#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <wait.h>

#ifdef EC\_NOKILL

int input\_size = 512;

pid\_t pid;

int status = 0;

void alarm\_handler(int signo){

    if (signo == SIGALRM){

        write(STDOUT\_FILENO, CATCHPHRASE, sizeof(CATCHPHRASE));

     }

}

void sigint\_handler(int signo){

    if (signo == SIGINT){

        if (status != 0){

            write(1, "\n", sizeof("\n"));

        }

        else {

            write(1, "\n", sizeof("\n"));

            write(STDOUT\_FILENO, PROMPT, sizeof(PROMPT));

        }

    }

}

void sig\_handler(){

    if (signal(SIGALRM, alarm\_handler) == SIG\_ERR){

        perror("Unable to catch SIGALRM\n");

        exit(EXIT\_FAILURE);

    }

    if (signal(SIGINT, sigint\_handler) == SIG\_ERR){

        perror("Unable to catch SIGINT\n");

        exit(EXIT\_FAILURE);

    }

}

//trim the white space at the front and in between the characters

char \*trimWhiteSpace(char \*str){

    int index = 0;

    int j = 0;

    int k = 0;

    char \*str1 = (char \*)malloc(input\_size);

    while (str[index] == ' '){

        index ++;

    }

    for (j = index; j < input\_size; j++){

        str1[k] = str[j];

        k++;

        if (str[j] == '\0'){

            return str1;

        }

        else if (str[j] == ' '){

            break;

        }

    }

    index = j;

    while (str[index] == ' ' || str[index] == '\0'){

        if (str[index] == '\0'){

            return str1;

        }

        index ++;

    }

    for (j = index; j < input\_size; j++){

        if (str[j] == '\n'){

            break;

        }

        str1[k] = str[j];

        k++;

    }

    return str1;

}

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

    int timeout = 0;

    char user\_command[input\_size];

    char user\_args[input\_size];

    if (argc > 2){

        perror("Please enter only one positive integer \n");

        exit(EXIT\_FAILURE);

    }

    if (argc == 2){

        int temp\_time = atoi(argv[1]);

        if (temp\_time < 0){

            perror("Please enter a positive integer\n");

            exit(EXIT\_FAILURE);

        }

        timeout = temp\_time;

    }

    while(1){

        write(STDOUT\_FILENO, PROMPT, sizeof(PROMPT));

        char \*inputBuffer = (char \*)malloc(input\_size);

        int n = read(STDIN\_FILENO, inputBuffer, input\_size);

        char cmd[input\_size];

        inputBuffer[n] = '\0';

        int a = 0;

        while (a < input\_size){

            cmd[a] = '\0';

            a++;

        }

        if (n == 0 && inputBuffer[0] != '\n'){

            free(inputBuffer);

            exit(0);

        }

        else if (n > 1 && inputBuffer[n - 1] != '\n'){

            free(inputBuffer);

            write(1, "\n", sizeof("\n"));

            continue;

        }

        int m = 0;

        for (m = 0; m < input\_size; m++){

            cmd[m] = inputBuffer[m];

            if (cmd[m] == '\n' || cmd[m] == '\0'){

                break;

            }

        }

        char \*command = trimWhiteSpace(cmd);

        free(inputBuffer);

        if (command[0] == '\n'){

            continue;

        }

        int i = 0;

        for (i = 0; i < input\_size; i++){

            if (command[i] == ' ' || command[i] == '\n'){

                user\_command[i] = '\0';

                break;

            }

            user\_command[i] = command[i];

        }

        int j;

        int k = 0;

        for (j = i + 1; j < input\_size; j++){

            user\_args[k] = command[j];

            k++;

            if (command[j] == '\0'){

                break;

            }

        }

        free(command);

        char \*args[] = {user\_command, user\_args, NULL};

        if (user\_args[0] == '\0'){

            args[1] = NULL;

        }

        pid = 0;

        pid = fork();

        status = 10;

        if (pid == -1){

            perror("an error occurs at fork\n");

            exit(EXIT\_FAILURE);

        }

        else if (pid == 0){

            signal(SIGALRM, SIG\_DFL);

            alarm(timeout);

            if (execve(user\_command, args, NULL) < 0){

                perror(user\_command);

                exit(EXIT\_FAILURE);

            }

        }

        else {

            sig\_handler();

            alarm(timeout);

            pid\_t cpid = wait(&status);

            if (cpid < 0){

                perror("wait error");

                exit(EXIT\_FAILURE);

            }

        }

        alarm(0);

        status = 0;

    }

    return 0;

}

#else

// regular credit code

int input\_size = 512;

pid\_t pid;

int status = 0;

void terminate(){

    if (kill(pid, SIGKILL) == -1){

        perror("Error occurs when terminating the function");

        exit(EXIT\_FAILURE);

    }

}

void alarm\_handler(int signo){

    if (signo == SIGALRM){

        terminate();

        write(STDOUT\_FILENO, CATCHPHRASE, sizeof(CATCHPHRASE));

     }

}

void sigint\_handler(int signo){

    if (signo == SIGINT){

        if (status != 0){

            write(1, "\n", sizeof("\n"));

            terminate();

        }

        else {

            write(1, "\n", sizeof("\n"));

            write(STDOUT\_FILENO, PROMPT, sizeof(PROMPT));

        }

    }

}

void sig\_handler(){

    if (signal(SIGALRM, alarm\_handler) == SIG\_ERR){

        perror("Unable to catch SIGALRM\n");

        exit(EXIT\_FAILURE);

    }

    if (signal(SIGINT, sigint\_handler) == SIG\_ERR){

        perror("Unable to catch SIGINT\n");

        exit(EXIT\_FAILURE);

    }

}

//trim the white space at the front and in between the characters

char \*trimWhiteSpace(char \*str){

    int index = 0;

    int j = 0;

    int k = 0;

    char \*str1 = (char \*)malloc(input\_size);

    while (str[index] == ' '){

        index ++;

    }

    for (j = index; j < input\_size; j++){

        str1[k] = str[j];

        k++;

        if (str[j] == '\0'){

            return str1;

        }

        else if (str[j] == ' '){

            break;

        }

    }

    index = j;

    while (str[index] == ' ' || str[index] == '\0'){

        if (str[index] == '\0'){

            return str1;

        }

        index ++;

    }

    for (j = index; j < input\_size; j++){

        if (str[j] == '\n'){

            break;

        }

        str1[k] = str[j];

        k++;

    }

    return str1;

}

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

    int timeout = 0;

    char user\_command[input\_size];

    char user\_args[input\_size];

    if (argc > 2){

        perror("Please enter only one positive integer \n");

        exit(EXIT\_FAILURE);

    }

    if (argc == 2){

        int temp\_time = atoi(argv[1]);

        if (temp\_time < 0){

            perror("Please enter a positive integer\n");

            exit(EXIT\_FAILURE);

        }

        timeout = temp\_time;

    }

    while(1){

        write(STDOUT\_FILENO, PROMPT, sizeof(PROMPT));

        char \*inputBuffer = (char \*)malloc(input\_size);

        int n = read(STDIN\_FILENO, inputBuffer, input\_size);

        char cmd[input\_size];

        inputBuffer[n] = '\0';

        int a = 0;

        sig\_handler();

        while (a < input\_size){

            cmd[a] = '\0';

            a++;

        }

        if (n == 0 && inputBuffer[0] != '\n'){

            free(inputBuffer);

            exit(0);

        }

        else if (n > 1 && inputBuffer[n - 1] != '\n'){

            free(inputBuffer);

            write(1, "\n", sizeof("\n"));

            continue;

        }

        int m = 0;

        for (m = 0; m < input\_size; m++){

            cmd[m] = inputBuffer[m];

            if (cmd[m] == '\n' || cmd[m] == '\0'){

                break;

            }

        }

        char \*command = trimWhiteSpace(cmd);

        free(inputBuffer);

        if (command[0] == '\n'){

            continue;

        }

        int i = 0;

        for (i = 0; i < input\_size; i++){

            if (command[i] == ' ' || command[i] == '\n'){

                user\_command[i] = '\0';

                break;

            }

            user\_command[i] = command[i];

        }

        int j;

        int k = 0;

        for (j = i + 1; j < input\_size; j++){

            user\_args[k] = command[j];

            k++;

            if (command[j] == '\0'){

                break;

            }

        }

        free(command);

        char \*args[] = {user\_command, user\_args, NULL};

        if (user\_args[0] == '\0'){

            args[1] = NULL;

        }

        pid = 0;

        pid = fork();

        status = 10;

        if (pid == -1){

            perror("an error occurs at fork\n");

            exit(EXIT\_FAILURE);

        }

        else if (pid == 0){

            if (execve(user\_command, args, NULL) < 0){

                perror(user\_command);

                exit(EXIT\_FAILURE);

            }

        }

        else {

            alarm(timeout);

            pid\_t cpid = wait(&status);

            if (cpid < 0){

                perror("wait error");

                exit(EXIT\_FAILURE);

            }

        }

        alarm(0);

        status = 0;

    }

    return 0;

}

#endif