**IPC: Interrupts and Signals**

**Subject - Unix Operating System**

**Name – Hemant Sharma**

**PRN – 22610001 Class – TYIT**

**Assignment No – 2d**

**Title-** Write application or program to trap a ctrl-c but not quit on this signal.

**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-**

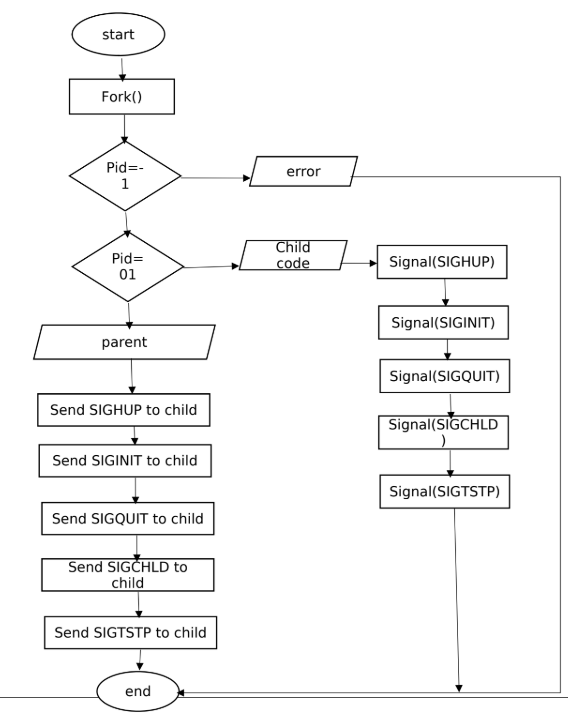
This program demonstrates how to trap the Ctrl-C (SIGINT) signal in a Linux environment without quitting the program. The program intercepts the signal using a custom signal handler and performs an action instead of terminating the process.

This functionality is useful in scenarios where you need to ensure graceful handling of interruptions or prevent the accidental termination of critical processes.

**How it works-**

1. Signal Handling: A custom signal handler function is defined to manage the SIGINT signal.
2. Signal Interception: The signal() function is used to intercept the SIGINT signal and associate it with the custom handler.
3. Custom Behavior: When the user presses Ctrl-C, the custom handler executes, printing a message and allowing the program to continue running.

**Flowchart-**

****

**Program-**

#include <stdio.h>  
#include <stdlib.h>  
#include <signal.h>  
#include <unistd.h>  
  
// Custom signal handler for SIGINT  
void handle\_sigint(int signum) {  
 printf("\nCaught signal %d (Ctrl-C). Continuing execution...\n", signum);  
}  
  
int main() {  
 // Set up the custom signal handler  
 if (signal(SIGINT, handle\_sigint) == SIG\_ERR) {  
 perror("Error setting up signal handler");  
 exit(EXIT\_FAILURE);  
 }  
  
 printf("Program running. Press Ctrl-C to test signal handling.\n");  
  
 // Keep the program running  
 while (1) {  
 printf("Working...\n");  
 sleep(2);  
 }  
  
 return 0;  
}

**Conclusion-**

This program demonstrates how to trap the Ctrl-C signal (SIGINT) and handle it gracefully without terminating the process. By using a custom signal handler, the program can override the default behavior and implement custom logic for handling interruptions. This technique is commonly used in robust applications that require reliable signal management.