HW Project #1: Introduction to Linux Kernel Programming
Real-Time Embedded System Assignment – CS251

Shixun Wu (swu264) **Anvaya B Narappa (an001)** **Simran Saha (ssaha031)**

Write-Up question:

1. How does a system call execute? Explain the steps from making the call in the user space process to returning from the call with a result.

A system call is a request made by a process for a service provided by the operating system. The process making the system call executes a specific instruction, such as the "int" instruction on x86-based systems, which triggers a switch from user mode to kernel mode. The operating system's kernel then handles the system call, performs the requested service, and returns control to the process, along with any result.

The specific steps involved:

- The process makes the system call by executing a specific instruction, such as "int" on x86-based systems.
- The instruction triggers a switch from user mode to kernel mode, allowing the kernel to handle the system call.
- The kernel retrieves the system call number and any arguments passed by the process from a designated location in memory, such as a register or system call table.
- The kernel uses the system call number to look up and execute the corresponding system call handler function, which performs the requested service.
- The kernel stores any result of the system call in a designated location in memory, such as a register or memory location pointed to by the process.
- The kernel switches back to user mode and returns control to the process, allowing it to continue execution with the result of the system call.

2. What does it mean for a kernel to be preemptive? Is the Linux kernel you are hacking on preemptive?

A preemptive kernel permits the scheduling of numerous processes to run concurrently and has the ability to pause the execution of one process to allow another to begin. In contrast, a non-preemptive kernel only executes one process at a time and doesn't stop it from running until the process voluntarily relinquishes control.

Because the Linux kernel is preemptive, it has the ability to halt the operation of one process to allow for the execution of another. As a result, resources can be used more effectively, and no one operation can monopolize the CPU.

The Completely Fair Scheduler (CFS), which has been the default scheduler since Linux 2.6, is the most popular preemption setting for the Linux kernel.

3. When does access to data structures in user space applications need to be synchronized?

When different threads or processes may be accessing the same data structure concurrently, access to the data structure in user space applications must be synchronized. A lack of synchronization could lead to inconsistent and unpredictable behavior since many threads or processes could read from and write to the data structure at the same time.

4. What synchronization mechanisms can be used to access shared kernel data structures safely? List them.

To access shared kernel data structures in a secure manner, a variety of synchronization techniques are available:

- Spinlocks: A spinlock is a lock that spins while it is busy waiting to be released. When there is little chance of lock contention and a brief hold period, spinlocks are employed in the kernel to safeguard shared data structures.
- A semaphore is a synchronization object that regulates access to a shared resource by numerous processes or threads. When the contention for the resource is anticipated to be high and the hold time is anticipated to be prolonged, semaphores can be employed to safeguard shared kernel data structures.
- Similar to a semaphore, mutexes are often employed to safeguard shared data structures within a single process. When it is anticipated that there will be little resource congestion and a brief hold time, mutexes can be utilized to safeguard shared kernel data structures.
- Read-write locks: A read-write lock prevents more than one writer from simultaneously accessing a shared data structure while allowing multiple readers to do so. This can be helpful when there is a lot of competition for the resource and many reads are anticipated, but writes are predicted to be infrequent.
- RCU (Read-Copy-Update): RCU is a synchronization mechanism that permits multiple readers to access a shared data structure at the same time and also permits updates to the data structure. However, it accomplishes these tasks by making a copy of the data structure, updating the copy, and then atomically swapping the updated copy in place of the original.
- Atomic operations: These methods let you carry out specific operations in a way known as an atomic operation. This means that the operation is carried out as a single, indivisible instruction, preventing multiple accesses to the same piece of data.
- Disabling interrupts when accessing a shared data structure, preventing concurrency from other sources.

5. Take a look at the container of a macro defined in include/linux/kernel.h. In rough terms, What does it do and how is it implemented?

The container of macro can be used to locate the address of the structure that contains a specific field in a given structure. It requires two arguments: the type of the container structure and a pointer to a field inside that structure. After that, it determines the address of the contained structure by deducting the field's offset from the structure.

The offset of macro, a common macro specified in stddef. h that determines the offset of a field within a structure, is used to implement it. By taking the address of the field pointer and subtracting the offset, the container of macro uses this offset to get the address of the contained structure.

6. Give a brief summary of the contributions of each member.

All the members of the group worked together towards the completion of the assignment by frequent meetings over zoom.
Simran Saha, worked towards 4.1 (hello world kernel module) and the writeup
Anvaya B Narappa worked towards 4.2 (virtual devices)
Shixun Wu worked towards 4.3 (system calls)