Managing Files and Directories

Chris Brown

In This Module …

File System Structure

Links, inodes, directories

Working with Directories

Creating, deleting, listing

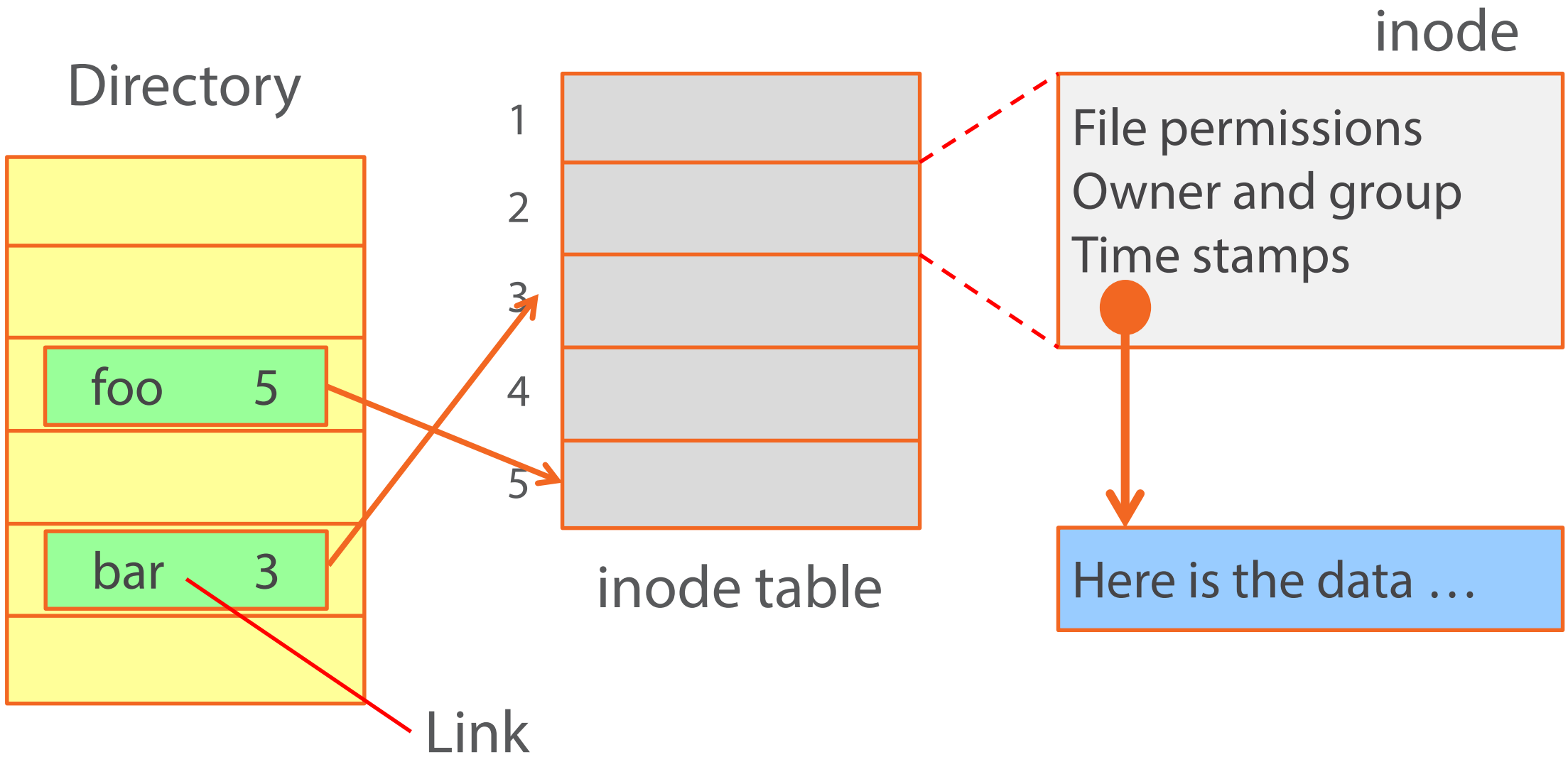
Advanced Techniques

Monitoring file events

Demonstration:

Directory listing

File System Structure



Examining File Attributes

Pathname of the file to be opened

Pointer to a stat structure

```
stat(name, buf)
fstat(fd, buf)
```

```
struct stat {
    …
};
```

Open file descriptor

Returns 0 on success, -1 on failure

The stat Structure

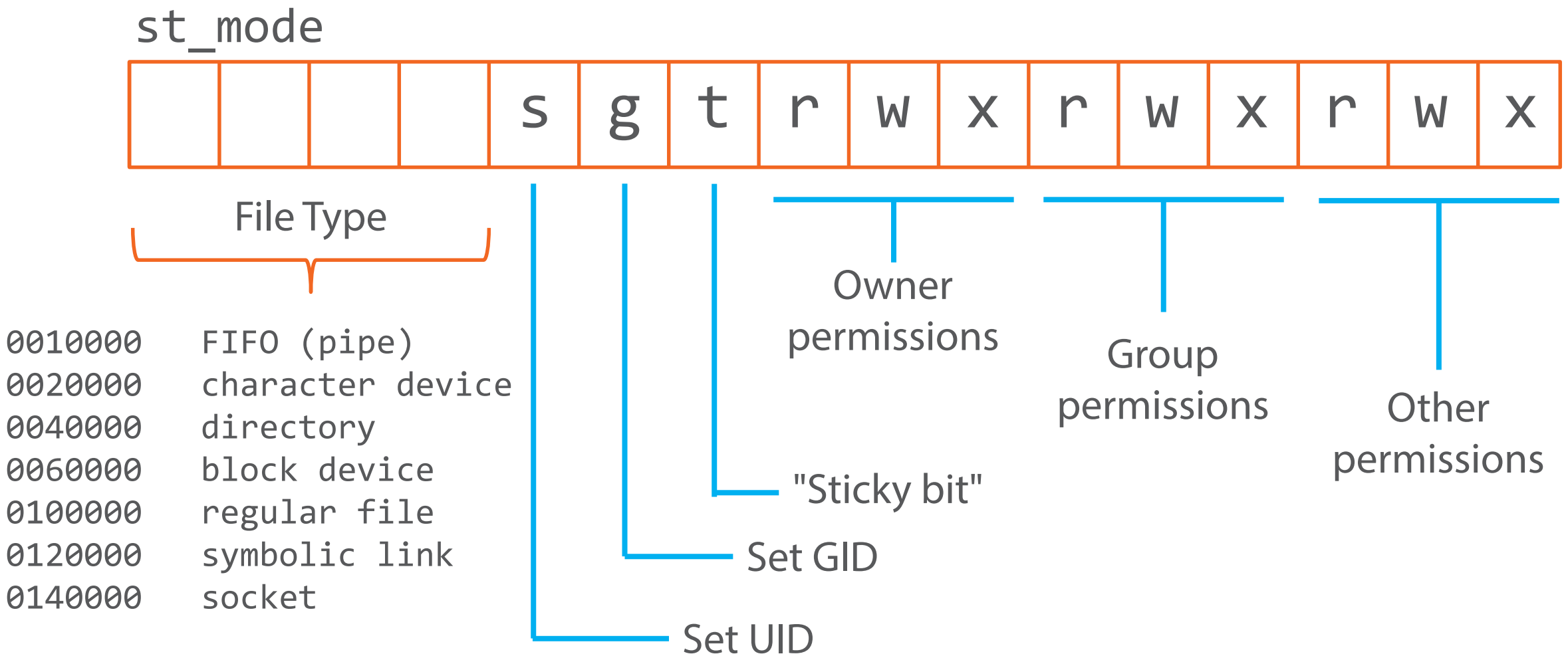
```
struct stat {
    dev_t     st_dev;     /* ID of device containing file */
    ino_t     st_ino;     /* inode number */
    mode_t    st_mode;    /* protection */
    nlink_t   st_nlink;   /* number of hard links */
    uid_t     st_uid;     /* user ID of owner */
    gid_t     st_gid;     /* group ID of owner */
    dev_t     st_rdev;    /* device ID (if special file) */
    off_t     st_size;    /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t  st_blocks;  /* number of 512B blocks allocated */
    time_t    st_atime;   /* time of last access */
    time_t    st_mtime;   /* time of last modification */
    time_t    st_ctime;   /* time of last status change */
};
```

Time Stamps

`st_atime`	Time of last access
`st_mtime`	Time of last modification
`st_ctime`	Time of last status change (inode)

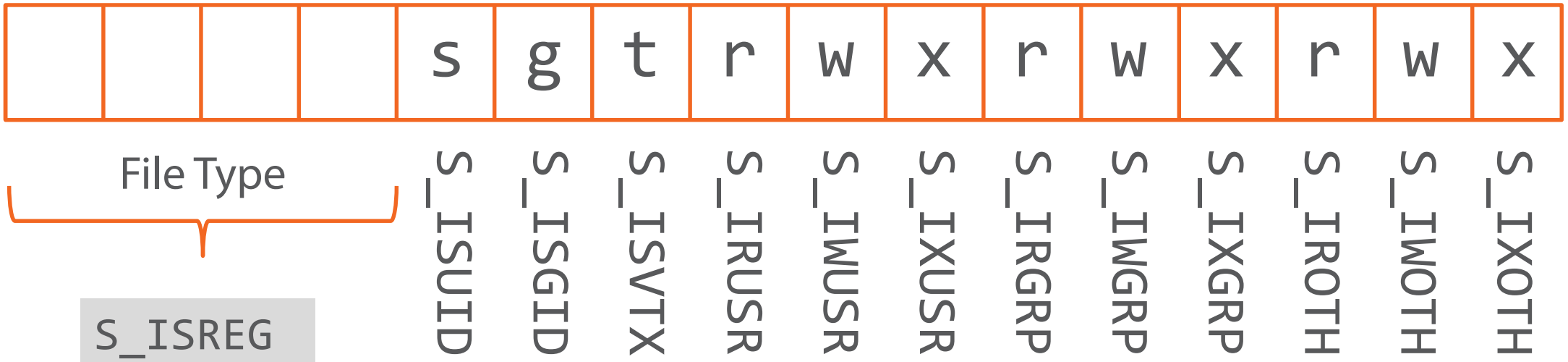
The *creation* time of a file is not recorded

File Type and Permissions



Useful Macros

st_mode




```
S_ISREG
S_ISDIR
S_ISCHR
S_ISBLK
S_ISFIFO
S_ISLNK
S_ISSOCK
```

```
if (statbuf.st_mode & S_IWOTH)
    printf("file is world writeable");
```

```
if (S_ISREG(statbuf.st_mode))
    printf("regular file");
```

Creating Files

Includes O_CREAT

```
open(name, flags, mode)
```

Pathname

Access mode of the file

```
creat(name, mode)
```

Creating and Removing Links

```
link(oldname, newname)
```

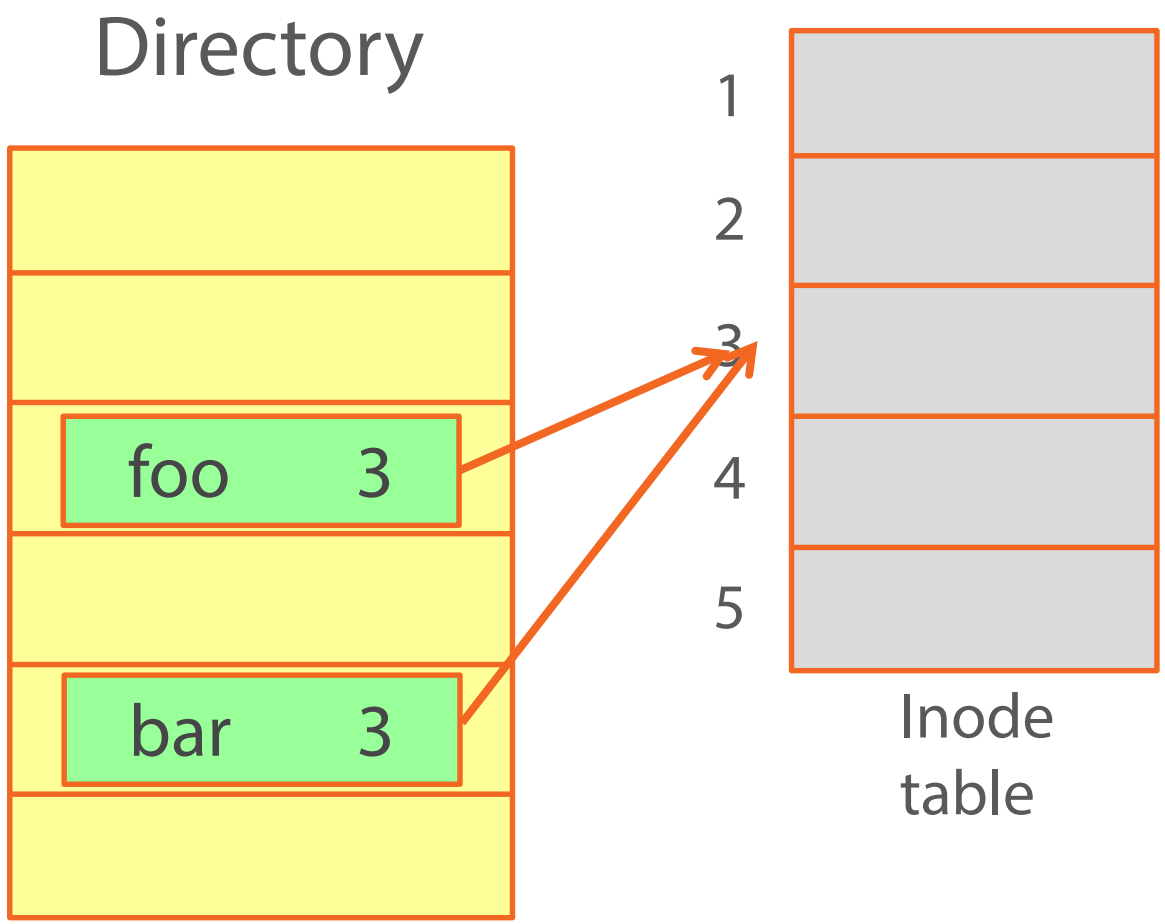
Gives the file an additional name

```
unlink(name)
```

If this is the last remaining link
the file is deleted

Deletion is postponed if a
process has the file open

Creating and Removing Links

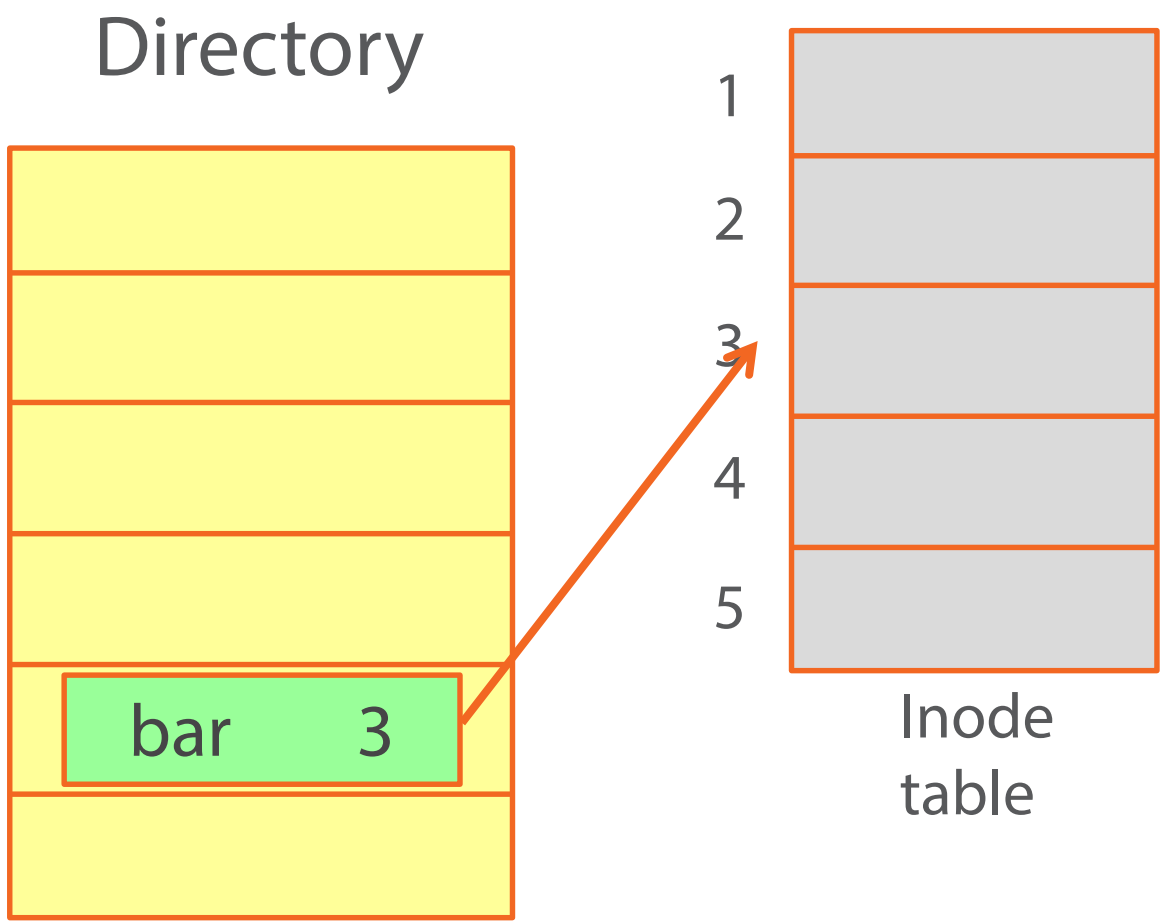



```
open("foo", O_CREAT, 0644);

link("foo", "bar");

unlink("foo");
```

Creating and Removing Links

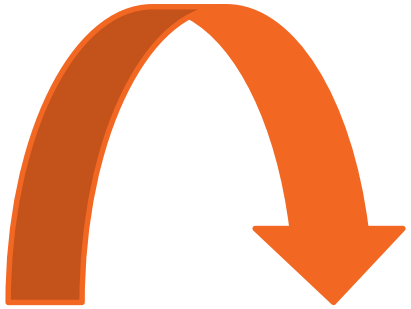



```
open("foo", O_CREAT, 0644);

link("foo", "bar");

unlink("foo");
```

Symbolic Links



A symbolic link is a file that contains the name of another file

Similar to a shortcut on Windows

Symbolic links can do things that hard links cannot

- Link to directories
- Link across file system boundaries

Creating Symbolic Links

```
symlink(oldname, newname)
```

```
unlink(name)
```

Removes the symlink, not the target file

Hard Links vs. Symbolic Links

Hard Link

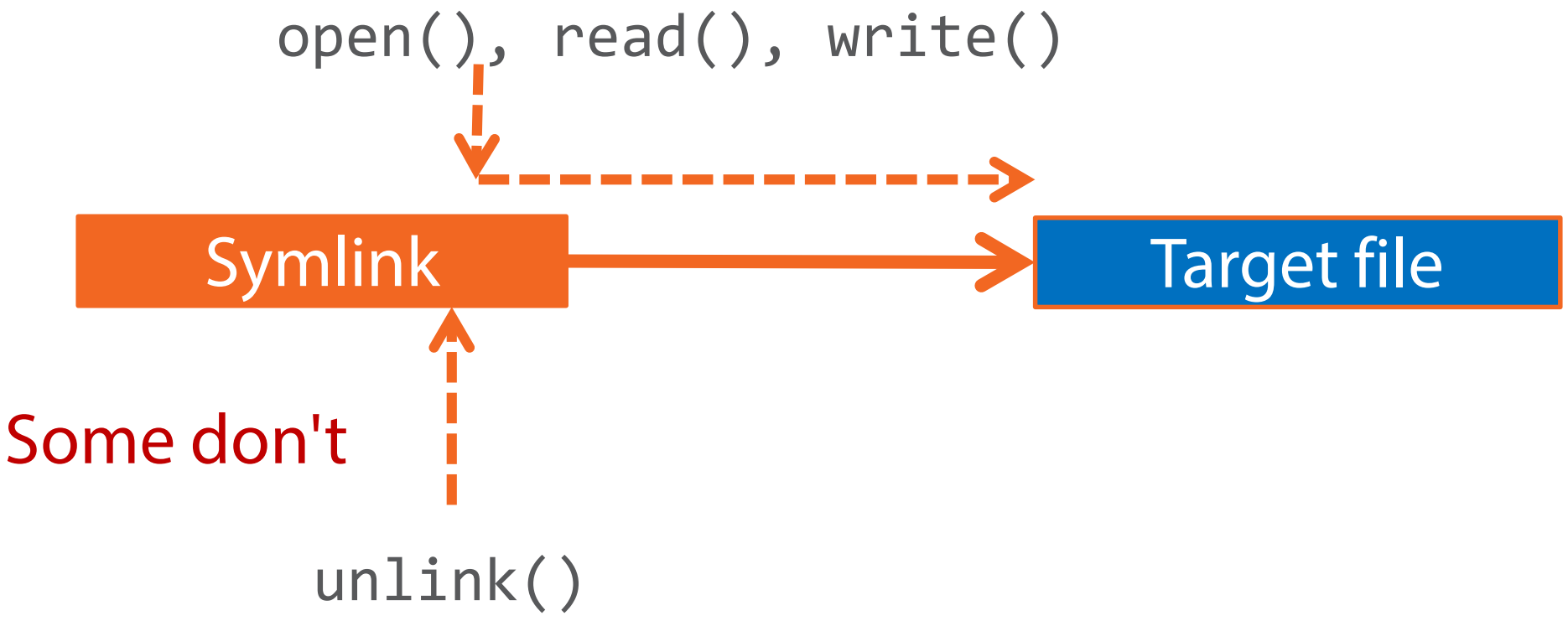
Name → inode number

Symbolic Link

Name → Another name

To Follow or Not to Follow

Some system calls follow symbolic links:



Interacting with Directories

The current directory

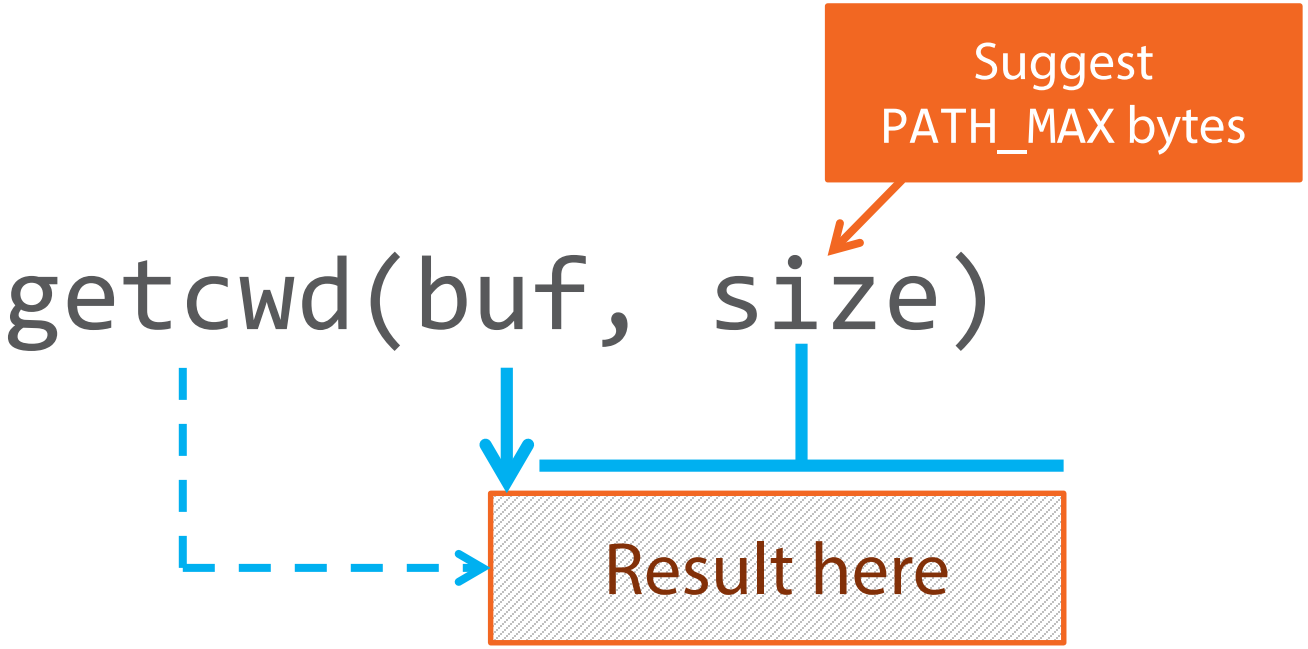
Creating and deleting directories

Reading directories

The Current Directory

Every process has a "current directory"

Relative path names are interpreted relative to this



The Current Directory

A process can change its current directory

```
chdir(pathname)
```

Creating and Deleting Directories

```
mkdir(name, mode)
```

This will only create *one* directory

`mkdir("a/b/c", 0755)` will fail unless a, b exist

```
rmdir(name)
```

Fails unless the directory is empty

Reading Directories

```
d = opendir(dirname)
```

struct dirent

```
d_ino  /* inode number */
d_type /* file type */
d_name /* file name */
```

```
info = readdir(d)
```

loop

```
closedir(d)
```

Directory Traversal Example

```
/* Add up sizes of all files in the current directory */

#include <stdio.h>
#include <sys/stat.h>
#include <dirent.h>

void main()
{
  DIR *d;
  struct dirent *info;  /* A directory entry */
  struct stat sb;       /* The stat buffer */
  long total = 0;       /* Total of file sizes */
```

No error checking !

Directory Traversal Example

```
  d = opendir(".");

  while ((info = readdir(d)) != NULL) {
    stat(info->d_name, &sb);
    total += sb.st_size;
  }

  closedir(d);

  printf("total size = %ld\n", total);
}
```

Doing It in Python

- Most of the calls we've seen here are available in the os module:

Function	Description
`os.stat(path)`	Returns a `stat_result` instance, similar to a stat structure
`os.link(src, dst)`	Create a new hard link
`os.symlink(src, dst)`	Create a symbolic link
`os.chdir(path)`	Change directory
`os.getcwd()`	Get current working directory

No direct equivalents to `opendir/readdir`, use `listdir` instead …

Python Directory Traversal

```
#! /usr/bin/python3
# Add up sizes of all files in the current directory

import os
total = 0

for file in os.listdir(".") :
  statinfo = os.stat(file)
  total = total + statinfo.st_size

print ("total is", total)
```

Monitoring File System Events

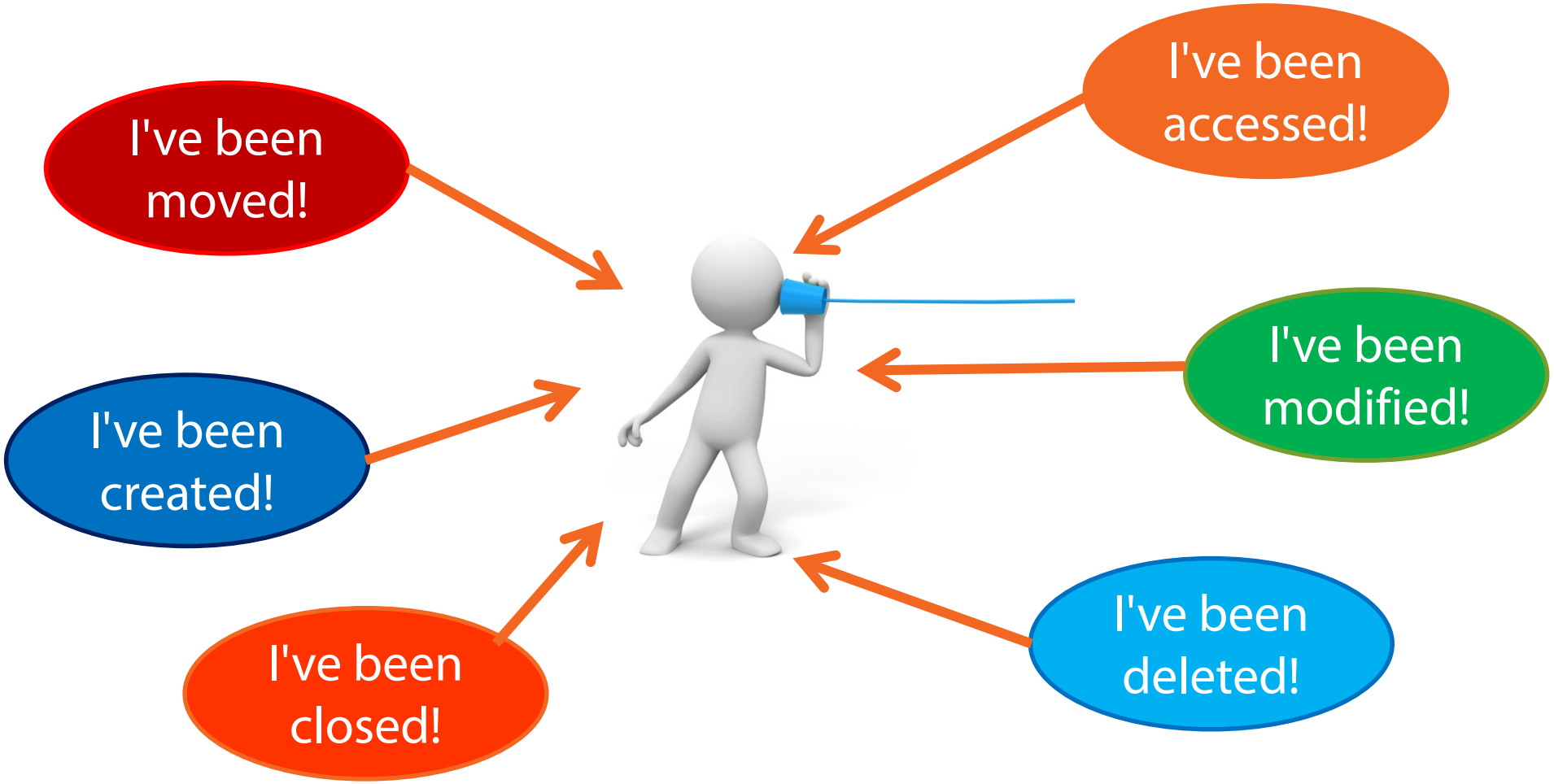
The `inotify` API

Creating an `inotify` instance

Adding to the watch list

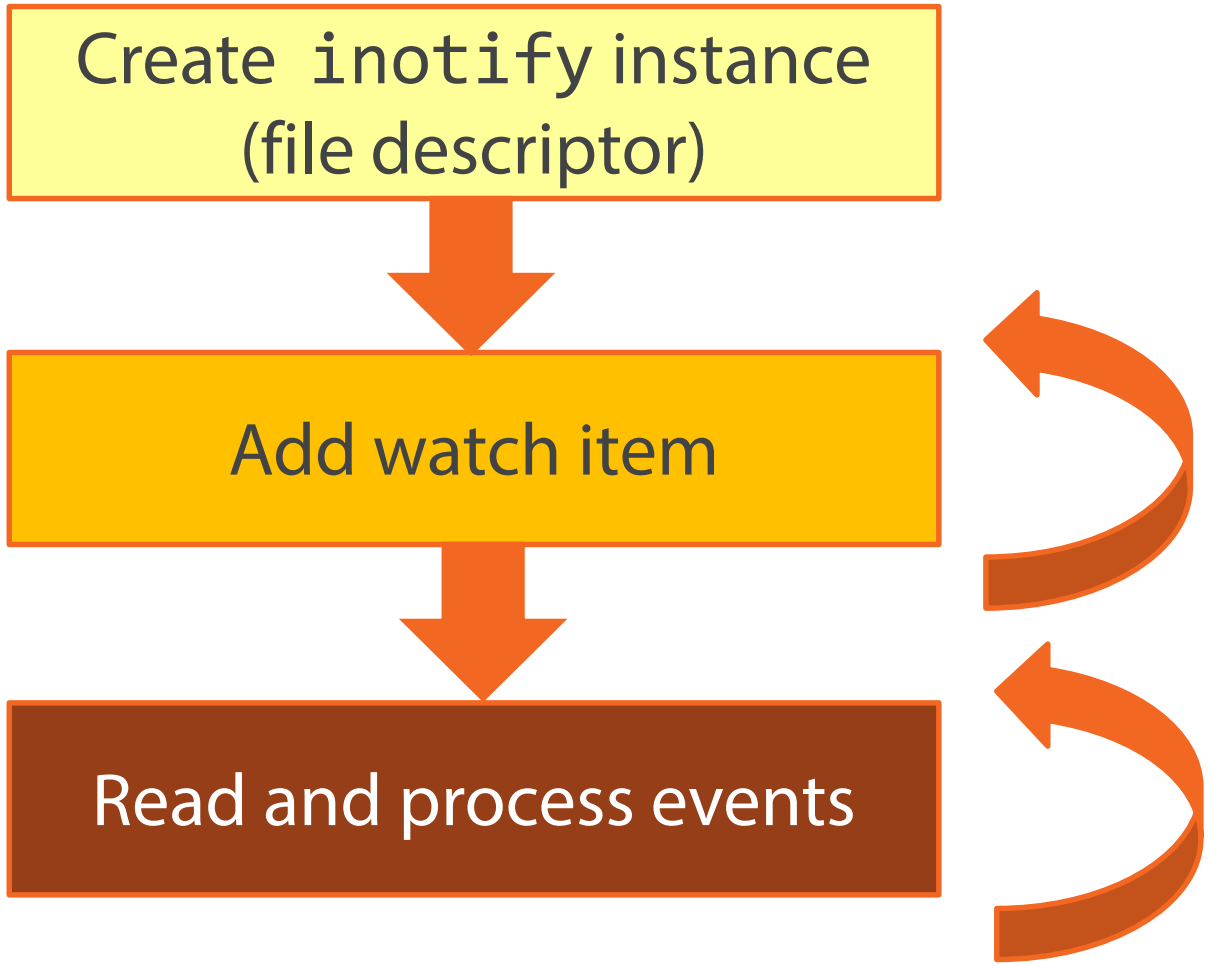
Reading events

File System Events



Individual files or whole directories can be watched

Three Steps



Creating an inotify Instance

```
fd = inotify_init()
```

Returns a file descriptor on which we can later `read()` the events.

Adding a watch Item

File descriptor of `inotify` instance

Name of file or directory to be watched.

```
wd = inotify_add_watch(fd, path, mask)
```

Returns a watch descriptor (small integer) identifying this watch

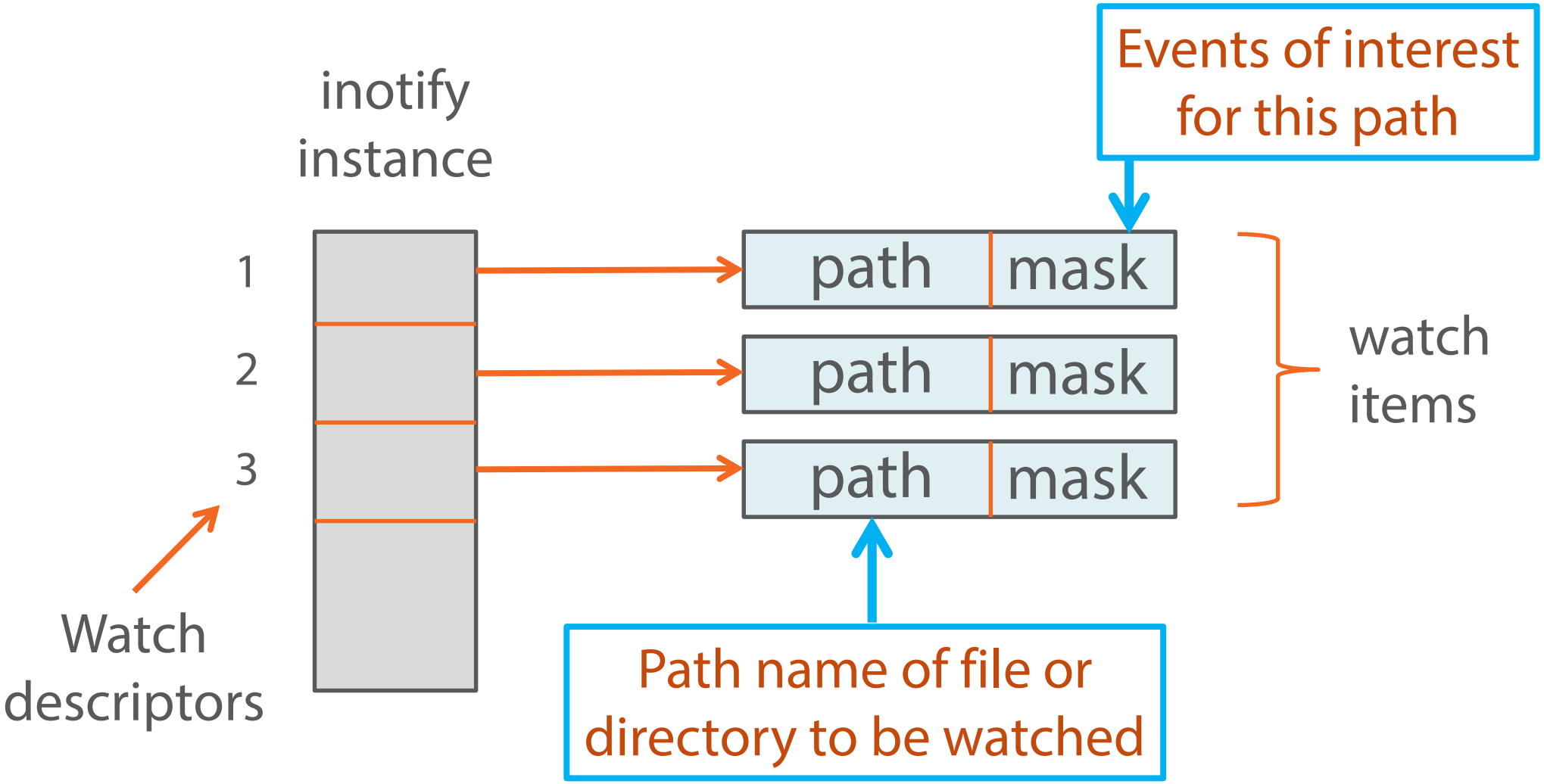
Bit mask specifying the events to be monitored

Watches and Events in Detail

Each event is specified by a single-bit constant
-- bitwise OR them together

Bit value	Meaning
IN_ACCESS	File was accessed
IN_ATTRIB	File attributes changed (ownership, permissions etc.)
IN_CREAT	File created inside watched directory
IN_DELETE	File deleted inside watched directory
IN_DELETE_SELF	Watched file deleted
IN_MODIFY	File was modified
IN_MOVE_SELF	File was moved

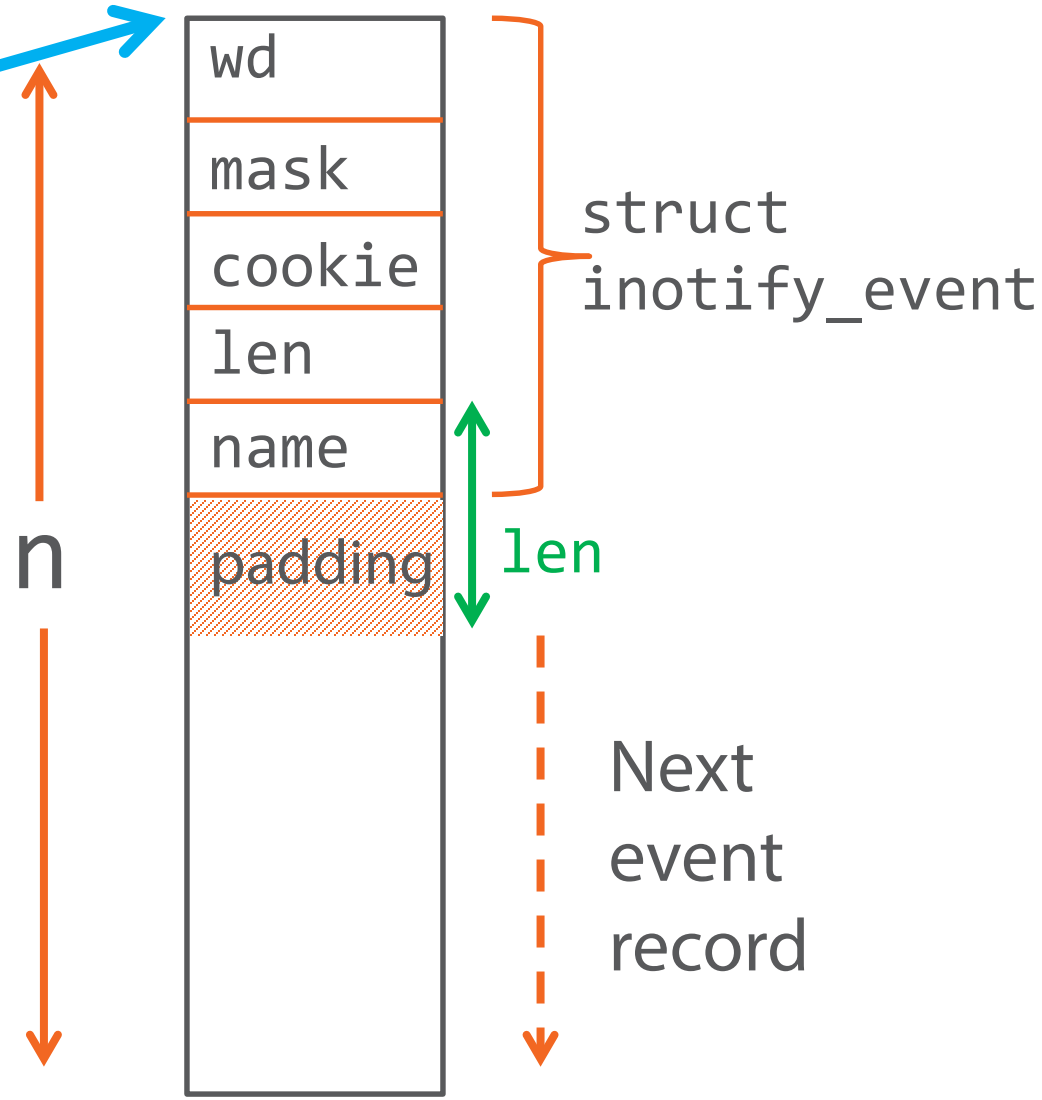
Watch Items



Reading Events

File descriptor
from `inotify_init()`

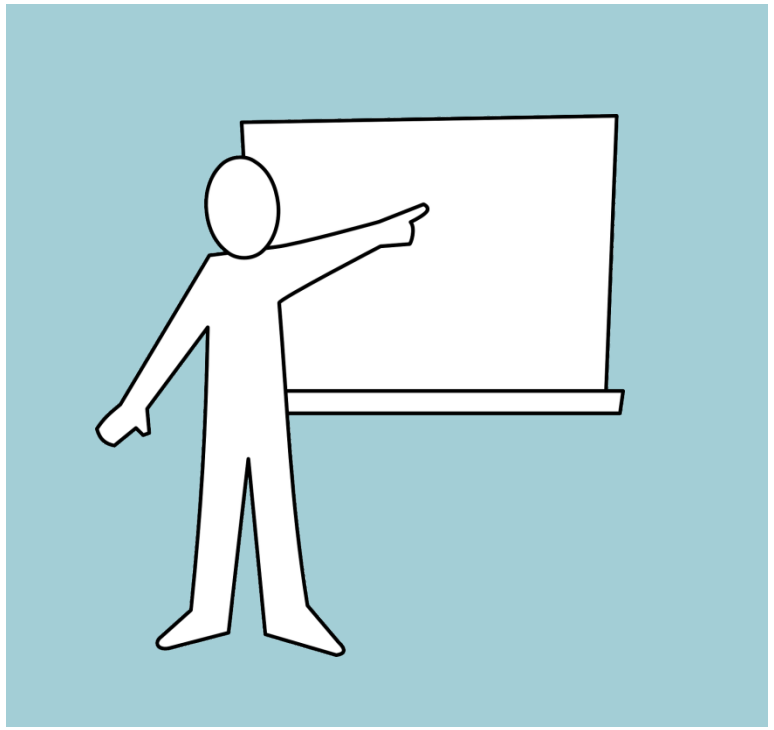
```
n = read(fd, buf, size)
```



The read will block until
a watched event occurs

More than one event may be
returned in a single read

Demonstration



Read a list of files (*not* directories) from a configuration file

Watch them for modification or deletion

Report changes to a log file

Module Summary

File system structure

— inodes and the stat structure

Links and symbolic links

Directories

— Current directory

— Creation, deletion

— Traversal

Monitoring file system events

— `inotify`

Coming up in the Next Module

Accessing the command line

The Environment

Time