The User Interface

- Files, directories, file descriptors, file systems
- File & Directories:
  - File: logically a container for data
  - Pathname: components in the path from the root to the node, by “/”
  - Special entries: “.” & “..”
  - Link: a directory entry for a file.

- A hierarchical, tree-structured name space

Giorgio Richelli
giorgio_richelli@it.ibm.com

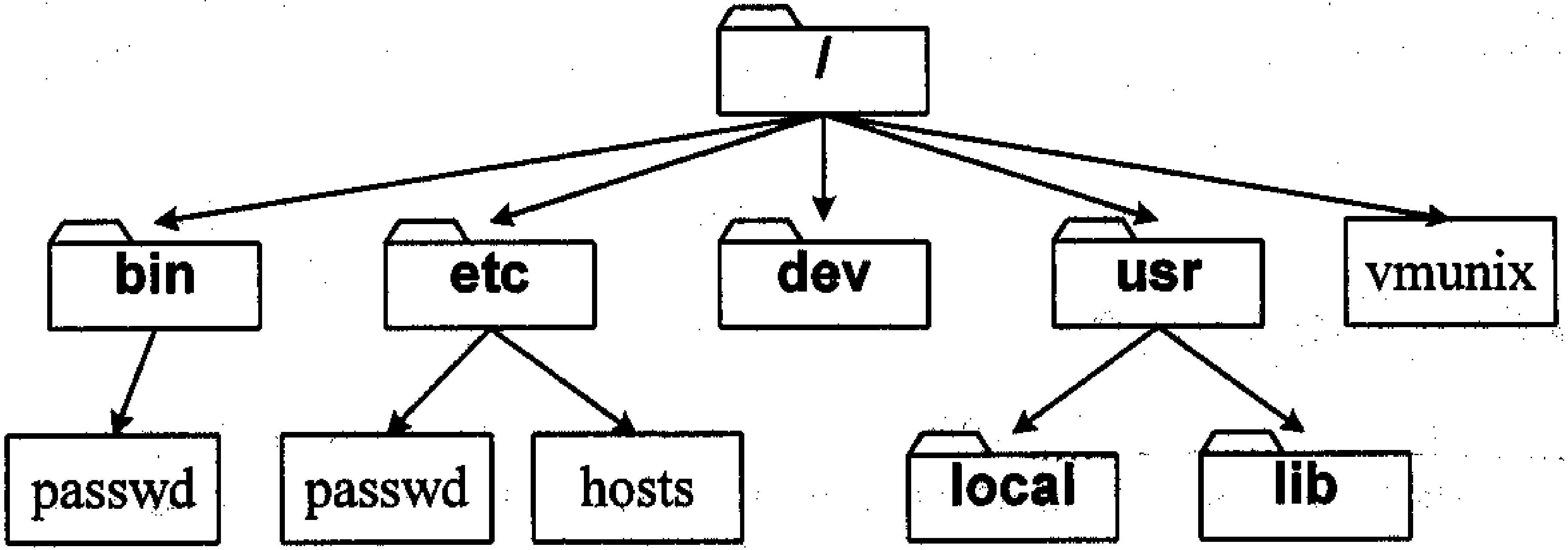


Figure 8-1. Files are organized in a directory tree.

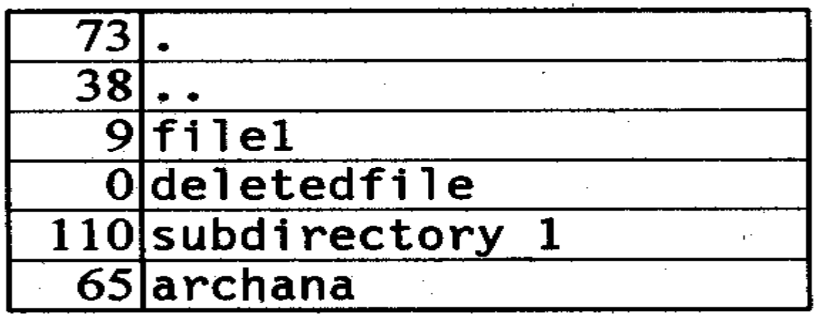
Giorgio Richelli
giorgio_richelli@it.ibm.com

Archaeology: System V File System(s5fs)

The layout of s5fs partition:

B	S	inode list	data blocks

Directory: a special file containing a list of files and subdirectories.



73	.
38	..
9	file1
0	deletedfile
110	subdirectory 1
65	archana

Figure 9-2. ***s5fs* directory structure.**

Giorgio Richelli
giorgio_richelli@it.ibm.com

Inodes

- The inode contains administrative information,or meta data.
- The inode list contains all the inodes
- An inode can have two copies:
    - On-disk
    - In-core

Giorgio Richelli
giorgio_richelli@it.ibm.com

Inode Fields

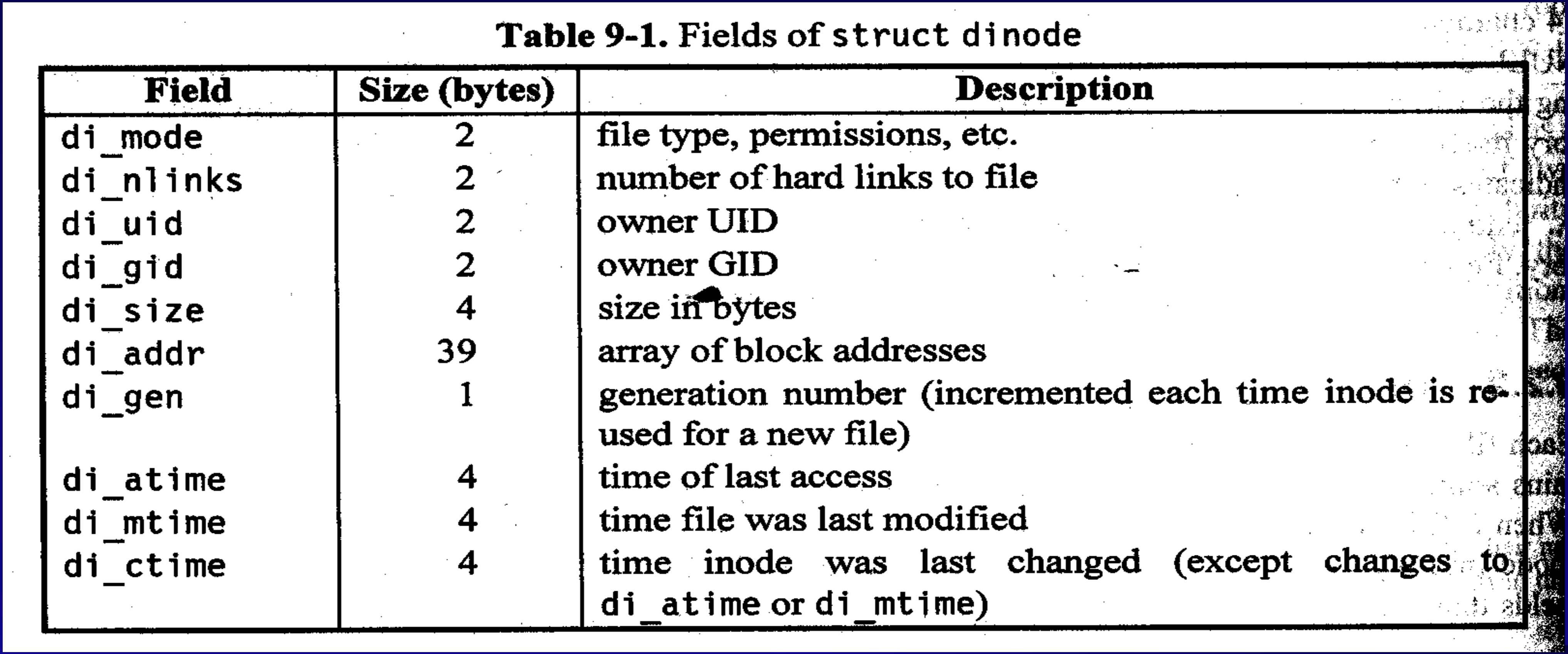
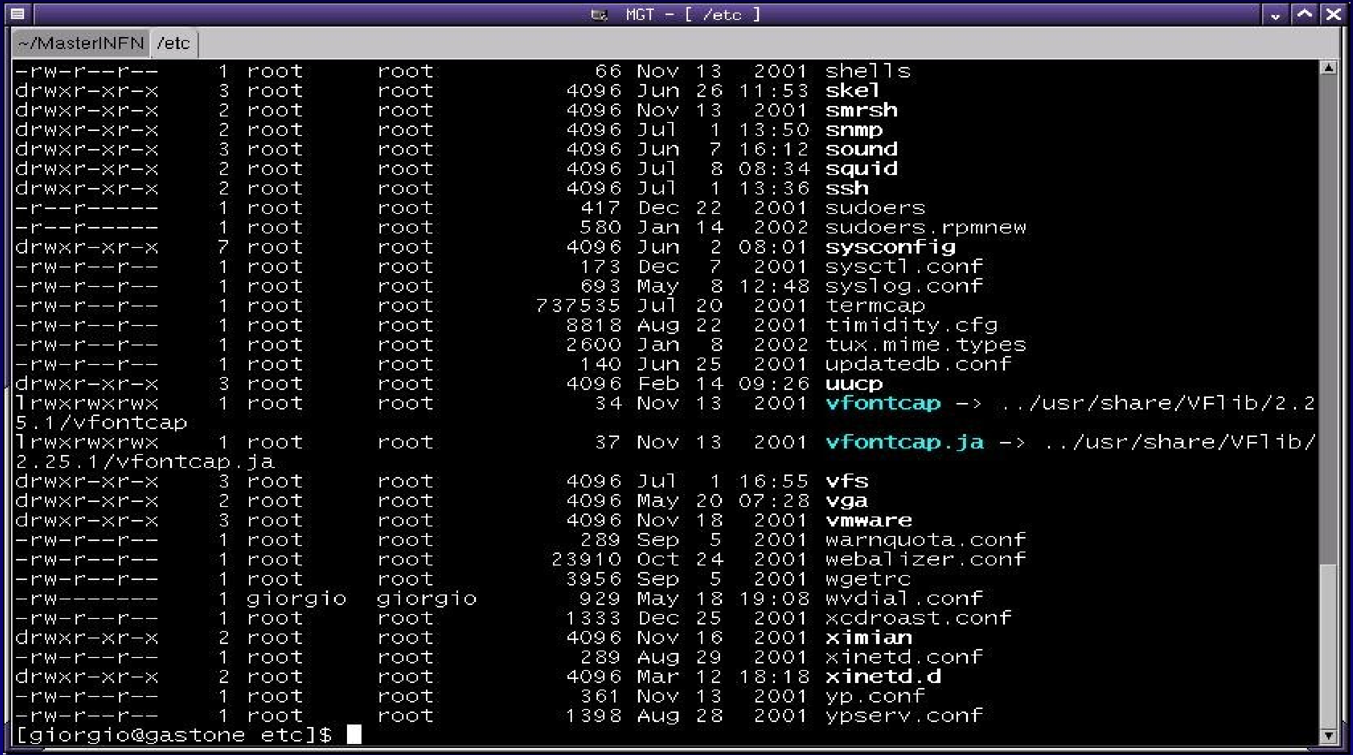


Table 9-1. Fields of struct dinode

Field	Size (bytes)	Description
di_mode	2	file type, permissions, etc.
di_nlinks	2	number of hard links to file
di_uid	2	owner UID
di_gid	2	owner GID
di_size	4	size in bytes
di_addr	39	array of block addresses
di_gen	1	generation number (incremented each time inode is re-used for a new file)
di_atime	4	time of last access
di_mtime	4	time file was last modified
di_ctime	4	time inode was last changed (except changes to di_atime or di_mtime)

Giorgio Richelli
giorgio_richelli@it.ibm.com

di_mode



```
-rw-r--r--    1 root     root          66 Nov 13  2001 shells
drwxr-xr-x    3 root     root        4096 Jun 26 11:53 skel
drwxr-xr-x    2 root     root        4096 Nov 13  2001 smrsh
drwxr-xr-x    2 root     root        4096 Jul  1 13:50 snmp
drwxr-xr-x    3 root     root        4096 Jun  7 16:12 sound
drwxr-xr-x    2 root     root        4096 Jul  8 08:34 squid
drwxr-xr-x    2 root     root        4096 Jul  1 13:36 ssh
-r--r-----    1 root     root         417 Dec 22  2001 sudoers
-r--r-----    1 root     root         580 Jan 14  2002 sudoers.rpmnew
drwxr-xr-x    7 root     root        4096 Jun  2 08:01 sysconfig
-rw-r--r--    1 root     root         173 Dec  7  2001 sysctl.conf
-rw-r--r--    1 root     root         693 May  8 12:48 syslog.conf
-rw-r--r--    1 root     root      737535 Jul 20  2001 termcap
-rw-r--r--    1 root     root        8818 Aug 22  2001 timidity.cfg
-rw-r--r--    1 root     root        2600 Jan  8  2002 tux.mime.types
-rw-r--r--    1 root     root         140 Jun 25  2001 updatedb.conf
drwxr-xr-x    3 root     root        4096 Feb 14 09:26 uucp
lrwxrwxrwx    1 root     root          34 Nov 13  2001 vfontcap -> ../usr/share/VFlib/2.2
5.1/vfontcap
lrwxrwxrwx    1 root     root          37 Nov 13  2001 vfontcap.ja -> ../usr/share/VFlib/
2.25.1/vfontcap.ja
drwxr-xr-x    3 root     root        4096 Jul  1 16:55 vfs
drwxr-xr-x    2 root     root        4096 May 20 07:28 vga
drwxr-xr-x    3 root     root        4096 Nov 18  2001 vmware
-rw-r--r--    1 root     root         289 Sep  5  2001 warnquota.conf
-rw-r--r--    1 root     root       23910 Oct 24  2001 webalizer.conf
-rw-r--r--    1 root     root        3956 Sep  5  2001 wgetrc
-rw-------    1 giorgio  giorgio      929 May 18 19:08 wvdial.conf
-rw-r--r--    1 root     root        1333 Dec 25  2001 xcdroast.conf
drwxr-xr-x    2 root     root        4096 Nov 16  2001 ximian
-rw-r--r--    1 root     root         289 Aug 29  2001 xinetd.conf
drwxr-xr-x    2 root     root        4096 Mar 12 18:18 xinetd.d
-rw-r--r--    1 root     root         361 Nov 13  2001 yp.conf
-rw-r--r--    1 root     root        1398 Aug 28  2001 ypserv.conf
[giorgio@gastone etc]$ 
```

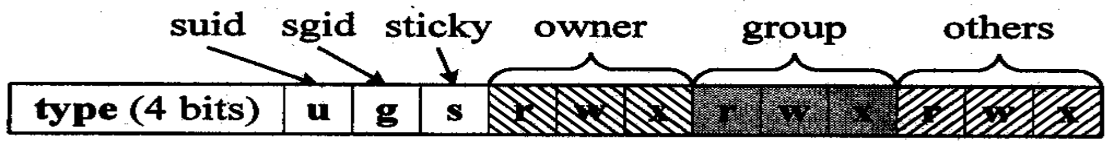


Figure 9-3. Bit-fields of di_mode.

Giorgio Richelli
giorgio_richelli@it.ibm.com

Disk Block Array: di_addr

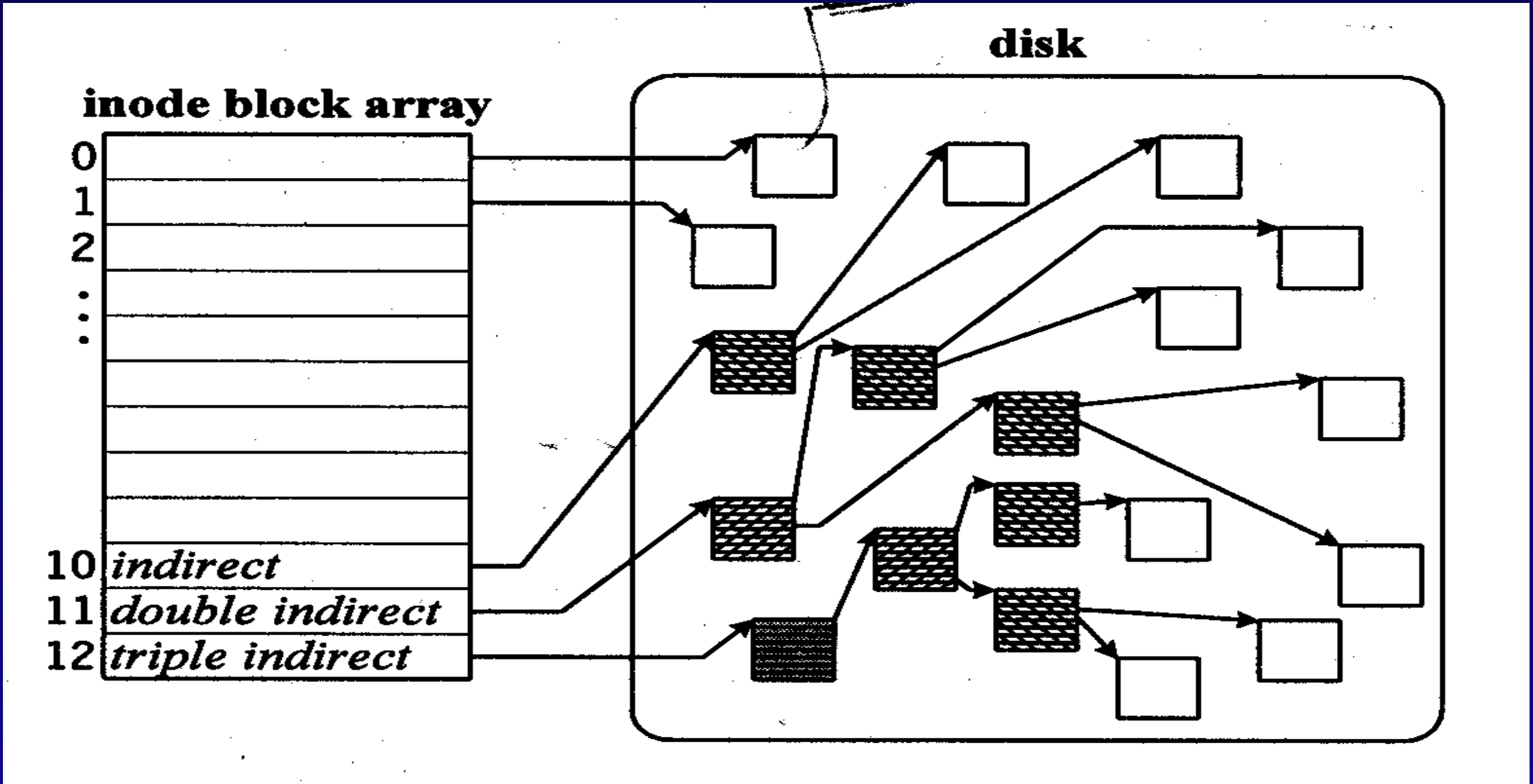


Figure 9-4. disk block array in *s5fs* inode.

rgio Richelli
giorgio_richelli@it.ibm.com

The superblock

- Size in blocks of the file system
- Size in blocks of the inode list
- Number of free blocks and inodes
- Free block list
- Free inode list

Giorgio Richelli
giorgio_richelli@it.ibm.com

Free block list

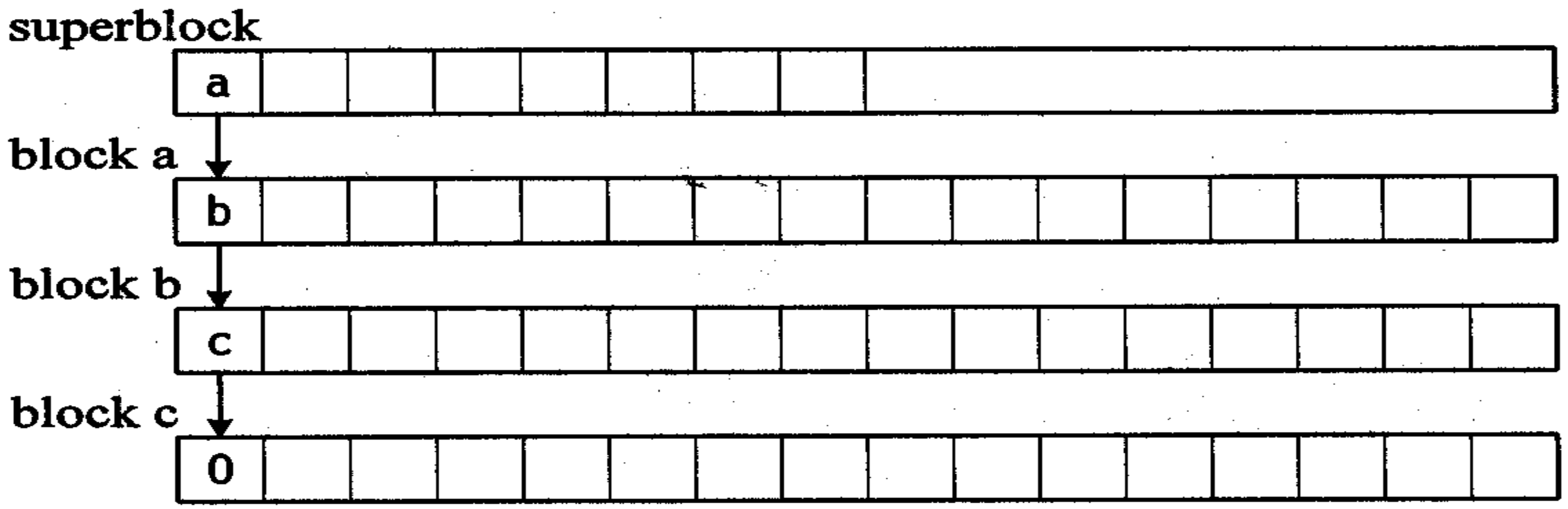


Figure 9-5. free block list in *s5fs*.

Giorgio Richelli
giorgio_richelli@it.ibm.com

Directory syscalls

```
dirp = opendir(char *filename);
direntp = readdir (dirp);
rewinddir(dirp);
status = closedir(firp);
struct dirent {
   int_t d_ino;
   char d_name[NAME_MAX +1];
}
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

File Information

```
int fstat(int fd, struct stat *buf);
```

- Returns information about an (open) file
- Parameters:
  - fd: file descriptor
  - buf: pointer to a struct stat
  - Returns
  - 0: if successful
  - -1: in case of errors
- Example:

```
sts=fstat(fd,&statbuf);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

File Information

```
int stat(const char *name, struct stat *buf);
```

- Returns information about an file
- Parameters:
  - name: path to file
  - buf: pointer to a struct stat
  - Returns
  - 0: if successful
  - -1: in case of errors
- Example:

```
sts=stat(“pippo.txt”,&statbuf);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

File Attributes

- Kept in the inode
- File attributes:
  - File type
  - Number of hard links
  - File size
  - Device ID
  - Inode number
  - User and Group Ids of the owner of the file.
  - Timestamps
  - Permissions and mode flags (suid, sgid, sticky)

File Descriptors

- ✔ An <handle> used to access a file or other I/O resources
- ✔ Usually a small integer, an entry in kernel resource tables
- ✔ It is a <**per-process**> object

✔
```
int open(const char *path,int oflag,mode_t mode);
fd=open("/path/to/file",O_RDWR,S_IRUSR|S_IWUSR);
```

✔
```
int creat(const char *pathname, mode_t mode);
fd=creat("./newfile",S_IRUSR|S_IWUSR|S_IRGRP);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

File Descriptors

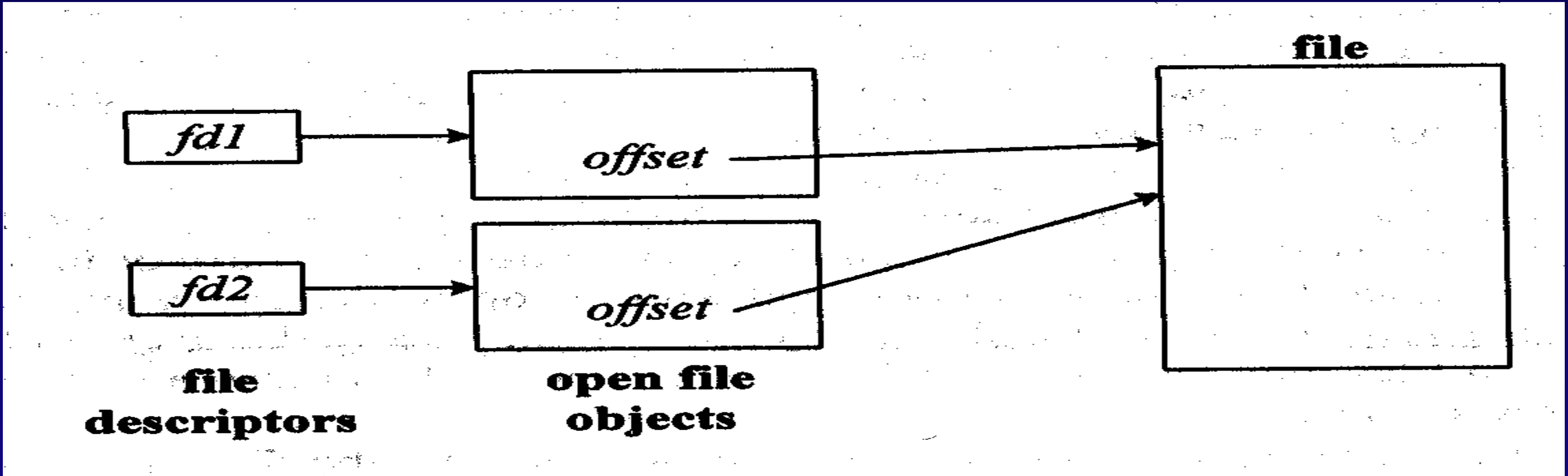


Figure 8-2. A file is opened twice.

Giorgio Richelli
giorgio_richelli@it.ibm.com

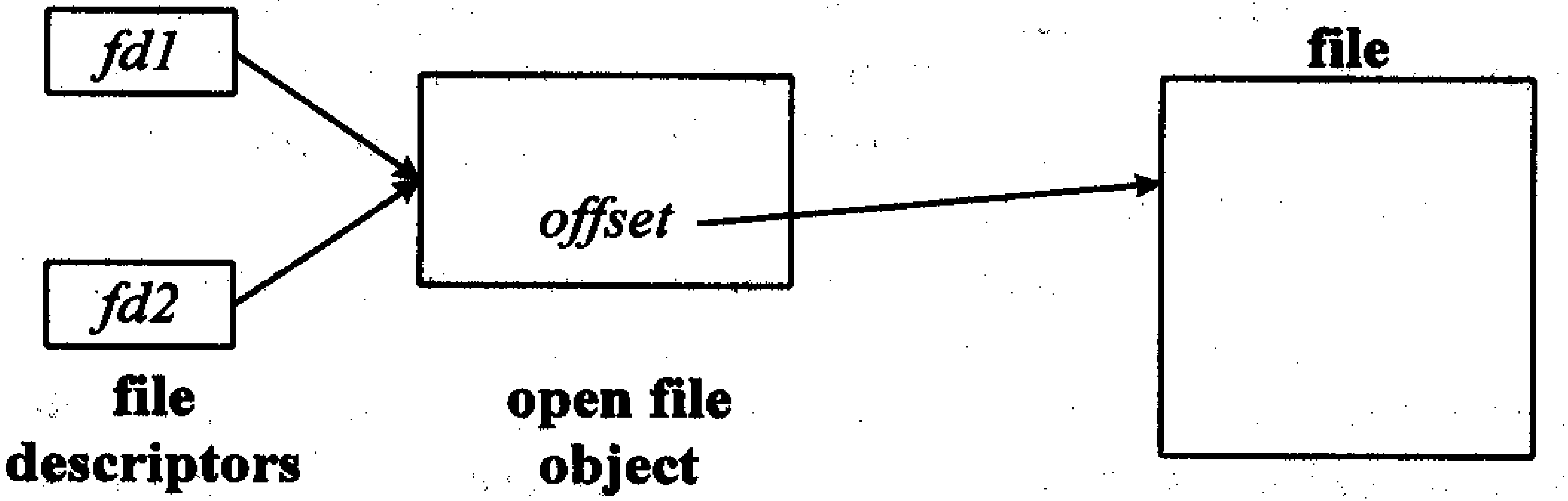


Figure 8-3. Descriptor cloned through *dup*, *dup2*, or *fork*.

Giorgio Richelli
giorgio_richelli@it.ibm.com

File I/O

- ✓ Position file pointer

```
off_t lseek(int fd, off_t offset, int whence);
```

- ✓ Read/Write

```
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
```

File I/O

- ✓ Scatter-Gather

```
ssize_t writev(int fd,const struct iovec *iov,int cnt);
ssize_t readv(int fd, const struct iovec *iov,int cnt);

     struct iovec {
       void  *iov_base;/* Starting address */
       size_t iov_len; /* Number of bytes to transfer */
     };
```

- ✓ Close

```
int close(int fd);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Link, Unlink, Rename

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

- ✓ creates a new (hard) link to file

```
int unlink(const char *pathname);
```

- ✓ unlinks a file (and possibly deletes it)

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

- ✓ renames a file, moving it between directories (if required)

All return <0> if successful or <-1> in case of errors

File Locking

- ✔ Operations on filesystem metadata are atomic, e.g:
  - file creation/removal
  - read/write
  - ..
- ✔ Locks:
  - Advisory locks: protect from cooperative processes
  - Mandatory locks: require kernel support

Giorgio Richelli
giorgio_richelli@it.ibm.com

Partitions & Logical Disks

- A <partition> is a portion of a physical disk drive
- A <logical disk>:
  - A storage abstraction
  - Linear sequence of fixed sized, randomly accessible blocks.

- Advanced Topics:
  - Disk mirror
  - Stripe sets
  - RAID 5,6, 10

Giorgio Richelli
giorgio_richelli@it.ibm.com

File system

- ✓ Create a file system:
  - newfs (BSD)
  - mkfs (SYSV, Linux)
- ✓ To be accessible, a file system must be <mounted> on a <mount point>
- ✓ The mount point (directory) is then <covered> by the mounted file system.
  - mount table & vfs list

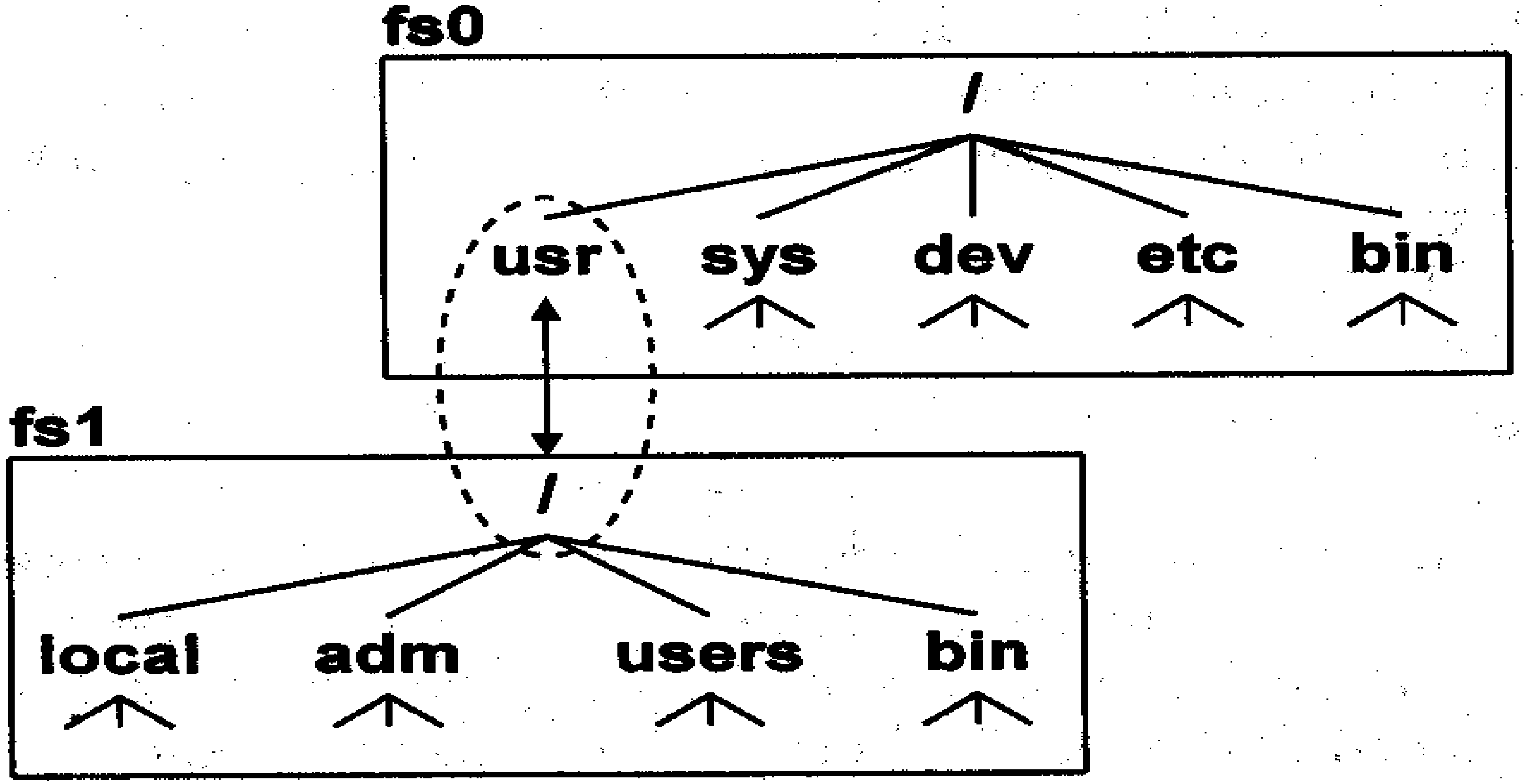


Figure 8-5. Mounting one file system onto another.

Giorgio Richelli
giorgio_richelli@it.ibm.com

Device I/O

- Block & character devices
- Major & minor device number:
  - indexes in the device table
- Character:

```
struct {
   int (*d_open)();
   int (*d_close)();
   int (*d_read)();
   int (*d_write)();
} cdevsw[];
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Opening a file

```
int fd = open(char *pathname, int flags, mode_t mode);
```

- pathname: filename (directory, ...)
- flags: read, write, ...
- mode: file access permissions (optional)
- returns:
  - fd: file descriptor
  - -1: in case of errors

```
fd=open(name,O_RDWR|O_CREAT,S_IRWXU);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Opening a file

```
fd = open(pathname, mode)
```

- ✓ Allocate a descriptor
- ✓ Allocate an open file object
- ✓ Lookup path name
- ✓ Check permissions
- ✓ Check operation
- ✓ Not exist ? O_Creat/VOP_CREAT, OK : ENOENT
- ✓ VOP_OPEN
- ✓ O_TRUNC ? VOP_SETATTR
- ✓ Initialize
- ✓ Return the index of the descriptor

Closing a File

```
int close(fd)
```

- fd: file descriptor
- Returns:
  - 0 if successful
  - -1 in case of errors

Example

```
sts=close(fd);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Reading from File

```
ssize_t read(int fd, void *buf, size_t count);
```

- fd: file descriptor
- buf: pointer to buffer space
- count: I/O size (in bytes)
- returns:
    - number of bytes read
    - -1 in case of error

Example:

```
nb=read(fd,buffer,sizeof(buffer));
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

Writing to File

```
ssize_t write(int fd, const void *buf, size_t count);
```

- fd: file descriptor
- buf: pointer to bufferspace
- count: I/O size
- Returns:
  - number of bytes written
  - -1 in case of error

Example:

```
nb=write(fd,string,strlen(string)+1);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

File Seek

```
off_t lseek(int fildes, off_t offs, int whence);
```

- Positions the <file pointer> for further I/O ops
- fd: file descriptor
- offs: offset (in bytes)
- whence: offs is relative to start of file, current position or end of file
- returns:
  - offset: resulting offset location
  - -1: in case of error

Example:

```
bytes=lseek(fd,offset,SEEK_SET);
```

Giorgio Richelli
giorgio_richelli@it.ibm.com

File I/O (1)

read(to a user buffer address)

- From <fd> get the open file object, verify mode → vnode → get the rw-lock → read()
- From <offset> → block number & the offset in blk→bmap()
- The page is not in memory ? page fault → VM handler→ getpage()→uiomove()→copyout()
- read() returns, unlock, advance the offset, return the number of bytes read

Giorgio Richelli
giorgio_richelli@it.ibm.com

File I/O (2)

write(from user addr.space):

- ✓ Not immediately to disk
- ✓ May increase the file size
- ✓ May require the allocation of data blocks
- ✓ Multiple steps:
  - Read the entire block
  - Write relevant data
  - Write back all the block

Giorgio Richelli
giorgio_richelli@it.ibm.com