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

int main() {
    while (1) {
        printf("Enter command: \n");
        char buffer[2048];
        char *line = fgets(buffer, 2048, stdin);

        char *command = NULL;
        int cmd_code = 0;
        if (0 == strncmp(line, "sort", 4)) {
            command = "/usr/sbin/sort";
            cmd_code = 1;
        } else if (0 == strncmp(line, "echo", 4)) {
            command = "/usr/sbin/echo";
            cmd_code = 2;
        } else if (0 == strncmp(line, "sleep", 5)) {
            command = "/usr/sbin/sleep";
            cmd_code = 3;
        } else if (0 == strncmp(line, "quit", 4)) {
            printf("User ended session: Bye!\n");
            break;
        }
        if (command == NULL) {
            printf("bad command: %s\n", line);
            continue;
        }
        printf("command is: %s\n", command);
        printf("line is: %s\n", line);

        // tokenise arguments (generic)
        // ...left as exercise for reader
        // hint: strsep or strtok with delimiter ' '

        // hackery: replace new line with null byte
        strchr(line, '\n')[0] = '\0'; // <-- scream about
         pointers here
        // hackery: get next string at first space char
        char *first_arg = strchr(line, ' ');
        if (first_arg)
            first_arg++; // exists, move after space
        int pid = fork();
        if (pid == 0) {
            int result = 0;
            switch(cmd_code) {
                case 1:
                    result = execl(command, "sort",
                     "numbers.txt", 0);
                    break;
                case 2:
                    result = execl(command, "echo",
                     first_arg, 0);
                    break;
                case 3:
                    result = execl(command, "sleep",
                     first_arg, 0);
                    break;
                default:
                    fprintf(stderr, "unknown command code:
                     %d\n", cmd_code);
                    break;
            }
            return 0;
        } else if (pid > 0) {
            // wait for child process to finish
            int status;
            int result = wait(&status);
        }
    } // while loop

    return 0;
}
```