

File Systems

- **Secondary storage management**
- **File**
 - Collection of related data
- **File types**
 - Executable, text, source, object, database, etc.

File Attributes

- **Name**
- **Type**
 - Not on all systems
- **Location and size**
- **Protection**
- **Ownership/usage information**

File Operations

- **Create/open**
- **Write/append**
- **Read**
- **Position (seek)**
- **Delete (or truncate)**

File as ADT

Low-Level File I/O

```
int open (const char* pathname,
         int oflag,
         ...
         /* mode_t mode */);
```

```
O_RDONLY  O_APPEND  O_NONBLOCK
O_WRONLY  O_CREAT   ...
O_RDWR   O_EXCL
O_TRUNC
```

Low-Level File I/O

```
int close (int filedes);
off_t lseek (int filedes,
            off_t offset,
            int whence);
```

```
SEEK_SET
SEEK_CUR
SEEK_END
```

Seek beyond end of file?

Low-Level File I/O

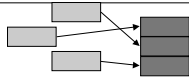
```
ssize_t read (int filedes,
             void* buff,
             size_t nbytes);
```

```
ssize_t write (int filedes,
              const void* buff,
              size_t nbytes);
```

```
STDIN_FILENO
STDOUT_FILENO
STDERR_FILENO
```

Low-Level File I/O

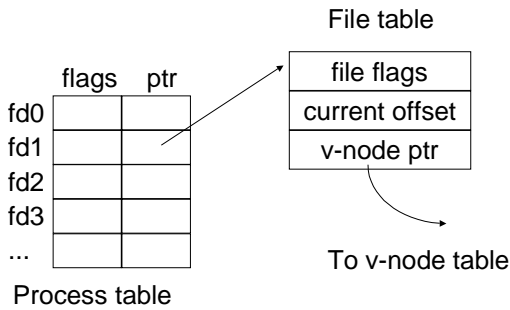
Scatter read,
gather write



```
ssize_t readv (int filedes,
               const struct iocb iov[],
               int iovcnt);
```

```
ssize_t writev (int filedes,
                const struct iocb iov[],
                int iovcnt);
```

OS File Data Structures



Low-Level Control

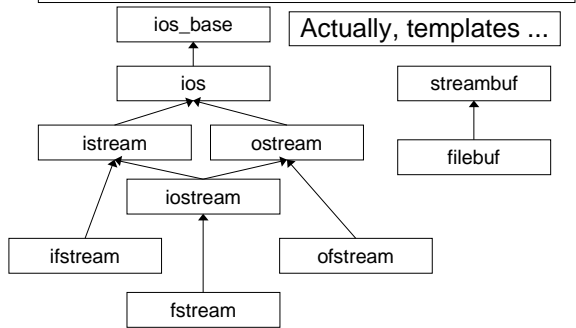
```
int fcntl (int filedes,
           int cmd,
           ...
           /* int arg */);
```

- | | |
|------------|---------|
| F_DUPFD | F_GETFL |
| F_GETFD | F_SETFL |
| F_SETFD | ... |
| FD_CLOEXEC | |

Memory-Mapped Files

- **File mapped to VAS**
 - File data appears as user memory
 - Requires “large enough” VAS
 - May be shared among processes
- **Program accesses directly**
- **Data flushed to file on close**

C++ File Streams



C++ fstream

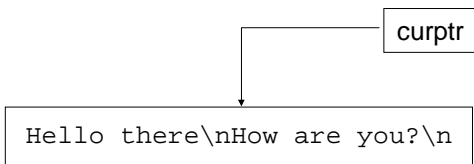
```

class fstream
{
  ...
  fstream();
  fstream(const char* s,
    ios_base::openmode mode = ...);
  filebuf* rdbuf() const;
  bool is_open() const;
  void open(const char* s,
    ios_base::openmode mode = ...);
  void close();
};
  
```

High-Level File I/O

- **Supports C++ file streams**
 - Originally used with C language
- **Buffers data inside process**
 - Unlike `read` and `write`
- **Uses file pointer, not descriptor**
 - `FILE* fp;`
 - Pointer to file structure
 - May vary from system to system

High-Level File I/O



High-Level File I/O

```
FILE* fopen (const char* pathname,
             const char* type);
FILE* freopen(const char* pathname,
             const char* type,
             FILE* fp);
FILE* fdopen (int filedes,
              const char* type);
int fileno (FILE* fp);
int fflush (FILE* fp);
int fclose (FILE* fp);
```

High-Level File I/O

```

int getc (FILE* fp);
int fgetc (FILE* fp);
int getchar (void);
int ferror (FILE* fp);
int feof (FILE* fp);
void clearerr (FILE* fp);
int ungetc (int c, FILE* fp);
int putc (int c, FILE* fp);
int fputc (int c, FILE* fp);
int putchar (int c);

```

Macro?

High-Level File I/O

```

char* fgets (char* buf,
             int n,
             FILE* fp);

int fputs (const char* str,
           FILE* fp);

int puts (const char* str);

```

High-Level File I/O

```

size_t fread (void* ptr,
              size_t size,
              size_t nobj,
              FILE* fp);

size_t fwrite (const void* ptr,
               size_t size,
               size_t nobj,
               FILE* fp);

```

Binary I/O: heterogeneous systems?

High-Level File I/O

```
long ftell (FILE* fp);
void rewind (FILE* fp);
int fseek (FILE* fp,
           long offset,
           int whence);
```

- SEEK_SET
- SEEK_CUR
- SEEK_END

File Structure

• Unstructured

- Simple byte stream (UNIX)
 - Embedded line breaks? (' \n ')

• Record structure

- Fixed- or variable-length records

• Indexed

- Lookup by key value

Record Structures

First record.-----
Second record of 3.-
Record 3.-----

Fixed

13	First record.
19	Second record of 3.
9	Record 3.

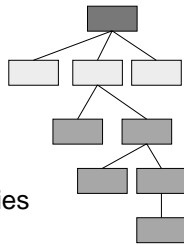
Variable

Access Methods

- **Sequential**
 - Advance from current file position
- **Direct access**
 - Record-structured: to record "n"
 - Unstructured: to byte "n"
- **Indexed**

Directory Structure

- **Single directory per device**
 - Early systems
- **Two-level**
 - One directory per user?
- **Tree-structured**
 - Directories within directories



File Protection



- **Ownership** (owner, group?)
- **Access lists**
 - Permissions by user
 - Owner, group member, other
 - Fixed mask, or variable list
- **File type** (e.g., executable)
