📘 `struct file_operations` Cheat Sheet (Linux Kernel)

This structure defines how your driver responds to file-related system calls like `open()`, `read()`, `write()`, etc. Found in `<linux/fs.h>`, commonly used in character drivers.

📌 Meta Information

`struct module *owner`

- **Purpose:** Prevents the module from being unloaded while it's in use.
- **Usage:** Always set to `THIS_MODULE`.

📖 File Position Control

`loff_t (*llseek)(struct file *filp, loff_t offset, int origin);`

- **Purpose:** Seek to a new file position.
- **Returns:** New offset (≥ 0), or error (< 0).
- **If NULL:** Kernel may allow strange seeking behavior.

📥 Reading

`ssize_t (*read)(struct file *filp, char __user *buf, size_t count, loff_t *offset);`

- **Purpose:** Read data from the device.

- **Returns:** Number of bytes read, or error.
- **If NULL:** `read()` system call returns `-EINVAL`.

`ssize_t (*aio_read)(struct kiocb *, char __user *, size_t, loff_t);`

- **Purpose:** Asynchronous version of `read`.
- **If NULL:** Falls back to synchronous `read`.

📤 Writing

`ssize_t (*write)(struct file *filp, const char __user *buf, size_t count, loff_t *offset);`

- **Purpose:** Write data to the device.
- **Returns:** Number of bytes written, or error.
- **If NULL:** `write()` returns `-EINVAL`.

`ssize_t (*aio_write)(struct kiocb *, const char __user *, size_t, loff_t);`

- **Purpose:** Asynchronous write.

📂 Directory Ops (Filesystems Only)

`int (*readdir)(struct file *, void *, filldir_t);`

- **Used by:** Filesystems, not device drivers.
- **If NULL:** Should be for device drivers.

📊 Polling

```
unsigned int (*poll)(struct file *filp, struct poll_table_struct *wait);
```

- **Purpose:** Supports `poll()`, `select()`, and `epoll()`.
- **Returns:** Bitmask indicating readiness for I/O.
- **If NULL:** Device assumed readable/writable without blocking.

🧪 IOCTL

```
int (*ioctl)(struct inode *, struct file *, unsigned int cmd, unsigned long arg);
```

- **Purpose:** Device-specific commands.
- **Returns:** `-ENOTTY` if unsupported.

🧠 Memory Mapping

```
int (*mmap)(struct file *, struct vm_area_struct *);
```

- **Purpose:** Map device memory into user space.
- **If NULL:** `mmap()` returns `-ENODEV`.

📂 Open/Close

```
int (*open)(struct inode *, struct file *);
```

- **Purpose:** Called when file is opened.
- **If NULL:** Open always succeeds silently.

`int (*release)(struct inode *, struct file *);`

- **Purpose:** Called when file is closed.
- **If NULL:** Close succeeds silently.

💾 Flushing & Syncing

`int (*flush)(struct file *);`

- **Purpose:** Flush data before closing a file descriptor.
- **Rarely used.**

`int (*fsync)(struct file *, struct dentry *, int datasync);`

- **Purpose:** Called by `fsync()` to flush pending data.
- **If NULL:** Returns `-EINVAL`.

`int (*aio_fsync)(struct kiocb *, int datasync);`

- **Purpose:** Async version of `fsync`.

📬 Async I/O & Notification

`int (*fasync)(int fd, struct file *filp, int mode);`

- **Purpose:** Asynchronous notification (e.g., SIGIO).

- **Advanced use case.**

🔐 File Locking

```
int (*lock)(struct file *, int cmd, struct file_lock *);
```

- **Purpose:** File locking.
- **Rarely used in device drivers.**

📦 Vector I/O

```
ssize_t (*readv)(struct file *, const struct iovec *,
unsigned long, loff_t *);
```

```
ssize_t (*writev)(struct file *, const struct iovec *,
unsigned long, loff_t *);
```

- **Purpose:** Scatter-gather I/O operations.
- **If NULL:** Falls back to normal `read`/`write`.

📡 Zero-copy Transfer

```
ssize_t (*sendfile)(struct file *, loff_t *, size_t,
read_actor_t, void *);
```

- **Purpose:** Efficient file transfer (e.g., for web servers).
- **If NULL:** `sendfile()` fails.

```
ssize_t (*sendpage)(struct file *, struct page *, int
offset, size_t size, loff_t *pos, int more);
```

- **Purpose:** Kernel uses this to send memory pages to a file descriptor.

🧭 Memory Area Hints

`unsigned long (*get_unmapped_area)(...)`

- **Purpose:** Helps the kernel find aligned memory regions for mmap.
- **Rarely implemented.**

⚙️ Miscellaneous

`int (*check_flags)(int flags);`

- **Purpose:** Validate flags in `fcntl(F_SETFL)`.

`int (*dir_notify)(struct file *, unsigned long);`

- **Purpose:** Filesystem-specific — notify on directory changes.

💡 Notes

- **NULL values** in most fields mean “not supported.”
- **Return values** follow the standard kernel convention: nonnegative = success, negative = error (`-EINVAL`, `-ENODEV`, etc.).
- Most **device drivers** only implement a subset: `open`, `release`, `read`, `write`, `mmap`, `poll`, `ioctl`.