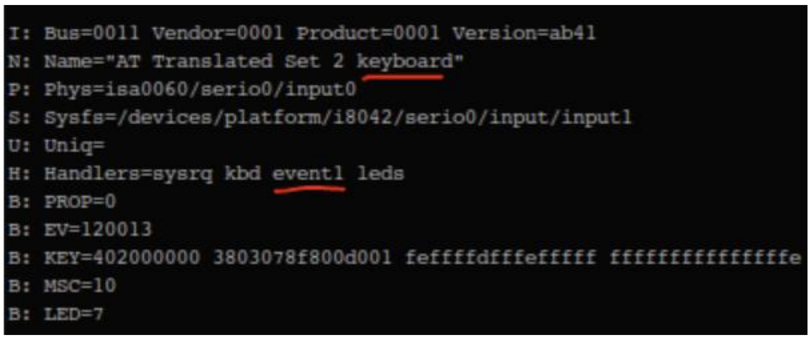
首先完成代码的编写：
如果检查作业需要编译代码验证结果，可参考以下步骤：
在 shell 中运行
$cat /proc/bus/input/devices
先在这个里边找到自己使用的键盘对应的事件号，比如下图：


```
I: Bus=0011 Vendor=0001 Product=0001 Version=ab41
N: Name="AT Translated Set 2 keyboard"
P: Phys=isa0060/serio0/input0
S: Sysfs=/devices/platform/i8042/serio0/input/input1
U: Uniq=
H: Handlers=sysrq kbd event1 leds
B: PROP=0
B: EV=120013
B: KEY=402000000 3803078f800d001 feffffdfffefffff fffffffffffffffe
B: MSC=10
B: LED=7
```

将代码中的键盘事件路径中的 event 的后缀编号改为对应的编号
编译这个文件，可以直接使用 gcc 编译
$gcc hw3.c -o hw3
然后使用 root 权限运行这个程序
sudo ./hw3
然后就可以看到结果了。

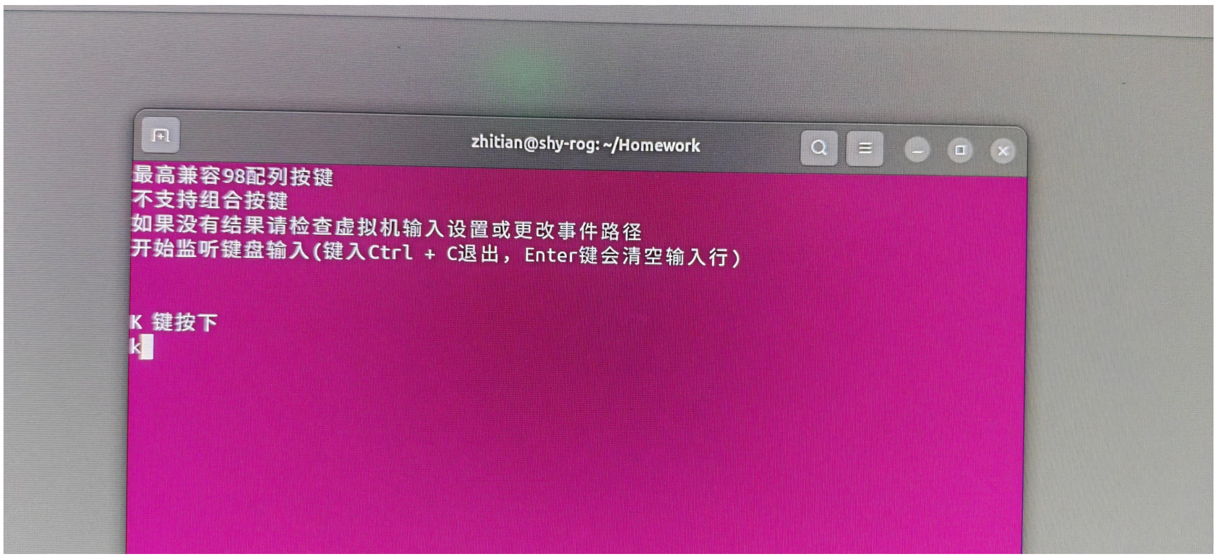
```c
#include <linux/input.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <pthread.h>
int rows = 8;
int currentKey = 0;
char dir[256] = "/dev/input/event20";//键盘事件路径，请根据自己使用的 keyboard 对应的 event 号修改



void init_layout(int rows)
{
    printf("\033[2J");
    printf("\033[1;1H");
    printf("最高兼容 98 配列按键\n");
    printf("不支持组合按键\n");
    printf("如果没有结果请检查虚拟机输入设置或更改事件路径\n");
    printf("开始监听键盘输入(键入 Ctrl + C 退出，按下 Enter 键会清空输入行)");
    printf("\033[%d;1H", rows);
    fflush(stdout);
}
```

```c
void get_input(char* input, int max_len, int rows)
{
    printf("\033[%d;1H", rows);
    printf("\033[2K");
    fgets(input, max_len, stdin);
    input[strcspn(input, "\n")] = 0;
}
void my_printf(const char* format, int rows, int cols)
{
    printf("\033[%d;%dH", rows - 1,cols);
    printf("\033[2K");
    printf("%s\n", format);
    printf("\033[%d;1H", rows);
    fflush(stdout);
}
```

```c
void listen_keyboard()
{
    int fd = open(dir, O_RDONLY);
    if (fd < 0)
    {
    perror("open event file failed!");
    exit(1);
    }
    printf("%d",currentKey);
    fflush(stdout);
    struct input_event ev;
    ssize_t n;
    while (1)
    {
        n = read(fd, &ev, sizeof(ev));
        if (n != -1)
        {
            if (ev.type == EV_KEY && ev.value == 1)
            {
                currentKey = ev.code;
            }
        }
    }
    close(fd);
}
void listen_currentKey()
{
    static int lastKey = 0;
    while (1)
    {
        if (lastKey != currentKey)
        {
            //printf("\033[10;1]CurrentKey: %d\n", currentKey);
            fflush(stdout);
            char output[128];
        bool mark = true;
            switch (currentKey)
            {
            case KEY_UP:
                strcpy(output,"↑");
                break;
            case KEY_DOWN:
                strcpy(output,"↓");
                break;
            case KEY_LEFT:
                strcpy(output, "←");
                break;
            case KEY_RIGHT:
                strcpy(output, "→");
                break;
            case KEY_ENTER:
                strcpy(output, "Enter");
                printf("\033[%d;1H", rows);
                printf("\033[2K");
                break;
            case KEY_BACKSPACE:
                strcpy(output, "Backspace");
                break;
            case KEY_DELETE:
                strcpy(output, "Delete");
                break;
            case KEY_HOME:
                strcpy(output, "Home");
                break;
            case KEY_END:
                strcpy(output, "End");
                break;
            case KEY_PAGEUP:
                strcpy(output, "PageUp");
                break;
            case KEY_PAGEDOWN:
```

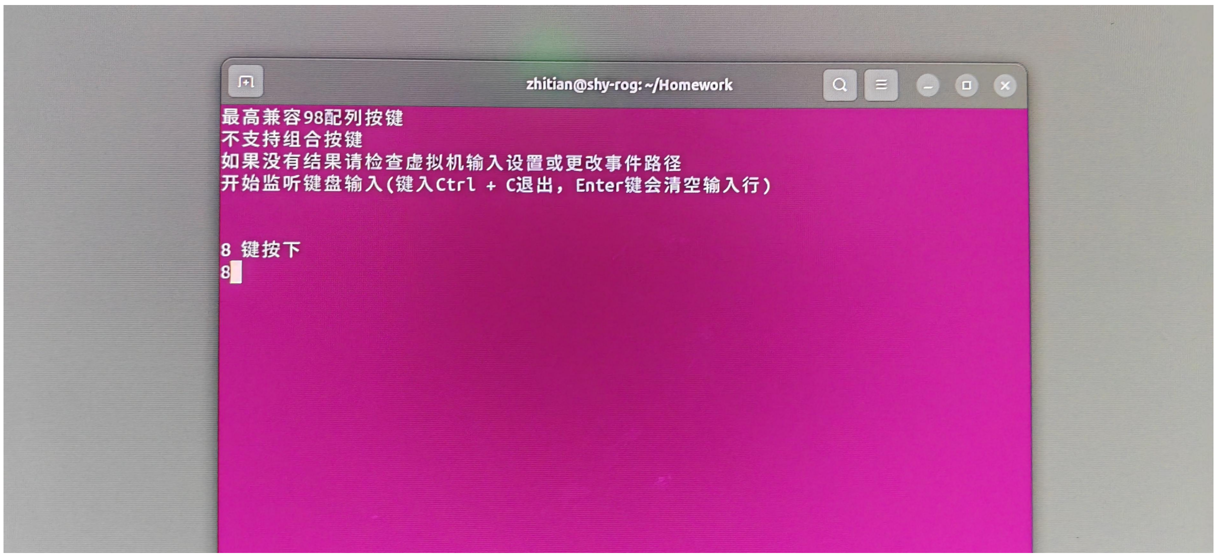
```
            strcpy(output, "PageDown");
            break;
        case KEY_1:
            strcpy(output, "1");
            break;
        case KEY_2:
            strcpy(output, "2");
            break;
        case KEY_3:
            strcpy(output, "3");
            break;
        case KEY_4:
            strcpy(output, "4");
            break;
        case KEY_5:
            strcpy(output, "5");
            break;
        case KEY_6:
            strcpy(output, "6");
            break;
        case KEY_7:
            strcpy(output, "7");
            break;
        case KEY_8:
            strcpy(output, "8");
            break;
        case KEY_9:
            strcpy(output, "9");
            break;
        case KEY_0:
            strcpy(output, "0");
            break;
        case KEY_A:
            strcpy(output, "A");
            break;
        case KEY_B:
            strcpy(output, "B");
            break;
        case KEY_C:
            strcpy(output, "C");
            break;
        case KEY_D:
            strcpy(output, "D");
            break;
            case KEY_E:
                strcpy(output, "E");
                break;
            case KEY_F:
                strcpy(output, "F");
                break;
            case KEY_G:
                strcpy(output, "G");
                break;
            case KEY_H:
                strcpy(output, "H");
                break;
            case KEY_I:
                strcpy(output, "I");
                break;
            case KEY_J:
                strcpy(output, "J");
                break;
            case KEY_K:
                strcpy(output, "K");
                break;
            case KEY_L:
                strcpy(output, "L");
                break;
            case KEY_M:
                strcpy(output, "M");
                break;
            case KEY_N:
                strcpy(output, "N");
                break;
            case KEY_O:
                strcpy(output, "O");
                break;
            case KEY_P:
                strcpy(output, "P");
                break;
            case KEY_Q:
                strcpy(output, "Q");
                break;
            case KEY_R:
                strcpy(output, "R");
                break;
            case KEY_S:
                strcpy(output, "S");
                break;
            case KEY_T:
                strcpy(output, "T");
                break;
            case KEY_U:
                strcpy(output, "U");
                break;
            case KEY_V:
                strcpy(output, "V");
                break;
            case KEY_W:
                strcpy(output, "W");
                break;
            case KEY_X:
                strcpy(output, "X");
                break;
            case KEY_Y:
                strcpy(output, "Y");
                break;
            case KEY_Z:
                strcpy(output, "Z");
                break;
            case KEY_F1:
                strcpy(output, "F1");
                break;
            case KEY_F2:
                strcpy(output, "F2");
                break;
            case KEY_F3:
                strcpy(output, "F3");
                break;
            case KEY_F4:
                strcpy(output, "F4");
                break;
            case KEY_F5:
                strcpy(output, "F5");
                break;
            case KEY_F6:
                strcpy(output, "F6");
                break;
            case KEY_F7:
                strcpy(output, "F7");
                break;
            case KEY_F8:
                strcpy(output, "F8");
                break;
            case KEY_F9:
                strcpy(output, "F9");
                break;
            case KEY_F10:
                strcpy(output, "F10");
                break;
            case KEY_F11:
                strcpy(output, "F11");
                break;
            case KEY_F12:
                strcpy(output, "F12");
                break;
            case KEY_ESC:
                strcpy(output, "Esc");
                break;
            case KEY_TAB:
                strcpy(output, "Tab");
                break;
            case KEY_SPACE:
                strcpy(output, "Space");
                break;
            case KEY_MINUS:
                strcpy(output, "-");
                break;
            case KEY_EQUAL:
                strcpy(output, "=");
                break;
            case KEY_LEFTBRACE:
                strcpy(output, "[");
                break;
            case KEY_RIGHTBRACE:
                strcpy(output, "]");
                break;
            case KEY_BACKSLASH:
                strcpy(output, "\\");
                break;
            case KEY_SEMICOLON:
                strcpy(output, ");");
                break;
            case KEY_APOSTROPHE:
                strcpy(output, "'");
                break;
            case KEY_GRAVE:
                strcpy(output, "`");
                break;
            case KEY_COMMA:
                strcpy(output, ",");
                break;
            case KEY_DOT:
                strcpy(output, ".");
                break;
            case KEY_SLASH:
                strcpy(output, "/");
                break;
            case KEY_CAPSLOCK:
                strcpy(output, "CapsLock");
                break;
            case KEY_LEFTSHIFT:
                strcpy(output, "Shift");
                break;
            case KEY_RIGHTSHIFT:
                strcpy(output, "Shift");
                break;
            case KEY_LEFTCTRL:
                strcpy(output, "Ctrl");
                break;
            case KEY_RIGHTCTRL:
                strcpy(output, "Ctrl");
                break;
            case KEY_LEFTALT:
                strcpy(output, "Alt");
                break;
            case KEY_RIGHTALT:
                strcpy(output, "Alt");
                break;
            case KEY_SYSRQ:
                strcpy(output, "SysRq");
                break;
            case KEY_SCROLLLOCK:
                strcpy(output, "ScrollLock");
                break;
            case KEY_NUMLOCK:
                strcpy(output, "NumLock");
                break;
            case KEY_PAUSE:
                strcpy(output, "Pause");
                break;
            case KEY_INSERT:
                strcpy(output, "Insert");
                break;
            case KEY_HOMEPAGE:
                strcpy(output, "HomePage");
                break;
            case KEY_MENU:
                strcpy(output, "Menu");
                break;
            case KEY_POWER:
                strcpy(output, "Power");
                break;
            case KEY_SLEEP:
                strcpy(output, "Sleep");
                break;
            case KEY_WAKEUP:
                strcpy(output, "WakeUp");
                break;
            case KEY_MUTE:
                strcpy(output, "Mute");
                break;
            case KEY_VOLUMEDOWN:
                strcpy(output, "VolumeDown");
                break;
            case KEY_VOLUMEUP:
                strcpy(output, "VolumeUp");
                break;
            case KEY_KP1:
                strcpy(output, "1");
                break;
            case KEY_KP2:
                strcpy(output, "2");
                break;
            case KEY_KP3:
                strcpy(output, "3");
                break;
            case KEY_KP4:
                strcpy(output, "4");
                break;
            case KEY_KP5:
                strcpy(output, "5");
                break;
            case KEY_KP6:
                strcpy(output, "6");
                break;
            case KEY_KP7:
                strcpy(output, "7");
                break;
            case KEY_KP8:
                strcpy(output, "8");
                break;
            case KEY_KP9:
                strcpy(output, "9");
                break;
            case KEY_KP0:
                strcpy(output, "0");
                break;
            case KEY_KPDOT:
                strcpy(output, ".");
                break;
            case KEY_KPENTER:
                strcpy(output, "Enter");
                break;
            case KEY_KPPLUS:
                strcpy(output, "+");
                break;
            case KEY_KPMINUS:
                strcpy(output, "-");
                break;
            case KEY_KPASTERISK:
                strcpy(output, "*");
                break;
            case KEY_KPSLASH:
                strcpy(output, "/");
                break;
            case KEY_KPCOMMA:
                strcpy(output, ",");
                break;
            case KEY_KPEQUAL:
                strcpy(output, "=");
                break;
            default:
                strcpy(output, "没有按下的按键或者不是适配的按键");
                mark = false;
        break;
            }
            if (mark)
        {
        strcat(output," 键按下");
        }
            my_printf(output, rows, 1);
            lastKey = currentKey;
        }
    }
}
void function()
{
    pthread_t for_keyboard, for_currentKey;
    pthread_create(&for_keyboard, NULL, (void*)listen_keyboard,
NULL);
    pthread_create(&for_currentKey, NULL,
(void*)listen_currentKey, NULL);
    pthread_join(for_keyboard, NULL);
    pthread_join(for_currentKey, NULL);
}
int main() {
    init_layout(rows);
    function();
    printf("\033[2J");
    printf("\033[1;1H");
    return 0;
}
```

以下为一些效果的参考图：

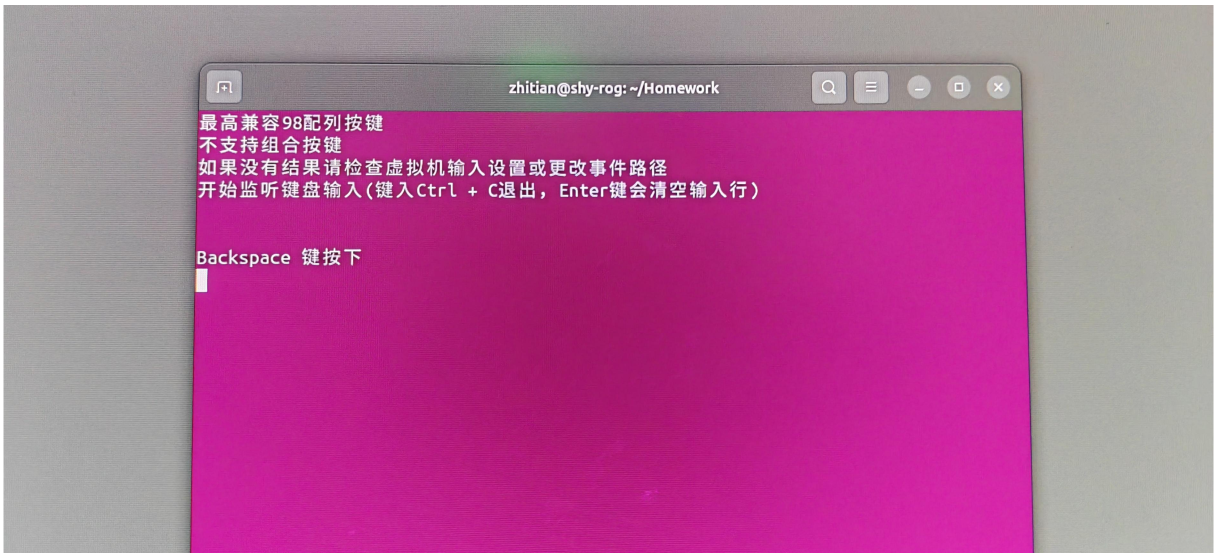


```
最高兼容98配列按键
不支持组合按键
如果没有结果请检查虚拟机输入设置或更改事件路径
开始监听键盘输入(键入Ctrl + C退出, Enter键会清空输入行)


K 键按下
k
```

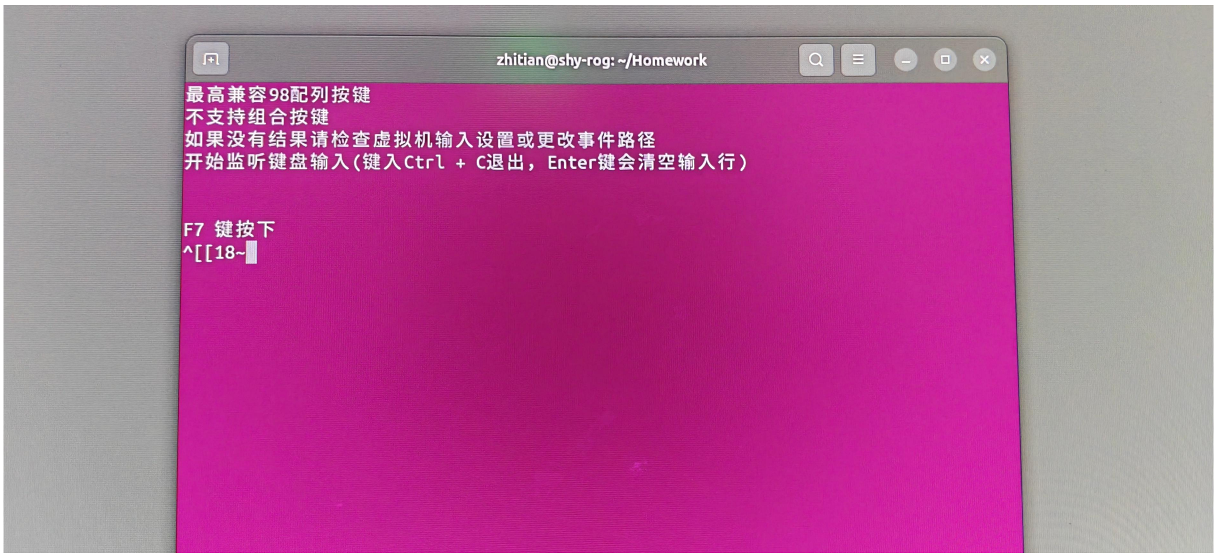
这个监听键盘输入不需要像这个程序输入任何内容，只需要按下按键即可，在系统的任何位置按键都可以，不需要给这个窗口焦点（缺点就是这种实现方式需要



```
最高兼容98配列按键
不支持组合按键
如果没有结果请检查虚拟机输入设置或更改事件路径
开始监听键盘输入(键入Ctrl + C退出，Enter键会清空输入行)


8 键按下
8
```



```
最高兼容98配列按键
不支持组合按键
如果没有结果请检查虚拟机输入设置或更改事件路径
开始监听键盘输入(键入Ctrl + C退出，Enter键会清空输入行)


Backspace 键按下
```



```
最高兼容98配列按键
不支持组合按键
如果没有结果请检查虚拟机输入设置或更改事件路径
开始监听键盘输入(键入Ctrl + C退出，Enter键会清空输入行)


F7 键按下
^[[18~
```

这个监听键盘输入不需要像这个程序输入任何内容，只需要按下按键即可，在系统的任何位置按键都可以，不需要给这个窗口焦点（缺点就是这种实现方式需要

root 权限运行程序）。