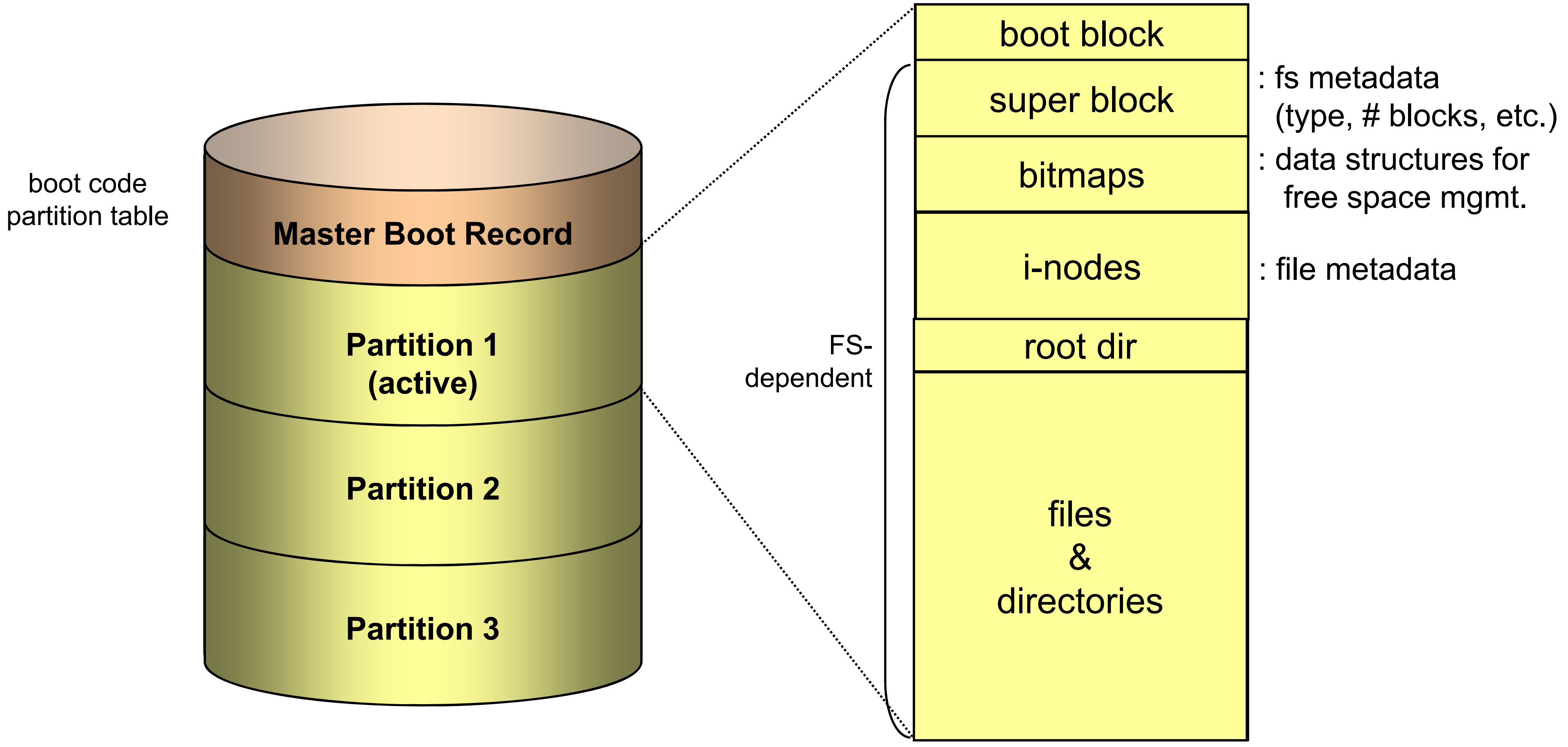
Files and Directories

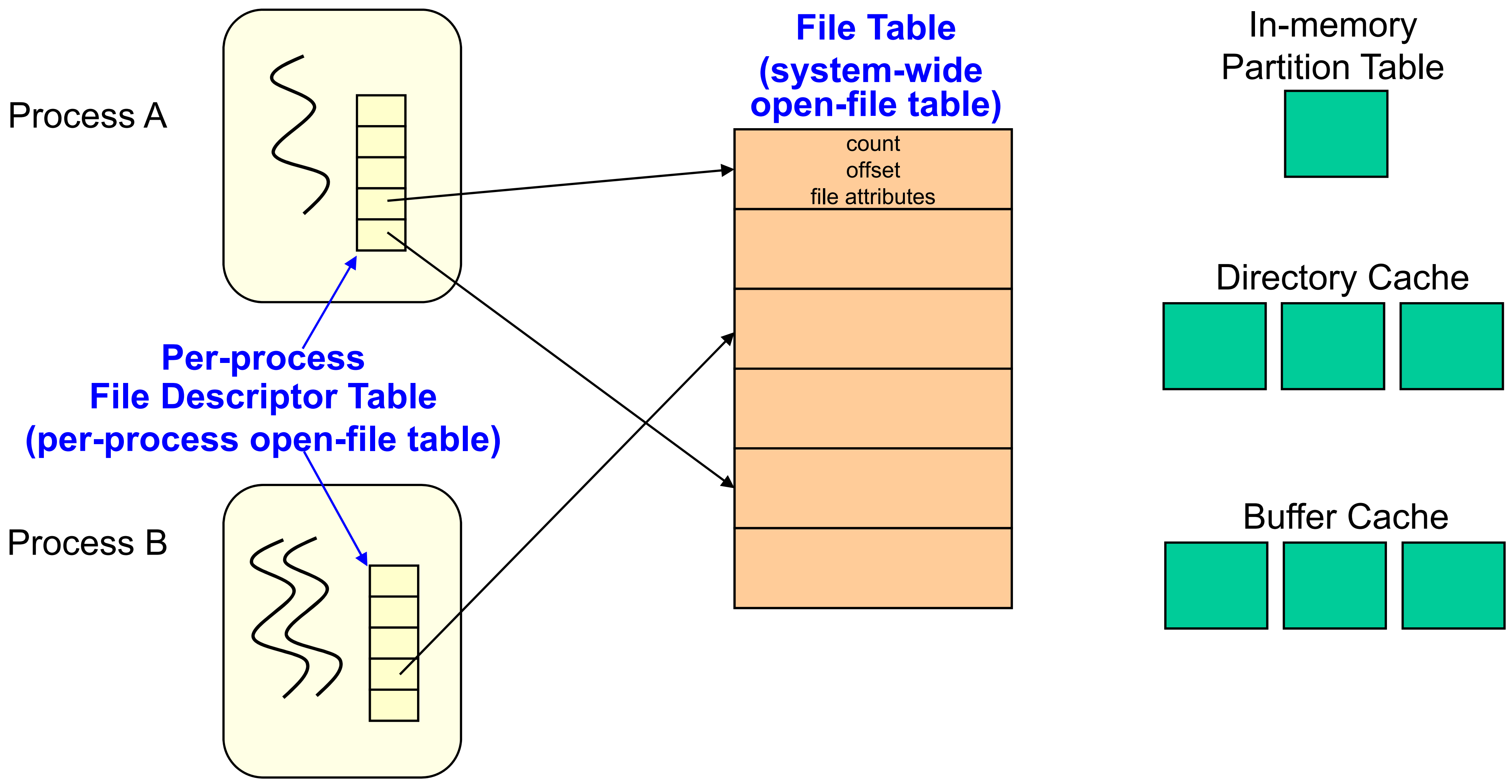
경희대학교 컴퓨터공학과

조 진 성

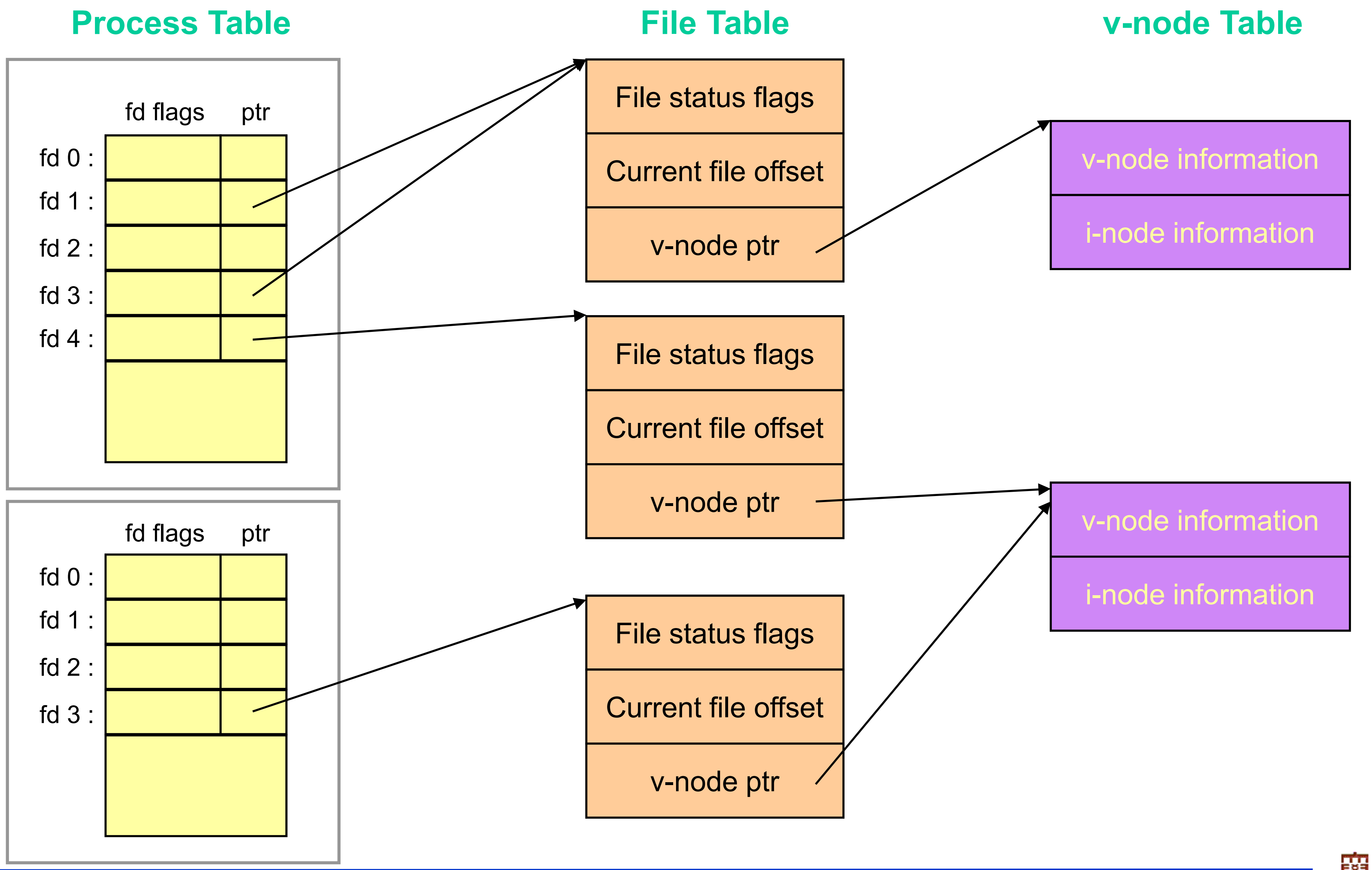
Linux File System: On-Disk Structure



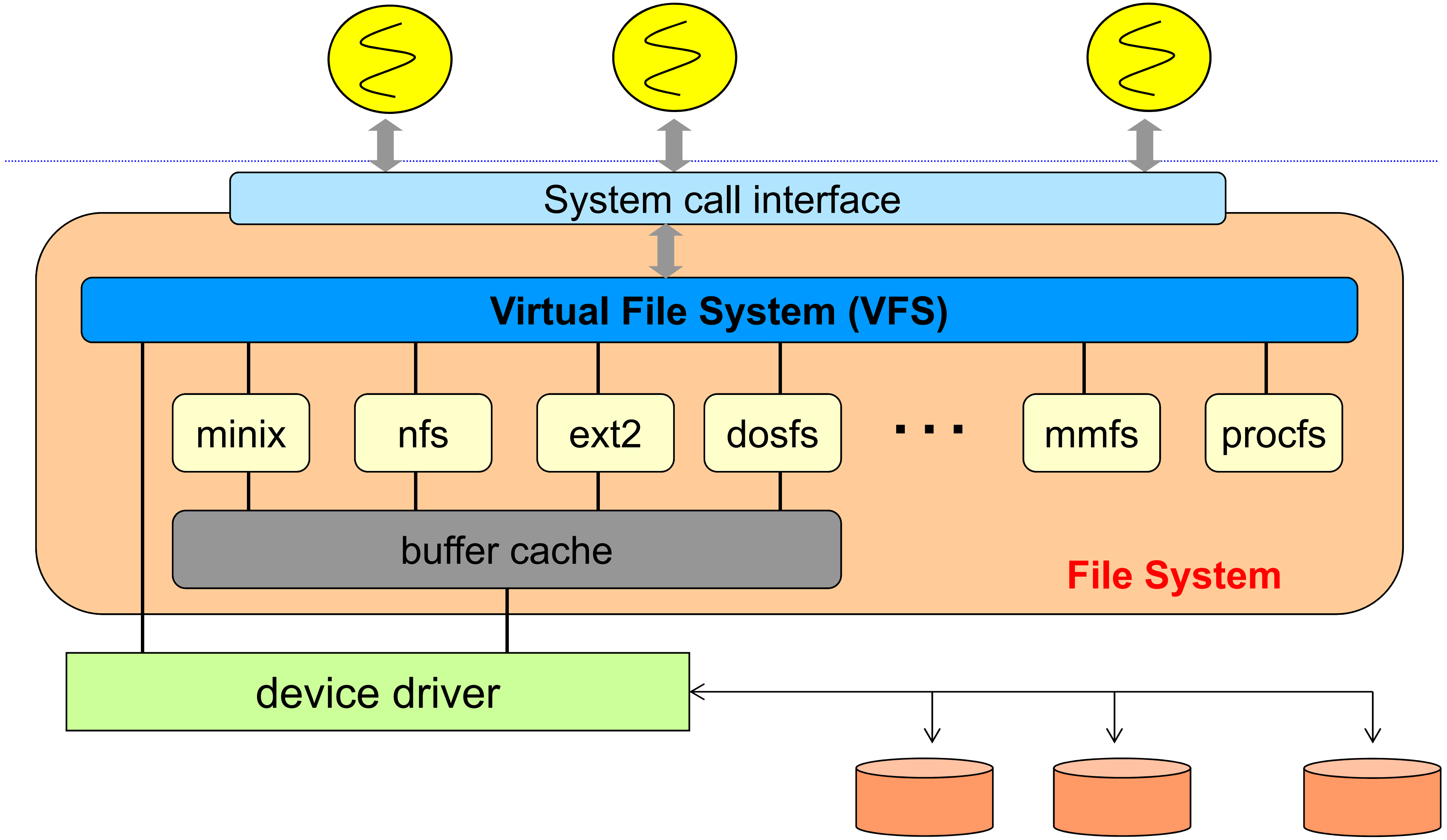
Linux File System: In-Memory Structure



Kernel Data Structure for Open Files

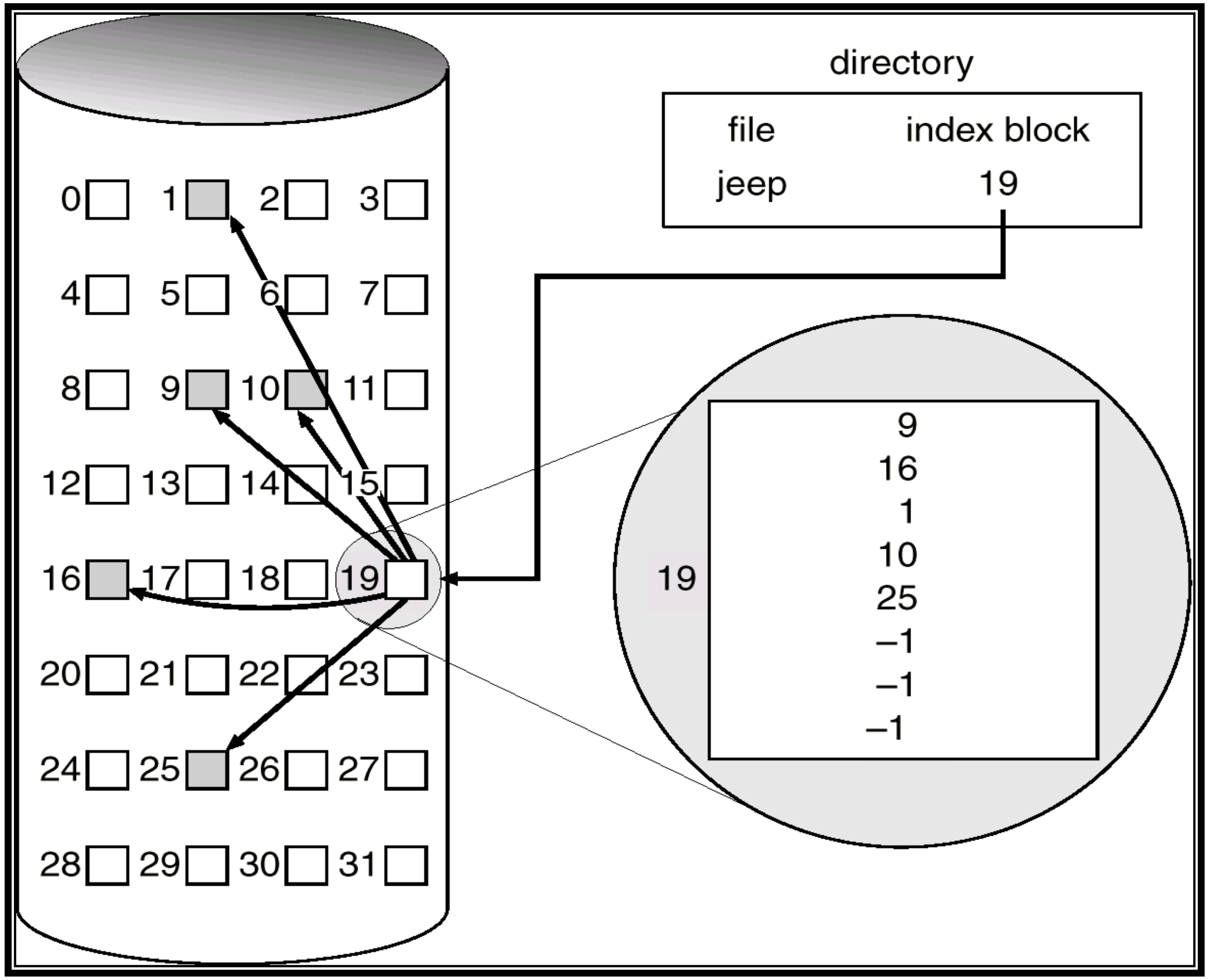


Virtual File System



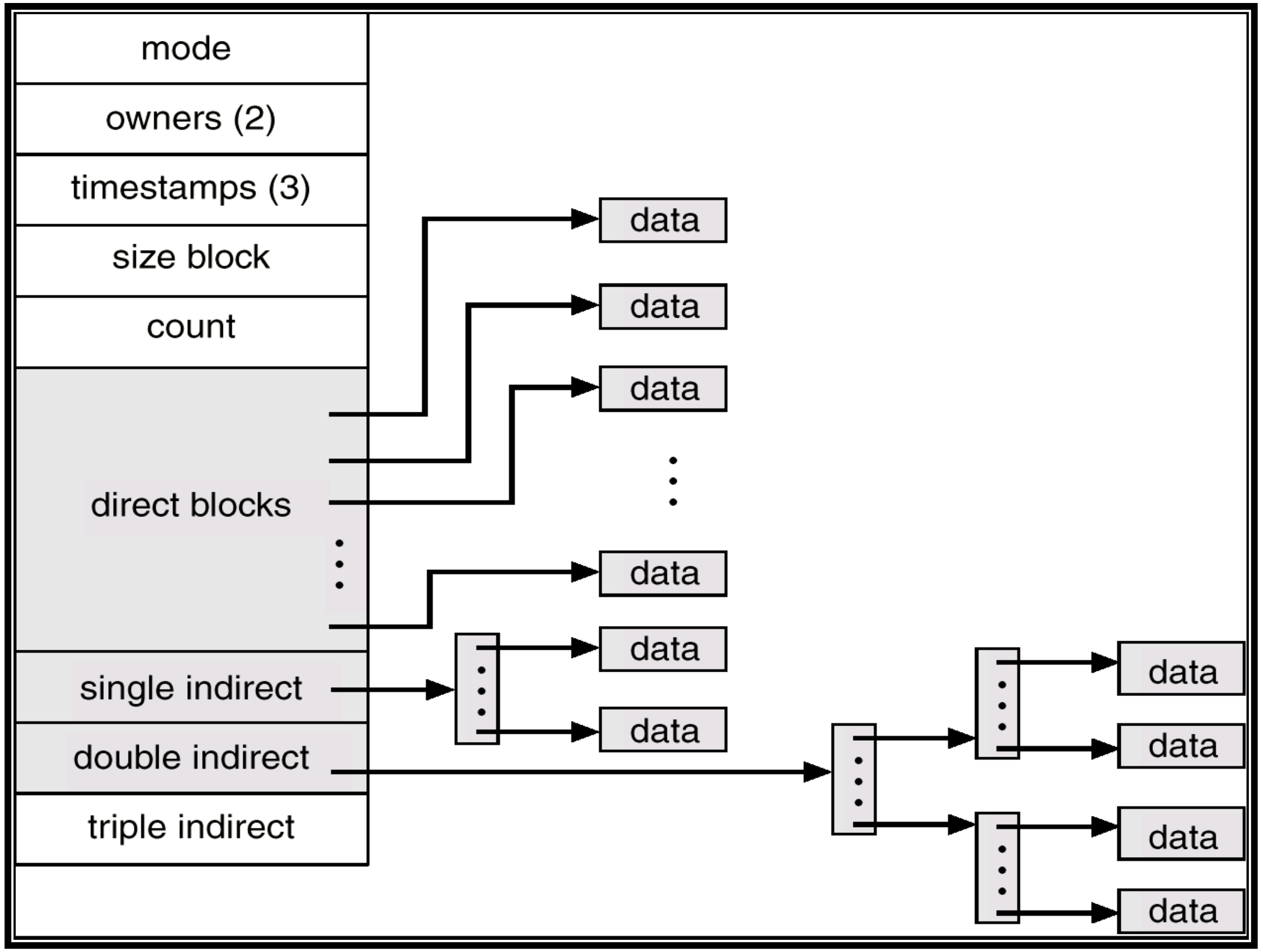
Linux File System Structure

- Indexed allocation



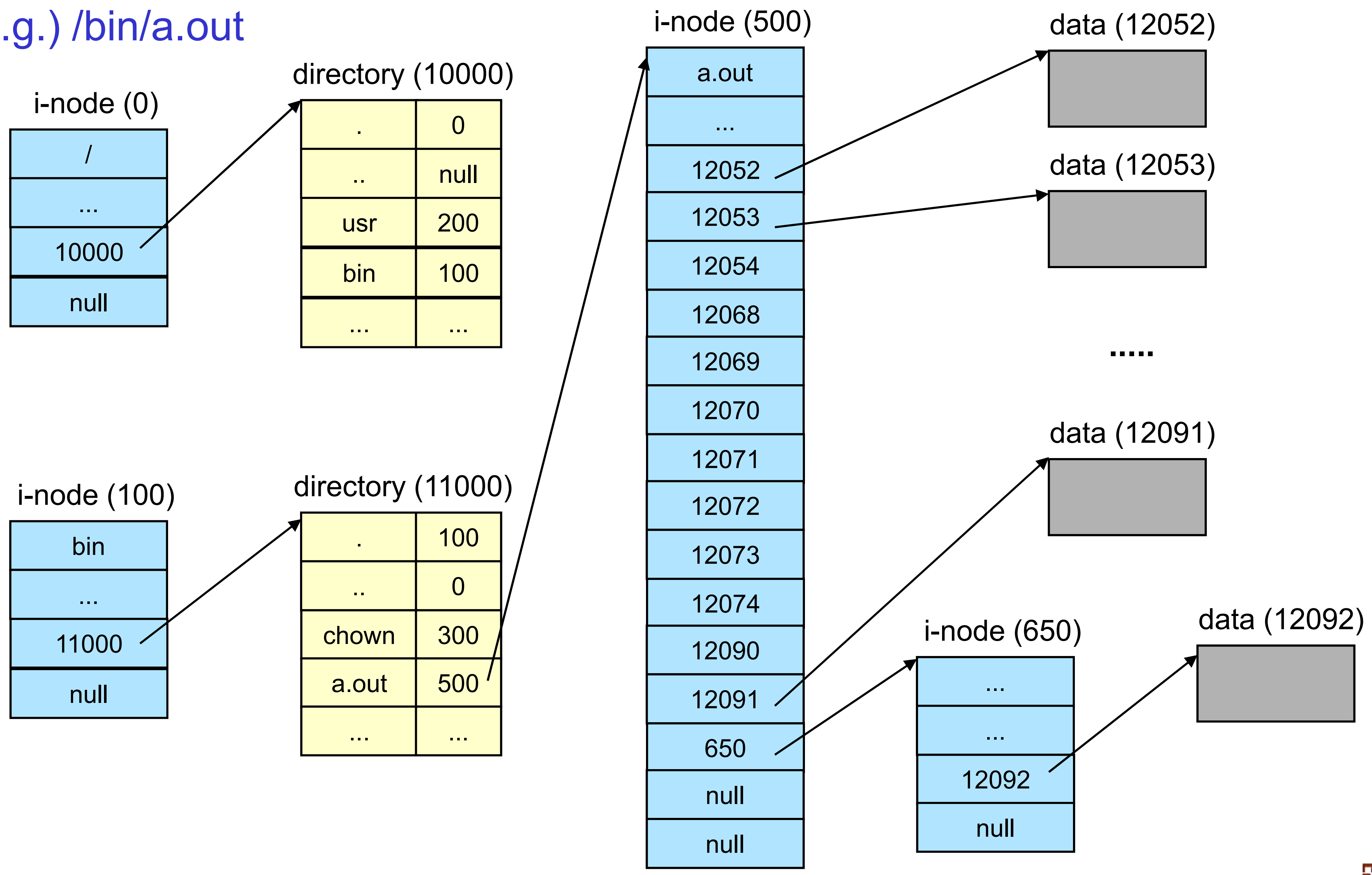
Linux File System Structure (Cont'd)

- i-node structure



Linux File System Structure (Cont'd)

- E.g.) /bin/a.out



System Calls for Files & Directories

- Get file status
  - ✓ `#include <sys/types.h>`
  - ✓ `#include <sys/stat.h>`
  - ✓ `int stat(char *pathname, struct stat *buf);`
  - ✓ `int fstat(int fd, struct stat *buf);`
  - ✓ `int lstat(char *pathname, struct stat *buf);`
  - ✓ all return: 0 if OK, –1 on error
  - ✓ File type macros
    - **`S_ISREG()`** : regular file
    - **`S_ISDIR()`** : directory file
    - **`S_ISCHR()`** : character special file
    - **`S_ISBLK()`** : block special file
    - **`S_ISFIFO()`** : pipe or FIFO
    - **`S_ISLNK()`** : symbolic link
    - **`S_ISSOCK()`** : socket

System Calls for Files & Directories (Cont’d)

```
struct stat  {
   mode_t     st_mode;      /* file type & mode (permissions) */
   ino_t      st_ino;       /* i-node number (serial number) */
   dev_t      st_dev;       /* device number (file system) */
   dev_t      st_rdev;      /* device number for special files */
   nlink_t    st_nlink;     /* number of links */
   uid_t      st_uid;       /* user ID of owner */
   gid_t      st_gid;       /* group ID of owner */
   off_t      st_size;      /* size in bytes, for regular files */
   time_t     st_atime;     /* time of last access */
   time_t     st_mtime;     /* time of last modification */
   time_t     st_ctime;     /* time of last file status change */
   long       st_blksize;   /* best I/O block size */
   long       st_blocks;    /* no. of 512-byte blocks allocated */
};
```

Exercise

- List the status of files (**`lstat`** example)

```
$ gcc –o stat stat.c (or make stat)
$ ./stat stat.c
$ ./stat .
$ ./stat *
$ ./stat .* * | more
```

System Calls for Files & Directories (Cont’d)

- Set file creation mask
  - ✓ `#include <sys/types.h>`
  - ✓ `#include <sys/stat.h>`
  - ✓ `mode_t umask(mode_t cmask);`
  - ✓ return: previous file mode creation mask
  - ✓ The parameter, **`cmask`**
    - **`S_ISUID, S_ISGID`**
    - **`S_IRUSR, S_IWUSR, S_IXUSR`**
    - **`S_IRGRP, S_IWGRP, S_IXGRP`**
    - **`S_IROTH, S_IWOTH, S_IXOTH`**

- Change permissions of a file
  - ✓ `#include <sys/types.h>`
  - ✓ `#include <sys/stat.h>`
  - ✓ `int chmod(char *pathname, mode_t mode);`
  - ✓ `int fchmod(int fd, mode_t mode);`
  - ✓ both return: 0 if OK, –1 on error

Exercise

- **umask** example

```
$ gcc –o umask umask.c (or make umask)
$ ./umask
$ ls –l bar foo
-rw-rw-rw-   1 cjs     other          0 Aug  9 10:55 bar
-rw-------   1 cjs     other          0 Aug  9 10:55 foo
```

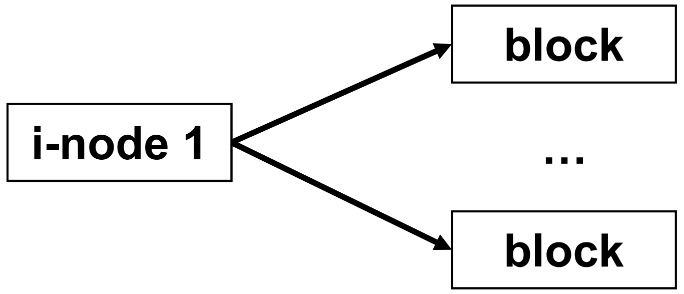
- **chmod** example

```
$ gcc –o chmod chmod.c (or make chmod)
$ ./chmod
$ ls –l bar foo
-rwSr--rw-   1 cjs     other          0 Aug  9 11:10 bar
-rw-r--r--   1 cjs     other          0 Aug  9 11:10 foo
```

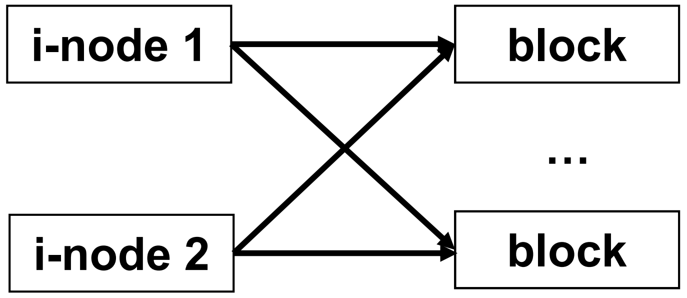
System Calls for Files & Directories (Cont'd)

- Change ownership of a file
  - ✓ `#include <sys/types.h>`
  - ✓ `#include <unistd.h>`
  - ✓ `int chown(char *pathname, uid_t owner, gid_t group);`
  - ✓ `int fchown(int fd, uid_t owner, gid_t group);`
  - ✓ `int lchown(char *pathname, uid_t owner, gid_t group);`
  - ✓ all return: 0 if OK, –1 on error
- Make a new name for a file (hard link)
  - ✓ `#include <unistd.h>`
  - ✓ `int link(char *existingpath, char *newpath);`
  - ✓ return: 0 if OK, –1 on error
- Make a new name for a file (symbolic link)
  - ✓ `#include <unistd.h>`
  - ✓ `int symlink(char *actualpath, char *sympath);`
  - ✓ return: 0 if OK, –1 on error

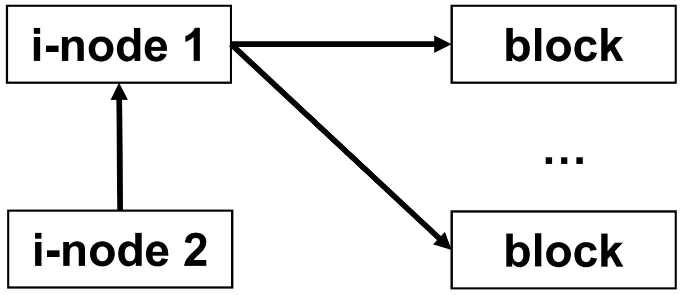
System Calls for Files & Directories (Cont'd)



- Hard link



- Symbolic link



Exercise

- Make my own **`ln`** program using **`link`** system call

```
$ gcc –o myln myln.c (or make myln)
$ ./myln myln.c myln.c.ln
$ ls –l myln.c myln.c.ln
$ vi myln.c.ln (& update it)
$ vi myln.c (& check the update)
$ ./stat myln.c myln.c.ln (& check st_nlink)
$ rm myln.c.ln
$ ./stat myln.c (& check st_nlink)
```

- Make my own **`ln –s`** program using **`symlink`** system call

```
$ gcc –o mysln mysln.c (or make mysln)
$ ./mysln mysln.c mysln.c.ln
$ ls –l mysln.c mysln.c.ln
$ Repeat the commands in the above exercise
```

System Calls for Files & Directories (Cont'd)

- Remove a file or directory
  - ✓ `#include <stdio.h>`
  - ✓ `int remove(char *pathname);`
  - ✓ return: 0 if OK, –1 on error

- Rename a file or directory
  - ✓ `#include <stdio.h>`
  - ✓ `int rename(char *oldname, char *newname);`
  - ✓ return: 0 if OK, –1 on error

Exercise

- Make my own **rm** program using **remove** system call

```
$ gcc –o myrm myrm.c (or make myrm)
$ ./myrm myrm.o
$ ls –l myrm.o
$ ./myrm *.o
$ ./myrm abc1234
```

- Make my own **mv** program using **rename** system call

```
$ gcc –o mymv mymv.c (or make mymv)
$ ./mymv mymv.o oops.o
$ ls –l mymv.o oops.o
$ ./mymv oops.o ../oops.o
$ ./mymv ../oops.o . (does it work?)
```

System Calls for Files & Directories (Cont'd)

- Create a directory
  - ✓ `#include <sys/types.h>`
  - ✓ `#include <sys/stat.h>`
  - ✓ `int mkdir(char *pathname, mode_t mode);`
  - ✓ return: 0 if OK, –1 on error
- Remove an empty directory
  - ✓ `#include <unistd.h>`
  - ✓ `int rmdir(char *pathname);`
  - ✓ return: 0 if OK, –1 on error

Exercise

- Make my own **`mkdir`** program using **`mkdir`** system call

```
$ gcc –o mymd mymd.c (or make mymd)
$ ./mymd test
$ ls –l | grep test
```

- Make my own **`rmdir`** program using **`rmdir`** system call

```
$ gcc –o myrd myrd.c (or make myrd)
$ ./myrd test
$ ls –l | grep test
$ ./myrd abc1234
```

System Calls for Files & Directories (Cont'd)

- Reading a directory
  - ✓ `#include <sys/types.h>`
  - ✓ `#include <dirent.h>`
  - ✓ `DIR *opendir(char *pathname);`
  - ✓ return: pointer if OK, **`NULL`** on error

  - ✓ `struct dirent *readdir(DIR *dp);`
  - ✓ return: pointer if OK, **`NULL`** at end of directory or error

```
struct dirent  {
  ino_t  d_ino;                /* i-node number */
  char   d_name[NAME_MAX+1]; /* Null-terminated file name */
};
```

  - ✓ `void rewinddir(DIR *dp);`

  - ✓ `int closedir(DIR *dp);`
  - ✓ return: 0 if OK, –1 on error

Exercise: myls.c, mylsr.c

- Make my own **`ls`** program using directory-related system calls

```
$ gcc -o myls myls.c (or make myls)
$ ./myls
$ ls
```

- Make my own **`ls -R`** program using directory-related system calls

```
$ gcc -o mylsr mylsr.c (or make mylsr)
$ ./mylsr
$ ls -R
```

System Calls for Files & Directories (Cont'd)

- Change working directory
  - ✓ `#include <unistd.h>`
  - ✓ `int chdir(char *pathname);`
  - ✓ `int fchdir(int fd);`
  - ✓ return: 0 if OK, –1 on error

- Get current working directory
  - ✓ `#include <unistd.h>`
  - ✓ `char *getcwd(char *buf, size_t size);`
  - ✓ return: **`buf`** if OK, **`NULL`** on error

- Commit buffer cache to disk
  - ✓ `#include <unistd.h>`
  - ✓ `void sync(void);`
  - ✓ `int fsync(int fd);`
  - ✓ return: 0 if OK, –1 on error

Exercise

- Make my own **`cd`** program using **`chdir`** system call

```
$ gcc –o mycd mycd.c (or make mycd)
$ pwd
$ ./mycd ..
$ pwd (where are you now?)
```

- Make my own **`mypwd`** program using **`getcwd`** system call

```
$ gcc –o mypwd mypwd.c (or make mypwd)
$ ./mypwd
$ pwd
```

Summary

- System calls in Linux for files and directories
  - ✓ `stat, fstat, lstat`
  - ✓ `umask`
  - ✓ `chmod, fchmod`
  - ✓ `chown, fchown, lchown`
  - ✓ `link, symlink`
  - ✓ `remove, rename`
  - ✓ `mkdir, rmdir`
  - ✓ `opendir, closedir, readdir, rewinddir`
  - ✓ `chdir, fchdir`
  - ✓ `sync, fsync`
  - ✓ `getcwd`