Sukhman Kaur

Laboratory Exercise 4

cpe 453 Spring 2025

Task 1: Build a Modified Kernel

Approach

To make sure I could revert any changes if needed, I started by creating a backup of the entire kernel source directory. I navigated to /usr/src and used the tar command to create a backup file named kernel_backup.tar. This allowed me to restore the original kernel source if my modifications caused any issues. Additionally, I made an individual backup of proc.c, with cp proc.c proc.c.backup which was the specific file I needed to modify.

After setting up these backups, I located the idle() function in proc.c. This function controls the system's idle state. I added the following code before the halt_cpu() function to make an "@" symbol appear whenever the system was genuinely idle:

```
if (1) {
   printf("@\b");
}
```

Initially, I had placed the code after halt_cpu(), but I realized that this was incorrect because the CPU is halted at that point, and no further instructions are executed. Moving the code before halt_cpu() ensured that the symbol would print before the processor entered its halted state.

Once I made this change, I navigated to /usr/src/tools and ran the make hdboot command to compile the modified kernel. After the build completed successfully, I rebooted the system using the shutdown -r now command. At the boot menu, I selected "Start Custom MINIX 3" by typing 1 and logged in as the root user to test the modification.

Problems Encountered

During this process, I encountered several challenges. Initially, I placed the printf("@\b"); statement after halt_cpu(), which resulted in the symbol never appearing because the CPU was halted and no further instructions were executed. Additionally, I encountered syntax errors in proc.c and discovered that the TRUE constant, which I had initially used, was not defined in the kernel. This caused a compilation error. Furthermore, while testing the kernel in QEMU, I noticed that the system would sometimes hang during shutdown, requiring a manual restart.

Solutions

To address these issues, I first moved the printf("@\b"); statement before halt_cpu() to ensure the code executed properly. I also reviewed the syntax errors in proc.c using the vi editor and replaced TRUE with 1, which resolved the compilation error. To handle the QEMU shutdown hang, I used Ctrl+Alt+G to release the keyboard grab, closed the QEMU window, and restarted the emulator manually.

Lessons Learned

This task taught me how important it is to understand the execution flow of kernel-level functions. Placing code after halt_cpu() was a mistake that helped me realize the importance of carefully considering the behavior of each function. I also learned that creating backups is essential for kernel development, as they allowed me to recover quickly from mistakes. Testing kernel modifications required patience, and I now better understand that the idle state occurs only when no processes are runnable, not just during moments of low activity. This experience highlighted the need for careful planning and thorough testing in kernel programming.

Task 2: Test the Modified Kernel

Approach

To verify that the "@" symbol only appeared during genuine idle periods, I designed three tests:

1. CPU-Intensive Test
    a. Command: while true; do let i=i+1; done
    b. Expected Result: The "@" symbol should not appear during computation.
    c. Observations:
        i. - During computation: No "@" symbol appeared while CPU was busy
        ii. - After stopping (Ctrl+C): "@" symbol reappeared when system returned to idle
2. Active System Test
    a. Command: find /
    b. Expected Result: No "@" symbol during file system search.
    c. Observations:
        i. During search: No "@" symbol appeared during file system traversal
        ii. After completion: "@" symbol reappeared when system became idle
3. Mixed Load Test
    a. Command: ls -R /usr
    b. Expected Result: No "@" symbol during listing, appearance only after completion.
    c. Observations:
        i. During listing: No "@" symbol during directory content listing

ii. Between operations: No "@" symbol during brief pauses
iii. After completion: "@" symbol appeared when system returned to idle

Problems Encountered

Initially, it was challenging to distinguish true idle states from brief pauses during command execution. The rapid appearance and disappearance of the "@" symbol made it difficult to track its behavior consistently.

Solutions

To resolve these issues, I used commands like sleep and CPU-intensive loops to establish clear baselines for idle and active states. I also followed a structured observation plan to carefully document the system's behavior before, during, and after each test. This approach helped me confirm that the "@" symbol appeared only during genuine idle periods.

Lessons Learned

This process taught me the importance of designing test scenarios that accurately reflect the system's behavior. I learned that idle states occur only when no processes are runnable, not just during pauses in activity. A structured test plan made it easier to track behavior and verify that the modification worked as intended. Overall, testing kernel modifications required patience, attention to detail, and a methodical approach.