# TYPELINE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

// Constants for command input
#define MAX_INPUT 80
#define MAX_ARGS 10

// Function to tokenize command input
int make_toks(char *input, char *args[]) {
    int i = 0;
    char *token;

    token = strtok(input, " ");
    while (token != NULL) {
        args[i++] = token;
        token = strtok(NULL, " ");
    }
    args[i] = NULL; // Null-terminate the arguments array
    return i; // Return number of tokens
}

// Function to handle the 'typeline' command
void typeline(char *option, char *filename) {
    FILE *file = fopen(filename, "r");
    if (!file) {
        printf("File %s not found.\n", filename);
        return;
    }

    if (strcmp(option, "a") == 0) {
        char line[256];
        while (fgets(line, sizeof(line), file)) {
            printf("%s", line);
        }
    } else {
        int n = atoi(option);
        for (int i = 0; i < n && !feof(file); i++) {
            char line[256];
```

```
            fgets(line, sizeof(line), file);
            printf("%s", line);
        }
    }
    fclose(file);
}

// Main shell loop
void myshell() {
    char input[MAX_INPUT];
    char *args[MAX_ARGS];

    while (1) {
        printf("myshell$ ");
        fflush(stdout);
        fgets(input, sizeof(input), stdin);

        // Remove trailing newline character
        input[strcspn(input, "\n")] = 0;

        // Tokenize the input
        int n = make_toks(input, args);

        // Handle built-in commands
        if (n > 0) {
            if (strcmp(args[0], "exit") == 0) {
                exit(0); // Exit the shell
            } else if (strcmp(args[0], "typeline") == 0 && n == 3) {
                typeline(args[1], args[2]); // Call typeline function
            } else {
                printf("Invalid command.\n");
            }
        }
    }
}

int main() {
    myshell(); // Start the shell
    return 0;
}
```