思考题

1

只展示修改部分的代码

```
switch (fork()) {

    case -1: /* Handle error */

        break;

    case 0: /* Child - writes to pipe */

        close(fildes[0]); /* Read end is unused */

        write(fildes[1], "Hello world\n", 12); /* Write data on pipe*/

        close(fildes[1]); /* Child will see EOF */

        exit(EXIT_SUCCESS);

    default: /* Parent - reads from pipe */

        close(fildes[1]); /* Write end is unused */

        read(fildes[0], buf, 100); /* Get data from pipe */

        printf("parent-process read:%s",buf); /* Print the data */

        close(fildes[0]); /* Finished with pipe */

        exit(EXIT_SUCCESS);

}
```

2

因为dup函数不是原子性的，我们先对fd的`pageref++`，再对pipe的`pageref++`的时候，可能存在`pageref(pipe)=pageref(fd)`的情况，这里的fd是fd[0]或者fd[1]，如果此时进程被切换到执行`_pipeclosed`函数，可能造成该函数的误判，导致无法正常完成读写任务。

3

在我们的实验中，一旦进入以`syscall_`开头的函数，时钟就会关闭，自然就不会被打断，可以保证原子性。
并非所有的系统调用都是原子操作，例如write ，两个进程同时往一个文本文件中不断的追加随机的数据，可能出现数据交叠的情制况，也就是在第一个进程的数据当中插入了第二个进程的写数据，简单的说，多个进程独立的往同一个文本文件中追加输出日志信息，可能发生该情况。

4

- 可以。这是因为通常pageref(pipe)>pageref(fd)，如果先unmap(fd)，那么`pageref(pipe)>pageref(fd)`的关系依然保持，所以不会出现二者相等而导致`_pipeclosed`误判的情况。
- 可以。原因同上

5

elf文件中每一段都有bin_size和sgsize，当bin_size < sgsize时，说明在sgsize - bin_size这一部分是需要用0来填充的，用`bzero`函数可实现置0功能，和lab3十分相似。

6

user的link script文件中约定了.text段的地址

7

在user中的init.c

实验难点

- 在此次的lab6中，我认为第一个难点是在实现管道的过程中如何判断读端或者写端是否关闭。通常情况下， 一旦建立起一个管道，两个fd与共享的那个pipe的pageref分别应该是1,1,2。当有一个进程用close函数关闭一个fd时，close实际上做了两个动作，让fd与pipe的pageref都减·，因此两个fd与pipe的pageref变成1,0,1或者0,1,1，我们通过判断某个fd的pageref与pipe的pageref是否相等就可以知道另一个fd是否关闭了。但是由于这种操作不是原子性的，在执行的过程中很容易被打断，导致结果发生错误，所以我们就需要调整unmap的顺序来避免这样的问题。
- 另一个难点是spawn函数的填写。在这个函数中需要加载elf文件，因为这个功能在lab3中已经实现，所以我原来想直接调用lab3的函数来实现，可是总是出错，浪费了很多时间；最终只能选择再实现一遍。其实不是难，但是很麻烦，毕竟重复的代码又多了。

心得和体会

- 在OO课程中有时要对自己的代码进行测试，就需要实现一个本地评测机，而控制程序运行结果输入至评测机中就用到了管道，之前只知道怎么用，这回在lab6真正实现了管道，理解了原理。
- 通过lab6我也理解了shell的实现方法，收获颇丰
- 伴随着lab6的结束，OS课程设计也就告一段落，因为疫情的影响本学期的OS课程设计没有课上部分，这确实降低了不少的难度。但是从lab0到lab6一路走过来，也遇到了很多的困难，或是自己思考，或是和同学助教老师讨论，一步一步前进，最终到达了终点。学习OS实验让我们对一个操作系统各种最基本的功能有一个清晰的认识，我相信这对于我们日后的学习和工作都有很大的帮助。

Extra

这次的extra其实和lab6没啥关系
难度适中
最重要的是模拟了一遍理论课需要重点掌握的P、V操作，让我们真正理解为什么P、V操作能够控制进程同步