Implemention Bodhi Linux system call(fstat)

To understand how the `fstat()` system call is implemented in Bodhi Linux, we need to look at a few things:

1. **Bodhi Linux Background**: Bodhi Linux is a lightweight Linux distribution based on Ubuntu (which is itself based on Debian). Therefore, Bodhi uses the same Linux kernel as Ubuntu, and system calls like `fstat()` are handled in the kernel.
2. **`fstat()` Overview**: The `fstat()` system call retrieves information about an open file referred to by a file descriptor. Its prototype in C is:

   ```
   c
   CopyEdit
   int fstat(int fd, struct stat *statbuf);
   ```

   It's implemented in the C library (glibc), which makes a system call to the Linux kernel.

High-Level Flow of `fstat()` in Linux

1. **User-space Call**:
   - Your C program calls `fstat(fd, &statbuf);`.
2. **glibc Wrapper**:
   - The `fstat()` in glibc translates the call into a syscall. On x86_64, it typically calls `syscall(SYS_fstat, fd, &statbuf)`.
3. **Kernel Handling**:
   - The actual system call in the kernel is `ksys_fstat()` (or `sys_newfstatat()` for modern implementations).
   - It eventually calls `vfs_fstat()` which populates the `stat` structure from the inode information.
4. **Return to User Space**:
   - The kernel returns the data to the glibc wrapper, which fills your `struct stat`.

Key Kernel Code (Linux Kernel)

Here's a simplified breakdown from the Linux source code:

Kernel entry point (in `fs/stat.c`):

```
c
CopyEdit
SYSCALL_DEFINE2(fstat, unsigned int, fd, struct stat __user *, statbuf)
{
    struct fd f = fdget(fd);
    int error = -EBADF;
    if (f.file) {
        error = vfs_fstat(f.file, statbuf);
        fdput(f);
    }
```

```
    return error;
}
```

vfs_fstat() (still in fs/stat.c):

```
c
CopyEdit
int vfs_fstat(struct file *file, struct stat __user *statbuf)
{
    struct kstat stat;
    int error = vfs_getattr(&file->f_path, &stat, STATX_BASIC_STATS,
AT_STATX_SYNC_AS_STAT);
    if (!error)
        error = cp_old_stat(&stat, statbuf);
    return error;
}
```

Even in Bodhi Linux, since it's based on Ubuntu and uses the Linux kernel, the implementation of `fstat()` is the same as any Linux system:

- In user space, `fstat()` is a wrapper in glibc.
- In the kernel, it's handled by `SYSCALL_DEFINE2(fstat, ...)` in the source file `fs/stat.c`.

If you're trying to **view or modify the implementation**, you'd look at the Linux kernel source, particularly

- `fs/stat.c`
- `include/linux/syscalls.h`
- `arch/x86/entry/syscalls/syscall_64.tbl` (for syscall number mapping

```
#include <stdio.h>

#include <sys/stat.h>

#include <unistd.h>

#include <fcntl.h>



int main() {

   const char* filename = "example.txt";

   int fd = open(filename, O_RDONLY);



   if (fd == -1) {

      perror("open");
```

```
        return 1;
    }

    struct stat fileStat;
    if (fstat(fd, &fileStat) == -1) {
        perror("fstat");
        close(fd);
        return 1;
    }

    std::cout << "File size: " << fileStat.st_size << " bytes\n";
    std::cout << "File permissions: " << (fileStat.st_mode & 0777) << "\n";
    std::cout << "Last modified: " << fileStat.st_mtime << "\n";

    close(fd);
    return 0;
}
```