**IPC: Interrupts and Signals**

**Subject - Unix Operating System**

**Name – Hemant Sharma**

**PRN – 22610001 Class – TYIT**

**Assignment No – 2e**

**Title-** Write application of signal handling in Linux OS and program any one.

**Objectives-**

1. To learn about IPC through signal.
2. To know the process management of Unix/Linux OS
3. Use of system call to write effective application programs

**Theory-**

Signal handling is a crucial mechanism in Linux operating systems that allows processes to handle asynchronous events like interrupts, exceptions, or notifications. Signals provide a way to communicate between the kernel and user space or between processes, enabling responsive and robust applications.

Common applications of signal handling include:  
1. Gracefully terminating processes (e.g., SIGTERM).  
2. Handling user interrupts (e.g., SIGINT from Ctrl-C).  
3. Implementing timers (e.g., SIGALRM).  
4. Debugging and logging.  
5. Synchronization between processes.

The following example demonstrates how to handle the SIGTERM signal to gracefully terminate a running process. The program sets up a custom signal handler that performs cleanup operations before exiting.

**Program-**

#include <stdio.h>  
#include <stdlib.h>  
#include <signal.h>  
#include <unistd.h>  
  
// Signal handler for SIGTERM  
void handle\_sigterm(int signum) {  
 printf("\nReceived SIGTERM (signal %d). Cleaning up and exiting...\n", signum);  
 // Perform cleanup tasks here  
 exit(0);  
}  
  
int main() {  
 // Set up the SIGTERM handler  
 if (signal(SIGTERM, handle\_sigterm) == SIG\_ERR) {  
 perror("Error setting up SIGTERM handler");  
 exit(EXIT\_FAILURE);  
 }  
  
 printf("Process running. Send SIGTERM to terminate gracefully.\n");  
  
 // Simulate a long-running process  
 while (1) {  
 printf("Working...\n");  
 sleep(2);  
 }  
  
 return 0;  
}

**Conclusion-**

This example demonstrates how to use signal handling to manage the SIGTERM signal in Linux. The program registers a custom signal handler to clean up resources and exit gracefully when the signal is received. Signal handling provides a robust mechanism for process control and enhances the reliability of applications.