Proj 5: 完善键盘驱动，添加设备

> 现场验收时间（两次）：2025.6.15晚上18:00-21:00（周日、同proj4一起），或者2025.6.18晚上18:00-21:00（周三）；同时，2025.6.18晚上同时接受前面未验收的实验的补验收（成绩最高为相应内容的及格分），实验楼103
>
> 报告提交截止时间：2025.6.18晚上23:59

本项目涉及IO和设备。

1. 完善键盘驱动（40%）

实验内容

为xv6增加ctrl+c功能：当用户在键盘上同时按下Ctrl和C键时，屏幕上显示 ^C，同时换行，OS将正在执行或者正在等待输入的进程杀死。有以下情况需要考虑。

- 按下组合键时，如果有用户进程正在执行，则将该用户进程杀死。
    - 将用户进程杀死后，屏幕需要显示换行（用consputc），同时注意，不应当将Ctrl+C字符存储到输入缓冲区input中，否则会留给其他进程，导致出错。
- 如果不存在正在运行的用户进程，即，如果用户进程均在阻塞状态，则将正在等待命令行输入的用户进程杀死。(Hint: 如果用户进程在等待命令行输入，则它的chan指向的是&input.r)
    - 如果等待命令行输入的是sh进程（pid为2，name为sh），则不可将其杀死，可以将其等价于用户按了回车键。
    - 建议在杀死进程前，在屏幕上打印提示信息，但请注意，在consoleintr中使用cprintf前需要释放cons.lock，因为cprintf中会拿cons.lock锁，两次拿同一个锁会panic
- ^的ASCII码在C语言中可直接用'^'表示。
- 你需要修改console.c文件

验收要求

运行deadloop命令。此时按Ctrl+C，屏幕显示^C，deadloop进程退出，屏幕重新开始接受用户输入。

```
$ deadloop
^C
$
```

运行infread命令。此时按Ctrl+C，屏幕显示^C，infread进程退出，屏幕接受用户输入。

```
$ infread
^C
$
```

在命令行界面直接按Ctrl+C时，屏幕显示^C，同时换行。

```
$ ^C
$
```

多测试几次，如：

```
$ deadloop
^C
$ infread
^C
$ deadloop
^C
$ deadloop
^C
$ deadloop
^C
^C
$ $ infread
^C
$
```

注意，运行deadloop时，偶尔会出现按了一两次ctrl+c后进程并未终止，需要多执行几次才会终止。这是允许的，可能deadloop恰好时间片到达，cpu去执行scheduler，导致myproc指向了0，也即当前进程不是deadloop。

报告要求

- 介绍你的实现。
- 阅读infread.c，它是从控制台读入一个字符，然后打印到屏幕上。但如果你在屏幕上连续输入abcdefg，屏幕没有反应，敲回车后，屏幕才会打印abcdefg。为何不是输入a后立即显示a？指出造成这个现象的原因及相应的内核源码。（提示：main函数中调用的consoleinit会将devsw[CONSOLE].read指向consoleread()，此函数在fs.c的readi中会被调用。）

2. 添加设备（40%）

实验内容

为xv6添加一个名为`osdev`的设备，该设备为虚拟设备，实际上是内核中一个大小为1024字节的缓冲区。该设备支持open、read和write操作，这三个系统调用是面向文件的，已经实现，你需要扩展它们在内核中的实现（增加if判断）。

典型使用方式如下：

- 在系统启动、进入scheduler前，初始化缓冲区，比如读写头位置，自旋锁等，参考pipeinit。读时不能写、写时不能读，用spinlock保护osdev。
- fd=open("osdev",O_RDWR)，返回int型的文件描述符，不需要考虑其他权限，即，你可以忽略O_RDWR域，但需要借助文件系统
    - 你需要扩充file结构体中type的值域，加一个FD_OSDEV类型，以方便后续read和write快速判断设备类别，设置readable和writable均为1。

    - 每次open，你需要申请一个struct file结构体(filealloc)，将type设置为FD_OSDEV
    - 你还需要申请一个文件描述符，将文件描述符指向file结构体，即调用fdalloc()
    - 返回申请到的文件描述符
    - 参考sys_pipe
- nbytes_read=read(fd, (char *)c, len)，从缓冲区读数据，如果没有内容，则用户进程阻塞，否则，返回实际读到的字节数（可能小于等于用户请求的字节数len，有多少读多少）；
    - sys_read在做完权限检查后，会调用fileread，你可以在fileread中判断文件类型，做特殊处理
    - 参考piperead，注意不需要唤醒写进程
- nbytes_write=write(fd,(char *)c, len)，将c指向的字符串，写最多len个字符到缓冲区中，返回实际写入缓冲区的字符个数（如果缓冲区满则少于len个字符），此外，
    - 写结束时唤醒读进程；
    - 仿照pipewrite，但注意，osdev的write不阻塞

可以仿照pipe实现，但有以下区别

- pipe随进程消亡，osdev常驻内存；
- pipe每次会创建一个独立的个体（比如pipealloc中需要kalloc以一个页面），而osdev不是，整个系统中osdev只有一个，所有进程打开osdev时，指向的是同一个缓冲区；
- pipe通过pipe()系统调用创建，返回两个文件描述符；而osdev通过open调用打开，返回1个文件描述符。
- pipe读和写都会陷入阻塞，osdev读会陷入阻塞，写不会。
- 如果make报错，找不到函数定义，可能你添加的函数在一个新的内核文件中，但该文件并没有在Makefile中声明。声明方式是加到OBJS中。

验收要求

运行osdevtest，典型输出如下

```
$ osdevtest
TC1 success: open osdev with fd=3
TC2: write: 0123456789 (10) read: 0123456789 (10)
TC2 success
TC3: one-two
TC4: Please type a string from keyboard:
good study
good study
```

特征

- TC1显示success
- TC2在第三行显示success（第二行为debug）
- TC3显示one-two
- TC4，需要手动输入一个字符串，敲回车后会打印该字符串

报告要求

- 描述你的实现过程和遇到的困难。

3. 实验报告和代码（20%）

需要准备两份文件：

- 源代码：zip压缩包
  - 进入`proj5-revise`目录，执行`make clean`，然后在proj5-revise的父目录下执行`zip -r proj5-revise.zip proj5-revise`，此命令会创建一个源代码压缩文件`proj5-revise.zip`。提交此文件。
  - 请确保提交的代码压缩包内无其他文件（比如.o文件，或者vscode目录）。
- 实验报告：pdf，
  - 在实验报告中回答上面提到的问题；
  - 给出参考资料（网址）和工具（哪个大模型）；
  - 报告中需要添加姓名学号。不限定模板，尽量整洁、美观。