```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void skipblank()
{
      char c = ' ';
      while ((c = getchar()) == ' ' || c  == '\t') ;
      ungetc(c, stdin);
}


int main()
{
      while(1)
      {
            char E_name[101];
            // char arg[101][101];
            char **arg;
            arg = (char **)malloc(sizeof(char *) * 101);

            printf(">");

            scanf("%s", E_name);
            arg[0] = (char*)malloc(sizeof(E_name));
            strcpy(arg[0], E_name);    //first argument should be process name

            int i, j = 1;
            char c;
            arg[j] = (char*)malloc(sizeof(char)*101);
            skipblank();
            for(i = 0; i < 100 && j < 100 && (c = getchar()) != '\n'; i++)
            {
```

```
//printf("111111=%c.\n", c);

                if(c == ' ' || c == '\t')
                {
                        arg[j][i] = '\0';
                        skipblank();
                        i = -1;
                        j++;
                        arg[j] = (char*) malloc(sizeof(char) * 101);
                }
                else
                {
                        arg[j][i] = c;
                }
        }

        int no_wait;
        if(arg[j][0] == '&')
        {
                // arg[j][0] = (char)NULL;
                free(arg[j]);
                arg[j] = (char*)NULL;
                no_wait = 1;
        }
        else
        {
                if (i == 0)
                {
                        free(arg[j]);
                        arg[j] = (char *)NULL;
                }
                else
                {
                        arg[j][i] = '\0';
                        // arg[j+1][0] = (char)NULL;
                        arg[j+1] = (char *)NULL;
                }

                no_wait = 0;
        }
/*
printf("%s\n", E_name);
for(i = 0; arg[i] != NULL; i++)
{
        printf("i= %d. %s.\n", i, arg[i]);
}
*/

        pid_t pid = fork();

        if(pid == 0)    //child process
        {
                if(no_wait == 1) //no wait, creating a grandchild process and kill this child
process
                {
                        pid_t G_pid = fork();
                        if(G_pid == 0)  //grandchild process
                        {
                                execvp(E_name, (char* const *)arg);   //execution of grandchild process
                                return 0;
                        }
                        else
                        {
                                return 0;  //kill child process
                        }
                }
                else  //wait, no need of grandchild process
                {
                        execvp(E_name, (char* const *)arg);   //execution of child process
                        return 0;
                }
        }
        else if(pid > 0) //parent process
        {
            waitpid(pid, NULL, 0);
        }
        else
        {
            //error
        }

        for(i = 0; arg[i] != NULL; i++)
        {
            free(arg[i]);
        }
    }   //end of while

    return 0;
}
```