: Note the inode number for ".", call it $n$ (use stat).
- Step 2: chdir .. (use chdir).
- Step 3: Find the name of the link with inode $n$ (use opendir, readdir, closedir).
- Repeat step 2 and step 3 until you reach the top of the tree.

# How pwd works

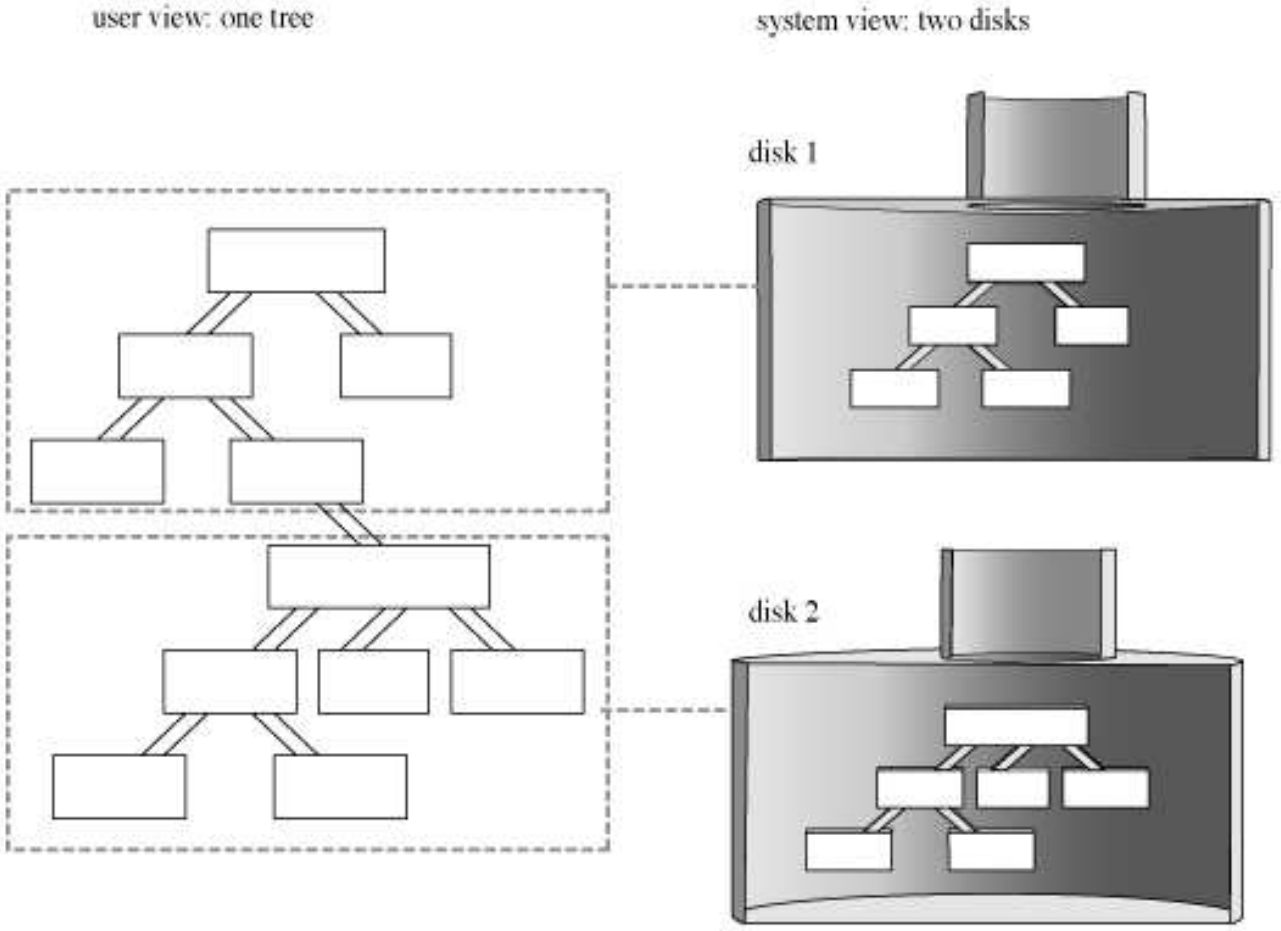
```
/* spwd.c: a simplified version of pwd
 *
 * starts in current directory and recursively
 * climbs up to root of filesystem, prints top part
 * then prints current part
 *
 * uses readdir() to get info about each thing
 *
 *      bug: prints an empty string if run from "/"
 **/
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>

ino_t get_inode(char *);
void    printpathto(ino_t);
void    inum_to_name(ino_t , char *, int );

int main()
{
        printpathto( get_inode( "." ) );        /* print path to here */
        putchar('\n');                          /* then add newline */
```

```
        return 0;
}

void printpathto( ino_t this_inode )
/*
 *      prints path leading down to an object with this inode
 *      kindof recursive
 */
{
        ino_t my_inode ;
        char its_name[BUFSIZ];

        if ( get_inode("..") != this_inode )
        {
                chdir( ".." );                          /* up one dir */

                inum_to_name(this_inode,its_name,BUFSIZ);/* get its name*/

                my_inode = get_inode( "." );            /* print head */
                printpathto( my_inode );                /* recursively */
                printf("/%s", its_name );               /* now print */
                                                        /* name of this */
        }
}
```

```
void inum_to_name(ino_t inode_to_find , char *namebuf, int buflen)
/*
 *      looks through current directory for a file with this inode
 *      number and copies its name into namebuf
 */
{
        DIR             *dir_ptr;               /* the directory */
        struct dirent *direntp;                /* each entry  */

        dir_ptr = opendir( "." );
        if ( dir_ptr == NULL ){
                perror( "." );
                exit(1);
        }

        /*
         * search directory for a file with specified inum
         */

        while ( ( direntp = readdir( dir_ptr ) ) != NULL )
                if ( direntp->d_ino == inode_to_find )
                {
                        strncpy( namebuf, direntp->d_name, buflen);
                        namebuf[buflen-1] = '\0';   /* just in case */
```

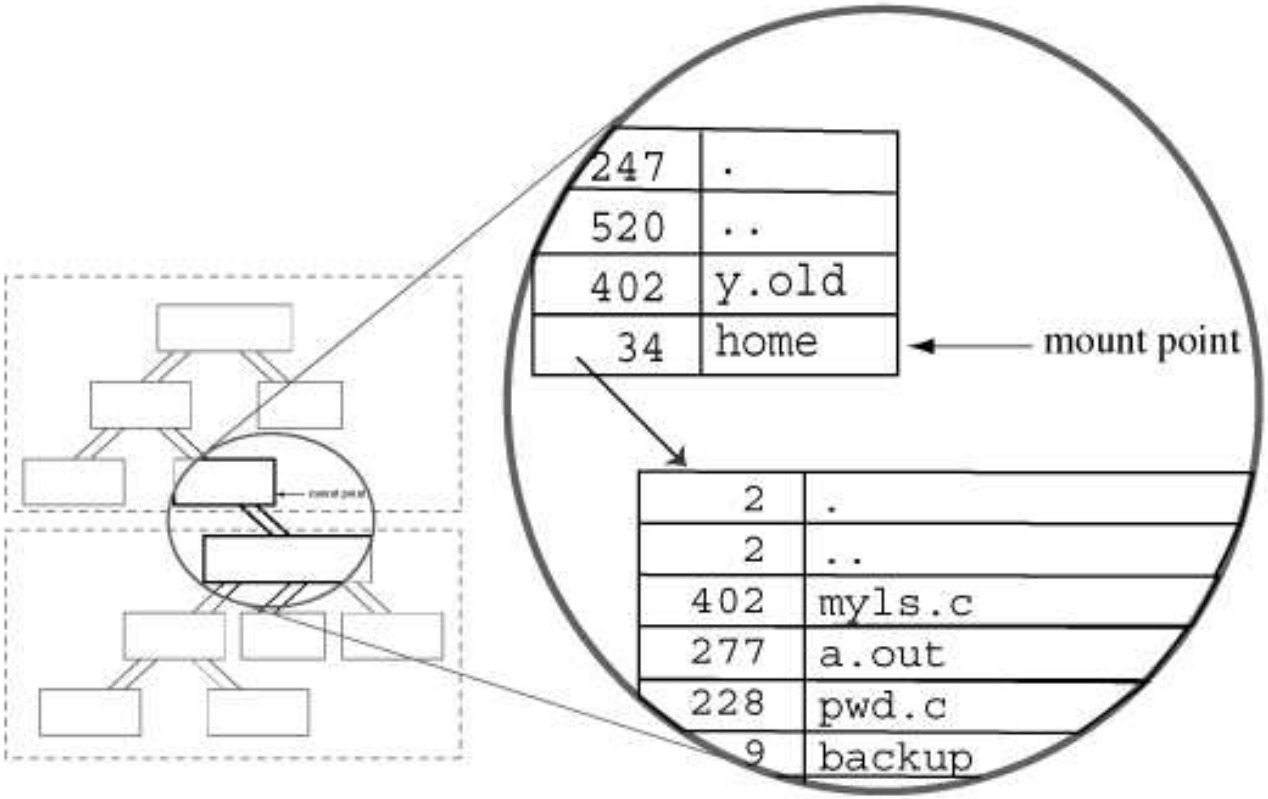
```
                        closedir( dir_ptr );
                        return;
                }
        fprintf(stderr, "error looking for inum %d\n", inode_to_find);
        exit(1);
}

ino_t get_inode( char *fname )
/*
 *      returns inode number of the file
 */
{
        struct stat info;

        if ( stat( fname , &info ) == -1 ){
                fprintf(stderr, "Cannot stat ");
                perror(fname);
                exit(1);
        }
        return info.st_ino;
}
```

# Multiple file systems: A tree of tree



# Mount a one file system to a mount point of another file system

$ mount options devicename mountpoint



## Hard link and symbolic link

- Due to the fact that files in different file systems may have the same inode number, therefore hard link can only point to inodes in other file systems.
- symbolic link refers a file by name, not by inode number, therefore can point to a file in a different file system.

```
[wch@localhost chinese_story]$ ln /home/wch/software/jgrasp183.zip  jgrap183.zip
ln: creating hard link ‘jgrap183.zip’ to ‘/home/wch/software/jgrasp183.zip’:
    Invalid cross-device link

[wch@localhost software]$ ls -l
total 5260
drwxr-xr-x   9 root root     4096 Nov 10 17:19 jdk1.5.0_06
drwxr-xr-x  10 root root     4096 Aug 17  2005 jgrasp
-rw-rw-r--   2 wch  wch   2669095 Jan  5 20:52 jgrasp183_link.zip
lrwxrwxrwx   1 wch  users      13 Mar 13 11:45 jgrasp183_slink.zip -> jgrasp183.zip
-rw-rw-r--   2 wch  wch   2669095 Jan  5 20:52 jgrasp183.zip
drwxrwxr-x  16 wch  wch      4096 Oct 21 21:28 matlab
```