UTSA The University of Texas at San Antonio™

EE3233 Systems Programming for Engrs

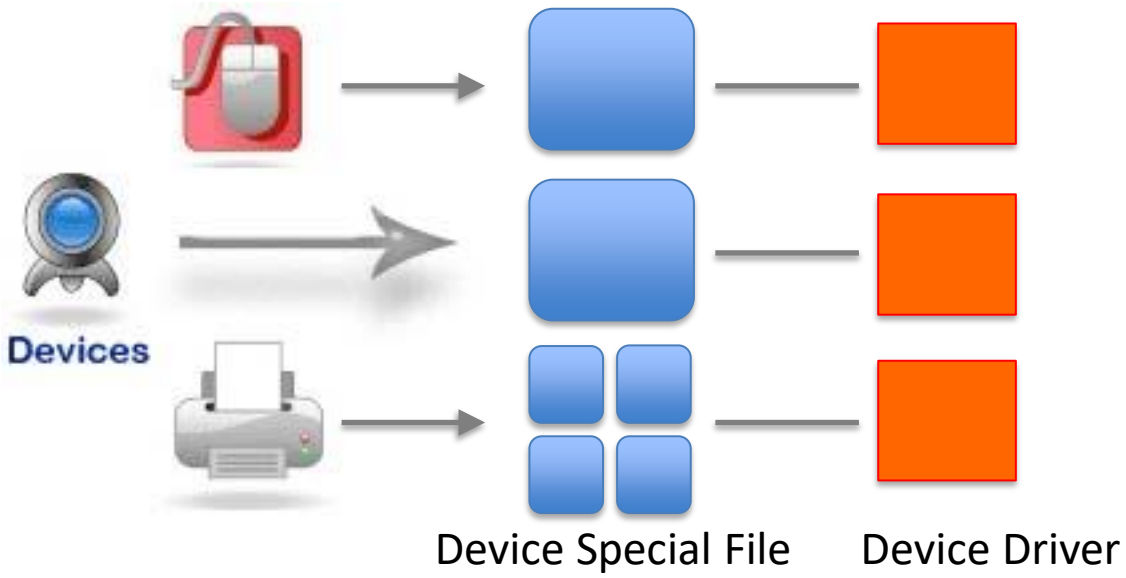
Reference: M. Kerrisk, The Linux Programming Interface

Lecture 8
File Systems

ECE Electrical & Computer Engineering

Device Special Files

- corresponds to a device on the system
- Within the kernel each device type has a corresponding device driver
- Device driver handles all I/O requests for the device



Device Types

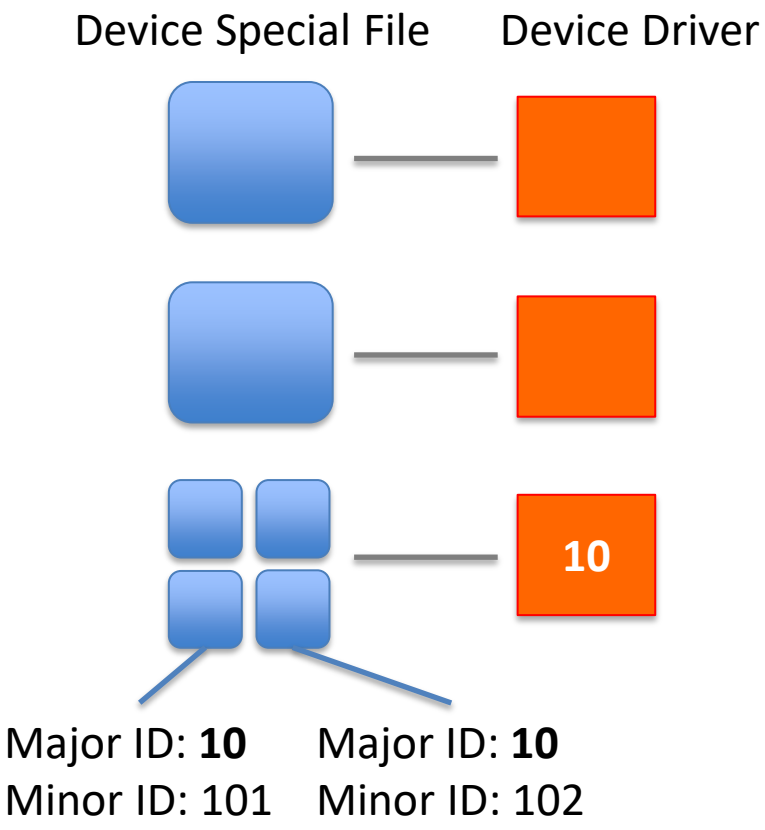
- Character devices
  - handle data on a character-by-character basis
  - Terminals and keyboards
- Block devices
  - handle data on a block at a time
  - The size of a block depends on the type of device
  - typically multiples of 512 bytes
  - Disks are a common example

/dev

- Device files appear within the file system, just like other files, usually under /dev

Device IDs

- Each device file has a major ID number and a minor ID number
- Major ID
  - identifies the general class of device
  - used by kernel to look up the appropriate driver for this type of device
- Minor ID
  - uniquely identifies a particular device within a general class
- Each device driver registers its association with a specific major device ID
  - results in connecting to device file



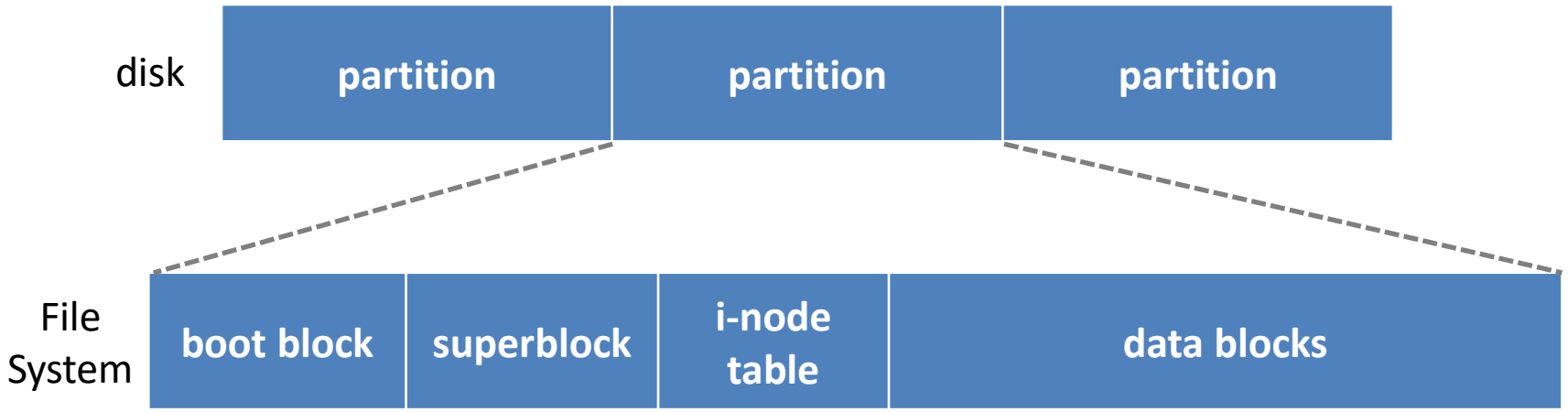
File Systems

- organized collection of regular files and directories
- is created using the *mkfs* command

File-system structure

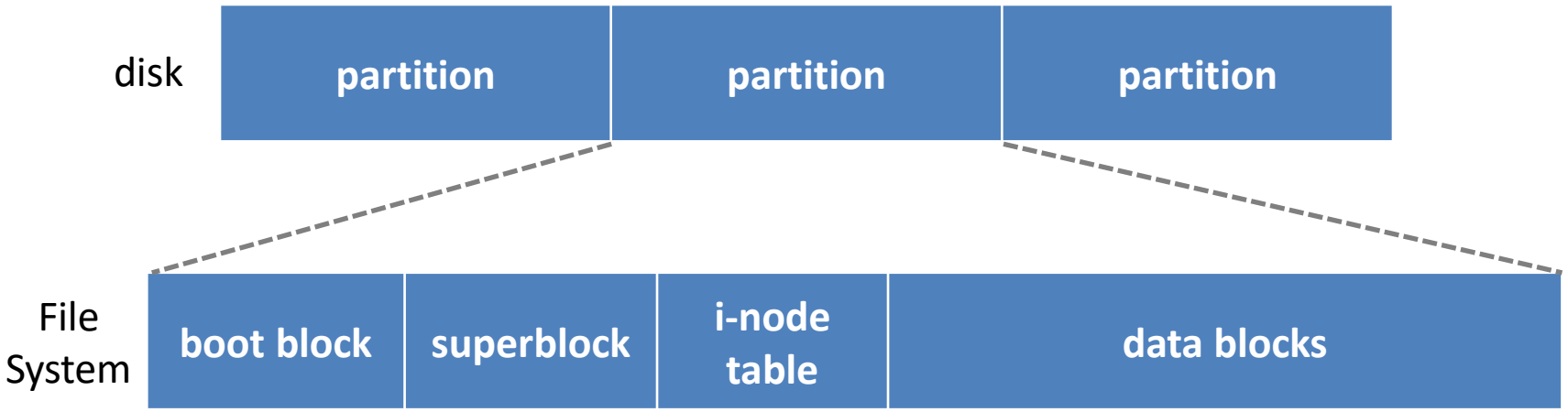
- Basic unit of allocating space is a logical block
  - multiple of contiguous physical blocks
  - the logical block size on *ext2* is 1024, 2048, or 4096 bytes
- logical block size is specified as an argument of the *mkfs*

Layout of disk partitions and a file system



- boot block
  - contains information used to boot the OS
  - All file systems have a boot block
- superblock: contains parameter information about the file system
  - Single block
  - size of the i-node table
  - size of logical blocks in this file system
  - size of the file system in logical blocks
- Different file systems residing on the same physical device can be different types and sizes, and have different parameter settings

Layout of disk partitions and a file system



- i-node table (*i-list*)
  - Each file or directory in the file system has a unique entry in the i-node table
  - this entry records various information about the file
- data blocks
  - forms files and directories

I-nodes

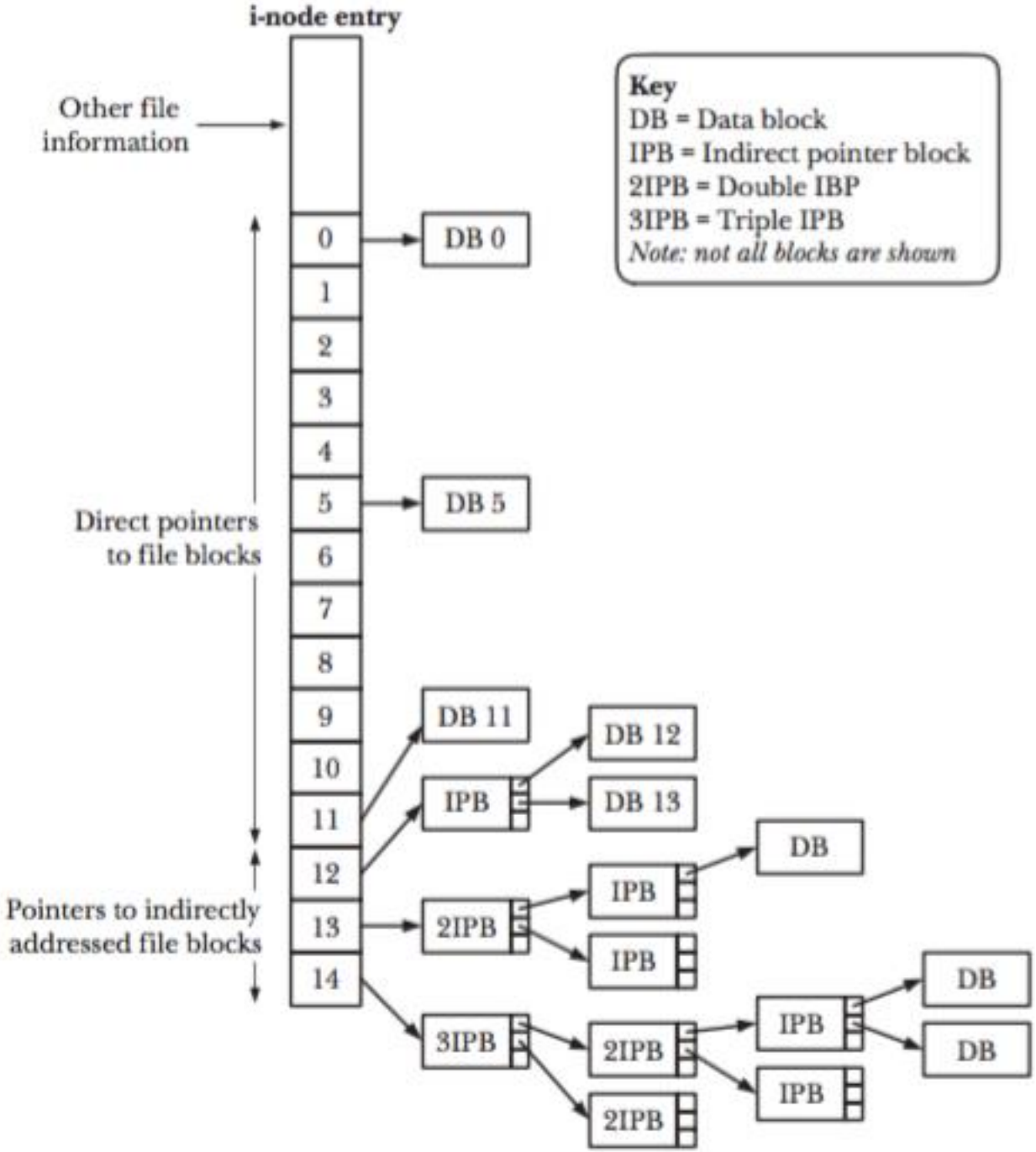
- i-node table contains one i-node (index node) for each file residing in the file system
  - identified numerically by their sequential location
  - i-node number of a file is the first field displayed by the “ls –li”

```
$ ls -li
```

```
 6999136 drwxr-xr-x   21 wonjunlee  staff     714 Aug 24 11:36 temp
11162855 -rw-r--r--    1 wonjunlee  staff     982 Aug 25 17:52 temp.txt
 6615224 -rwxr-xr-x    1 wonjunlee  staff   45104 Jan 30  2016 test
```

I-nodes

- The information maintained in an i-node
  - File type (regular file, directory, symbolic link, char dev.)
  - Owner for the file
  - Group for the file
  - Access permissions for three categories of user
  - Three timestamp: last access, last modification, last status change
  - Number of hard links to the file
  - Size of file in bytes
  - Number of blocks actually allocated to the file
  - Pointers to the data blocks of the file


i-node entry
Other file information
Key
DB = Data block
IPB = Indirect pointer block
2IPB = Double IBP
3IPB = Triple IPB
Note: not all blocks are shown
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
DB 0
DB 5
DB 11
DB 12
IPB
DB 13
DB
IPB
2IPB
IPB
DB
IPB
3IPB
2IPB
DB
IPB
2IPB
Direct pointers to file blocks
Pointers to indirectly addressed file blocks

data block pointers in *ext2*

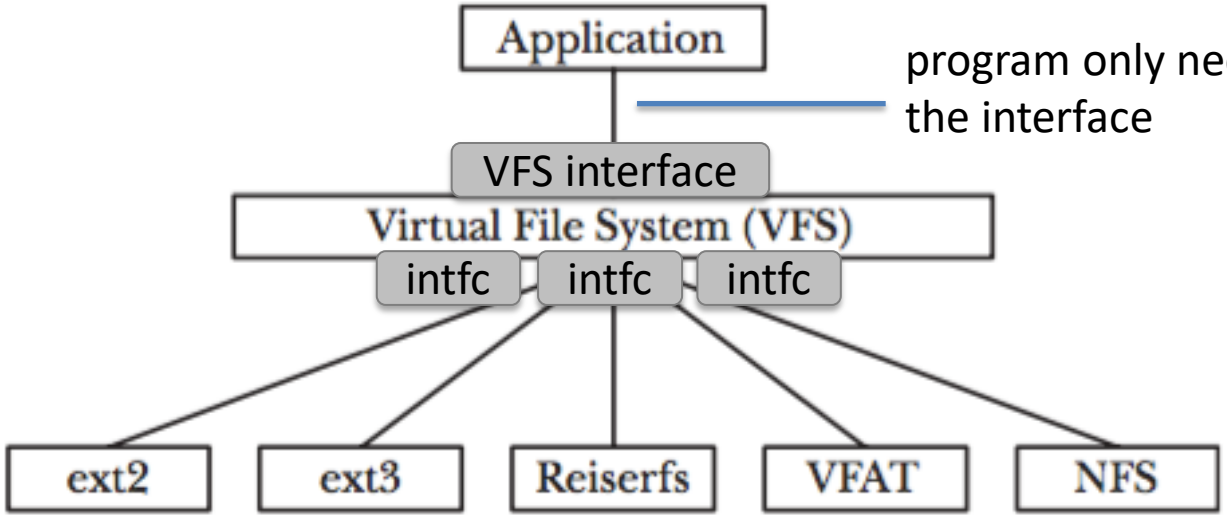
- *ext2* doesn't store data blocks of a file contiguously or even in sequential order
  - To locate the file data blocks, the kernel maintains a set of pointers in the i-node
- Under *ext2*, each i-node contains 15 pointers
  - 0 ~ 11 points to the location in the file system of the first 12 blocks
  - The next pointer is a pointer to a block of pointers that give the locations of the 13^{th} and subsequent data blocks of the file
  - Element 12: If each pointer requires 4 bytes, and one block size is 1024B → 256 (1024/4) pointers are possible : allows quite large files (256*1024B)
  - Element 13: For even larger files, use double indirect pointer
  - Element 14: For a block size, 4096 bytes, largest file size is theoretically 1024*1024*1024*4096 bytes = 4096 GB (4 TB)

Virtual File System (VFS)

- Each of the file systems in Linux differs
  - in the way that blocks of a file are allocated
  - in the manner that directories are organized
- Every program needs to understand specific details of each file system?
  - task of writing program to work with different file systems is nearly impossible
- VFS is a kernel feature resolving this problem
  - it creates abstraction layer for file-system operation

Virtual File System (VFS)

- Idea behind the VFS
  - VFS defines a generic interface for file-system operations
  - Each file system provides an implementation for VFS interface



VFS Interface

- includes operations corresponding to all of the usual system calls for file system
  - open(), read(), write(), lseek(), close(), mount(), link(), mkdir(), rename(), etc.
- Some file systems don't support all of the VFS operations
  - MS VFAT does not support *symlink()*
  - In such case, underlying file system passes an error back to VFS, and VFS passes this error to application

File inconsistency

- After a system crash, a file-system consistency check must be performed on reboot for integrity
  - a file update may have been only partially completed
  - file system metadata (directory entries, *i-node* information, file data block pointers) may be in an inconsistency state

  ➔ file system may be damaged if these consistencies are not repaired
- Thus requires examining entire file system
  - where possible, repairs are performed
  - it takes much time for large file system

  : a few second for small, an hour for large system

File inconsistency

- For example, deleting a file on a Unix file system involves three steps
  - step 1. removing its directory entry
  - step 2. releasing the i-node to the pool of free i-nodes
  - step 3. returning all used disk blocks to the pool of free blocks
- If a crash occurs after step 1 and before step 2, there will be an orphaned i-node leaking storage
- If only step 2 is performed before the crash, the not-yet-deleted file will be marked free

Journaling File Systems

- Journaling file systems eliminate the need for lengthy consistency check after crash
  - it logs all metadata updates to a special on-disk journal file before they are actually carried out
  - after crash, the log can be used to rapidly redo any incomplete updates and bring the file system back to a consistent state on reboot

Single Directory Hierarchy and Mount point

- On Linux, all files from all file systems reside under a single directory tree (based of /)
- Other file systems are *mounted* under the root directory and appear as subtrees
- Superuser uses a command to mount

```
$ mount  device  directory
```

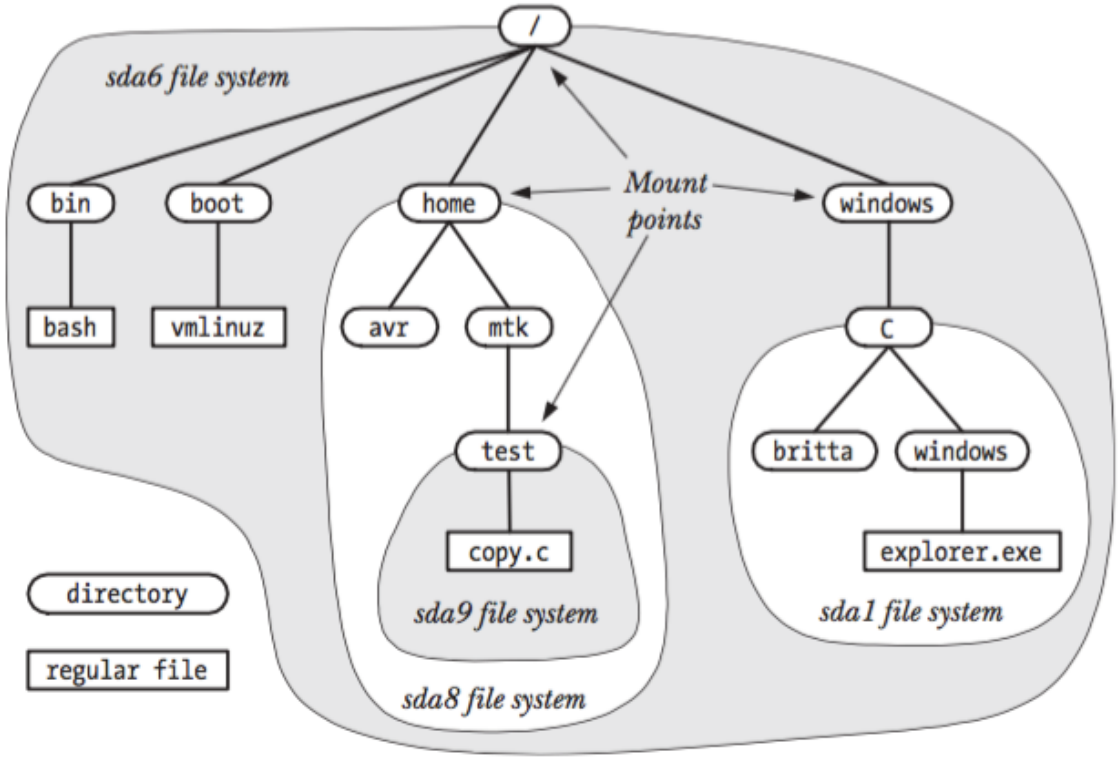
: attaches the file system (device) into the directory hierarchy (directory)

Single Directory Hierarchy and Mount point

- With Linux 2.4.19 and later, kernel supports per-process *mount namespaces*
  - each process potentially has its own set of file-system mount points
  - thus, may see a different single directory hierarchy from other processes

Single Directory Hierarchy and Mount point

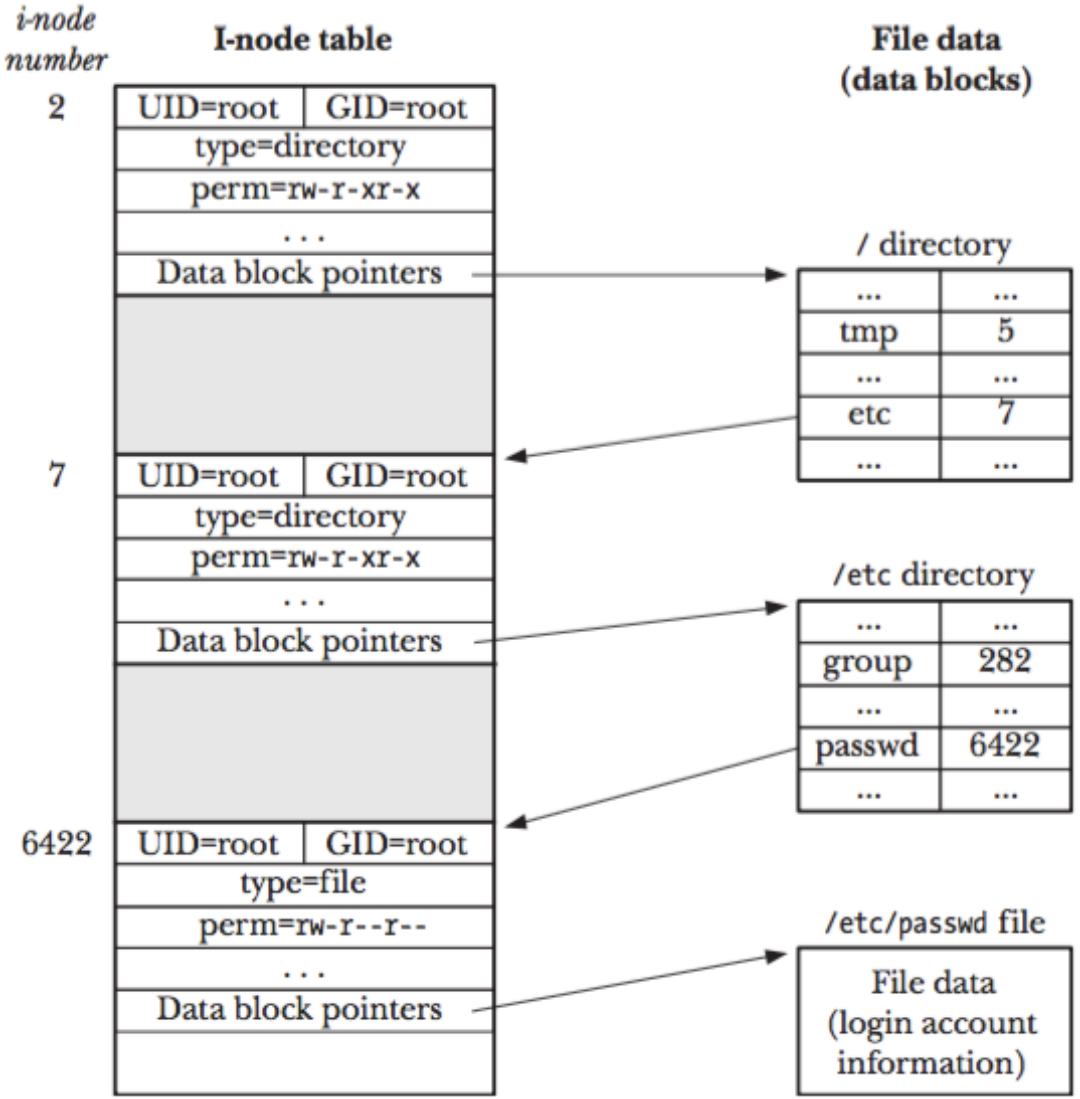
```
$ mount
/dev/sda6 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
/dev/sda8 on /home type ext3 (rw,acl,user_xattr)
/dev/sda1 on /windows/C type vfat (rw,noexec,nosuid,nodev)
/dev/sda9 on /home/mtk/test type reiserfs (rw)
```



Directories and Links

- A directory is stored in the file system in a similar way to a regular file
- Two things are different
  - marked with a different file type in its i-node entry
  - a file with a special organization (a table consisting of filenames and i-node numbers)

Relationship between i-node and directory structures for the file /etc/passwd



Hard Links

- known as multiple names in the same or in different directories, each of which refers to the same *i-node*

```
$ echo  'Hello ee 3233 students ~' >  org.txt
$ ls –li org.txt
  1195694 –rw-rw-r– 1 user1 user1 24 Oct 12 08:00 org.txt
$ ln  org.txt  dest.txt
$ echo ' Good afternoon!'  >>  dest.txt
$ cat org.txt
   Hello ee 3233 students ~ Good afternoon!
$ ls –li org.txt  dest.txt
  1195694 –rw-rw-r– 2 user1 user1 37 Oct 12 08:00 org.txt
  1195694 –rw-rw-r– 2 user1 user1 37 Oct 12 08:00 dest.txt
```

org.txt and dest.txt refer to the same *i-node* entry, and hence to the same file

Hard Links

- If one of these filenames is removed, the other name, and the file itself, continue to exist

```
$ rm  org.txt
$ ls –li dest.txt
  1195694 –rw-rw-r– 1 user1 user1 37 Oct 12 08:00 dest.txt
```

  - *i-node* count becomes 1 from 2
  - the physical file continued to exist
- *i-node* count 0 : *i-node* entry and data blocks for the file are removed (when all of the names have been removed)

Hard Links

- All of the names (links) for a file are equivalent
  - none of the names has priority over any of the others
- Two limitations
  - a hard link must reside on the same file system as the file to which it refers : *i-node* numbers are unique only within a file system
  - A hard link can't be made to a directory: this prevents the creation of circular links, which confuse many system programs

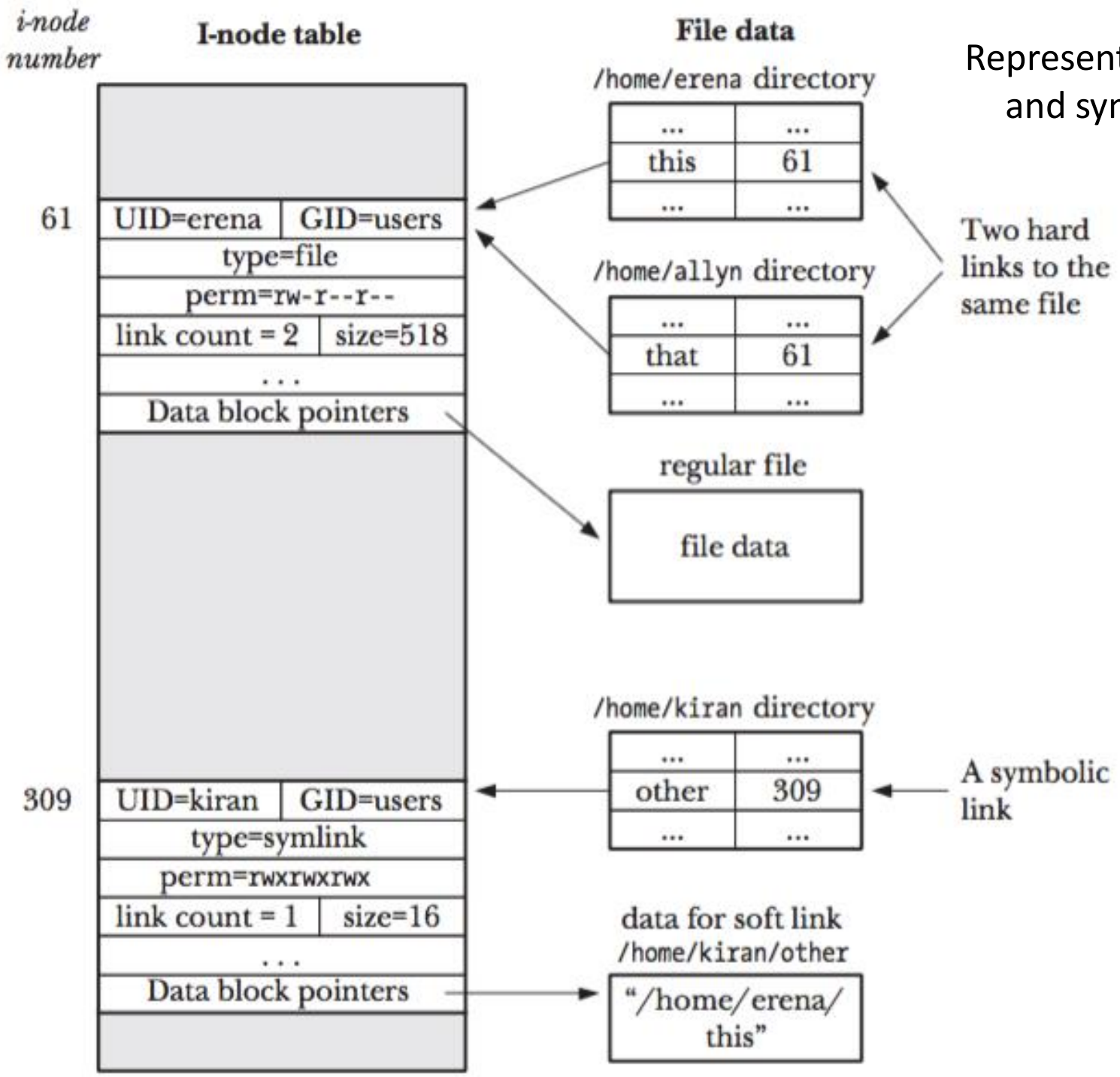
Symbolic (Soft) Links

- Symbolic links are created using the ***ln –s*** command

```
$ ln –s org.txt dest.txt
  1195694 –rw-rw-r--   1 user1 user1 23 Oct 12 08:00 org.txt
  1195700 lrwxrwxrwx 1 user1 user1 23 Oct 12 08:00 dest.txt -> org.txt
```

- A symbolic link is not included in the link count of file
- A symbolic link refers to a filename, rather than an *i-node* number
  - it can be used to link to a file in a different file system

Representation of hard and symbolic links



Symbolic (Soft) Links

- If the filename to which the symbolic link refers is removed, the symbolic link itself continues to exist (dangling link)

```
$ rm org.txt
$ ls –li dest.txt
  1195700 lrwxrwxrwx 1 user1 user1 23 Oct 12 08:00 dest.txt -> org.txt
$ cat dest.txt
  cat: dest.txt: No such file or directory
```

Symbolic (Soft) Links

- Since a symbolic link refers to a filename, rather than an i-node number, it can be used to link to a file in a different file system
- can also create symbolic links to directories
  - don’t follow symbolic links by default, or avoid getting trapped in circular references created using symbolic links