实验思考题

6.1

- ```
  #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:
          yield();                            /* Parent - reads from pipe */
          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

- dup函数本身并不是原子性的。而在原版代码中，先共享了fd本身，后共享了data，如果中断发生在两个共享之间，就会发生描述中出现的错误

6.3

- 系统调用分两部分，内核态和用户态
- 内核态是原子的，毕竟已经把钟关了
- 用户态被中断好像也没什么事，毕竟只syscall这一条指令卡在了中间

6.4

- 可以。因为此时不正确的配套fd与data中，data的pageref严格大于fd（本来有可能相等的）
- 可以。与上文分析类似
  ```

6.5

- 加载的时候，不是简简单单的把elf本身加进去，针对每个phdr，直接读bin_size和sg_size，仿照lab3写即可。bss端需要的空间已经在phdr里面标好了

6.6

- user的link script文件标记好了

6.7

- 在user的init.c中

难点图示

- lab6前面的部分还挺简单的，重点就在于线程安全相关问题。
- 老版指导书有关spawn的描述太差了，完全不知道出题人想干什么。经过反馈，指导书进行了更新。新版指导书总体感觉还行，起码照着指导书能把代码填出来了
- 有两个坑点，一个是spawn从文件读的时候，要先copy到一个tmp页，然后再map过去，否则会引发写回时的错误（因为这一页被共享了）。另一个是官方下发的fwritef是错的

指导书反馈

- spawn里面为啥要用fork创建而不是syscall_env_alloc呢？在群里询问之后也没有得到好的解释。我目前倾向于指导书写错了

感想

- 相比于lab4-1神奇的换文件操作，lab4-2-exam的神奇评测，lab5-2-extra神奇的反斜杠，lab6在很多地方做的都比其他lab好
- 第一点，终于有助教出来主动答疑了。虽然lab6的fwritef是官方bug，但最后做出了合情合理的解释。
- 第二点，终于公开部分评测机制了。连评测机怎么测都不知道，如何debug？构造样例之后本地怎么搭对拍器？虽然我在公开评测大致信息之前就通过了课下，但我仍对这一行为表示赞同

建议

- 放几个能用的链接，方便同学本地搭建OS实验环境，现有链接好几个失效的
- 评测白盒化，这样可以本地搭评测机
- 公开git仓库，连公网，允许学生本地实验，提交
- 在以上建议的基础上，增加查重力度。坦白讲，今年的查重就是个笑话