Practical - 7

Aim : Write a program to demonstrate the handling of the signals: SIGINT,SIGALRM & SIGQUIT.

Code:

```
#include<bits/stdc++.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h> // sigaction(), sigsuspend(), sig*()
#include <unistd.h> // alarm()
using namespace std;


/* How to use?
 *
 * First, compile and run this program:
 *     $ g++ signal.cpp
 *     $ ./a.out
 *
 * It will print out its pid. Use it from another terminal to send signals
 *     $ kill -s signal <pid>
 *     For example,
 *     $ kill -s SIGQUIT <pid>
 *
 * Exit the process with ^C ( = SIGINT) or ^\ (= SIGQUIT)
 */


/*
 *  Please note that printf et al. are NOT safe to use in signal handlers.
 *  Look for async safe functions.
 *  http://man7.org/linux/man-pages/man7/signal-safety.7.html
 */

void handle_signal(int signal) {
    // Find out which signal we're handling
    switch (signal) {
        case SIGINT:
            printf("Caught SIGINT!\n");
            break;
        case SIGQUIT:
            printf("Caught SIGQUIT, terminating!\n");
            exit(0);
        default:
            fprintf(stderr, "Caught wrong signal: %d\n", signal);
            return;
    }
}

void handle_sigalrm(int signal) {
    if (signal != SIGALRM) {
        fprintf(stderr, "Caught wrong signal: %d\n", signal);
```

```
    }

    printf("Got sigalrm, do_sleep() will end\n");
}

void do_sleep(int seconds) {
    struct sigaction sa;
    sigset_t mask;

    sa.sa_handler = &handle_sigalrm; // Intercept and ignore SIGALRM
    sa.sa_flags = SA_RESETHAND; // Remove the handler after first signal
    sigfillset(&sa.sa_mask);
    sigaction(SIGALRM, &sa, NULL);

    // Get the current signal mask
    sigprocmask(0, NULL, &mask);

    // Unblock SIGALRM
    sigdelset(&mask, SIGALRM);

    // Wait with this mask
    alarm(seconds);
    sigsuspend(&mask);

    // printf("sigsuspend() returned\n");
}
int main() {
    struct sigaction sa;

    // Print pid, so that we can send signals from other terminals
    printf("My pid is: %d\n", getpid());

    // Setup the sighub handler
    sa.sa_handler = &handle_signal;

    // Restart the system call, if at all possible
    sa.sa_flags = SA_RESTART;

    // Block every signal during the handler
    sigfillset(&sa.sa_mask);


    // Will always fail, SIGKILL is intended to force kill your process
    if (sigaction(SIGKILL, &sa, NULL) == -1) {
        perror("Cannot handle SIGKILL"); // Will always happen
        printf("You can never handle SIGKILL anyway...\n");
    }

    if (sigaction(SIGINT, &sa, NULL) == -1) {
        perror("Error: cannot handle SIGINT"); // Should not happen
    }
```

```
    if (sigaction(SIGQUIT, &sa, NULL) == -1) {
        perror("Error: cannot handle SIGQUIT"); // Should not happen
    }
    while(1) {
        printf("\nSleeping for ~3 seconds\n");
        do_sleep(3); // Later to be replaced with a SIGALRM
    }
    return 0;
}
```

Output:

```
bhishm@BhishmDaslaniya: /media/bhishm/Projects/Internals_of_Operating_System_Lab/Pr
bhishm@BhishmDaslaniya: /media/bhishm/Projects/Internals_of_Operating_System_Lab/Practical_7 93x22
bhishm@BhishmDaslaniya:/media/bhishm/Projects/Internals_of_Operating_System_Lab/Practical_7$
./a.out
My pid is: 13088
Cannot handle SIGKILL: Invalid argument
You can never handle SIGKILL anyway...

Sleeping for ~3 seconds
Got sigalrm, do_sleep() will end

Sleeping for ~3 seconds
Caught SIGINT!

Sleeping for ~3 seconds
Got sigalrm, do_sleep() will end

Sleeping for ~3 seconds
Got sigalrm, do_sleep() will end

Sleeping for ~3 seconds
Caught SIGQUIT, terminating!
bhishm@BhishmDaslaniya:/media/bhishm/Projects/Internals_of_Operating_System_Lab/Practical_7$
```

```
bhishm@BhishmDaslaniya: /media/bhishm/Projects/Internals_of_Operating_System_Lab/Practical_7 93x8
bhishm@BhishmDaslaniya:/media/bhishm/Projects/Internals_of_Operating_System_Lab/Practical_7$
kill -s SIGINT 13088
bhishm@BhishmDaslaniya:/media/bhishm/Projects/Internals_of_Operating_System_Lab/Practical_7$
kill -s SIGQUIT 13088
bhishm@BhishmDaslaniya:/media/bhishm/Projects/Internals_of_Operating_System_Lab/Practical_7$
```

Conclusion: From this practical I have learnt about different types of signals and how to handle/use it in program for better execution.