操作系统lab6实验报告

17373452 单彦博

一、思考题

6.1 示例代码中，父进程操作管道的写端，子进程操作管道的读端。如果现在想让父进程作为“读者”，代码应当如何修改?

因为fork返回后会先执行父进程，所以应该在父进程开始执行时，先切换到子进程进行写入。

```
#include <stdlib.h>
#include <unistd.h>

int fildes[2];
/* buf size is 100 */
char buf[100];
int status;

int main()
{
    status = pipe(fildes);
    if (status == -1) {
        /* an error occurred */
        printf("error\n");
    }

    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 */
            yield();
            close(fildes[1]); /* Write end is unused */
            read(fildes[0], buf, 100); /* Get data from pipe */
            printf("child-process read:%s", buf); /* Print the data */
            close(fildes[0]);
    }
}
```

6.2 上面这种不同步修改pp_ref 而导致的进程竞争问题在user/fd.c 中的dup 函数中也存在。请结合代码模仿上述情景，分析一下我们的dup 函数中为什么会出现预想之外的情况?

dup函数也不是原子性的，会被时钟中断打断，而在未修改的dup函数中，先map了fd，然后才map了data，如果在两步的中间发生了中断的话，就可能会出现上述错误。

6.3 阅读上述材料并思考：为什么系统调用一定是原子操作呢？如果你觉得不是所有的系统调用都是原子操作，请给出反例。希望能结合相关代码进行分析。

系统调用在陷入内核的时候，会关闭时钟中断，这样整个系统调用过程就不会被打断了，所以是原子操作。

6.4 仔细阅读上面这段话，并思考下列问题

- **按照上述说法控制pipeclose 中fd 和pipe unmap 的顺序，是否可以解决上述场景的进程竞争问题？给出你的分析过程。**
- **我们只分析了close 时的情形，那么对于dup 中出现的情况又该如何解决？请模仿上述材料写写你的理解。**

可以解决。因为pipe的pageref本来就是大于fd的，在unmap的时候，优先接触fd的映射，就会保证严格大于关系恒成立，即使被时钟中断打断，也不会发生错误。dup同理。

6.5 bss 在ELF 中并不占空间，但ELF 加载进内存后，bss 段的数据占据了空间，并且初始值都是0。请回答你设计的函数是如何实现上面这点的?

C 语言中未初始化的全局变量，我们需要为其分配内存，但它又不需要被初始化成特定数据。在lab3的load_icode_mapper函数中，可以看到，当bin_size < sgsize的时候，会往空的位置填充0，这个差值就是bss段，所以C 语言中全局变量会有默认值0。

6.6 为什么我们的*.b 的text 段偏移值都是一样的，为固定值?

user的link script文件中约定了.text段的地址。

6.7 在哪步，0 和1 被” 安排” 为标准输入和标准输出?

在user的init.c中。

二、实验难点

1. spawn函数是lab6的一大难点，因为lab6的shell部分实现主要就是用这个函数，在填写这个函数的时候我也琢磨了好久，然后通过模仿lab3的加载二进制文件，一步一步写完的。
2. 其次，lab6的槽点也很多。首先就是官方代码的bug，fwritef函数是错的，导致很多同学（包括我）第二部分始终不能AC，后来在修改后才能通过。

三、感想与体会

经历重重困难，终于通过了操作系统的所有实验。对于lab6，个人花费时间在20小时左右，包括读代码，理解流程，填写函数等等。我感觉收获还是挺大的，了解了shell里面各种命令的实现方法，还有管道、重定向操作等等。

os实验到目前为止也就告一段落了，从lab0到lab6一路走下来，遇到了很多困难，我也很感谢老师同学助教们帮我解决问题，帮助我一步一步地前进，学习OS实验让我对一个操作系统各种功能的最基本的实现有了一定的了解，或许这在以后能派上用场。另一方面，OS实验对C语言也是一个很好的复习。