بسمه تعالی

گزارش تمرین عملی چهارم – سوال سوم – درس سیستم‌های عامل دکتر اسدی – نیمسال اول 404-405

نویسندگان:

1. سیداحمد موسوی‌اول – 402106648
2. عرفان تیموری – 402105813

گزارشکار.

هدف از این پروژه، طراحی و پیاده‌سازی یک سیستم فایل (File System) در فضای کاربر با استفاده از رابط FUSE در سیستم‌عامل لینوکس است. این سامانه بدون نیاز به تغییر در هسته (Kernel)، یک فایل باینری حجیم (filesys.db) را به عنوان یک دیسک مجازی مدیریت کرده و عملیات استاندارد فایل (مانند ساخت، خواندن، نوشتن و حذف) را پشتیبانی می‌کند.

1. هسته عملیاتی سیستم (my_fs.c) – خلاصه:

این بخش به عنوان موتور اصلی سیستم عمل می‌کند و مسئولیت مدیریت مستقیم بایت‌ها بر روی دیسک مجازی را بر عهده دارد.

- ساختار دیسک: دیسک مجازی (filesys.db) دارای اندازه ثابت ۲ مگابایت است.
- سوپر-بلوک (Superblock): اولین بخش دیسک که اطلاعات حیاتی مانند نسخه سیستم‌فایل و آدرس آخرین بلوک نوشته شده را نگهداری می‌کند.
- ساختار پرونده‌ها (FileEntry): فایل‌ها به صورت یک لیست پیوندی (Linked List) ذخیره می‌شوند. هر مدخل شامل نام، سایز، وضعیت (فعال/غیرفعال) و آدرس محتوا است.
- توابع پایه: توابعی مانند fs_open برای یافتن یا ساخت فایل، fs_write نوشتن داده در آفست مشخص وfs_rm حذف منطقی فایل در این لایه پیاده‌سازی شده‌اند.

2. رابط FUSE و اتصال به کرنل (my_fuse.c) – شرح تفصیلی:

این بخش مهم‌ترین قسمت پروژه است که به عنوان مترجم بین درخواست‌های سیستم‌عامل (VFS) و منطق داخلی ما (my_fs) عمل می‌کند. ما توابع استاندارد لینوکس را بازنویسی کرده و به توابع خودمان متصل کردیم. توابع کلیدی پیاده‌سازی شده عبارتند از:

الف) دریافت ویژگی‌های پرونده (do_getattr):

این تابع حیاتی‌ترین بخش برای دستوراتی مثل ls -l است. وقتی سیستم‌عامل اطلاعات یک فایل را می‌خواهد، این تابع در دیتابیس ما جستجو می‌کند. اگر مسیر / باشد، آن را به عنوان دایرکتوری (S_IFDIR) معرفی می‌کند. اگر یک فایل باشد، آفست آن را پیدا کرده و آن را به عنوان فایل عادی (S_IFREG) با مجوزهای خواندن/نوشتن (0666) به سیستم‌عامل معرفی می‌کند.

ب) **خواندن دایرکتوری** (do_readdir):

این تابع مسئول پاسخ‌دهی به دستور ls است. این تابع ابتدا دایرکتوری‌های استاندارد . و .. را اضافه می‌کند. سپس تابع fs_list_files را فراخوانی می‌کند تا تمام فایل‌های فعال در دیتابیس را استخراج کرده و به بافر نمایش FUSE تحویل دهد.

ج) **خواندن و نوشتن** (do_write, do_read):

این توابع برای دستوراتی مثل cat و echo استفاده می‌شوند.

do_read: درخواست خواندن را از کرنل می‌گیرد، مکان فایل را در دیسک پیدا کرده و داده‌ها را از آفست مشخص شده به بافر کاربر منتقل می‌کند.

do_write: داده‌های کاربر را گرفته و دقیقاً در موقعیت دیتای فایل در filesys.db می‌نویسد.

د) **توابع تکمیلی و رفع خطاها:**

برای جلوگیری از خطای Function not implemented، توابع زیر نیز پیاده‌سازی شدند:

do_create: برای دستور touch. فایل را در حالت ایجاد باز می‌کند.

do_truncate: برای دستوراتی که فایل را بازنویسی می‌کنند (مثل echo > file) سایز فایل را تغییر می‌دهد.

do_utimens: یک تابع خالی (Dummy) برای مدیریت درخواست‌های تغییر زمان فایل (که توسط touch ارسال می‌شود) تا از بروز خطا جلوگیری شود.

3. استراتژی‌های آزمون و ارزیابی:

برای اطمینان از عملکرد صحیح، سیستم در دو سطح متفاوت مورد آزمون قرار گرفت:

سطح 1: آزمون واحد و مستقل (Unit Testing - test_standalone.c):

این فایل تست، منطق داخلی (my_fs.c) را بدون دخالت FUSE و عملیات ماونت بررسی می‌کند. اهمیت این تست در ایزوله کردن باگ‌های منطقی از باگ‌های سیستمی است.

شبیه‌سازی FUSE: تابعی به نام mock_filler نوشته شد تا رفتار تابع readdir در FUSE را شبیه‌سازی کند و لیست فایل‌ها را در خروجی چاپ کند.

```
→ MyFUSE ./test_fs.sh
---  Starting the FUSE file system testing process  ---
1. Cleaning up previous runs...
2. Compiling...
Compilation successful.
3. Mounting filesystem...
Disk not found. Creating new disk (Fixed 2MB)...
Filesystem mounted successfully.
4. Testing File Creation (touch)...
Pass: File created.
5. Testing Write (echo)...
Pass: Write operation successful.
6. Testing Read (cat)...
./test_fs.sh: line 65: warning: command substitution: ignored null byte in input
Pass: Read data matches written data (Hello_FUSE_World).
7. Testing Directory List (ls)...
Pass: File found in list.
8. Testing Delete (rm)...
Pass: File deleted.
9. Persistence Test...
Pass: File created.
Pass: Write operation successful.
9. Unmounting and Cleaning up...
--- ALL TESTS PASSED SUCCESSFULLY ---
```

سطح 2: آزمون یکپارچگی و پایداری (Integration Scripts):

این آزمون‌ها از طریق اسکریپت‌های Bash و در محیط واقعی لینوکس انجام شدند:

1. test_fs.sh: سیستم فایل را ماونت می‌کند. دستورات واقعی شل (touch, echo, cat, ls, rm) را روی دایرکتوری ماونت شده اجرا می‌کند و در نهایت خروجی هر دستور را با خروجی مورد انتظار مقایسه کرده و در صورت مغایرت، خطا می‌دهد.

```
→ MyFUSE ./test_standalone
--- Starting Standalone FS Test ---

1. Initializing File System...
Disk not found. Creating new disk (Fixed 2MB)...
[FAIL] filesys.db size is incorrect. Got: 2097152
→ MyFUSE gcc test_standalone.c my_fs.c -o test_standalone -D_FILE_OFFSET_BITS=64 $(pkg-config fuse --cflags --libs)
→ MyFUSE ./test_standalone
--- Starting Standalone FS Test ---

1. Initializing File System...
Disk not found. Creating new disk (Fixed 2MB)...
[PASS] filesys.db created with correct size (10MB).

2. Creating 'unit_test.txt'...
[PASS] File 'unit_test.txt' created/opened successfully.

3. Writing data...
[PASS] Write operation completed.

4. Reading data back...
   Read content: 'Standalone Logic Check 123'
[PASS] Read data matches written data.

5. Listing files (using mock filler)...
   [Found File]: unit_test.txt
[PASS] List files executed.

6. Deleting 'unit_test.txt'...
[PASS] File successfully deleted (cannot open anymore).

7. Shutting down...

--- ALL STANDALONE TESTS PASSED ---
→ MyFUSE
```

2. test_persistence.sh (تست ماندگاری): بررسی می‌کند که آیا اطلاعات پس از unmount شدن و اجرای مجدد برنامه (Mount دوباره)، همچنان در filesys.db باقی می‌مانند یا خیر. این تست تایید کرد که سیستم‌فایل ما فرار (Volatile) نیست و داده‌ها را به درستی روی دیسک ذخیره می‌کند.

```
→ MyFUSE ./test_persistence.sh
--- Starting the Persistence Test ---
1. Ensuring filesystem is unmounted...
2. Re-compiling...
3. Mounting existing filesystem...
Filesystem mounted successfully.
4. Checking for 'hello.txt' from previous run...
Pass: Old file 'hello.txt' found!
./test_persistence.sh: line 57: warning: command substitution: ignored null byte in input
Fail: Content mismatch! Got 'Hello_FUSE_World
q◆', expected 'Hello_FUSE_World'.
5. Adding a NEW file (append test)...
Pass: New file created successfully alongside old data.
Current files in filesystem:
total 0
-rw-rw-rw- 1 root root 1024 Jan  1  1970 hello.txt
-rw-rw-rw- 1 root root 1024 Jan  1  1970 new_persistent.txt
6. Unmounting...
--- PERSISTENCE TEST PASSED ---
Your filesystem successfully remembered data across restarts!
→ MyFUSE
```