```
/* Simple program to illustrate the use of fork-exec-wait pattern.
 * This version uses execvp and command-line arguments to create a new process.
 * To Compile: gcc -Wall forkexecvp.c
 * To Run: ./a.out <command> [args]
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>


pid_t pid;
static void sig_usr(int signo)
{ /// Sighandler.c
    switch (signo)
    {
    case SIGINT:
        kill(pid, SIGKILL);
        printf("received SIGINT signal %d\n", signo);
        break;
    case SIGQUIT:
        printf("received SIGQUIT signal %d\n", signo);
        kill(0, SIGTERM); //sigint.c
        break;
    case SIGTSTP:
        kill(pid, SIGTSTP);
        printf("received SIGTSTP signal %d\n", signo);
        break;
    default:
        printf("received signal %d\n", signo);
    }
}

int main(int argc, char **argv)
{

    if (argc < 2)
    {
        printf("Usage: %s <command> [args]\n", argv[0]);
        exit(-1);
    }
    if (signal(SIGINT, sig_usr) == SIG_ERR)
    {
        printf("can’t catch SIGINT\n");
        exit(-1);
    }
    if (signal(SIGTSTP, sig_usr) == SIG_ERR)
    {
        printf("can’t catch SIGTSTP\n");
        exit(-1);
    }
    if (signal(SIGQUIT, sig_usr) == SIG_ERR)
    {
        printf("can’t catch SIGINT\n");
        exit(-1);
    }
    int status;
    // forkexecvp.c
    pid = fork();
    if (pid == 0)
    { /* this is child process */
```

```
        execvp(argv[1], &argv[1]);
        printf("If you see this statement then execl failed ;-(\n");
        exit(-1);
    }
    else if (pid > 0)
    { /* this is the parent process */
        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");
            for(; ;);
            pause();

        }
    }
    else
    {                  /* we have an error */
        perror("fork");

        exit(EXIT_FAILURE);
    }

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

    return 0;
}

// http://www2.cs.uidaho.edu/~krings/CS270/Notes.S10/270-F10-26.pdf
// https://stackoverflow.com/questions/13273836/how-to-kill-child-of-fork
```