```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/wait.h>
#include <unistd.h>
#include <sched.h>
#include <bits/waitflags.h>

char * sh_read_line(void);
char ** sh_tokenize(char * line);
int sh_launch(char ** args);
int sh_execute(char ** args);
// builtin commands
int sh_cd(char ** args);
int sh_help(char ** args);
int sh_exit(char ** args);

// function which loops until user exits and reads, parses and executes commands
void sh_loop(void) {
    char * line;
    char ** args;
    int status;
    do {
        printf("> ");
        line = sh_read_line();
        args = sh_tokenize(line);
        status = sh_execute(args);
        free(line);
        free(args);
    } while (status);
}

// function to read a line from the command line
char * sh_read_line(void) {
    char * line = NULL;
    size_t len = 0;
    getline(&line, &len, stdin);
    return line;
}

// function to tokenize the line into arguments
char ** sh_tokenize(char * line) {
    char ** tokens = malloc(64 * sizeof(char*));
    char * token;
    int i = 0;
    token = strtok(line, " \t\r\n\a");
    while (token != NULL) {
        tokens[i] = token;
        i++;
        token = strtok(NULL, " \t\r\n\a");
    }
    tokens[i] = NULL;
    return tokens;
}

// function to launch a program
int sh_launch(char ** args) {
    pid_t pid, wpid;
    int status;
    pid = fork();
    if (pid == 0) {
        if (execvp(args[0], args) == -1) {
            perror("myshell");
        }
        exit(EXIT_FAILURE);
    } else if (pid < 0) {
        perror("myshell");
    } else {
        do {
            wpid = waitpid(pid, &status, WUNTRACED);
        } while (!WIFEXITED(status) && !WIFSIGNALED(status));
    }
    return 1;
}

// builtin commands
char *builtin_str[] = {
    "cd",
    "help",
    "exit"
};

int (*builtin_func[]) (char **) = {
    &sh_cd,
    &sh_help,
    &sh_exit
};

int sh_num_builtins() {
    return sizeof(builtin_str) / sizeof(char *);
}

int sh_cd(char ** args) {
    if (args[1] == NULL) {
        fprintf(stderr, "myshell: expected argument to \"cd\"\n");
    } else {
        if (chdir(args[1]) != 0) {
            perror("myshell");
        }
    }
    return 1;
}

int sh_help(char ** args) {
    printf("myshell\n");
    printf("Type program names and arguments, and hit enter.\n");
    printf("The following are built in:\n");
    for (int i = 0; i < 3; i++)
        printf("  %s\n", builtin_str[i]);
    printf("Use man command for other programs.\n");
    return 1;
}

int sh_exit(char ** args) {
    return 0;
}

// function to execute the command
int sh_execute(char ** args) {
    if (args[0] == NULL) {
        return 1;
    }
    for (int i = 0; i < sh_num_builtins(); i++) {
        if (strcmp(args[0], builtin_str[i]) == 0) {
            return (*builtin_func[i])(args);
        }
    }
    return sh_launch(args);
}

int main(int argc, char ** argv) {
    sh_loop();
    return EXIT_SUCCESS;
}
```