一、触摸屏

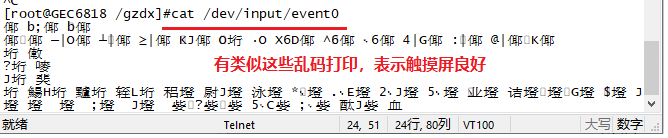
1、几个重要的概念

文件：触摸屏对应的驱动设备文件 ---/dev/input/event0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*查看自己的触摸屏是不是正常使用的\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

命令行敲：cat /dev/input.event0

然后手指在屏幕上滑动，如果有一些乱码打印出来，如下所示。表示触摸屏正常使用，如果什么打印都没有，表示触摸屏坏的，需要换一块开发板



事件：当外接设备（比如触摸屏）接入ARM开发板的时候，而且这些设备的状态发生改变的时候，称之为事件的发生

输入子系统：当事件发生的时候，输入子系统就会帮助我们应用层自动计算这个事件对应的值

2、尝试读取触摸屏的数据

触摸屏对应的头文件：/usr/include/linux/input.h

struct input\_event {

\_\_u16 type;//事件的类型

\_\_u16 code;//事件的编码，对事件进一步描述

\_\_s32 value;//事件的值

};

type:事件的类型,也就是判断发送的是哪个事件

/\*

\* Event types

\*/

#define EV\_SYN 0x00

#define EV\_KEY 0x01 按键事件 压力事件

#define EV\_REL 0x02

#define EV\_ABS 0x03 触摸屏事件

#define EV\_MSC 0x04 鼠标事件

#define EV\_SW 0x05

#define EV\_LED 0x11

#define EV\_SND 0x12

#define EV\_REP 0x14

#define EV\_FF 0x15

#define EV\_PWR 0x16

#define EV\_FF\_STATUS 0x17

#define EV\_MAX 0x1f

#define EV\_CNT (EV\_MAX+1)

code;//事件的编码，对事件进一步描述

#define ABS\_X 0x00 触摸屏事件中的X轴事件

#define ABS\_Y 0x01 触摸屏事件中的Y轴事件

#define BTN\_TOUCH 0x14a 表示压力事件

value： 压力事件中 按下 表示 1

松开 表示 0

思考：判断当手松开的时候，才打印坐标值

也就是你按一次，只打印一次坐标

练习1：当左滑动的时候，显示上一张图片；

当右滑动的时候，显示下一张图片