OPERATING SYSTEM DESIGN

S Thenmozhi

Department of Computer Applications

OPERATING SYSTEM DESIGN

OS Structures & Kernel Programming

S Thenmozhi
Department of Computer Applications

Kernel Headers

1. linux/module.h

- This header provides macros and functions for loading and unloading kernel modules. It includes definitions for module initialization, cleanup, and module metadata.

Important Methods:

- MODULE_AUTHOR() - Sets the author of the module.
- MODULE_DESCRIPTION() - Sets a description for the module.
- MODULE_LICENSE() - Sets the license for the module.

Kernel Headers

2. linux/kernel.h

- This header contains fundamental kernel macros and functions, such as printk for logging messages to the kernel buffer.

Important Methods

- printk() - logging mechanism in the Linux kernel.
- It allows kernel developers to print messages to the kernel log buffer,
- It can be viewed using the dmesg command or by checking log files like /var/log/kern.log.

Syntax: int printk(const char *format, ...);

Kernel Headers

KERN_EMERG: Emergency messages, system is unusable.

KERN_ALERT: Action must be taken immediately.

KERN_CRIT: Critical conditions.

KERN_ERR: Error conditions.

KERN_WARNING: Warning conditions.

KERN_NOTICE: Normal but significant condition.

KERN_INFO: Informational messages.

KERN_DEBUG: Debug-level messages.

Example: printk(KERN_INFO "Hello, Kernel World!\n");

Kernel Headers

3. linux/init.h

- This header is used for module initialization and cleanup functions. It defines macros like module_init and module_exit.

Important Methods

- module_init() - Marks the initialization function for a module.
- module_exit() - Marks the cleanup function for a module.

4. linux/sched.h

- This header provides access to scheduling functions and process-related structures, such as the task structure.

Important Methods

- schedule() - Invokes the scheduler.
- task_waking() - Marks a task as waking up.
- task_forked() - Marks a task as forked.

5. linux/timer.h

- This header includes functions and macros for working with timers within the kernel, such as setting up and managing timer callbacks.

Important Methods

- setup_timer() - Sets up a timer
- del_timer() - Deletes a timer
- mod_timer() - Modifies a timer

6. linux/pthread.h

- This header file is part of the POSIX standard and provides an interface for creating and managing threads in user-space.

Important Methods

- pthread_create(): Creates a new thread.
- pthread_join(): Waits for a thread to terminate.
- pthread_exit(): Terminates the calling thread.
- pthread_cond_wait(): Waits on a condition variable.
- pthread_cond_signal(): Signals a condition variable.

7. linux/kthread.h

- This header provides a simple interface for creating and managing kernel threads.

Important Methods

- kthread_create(): Creates a kernel thread on the current node.
- kthread_run(): Creates and wakes up a kernel thread.
- kthread_stop(): Stops a kernel thread.
- kthread_should_stop(): Checks if the thread should stop.
- kthread_bind(): Binds a kernel thread to a specific CPU.

8. linux/spinlock.h

- This header provides functions and macros for spinlock synchronization primitives, essential for protecting shared data in a multitasking environment.

Important Methods

- spin_lock() - Acquires a spinlock.
- spin_unlock() - Releases a spinlock.
- spin_trylock() - Tries to acquire a spinlock without blocking.

9. linux/slab.h

- This header contains functions for memory allocation within the kernel, such as kmalloc and kfree.

Important Methods

- kmem_cache_create() - Creates a cache for kernel objects.
- kmem_cache_alloc() - Allocates an object from a cache.
- kmem_cache_free() - Frees an object back to a cache.

10. linux/fs.h

- This header includes definitions for interacting with the filesystem, such as file operations and inode structures. It's essential for writing file system modules or device drivers.

Important Methods

- vfs_read() - Reads data from a file.
- vfs_write() - Writes data to a file.
- open() - Opens a file.
- close() - Closes a file.

11. linux/cdev.h

- This header provides the character device framework, including functions and macros for registering and managing character devices.

Important Methods

- cdev_add() - Adds a character device to the system.
- cdev_del() - Removes a character device from the system.

Kernel Data Structures

12. linux/errno.h

- This header defines standard error codes used within the kernel

Important Error Macros

- EIO:Input/output error
- ENODEV:No such device
- ENOMEM:Not enough space
- EISDIR:Is a directory
- EMFILE:Too many open files
- EEXIST:File exists
- EPERM:Operation not permitted

THANK YOU

S Thenmozhi

Department of Computer Applications

thenmozhis@pes.edu

+91 80 6666 3333 Extn 393