Lab6实验报告

思考题

Thinking 6.1

- 交换写端和读端代码即可

```
    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);
    }
```

Thinking 6.2

- `dup` 中包含两个过程：
  - 将 `newfd` 所在的虚拟映射到 `oldfd` 所在的物理页
  - 将 `newfd` 的数据所在的虚拟页映射到 `oldfd` 的数据所在的物理页

  如果在执行过程中，父进程刚执行完第一个过程，此时因为时钟中断让子进程继续执行，则子进程会判断写进程已关闭

Thinking 6.3

- 每次执行系统调用前都会关中断，因此可以直到系统调用一定是原子操作，从而避免再次触发异常处理，确保当前异常处理函数正常执行完毕

Thinking 6.4

- 可以解决。如果程序正常执行，那么 `pipe` 的 `pageref` 要大于 `fd` 的，在执行 `unmap` 操作时，优先解除 `fd` 的映射，可以保证大于关系恒成立，即使发生了时钟中断，也不会出现错误
- 在 `dup` 引用次数增加时，先将 `pipe` 的引用次数增加，保证不会出现两者相等的情况

Thinking 6.5

- 对于 `bss` 段中和 `text/data` 段共同占据一个页面的部分，就将这一部分置零；对于 `bss` 段其它部分，仅使用 `syscall_mem_malloc` 分配页面而不映射到任何内容

Thinking 6.6

- 在 `user/init.c` 中有如下代码

```
// stdin should be 0, because no file descriptors are open yet
if ((r = opencons()) != 0) {
    user_panic("opencons: %d", r);
}
// stdout
if ((r = dup(0, 1)) < 0) {
    user_panic("dup: %d", r);
}
```

Thinking 6.7

- shell 命令是外部命令，因为执行 shell 命令时，当前进程会通过 `fork` 产生一个子 shell，符合外部命令的定义
- Linux 中的内部命令由 shell 程序识别并在 shell 程序内部完成运行，通常在 linux 系统加载运行时就被加载并驻留在系统内存中，`cd` 指令非常简单，将其作为内部指令可以避免每次执行都需要 `fork` 并加载程序，提高执行效率

Thinking 6.8

- 生成 2 次 `spawn`，分别对应 `ls.b` 和 `cat.b`
- 销毁了 4 次

```
$ ls.b | cat.b > motd
[00002803] pipecreate
[00003805] destroying 00003805
[00003805] free env 00003805
```

```
i am killed ...
[00004006] destroying 00004006
[00004006] free env 00004006
i am killed ...
[00003004] destroying 00003004
[00003004] free env 00003004
i am killed ...
[00002803] destroying 00002803
[00002803] free env 00002803
i am killed ...
```

实验难点

重点在于对管道代码的底层原理以及实现，最后封装好一整个完整的操作系统供用户使用

在实验过程中，对于函数的调用过程，一般都是覆盖了前边的所有 Lab，可能需要很多的回顾以及反复琢磨观察，且大部分函数需要补全的部分非常多，需要跟着提示一步一步进行推进

体会与感想

整个学期的 OS 基础 Lab 也是告一段落了，无论是理论还是实验（特别是上机）难度都是蛮高的，但是可以在一学期结束时基本完整地搞定一个操作系统还是蛮爽地，也是感谢助教在讨论区的帮助！