```
/*
Name: John Robertson
BlazerId: jprob
Lab #: 9
To compile: gcc lab9.c
To run: a.out <command> [args]
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>

// Globals
pid_t pid;

static void sig_handle_z(int signo) {
    signal(signo, SIG_IGN);
    fflush(stdout);
    kill(pid, SIGTSTP);
    printf("child process with pid=%ld stopped\n", (long)pid);
    fflush(stdout);
    signal(signo, sig_handle_z);
}

static void sig_handle_c(int signo) {
    signal(signo, SIG_IGN);
    fflush(stdout);
    kill(pid, SIGKILL);
    printf("child process with pid=%ld killed\n", (long)pid);
    fflush(stdout);
    signal(signo, sig_handle_c);
}

int main(int argc, char **argv) {
    if (argc < 2) {
        printf("Usage: %s <command> [args]\n", argv[0]);
        exit(-1);
    }

    int status;
    signal(SIGINT, sig_handle_c);
    signal(SIGTSTP, sig_handle_z);
    pid = fork();
    if (pid == 0) {
        execvp(argv[1], &argv[1]);
        perror("execvp");
        exit(-1);
    }
    else if (pid > 0) {
        printf("Wait for the child process to terminate\n");
        wait(&status); /* wait for the child process to terminate */
        if (WIFEXITED(status)) { /* child process terminated normally */
            printf("Child process exited with status = %d\n", WEXITSTATUS(status));
        } else { /* child process did not terminate normally */
            printf("Child process did not terminate normally!\n");
            /* look at the man page for wait (man 2 wait) to determine
               how the child process was terminated */
                        for ( ; ; )
                                pause();
        }
    }
```

```
    printf("[%ld]: Exiting prog.\n", (long)getpid());

    return 0;
}
```