EE-255 Real Time Embedded Systems Project 1

Writeup

(Group No- 17)

TEAM MEMBERS:

NAME	STUDENT ID	NET ID
Srushti Hembade	862395839	shemb001
Piyush kanadje	862393475	pkana006

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

➔ **Trap to kernel mode**: The process makes a system call that results in a software interrupt. Processes can perform privileged operations by switching from user mode to kernel mode using the CPU hardware.

➔ **System call handler**: When an interrupt is triggered, the control is transferred to the operating system's interrupt handler, which then directs it to the system call handler, a specific section of the kernel code.

➔ **System call dispatch:** The system call handler selects the appropriate service based on the system call number provided as an argument and directs control to the corresponding kernel routine that performs the service.

➔ **Service execution**: When a service request is made, the kernel procedure responds by typically accessing system resources and delivering data to the user process via the proper system call arguments.

➔ **Return from system call**: The interrupt handler receives control after the kernel routine passes control back to the system call handler. The system call returns to the user process with the service's outcome once the interrupt handler restores the process's previous mode (user or kernel).

➔ **User process continuation**: The outcome of the system call is used to continue the user process's execution.

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

- ➔ An operating system kernel known as a "preemptive kernel" permits many processes to run concurrently and has the ability to pause the execution of one process in favor of another that has a higher priority. This means that the kernel can transfer control to another process while a process is still running, even if the first process did not freely release control.
- ➔ It is true that the Linux kernel is preemptive. This means that, if necessary, the Linux kernel can halt the operation of one process and launch another, for example, to guarantee the system's responsiveness or the successful performance of real-time activities.

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

- ➔ When several threads or processes simultaneously access and alter the same data structures, access to such data structures in userspace programs must be synchronized. Inappropriate synchronization might result in race circumstances and unpredictable outcomes.
- ➔ A variety of synchronization methods, including semaphores, mutexes, monitors, and critical sections, can be used in userspace applications to synchronize access to data structures. Depending on the particular needs of the application, such as the level of concurrency, the degree of data sharing, and the required responsiveness, the best synchronization mechanism will be determined.

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

- ➔ Spinlocks:To prevent concurrent access to shared data structures, synchronization techniques called spinlocks are employed. A lock that is retained in a spinning position while it is locked is called a spinlock. When there is no longer any access to the data structure, the lock is released.
- ➔ Mutexes: To make sure that only one thread at a time can access a shared data structure, synchronization techniques called mutexes, or mutual exclusion, are used. The use of mutexes helps to maintain data consistency and protect shared data structures from concurrent access.
- ➔ Read-Write locks: Access to a shared data structure is controlled by a read-write lock, a synchronization mechanism. One thread can only edit data at a time while using a read-write lock, which enables several threads to read a shared data structure simultaneously.
- ➔ Semaphores: An access control method for a shared data structure is called a semaphore. Semaphores are used to prevent concurrent access to shared data

structures and to guarantee that only a limited number of threads can access the data at once.

➔ Atomic operations: By using atomic operations, shared data structures can be accessed and modified in a way that prevents concurrent access. Data consistency is maintained even in the face of concurrent access thanks to atomic operations, which are carried out as a single, indivisible operation.

5. **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?**

➔ The Linux kernel's container_of macro can be used to locate the address of a structure that holds a specific field. Finding the parent structure of a field that is referenced by a pointer is frequently used in the Linux kernel.

➔ The macro takes three arguments:
ptr: A pointer to the field within the structure.
type: The type of structure that contains the field.
member: The name of the field within the structure.

➔ To compute the member field's offset within the type structure, the container of macro uses the offsetof macro. The offsetof macro, which returns a field's offset in bytes within a structure, is specified in the stddef.h header file.

➔ The container of macro subtracts the offset after calculating it from the address of the ptr argument, which is a pointer to the field in the structure. The structure's address, which houses the field, is the outcome.

➔ With a pointer to a struct device driver, the container of macro can be used in the Linux kernel to access the parent structure of a field, such as a struct device.

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

WORK	PIYUSH	SRUSHTI
SETUP	☑	☑
4.1.1	☐	☑
4.1.2	☐	☑
4.2.1	☑	☐
4.2.2	☑	☐
4.3.1	☑	☑

4.3.2	☑	☑
4.3.3	☑	☑
4.4	☐	☑