CS251/EE255: Real-Time Embedded System

Names: Akash Suresh Bilgi(862395080), Anish More(862324523),
Ritesh Singh(862395274)

HW #1 Report

Q. How does a system call execute? Explain the steps from making the call in the userspace process to returning from the call with a result.

Ans. System call is used to request a service from the kernel. The steps for executing a system call is described below:-

System call initialization- The userprocess process makes a system call by executing specific assembly instruction such as 'syscall' or 'int 0x80', this instructions triggers a software interrupt and the control is transferred to the kernel

Interrupt handling- the kernel identifies the syscall being called by the interrupt number or systemcall number passed.

System call dispatch- the kernel sends the syscall to appropriate handler where it implements the requested services

System call execution- system call handler execute the requested service and returns the value for system call

Return to Userspace- the control returned to the userspace process after returning the value from the interrupt Handler.

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

Ans. A kernel is called preemptive if it allows the scheduler to temporarily preempt or interrupt a running process to allow other processes to run. The linux kernel used for this homework is a preemptive kernel, meaning we can interrupt any running task and switch to any other task at any time.

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

Ans. Access to data structures in user space applications needs to be synchronized when multiple thread or processes are accessing the same data structure simultaneously. This is done to prevent the race conditions And ensure that the data is consistent for each process.

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

Ans. Following synchronization mechanisms can be used to access the shared kernel data structure safely-

1. Spinlock
2. Mutexes
3. Semaphores
4. Read-Write lock
5. Read-Copy-Update(RCU)

Q. Take a look at the `container_of` macro defined in `include/linux/kernel.h`. In rough terms, what does it do and how is it implemented?

Ans. The container_of macro defined in include/linux/kernel.h is used when given a pointer to a field inside a containing structure, this function finds the address of that structure.
The implementation of container_of typically involves some pointer arithmetic and type casting. It works by Subtracting the offset of the desired field within the structure from the address of the field and then Type-casting the result back to a pointer to the containing structure.

```
((type *)(__mptr - offsetof(type, member)));
```

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

Ans. Akash Suresh Bilgi - Hello world Kernel Module(4.1), System Calls(4.3)

Anish More - Virtual device(4.2), System Calls(4.3)

Ritesh Singh - Hello world Kernel Module(4.1), Virtual Device(4.2)

We all collaborated together from time to time to debug the code whenever we faced error and contributed equally for the writeup as well.