Problem V1: IO (16 points)

1. iocic (2′)

2. [1] File A (or FILE B) [2] foo.txt (or bar.txt)

   [3] 3 [4] 0

每空一分，若[1]回答 File A，则[2]必须为 foo.txt；若[1]回答 FILE B，则[2]必须为 bar.txt。

3. foo.txt: oocococicam

   bar.txt: icsy

每空 3 分，foo.txt 中错一个字母（其余字母均正确）给 2 分

4. ociic (or ocici) (2′)

   FIX: fflush(stdout); (2′)

   前者两种答案均得 2 分，第二问只要出现了 fflush 均得 2 分

Problem V2: IO (16 points)

1. NIONO

2. [1] File A (or FILE B) [2] foo.txt (or bar.txt)

   [3] 3 [4] 0

3. foo.txt: IIOIOIONO

   bar.txt: NOTU

4. IONNO (or IONON)

   FIX: fflush(stdout);

   评分细则与 V1 相同

Problem W1: Schedule (16 points)

1.

[1] 10ms [2] 4.5ms [3] 9.5ms [4] 4ms

[5] 9.5ms [6] 4ms (或 3ms)

7.8 两校空三种答案都接受

[7] 13ms [8] 1ms

[7] 11.5ms [8] 2.5ms

[7] 12ms [8] 1.5ms

2.

[1] A [2] B [3] C [4] C

[5] D [6] C [7] 15 [8] 0.5

原子性，每空要么 1 分，要么 0 分。

Problem W2: Process (16 points)

1.

3 times. 6 times.

2.

No.

3.

[C,0,4][C,1,9][P,1,8][P,0,3][C,1,7][P,1,6]

答案唯一，顺序不对，内容对，能得1.5分

4.

父进程的输出需要等到子进程打印出所有输出并终止后才会打印。

需要提及子进程终止前所有输出，而不只是fork后的两句输出的相对关系，否则扣一分。

Problem X1: Signal (18 points)

1. (3')

Line1	Line2	Line3	Line4	Line5	Line6	Line7	Line8
S1	1	S2	1	S2	S3	S3	2
Line9	Line10	Line11	Line12	Line13	Line14	Line15	Line16
S4	2	S4	S4				

输出行数为 12 行且输出内容相同，顺序不同均得 3 分。其他情况酌情给分。

2. [1] 15 [2] 3 每空 2 分，答错不给分

3. [1] 42 [2] 5 每空 2 分，答错不给分

4. [1] 15 [2] 4 每空 2 分，答错不给分

[3] No. The memory spaces of the parent and child are isolated. (3')

回答 Yes 不给分；只回答 No 得 1 分；说父子进程 cnt 变量是独立的给 3 分。

Problem X2: Signal (18 points)

1.

Line1	Line2	Line3	Line4	Line5	Line6	Line7	Line8
S1	1	S2	1	S2	S3	S3	0
Line9	Line10	Line11	Line12	Line13	Line14	Line15	Line16
S4	0	S4	S4				

2. [1] 21 [2] 4
3. [1] 37 [2] 5
4. [1] 15 [2] 4
   [3] No. The memory spaces of the parent and child are isolated.
   评分细则与 X2 相同。

Problem Y1: Network (24 points)

ANS:

```
    /* (2) correctly open connection */
    int clientfd = Open_clientfd("ipads.se.sjtu.edu.cn", 8080);
    /* (2) correctly prepare message */
    sprintf(send_buf, "LABHISTORY\n%s\n%s\n", stu_id, pass);
    /* (2) correctly send out message */
    Rio_writen(clientfd, send_buf, strlen(buf));
```

```
    rio_t rio;
    Rio_readinitb(&rio, clientfd);

    int valid = 0;
    while (1) {
        /* (2) correctly readline */
        Rio_readlineb(&rio, recv_buf, 1024);
        /* (2) correctly handle INCORRECT ACCOUNT */
        if (!strcmp(recv_buf, "INCORRECT ACCOUNT\n")) {
            printf("Incorrect account!\n");
            Close(clientfd);
            return -1;
        }
```

```
        /* (2) correctly iterate each line until reach the end */
        if (!strcmp(recv_buf, "END LABHISTORY\n")) {
            break;
        }
        /* (2) correctly match lab name */
        if (!strcmp(recv_buf, labname)) {
            valid = 1;
            break;
        }
    }
    /* (2) correctly handle missing lab name */
    if (!valid){
        printf("Invalid labname!\n"); Close(clientfd); return -1;
    }
    /* (1) no other incorrect coding */
```

```
    /* (2) correctly prepare GRADE request */
    sprintf(send_buf, GRADE\n%s\n%s\n%s\n", labname, stu_id, pass);
    /* (2) correctly send out message */
    Rio_writen(clientfd, send_buf, strlen(buf));
```

```
    /* (1) correctly read message */
    Rio_readlineb(&rio, recv_buf, 1024);
    /* (2) correctly print grade */
    printf("%s", recv_buf);
    Close(clientfd);
```

Problem Y2: Network (24 points)

ANS:

```
/* (2) correctly open connection */
int clientfd = Open_clientfd("ipads.se.sjtu.edu.cn", 8888);
/* (2) correctly prepare message */
sprintf(send_buf, "LABHISTORY\n%s\n%s\n", stu_id, pass);
/* (2) correctly send out message */
Rio_writen(clientfd, send_buf, strlen(buf));
```

```
rio_t rio;
Rio_readinitb(&rio, clientfd);

int valid = 0;
while (1) {
    /* (2) correctly readline */
    Rio_readlineb(&rio, recv_buf, 1024);
    /* (2) correctly handle INCORRECT ACCOUNT */
    if (!strcmp(recv_buf, "INCORRECT ACCOUNT\n")) {
        printf("Incorrect account!\n");
        Close(clientfd);
        return -1;
    }
    /* (2) correctly iterate each line until reach the end */
    if (!strcmp(recv_buf, "END LABHISTORY\n")) {
        break;
    }
    if (!strcmp(recv_buf, "BEGIN LABHISTORY\n")) {
        continue;
    }
    /* (2) correctly match lab name */
    if (!strcmp(recv_buf, labname)) {
        valid = 1;
        break;
    }
}

/* (2) correctly handle missing lab name */
if (!valid){
    printf("Invalid labname!\n"); Close(clientfd); return -1;
}
```

```
/* (1) no other incorrect coding */
```

```
/* (2) correctly prepare GRADE request */
sprintf(send_buf, GRADE\n%s\n%s\n%s\n", stu_id, pass, labname);
/* (2) correctly send out message */
Rio_writen(clientfd, send_buf, strlen(buf));
```

```
/* (1) correctly read message */
Rio_readlineb(&rio, recv_buf, 1024);
/* (2) correctly print grade */
printf("%s", recv_buf);
Close(clientfd);
```

Problem Z1: Lock (26 points)

下面所有小题，主体正确，但是回答中有事实错误的-1 分，直接答到 ticket lock 那边不给分

1. Yes. IDEL only make the thread wait for a while, which will not influence the correctness of the spin lock. (4') 只回答 yes 不解释给 3 分
2. No. Thread A may acquire the lock ealier than thread B. However, the lock has been hold when thread A TAS the flag. When thread A being schedule out, the holder release the lock and thread B can successfully acquire the lock before thread A does. (4')

   只回答 No 不解释给 3 分
3. Reduce the contention and memory references towards the flag, thus the TAS operation can run faster. (3')
   只说 memory access 或者只抽象的说节省 CPU 的给 2'
4. When the IDEL time (c) is set to an inappropriate value, the lock competitor may still in IDEL state when the lock is free. In such case, the time will be wasted. (3')
5. No. Suppose we have 2 threads. Thread A hold the lock and thread B is going to acquire the lock. If thread A check the **wait** and call **unpark** before thread B call **park** in line 5, thread A will not wake the thread B successfully. And thread B will wait forever. (6')
6. 6'

```
void lock(lock_t *lock) {
     do {
        while(lock->flag); // spin
     } while (TAS(&lock->flag, 1));
}
```

其他思路 解释对给 3' 代码对给 3'

强行说要用 ticketlock 的给 2'

其他能保证正确性，但是没优化的给 3’

Problem Z2: Lock (26 points)

下面所有小题，主体正确，但是回答中有事实错误的-1 分

1. Yes. IDEL only make the thread wait for a while, which will not influence the correctness of the ticket lock. (4’) 只回答 yes 不解释给 3 分
2. Yes. Each lock competitor will be assigned a unique ticket according to time sequence of lock acquisition, and the lock will be passed according to the ticket. (4’) 只回答 yes 不解释给 3 分
3. Reduce the contention and memory references towards the **`turn`**. (3’)
4. When the IDEL time (c) is set to an inappropriate value, the lock competitor may still in IDEL state when the lock is free. In such case, the time will be wasted. (3’)
5. No. Suppose we have two threads: thread A and thread B. Thread A holds the lock and thread B is acquiring the lock. When those threads have the following interleaving: thread B line 3, 4 => thread A line 9, 10, 11 => thread B line 5

   In such a case, thread B will never be woke up. (6’)

   说不对给 3 分，解释正确再给 3 分 0
6. (6’)

```
void lock(lock_t *lock) {
   int myturn =
       FAA(&lock->ticket);
   while (lock->turn != myturn)
       IDLE(myturn-lock->turn);
}
```

其他思路 解释对给 3’ 代码对给 3’

其他能保证正确性，但是没优化的（给了代码的）给 3’