装订线内

不要答题

北京大学信息科学技术学院考试试卷

考试科目： 计算机系统导论 **姓名：** _____ **学号：** _____

考试时间： 2014 年 1 月 7 日 **任课教师：** _____

题号	一	二	三	四	五	六	七	八	**总分**
分数									
阅卷人									

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题和答题纸，共 20 页。

得分

第四题（10 分）

考虑如下两个程序（fact1.c和fact.c）：

```
/* fact1.c */
#define MAXNUM 12
int table[MAXNUM];
int fact(int n);
int main(int argc, char **argv) {
    int n;
    table[0] = 0;
    table[1] = 1;
    if (argc == 1) {
        printf("Error: missing argument\n");
        exit (0);
    }
    argv++;
    if (sscanf(*argv, "%d", &n) != 1 || n < 0 || n >= MAXNUM)
    {
        printf ("Error: %s not an int or out of range\n",
    *argv);
        exit (0);
    }
    printf("fact(%d) = %d\n", n, fact(n));
}

/* fact2.c */
int* table;
int fact(int n) {
    static int num = 2;
    if (n >= num) {
        int i = num;
        while (i <= n) {
            table[i] = table[i-1] * i;
            i++;
        }
```

```
        num = i;
    }
    return table[n];
}
```

（1）对于每个程序中的相应符号，给出它的属性（局部变量、强全局变量或弱全局变量），以及它在链接后位于 ELF 文件中的什么位置？（提示：如果某表项中的内容无法确定，请画 X）（6 分）

fact1.c

变量	类型	ELF Section
table		
fact		
num		

fact2.c

变量	类型	ELF Section
table		
fact		
num		

（2）对上述两个文件进行链接之后，会对每个符号进行解析。请给出链接后下列符号被定义的模块（fact1 or fact2）。（2 分）

	定义模块
table	
fact	
num	

（3）使用 gcc（命令：gcc -o fact fact.c fact.c）来编译之后得到的可执行文件是否能够正确执行？为什么？（2 分）

得分

第五题（10 分）

Part I

请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行）：

```
int main() {
    printf("A\n");
    if (fork() == 0) {
        printf("B\n");
    }
    else {
        printf("C\n");
        A
    }
    printf("D\n");
    exit(0);
}
```

（1）如果程序中的 A 位置的代码为空，列出所有可能的输出结果：（1 分）

（2）如果程序中的 A 位置的代码为：waitpid(-1, NULL, 0);
列出所有可能的输出结果：（2 分）

（3）如果程序中的 A 位置的代码为：printf("E\n");
列出所有可能的输出结果：（2 分）

Part II

请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行，且每条语句都是原子动作）：

```
pid_t pid;
int even = 0;
int counter1 = 0;
int counter2 = 1;
void handler1(int sig) {
    if (even % 2 == 0) {
        printf("%d\n",
    counter1);
        counter1 = ____A____ ;
        } else {
        printf("%d\n",
    counter2);
        counter2 = ____B____ ;
        }
        even = ____C____ ;
}
void handler2(int sig) {
        if (____D____) {
        counter1 = even * even;
        } else {
        counter2 = even * even;
        }
}
int main() {
    signal(SIGUSR1, handler1);
    signal(SIGUSR2, handler2);
    if ((pid = fork()) == 0) {
        while (1) {};
        }
        while (even < 20) {
            kill(pid, ___E___);
            sleep(1);
            kill(pid, ___F___);
            sleep(1);
            even += 2;
        }
        kill(pid, SIGKILL);
        exit(0);
    }
```

1)．完成程序，使得程序在输出前 20 个斐波那契(Fibonacci)数列，即 $F_0=0$, $F_1=1$, …, $F_n=F_{n-1}+F_{n-2}$。（如果存在对本次程序执行结果没有影响的语句，请在相应位置填写“无关”）（3 分）

A: _____

B: _____

C: _____

D: _____

E: _____

F: _____

2)．完成程序，其中 A，B 处保持不变，使得程序可以分别输出前几个奇数或偶数的平方和。（如果存在对本次程序执行结果没有影响的语句，请在相应位置填写“无关”）（2 分）

其中：

若要输出奇数的平方和：even 的初始值为 **3**；

若要输出偶数的平方和，even 的初始值为 **2**。

C: _____

D: _____

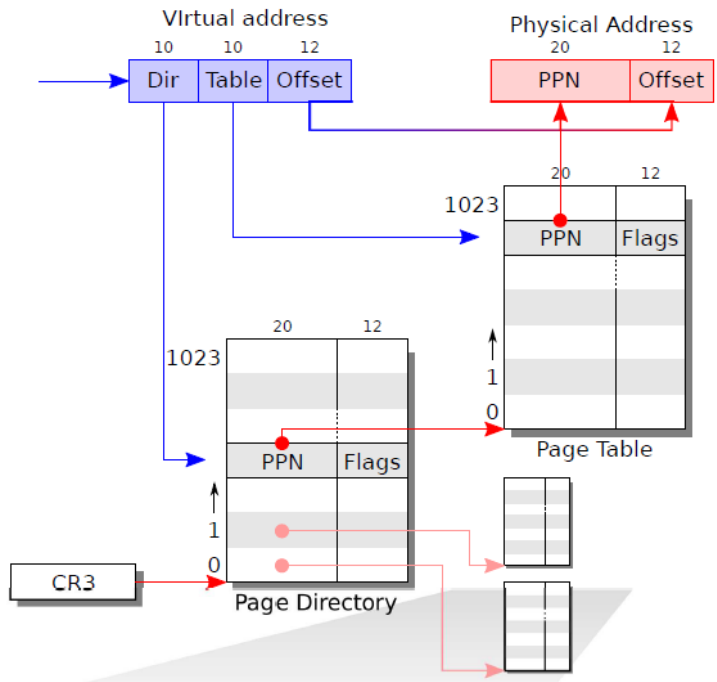
E: _____

F: _____

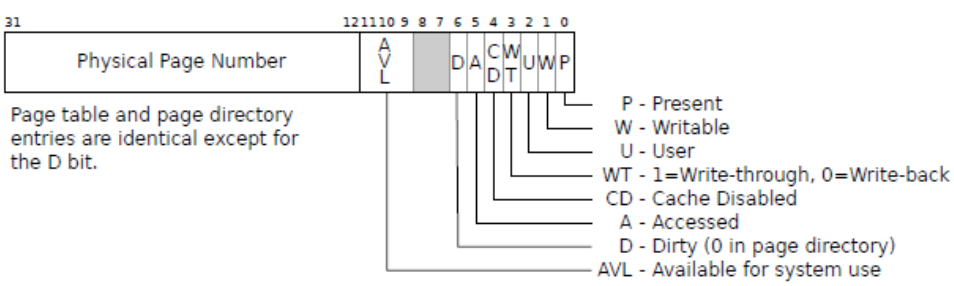
得分

第六题（10 分）

Intel 的 IA32 体系结构采用二级页表，称第一级页表为页目录(Page Directory)，第二级页表为页表(Page Table)。其虚拟地址到物理地址的翻译方式如下图。先根据 CR3 找到页目录地址，然后依据偏移 Dir 找到一个页目录项，页目录项的高 20 位(PPN)为二级页表地址；在二级页表中根据偏移 Table 找到页表项，页表项中的高 20 位(PPN)即为物理地址的高 20 位，将这 20 位与虚拟地址的低 12 位拼在一起形成完整的物理地址。



页目录和页表均有 1024 项，每一项为 4 字节，含义如下：



页目录和页表由操作系统维护，通常只能在内核态下访问，为了给用户提供一个访问页表项和页目录项内容的接口，假设操作系统中已经执行过如下代码段：

```
#define UVPT 0xef400000
#define PDX(la) ((((unsigned int) (la)) >> 22) & 0x3FF)
……
kern_pgdir[PDX(UVPT)] = PADDR(kern_pgdir) | PTE_U | PTE_P;
```

其中 kern_pgdir 是操统系统维护的页目录数组，共 1024 项，每一项的类型为 unsigned int。PADDR(kern_pgdir)用于获得 kern_pgdir 的物理地址，页目录在物理内存中正好占一页，所以 kern_pgdir 的物理地址是 4KB 对齐的。PTE_U 和 PTE_P 代表了这个页目录项的权限，即用户态可访问(只读)。可以看到，这条语句将页目录的第 PDX(UVPT)项指向了页目录自身。

利用这一点，对于给定的虚拟地址 va，可以获得 va 对应的页目录项和页表项内容，分别对应于函数 get_pde 和 get_pte，请完成这两个函数(每空 2 分)：

```
#define UVPT 0xef400000
// get_pde(va)：获取虚拟地址 va 对应的一级页表(页目录)中的页目录项内容
unsigned int get_pde(unsigned int va) {
    unsigned int pdx = (va >> _[1]_____ ) & _[2]_____;
    unsigned int addr = UVPT + (_[3]_____) + pdx * 4;
            return *((unsigned int *)(addr));
}
// get_pte(va)：获取虚拟地址 va 对应的二级页表中的页表项内容
unsigned int get_pte(unsigned int va) {
    unsigned int PGNUM = va >> ___[4]_____;
    unsigned int addr = ___[5]_____ + PGNUM;
    return *((unsigned int *)(addr));
}
```

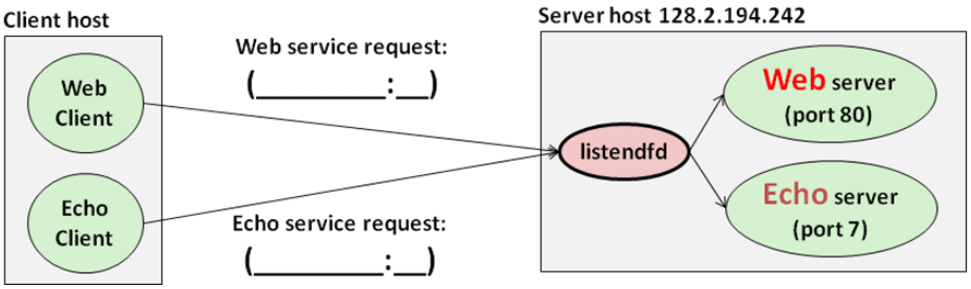
得分

第七题（10 分）

（1） 一个服务器拥有两个独立的固定 IP 地址，那么它在 web 应用端口 80 上最多可以监听多少个独立的 socket 连接？（2 分）

（2）该服务器在所有有 web 应用端口上最多可以监听多少个独立的 socket 连接？（2 分）

（3） 在下图中连线上填入正确的目标服务器的 socket 标识符（2 分）



（4）在 Echo server 范例中，server 端通过 accept 函数接受了一个 client 的连接请求，从而将网络描述符与该网络连接、socket 绑定，然后进行网络数据传输。在下面的空格处填写正确的网络描述符，每个空填写 **listenfd** 或 **connfd**（共 4 分，每空 1 分）。

```
int main(int argc, char **argv) {

    int listenfd, connfd, port, clientlen;

    struct sockaddr_in clientaddr;

    struct hostent *hp;

    char *haddrp;

    unsigned short client_port;

      ……

      while (1) {

          clientlen = sizeof(clientaddr);

          __________ = Accept(__________, (SA *)&clientaddr,

                      &clientlen);

          hp = Gethostbyaddr((const char*)

               &clientaddr.sin_addr.s_addr,

               sizeof(clientaddr.sin_addr.s_addr), AF_INET);

          haddrp = inet_ntoa(clientaddr.sin_addr);

          client_port = ntohs(clientaddr.sin_port);

          printf("server connected to %s (%s), port %u\n",

                    hp->h_name, haddrp, client_port);

          echo(__________);

          Close(__________);

          }

      }
```

得分

第八题（10 分）

某个城市为了解决市中心交通拥堵的问题，决定出台一项交通管制措施，对进入市中心区的机动车辆实行单双日限制行驶的办法。具体要求是，逢单日，只允许车辆牌号号码为单数的机动车进入市中心区；同样，逢双日，只允许车辆牌号号码为双数的机动车进入市内中心区。有一个进入市中心区的交通路口，进入该路口的道路有一条，离开该路口的道路有两条，其中一条是通往市中心区的道路，而另一条是绕过市中心区的环路，在进入路口处设置了自动识别车辆牌号的识别设备与放行栅栏控制设备。在单日，遇到单号车辆进入路口车辆号码识别区，号码识别设备打开通往市中心区道路的放行栅栏；而遇到双号车辆，则打开绕过市中心区环路的放行栅栏。反之亦然。显然，只有在该路口车辆号码识别区中无车时，才允许一辆车进入车辆号码识别区。同时为了防止有车辆混过路口，两个放行栅栏平时处于关闭状态，只有在车辆号码识别区中的车辆已被识别出单双号之后，放行栅栏才会在识别设备的控制下，打开对应的放行栅栏，在车辆通过之后，该放行栅栏自行关闭。

```
vehicle_n  int;                    /* 车辆号码 */
```

检查车辆牌号线程 T1：

```
 while (1) {
    车辆到达识别区路口；
    ①
    车辆进入号码识别区；
    if  (vehicle_n == 奇数)
       {    ②       }
    else
       {    ③       }
  };
```

市区放行栅栏线程 T2：

```
 while (1) {
    ④
    允许车辆进入市中心区；
    ⑤
  };
```

环路放行栅栏线程 T3：

```
while (1) {
   ⑥
   允许车辆绕行环路；
   ⑦
};
```

（1）请设计若干信号量，给出每一个信号量的作用和初值。（3 分）

（2）请将信号量上对应的 PV 操作填写在代码中适当位置。（7 分）

装订线内

不要答题

北京大学信息科学技术学院考试试卷

考试科目： 计算机系统导论 **姓名：** _____ **学号：** _____

考试时间： 2015 年 1 月 13 日 **小班教师：** _____

题号	一	二	三	四	五	六	七	八	九	**总分**
分数										
阅卷人										

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题和答题纸，共 24 页。

得分

第四题（10 分）链接

考虑如下3个文件：main.c, fib.c和bignat.c：

```
/* main.c */
void fib (int n);
int main (int argc, char** argv) {
    int n = 0;
    sscanf(argv[1], "%d", &n);
    fib(n);
}
```

```
/* fib.c */
#define N 16

static unsigned int ring[3][N];

static void print_bignat(unsigned int* a) {
    int i;
    for (i = N-1; i >= 0; i--)
        printf("%u ", a[i]); /* print a[i] as unsigned int
        */
    printf("\n");
}
void fib (int n) {
    int i, carry;
    from_int(N, 0, ring[0]); /* fib(0) = 0 */
    from_int(N, 1, ring[1]); /* fib(1) = 1 */
    for (i = 0; i <= n-2; i++) {
        carry   =   plus(N,   ring[i%3],   ring[(i+1)%3],
        ring[(i+2)%3]);
        if (carry)
            {   printf("Overflow   at   fib(%d)\n",   i+2);
        exit(0); }
    }
    print_bignat(ring[n%3]);
}
```

另外，假设在文件 bignat.c 中定义了如下两个函数 plus 和 from_int（具体定义略）：

```
int plus (int n, unsigned int* a, unsigned int* b, unsigned
int* c);
void from_int (int n, unsigned int k, unsigned int* a);
```

1．（5 分）对于每个程序中的相应符号，给出它的属性（局部或全局，强符号或弱符号）（提示：如果某表项中的内容无法确定，请画 X。）

main.c

	局部或全局?	强或弱?
fib		
main		

fib.c

	局部或全局?	强或弱?
ring		
fib		
plus		

2．（3 分） 假设文件 bignat.c 被编译为一个静态库 bignat.a，对于如下的 gcc 调用，会得到什么样的结果（请选择）？

（A） 编译和链接都正确

（B） 链接失败（原因是包含未定义的引用）

（C） 链接失败（原因是包含重复定义）

命令	结果（A，B 或 C）
gcc -o fib main.c fib.c bignat.a	
gcc -o fib bignat.a main.c fib.c	
gcc -o fib fib.c main.c bignat.a	

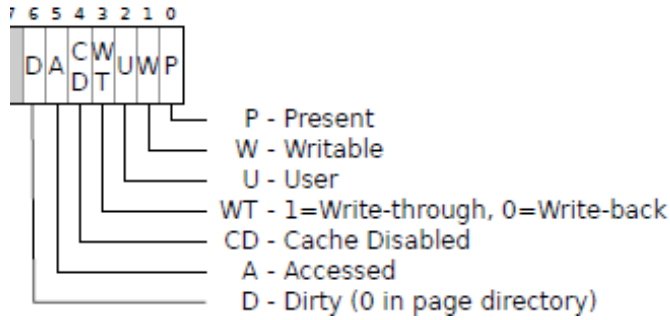
3．（2 分）如果在文件 fib.c 中，程序员在声明变量 ring 时，不小心把它写成了：static int ring[3][N];

会不会影响这些文件的编译、链接和运行结果？为什么？

得分

第五题（10 分）虚拟存储

Intel 的 IA32 体系结构采用小端法和二级页表。其中两级页表的大小相同，页大小为 4KB。一级页表和二级页表的表项结构相同，其中页表项后六位的含义如下。



已知一级页表的地址为 0x0c23b000，物理内存中的部分内容如下图所示。

地址	内容	地址	内容	地址	内容	地址	内容
00023000	E0	00023001	BE	00023002	EF	00023003	BE
00023120	83	00023121	C8	00023122	FD	00023123	12
00023200	23	00023201	FD	00023202	BC	00023203	DE
00023320	33	00023321	29	00023322	E5	00023323	D2
00023FF8	29	00023FF9	FF	00023FFA	DE	00023FFB	BC
00055004	03	00055005	D0	00055006	74	00055007	89
0005545C	97	0005545D	C2	0005545E	7B	0005545F	45
00055460	97	00055461	D2	00055462	7B	00055463	45
00055464	97	00055465	E2	00055466	7B	00055467	45
0C23B020	55	0C23B021	EB	0C23B022	AE	0C23B023	24
0C23B040	55	0C23B041	AB	0C23B042	2A	0C23B043	01
0C23B080	05	0C23B081	5D	0C23B082	05	0C23B083	00
0C23B09D	05	0C23B09E	D3	0C23B09F	F2	0C23B0A0	0F
0C23B274	05	0C23B275	3D	0C23B276	02	0C23B277	00
0C23B9FC	25	0C23B9FD	D2	0C23B9FE	14	0C23B9FF	23
2314D200	23	2314D201	12	2314D202	DC	2314D203	0F
2314D220	A9	2314D221	45	2314D222	13	2314D223	D2
2314D4A0	BD	2314D4A1	BC	2314D4A2	88	2314D4A3	D3

2314D890	00	2314D891	2D	2314D892	B3	2314D893	00
24AEE001	07	24AEE002	A0	24AEE003	37	24AEE004	C2
24AEE520	D1	24AEE521	DA	24AEE522	8C	24AEE523	B5
29DE2504	02	29DE2505	AD	29DE2506	FF	29DE2507	56
29DE4400	D0	29DE4401	5C	29DE4402	B4	29DE4403	2A
29DE9402	00	29DE9403	20	29DE9404	73	29DE9405	D4
29DEE500	B0	29DEE501	CD	29DEE502	23	29DEE503	1A

TLB 采用直接映射，TLB 的内容如下所示。

索引	TLB 标记	内容	有效位
0	0x08001	2314d220	1
1	0x01000	24aee520	0
2	0x005AE	00055004	0
3	0x016BA	0c23b09d	1
4	0x0AA00	0005545c	1
5	0x0000A	29dee500	0
6	0x5AE82	00023320	1
7	0x28DFC	00023000	1

1．（2 分）某用户态进程试图写入虚拟地址：0x080016ba。该访问的最后结果是_____。

(a) 该进程成功写入，未触发异常

(b) 该进程触发了一个缺页异常

(c) 该进程触发了一个非法访问异常

2.下面描述了具体的访问过程，请填空。如果某个空在访问过程中已不可用，请填入“--”

2.1 TLB 的索引为_____，访问为_____ (a) 命中 (b) 不命中（2 分）

2.2 一级页表表项地址为_____。（2 分）

2.3 二级页表表项地址为_____。（2 分）

2.4 最后物理地址为_____。（2 分）

得分

第六题（10 分）异常

1.（5 分）以下程序运行时系统调用全部正确执行，buffer.txt 文件的内容为 pekinguniv。请给出代码运行后打印输出的结果，并给出程序运行结束后 buffer.txt 文件的内容。

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

int main() {
    char c;
    int file1 = open("buffer.txt", O_RDWR);
    int file2;

    read(file1, &c, 1);
    file2 = dup(file1);
    write(file2, &c, 1);
    printf("1 = %c\n", c);

    int pid = fork() ;
    if (pid == 0) {
        read(file1, &c, 1);
        write(file2, &c, 1);
        printf("2 = %c\n", c);
        read(file1, &c, 1);
        printf("3 = %c\n", c);
        close(file1);
        exit(0);
    } else {
        waitpid(pid, NULL, 0);
        close(file2);
        dup2(file1, file2);
        read(file2, &c, 1);
        write(file2, &c, 1);
        printf("4 = %c\n", c);
    }
    return 0;
}
```

答：

2.(5 分)某程序员实现了一个课程实验用的操作系统 ICSNIX,其系统函数 sleep 用以下代码实现。请分析该代码存在哪些问题。

```
1 #include    <signal.h>
2 #include    <unistd.h>
3 static void sig_alrm(int signo)
4 {
5    /* nothing to do, just return to wake up the pause */
6 }
7
8 unsigned int sleep(unsigned int seconds)
9 {
10    if (signal(SIGALRM, sig_alrm) == SIG_ERR)
11       return(seconds);
12
13    alarm(seconds); /* start the timer */
14    pause(); /* next caught signal wakes us up */
15    return(alarm(0)); /* turn off timer, return unslept
time */
16}
```

答：

得分

第七题（10 分）系统 I/O

请阅读下面的代码：

```
 1:  int main(int argc, char** argv) {
 2:      int    fd1    =    open("ICS.txt",    O_CREAT|O_RDWR,
 3:  S_IRUSR|S_IWUSR);
 4:      write(fd1, "abc", 3);
 5:
 6:      int fd2 = fd1;
 7:      int fd3 = dup(fd2);
 8:      int fd4 = open("ICS.txt", O_APPEND|O_RDWR);
 9:      write(fd2, "defghi", 6);
10:      write(fd4, "xyz", 3);
11:
12:      int fd5 = fd4;
13:      dup2(fd3, fd5);
14:      write(fd4, "pqr", 3);
15:
16:      close(fd1);
17:
18:      return 0;
19:  }
```

1.请填写在第 16 行代码刚刚执行完之后，下面的打开文件表和 v-node 表中表项的部分值，并画出表项之间的指向关系。（6 分）

初始时，ICS.txt 文件不存在。程序执行时，所有的系统调用均会成功，所有表项均会从上到下依次分配，描述符表一开始被占用掉前 3 个表项。对于已经释放的打开文件表表项，请填写释放前那一刻的值和指向的 v-node 表表项。

对于多余的表项，请直接忽略。

描述符表 Descriptor table

...
3
4
5
6
7

打开文件表 Open file table

pos	refcnt	释放?

v-node 表 v-node table

文件名

2.请填写在第 16 行代码刚刚执行完之后，下列变量的值（2 分）

fd1	fd2	fd3	fd4	fd5

3.请写出程序执行完之后，ICS.txt 文件中的内容（2 分）

得分

第八题（10 分）网络

1.（1 分）以下问题默认为 IPv4 协议。一个服务器拥有四个独立的固定 IP 地址，那么它在 web 应用端口 80，理论上可以最多再监听_____个来自一个客户端独立的 socket 连接（客户端只有一个固定 IP 地址）。

2.（2 分）在 client-server 模型中，一个连接（connection）可以由 IP 地址，端口号的组合来表示。假设客户端 IP 地址为 162.105.192.178，内网 IP 为 192.168.100.121。HTTP 服务器端 IP 地址为 208.216.181.15。

服务器使用的是默认监听端口号。

指出下面这个网页浏览器应用的 Connection socket pair 有什么错误，并简要说明原因？

客户端 IP：端口号	服务器端 IP：端口号
192.168.100.121:15321	208.216.181.15:25

答：

3.(4 分)在 Echo Server 程序中，客户端（Client）与服务器端（Server）通过 socket 进行一系列的命令和数据交互。

注意：客户端 Connect 命令包含在其 Open_clientfd 命令中。

请在下图中用单向箭头标出这些交互步骤。例如，当 Client 给 Server 端发送某个命令或者数据时，则需要在 Client 端相应代码行，朝向 Server 端相应代码行画一条单向箭头。

Echo client code

```
int main(int argc, char **argv){
  int clientfd, port;
    char *host, buf[MAXLINE];
    rio_t rio;
    host = argv[1];
    port = atoi(argv[2]);


    clientfd = Open_clientfd(host, port);

    Rio_readinitb(&rio, clientfd);

    printf("type:");
    fflush(stdout);

    while(Fgets(buf,MAXLINE,stdin)!= NULL) {

    Rio_writen(clientfd, buf, strlen(buf));

    Rio_readlineb(&rio, buf, MAXLINE);
        printf("echo:");
        Fputs(buf, stdout);
        printf("type:"); fflush(stdout);
      }



    Close(clientfd);
    exit(0);
  }
```

Echo Server code

```
int main(int argc, char **argv) {
   int listenfd, connfd, port, clientlen;
   struct sockaddr_in clientaddr;
   struct hostent *hp;
   char *haddrp;
   unsigned short client_port;
   port = atoi(argv[1]);

   listenfd = open_listenfd(port);
   while (1) {
     clientlen = sizeof(clientaddr);

     connfd=Accept(listenfd, (SA*)&clientaddr, &clientlen);

     hp=Gethostbyaddr((const char*)&clientaddr.sin_addr.s_addr,
        sizeof(clientaddr.sin_addr.s_addr), AF_INET);

     haddrp = inet_ntoa(clientaddr.sin_addr);

     client_port = ntohs(clientaddr.sin_port);
     printf("server connected");

     size_t n; char buf[MAXLINE];  rio_t rio;

     Rio_readinitb(&rio, connfd);

     while((n=Rio_readlineb(&rio,buf, MAXLINE)) != 0) {
       upper_case(buf);

       Rio_writen(connfd, buf, n);
      }

     Close(connfd);
    }
}
```

4.关于 Tiny Server 程序，请回答下列问题。

a.（1 分）下面这段服务器代码用来生成内容的文件是哪个参数？

b.（1 分）所生成的内容是静态还是动态？请简述原因。

c.（1 分）如果支持多个客户端请求，下面程序需要添加一个什么功能？

```
/* Return first part of HTTP response */
   sprintf(buf, "HTTP/1.0 200 OK\r\n");
   Rio_writen(fd, buf, strlen(buf));
   sprintf(buf, "Server: Tiny Web Server\r\n");
   Rio_writen(fd, buf, strlen(buf));

   /* Real server would set all CGI vars here */
   setenv("QUERY_STRING", cgiargs, 1);
   Dup2(fd, STDOUT_FILENO); /*Redirect stdout to socket and
client */
   Execve(filename, emptylist, environ);/* Run CGI prog  */
```

得分

第九题（10 分）并发

桌子上有一个水果盘，能容纳一个水果。一家四口人：爸爸、妈妈、儿子、女儿。爸爸专门往盘子里放苹果，妈妈专门往盘子里放桔子；儿子专等盘子里的苹果吃，女儿专等盘子里的桔子吃。

```
dad() {
   while(1) {
      准备好一个苹果;
          ①
      往果盘中放苹果;
          ②
   }
}
mom() {
   while(1) {
      准备好一个桔子;
          ③
      往果盘中放桔子;
          ④
   }
}
```

```
boy() {
   while(1) {
          ⑤
      从果盘中拿走苹果;
          ⑥
      吃苹果;
   }
}
girl() {
   while(1) {
          ⑦
      从果盘中拿走桔子;
          ⑧
      吃桔子;
   }
}
```

1. (2 分)请设计若干信号量，给出每一个信号量的作用和初值。

2. (8 分)请将信号量上对应的 PV 操作填写在代码中适当位置。

装订线内

不要答题

北京大学信息科学技术学院考试试卷

考试科目：计算机系统导论 姓名：_____ 学号：_____

考试时间：2016 年 1 月 4 日 小班教师：_____

题号	一	二	三	四	五	六	七	八	总分
分数									
阅卷人									

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题和答题纸，共 21 页。

得分

第四题（10 分）

在x86_64环境下，考虑如下2个文件：main.c和foo.c：

```
/* main.c */
#include <stdio.h>

long long _____________________________;
const char* foo(int);

int main(int argc, char **argv){
   int n = 0;
   sscanf(argv[1], "%d", &n);
   printf(foo(n));
   printf("%llx\n", a);
}

/* foo.c */
#include <stdio.h>

int a[2];

static void swapper(int num){
   int swapper;
   if (num % 2){
      swapper = a[0];
      a[0] = a[1];
      a[1] = swapper;
   }
}

const char* foo(int num){
   static char out_buf[50];
   swapper(num);
   sprintf(out_buf, "%x\n", _____________________________);
```

```
    return out_buf;
}
```

1．对于每个程序中的相应符号，给出它的属性（局部或全局，强符号或弱符号）（提示：如果某表项中的内容无法确定，请画 X。）

main.c

	局部或全局？	强或弱？
a		
foo		

foo.c

	局部或全局？	强或弱？
a		
foo		
out_buf		

2．根据如下的程序运行结果，补全程序【在程序空白处填空即可】。

```
$ gcc -o test main.c foo.c
$ ./test 1
bffedead
cafebffedeadbeef

$ ./test 2
beefcafe
deadbeefcafebffe
```

3．现在有一位程序员要为这个程序编写头文件。假设新的头文件名称为 foo.h，内容如下：

```
extern long long a;
extern char *foo(int);
```

然后在 main.c 和 foo.c 中分别引用该头文件，请问编译链接能通过吗？请说明理由。

得分

第五题（10 分）

以下程序运行时系统调用全部正确执行，且每个信号都被处理到。请给出代码运行后所有可能的输出结果。

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

int c = 1;

void handler1(int sig) {
   c++;
   printf("%d", c);
}


int main() {

   signal(SIGUSR1, handler1);
   sigset_t s;
   sigemptyset(&s);
   sigaddset(&s, SIGUSR1);
   sigprocmask(SIG_BLOCK, &s, 0);

   int pid = fork()?fork():fork();

   if (pid == 0) {
      kill(getppid(), SIGUSR1);
      printf("S");
      sigprocmask(SIG_UNBLOCK, &s, 0);
      exit(0);
   } else {
      while (waitpid(-1, NULL, 0) != -1);
```

```
            sigprocmask(SIG_UNBLOCK, &s, 0);
            printf("P");
        }
        return 0;
    }
```

答：

得分

第六题（15 分）

为了提升虚拟内存地址的转换效率，降低遍历两级页表结构所带来的地址转换开销，英特尔处理器中引入了大页 TLB，即一个 TLB 项可以涵盖整个 4MB 对齐的地址空间（针对 32 位模式）。只要设置页目录页中页目录项（PDE）的大页标志位，即可让 MMU 识别这是一个大页 PDE，并加载到大页 TLB 项中。大页 PDE 中记录的物理内存页面号必须是 4MB 对齐的，并且整个连续的 4MB 内存均可统一通过该大页 PDE 进行地址转换。

在 32 位的 Linux 系统中，为了方便访问物理内存，内核将地址 0~768MB 间的物理内存映射到虚拟内存地址 3GB~3GB+768MB 上，并通过大页 PDE 进行进行该区间的地址转换。任何 0~768MB 的物理内存地址可以直接通过加 3G（0xC0000000）的方式得到其虚拟内存地址。在内核中，除了该区间的内存外，其他地址的内存通常都通过普通的两级页表结构来进行地址转换。

假设在我们使用的处理器中有 2 个大页 TLB 项，其当前状态如下：

索引号	TLB **标记**	**页面号**	**有效位**
0	0xC48	0x04800	1
1	0xC9C	0x09C00	1

有 4 个普通 TLB 项，当前的状态如下：

索引号	TLB 标记	页面号	有效位
0	0xF8034	0x04812	1
1	0xF8033	0x09812	1
2	0xF4427	0x12137	1
3	0xF44AE	0x17343	1

当前页活跃的目录页（PD）中的部分 PDE 的内容如下：

PDE 索引	页面号	其他标志	大页位	存在位
786	0x04800	…	1	1
807	0x09C00	…	1	1
977	0x09C33	…	0	1
992	0x09078	…	0	1

注：普通页面大小为 4KB，并且 4KB 对齐。每个页面的页面号为其页面起始物理地址除以 4096 得到。大页由连续 1024 个 4KB 小页组成，且 4MB 对齐。

1．分析下面的指令序列，

```
movl $0xC48012024, %ebx
movl $128, (%ebx)
movl $0xF8034000, %ecx
movl $36(%ecx), %eax
```

请问，执行完上述指令后，eax 寄存器中的内容是（　　）；在执行上述指令过程中，共发生了（　　）次 TLB miss？同时会发生（　　）次 page fault？

注：不能确定时填写“--”。

2．请判断下列页面号对应的页面中，哪些一定是页表页？哪些不是？哪些不确定？

页面号	是否为页表页（是/不是/不确定）
0x04800	4
0x09C33	5
0x09812	6

3．下列虚拟地址中哪一个对应着够将虚拟内存地址 0xF4427048 映射到物理内存地址 0x14321048 的页表项（　　）？

（A）0x09C33027　　（B）0xC9C3309C

（C）0xC9C33027　　（D）0x09C3309C

通过上述虚拟地址，利用 movl 指令修改对应的页表项，完成上述映射，在此过程中，是否会产生 TLB miss？（　　）（回答：会/不会/不确定）

修改页表项后，是否可以立即直接使用下面的指令序列将物理内存地址 0x14321048 开始的一个 32 位整数清零？为什么？

```
movl $0xF4427048, %ebx
movl $0, (%ebx)
```

答：

得分

第七题（10 分）

1．请根据 web 应用在计算机网络中的定义以及其在协议栈自上而下在软件中的实现，把以下关键字填入表格

注：同一个关键词可能被填入多次；不是每一个关键词都必须被填入

Streams (end to end), Datagrams, web content, IP, TCP, UDP, Kernel code, User code

协议	数据包类型	软件实现
HTTP		

2．以下关于互联网的说法中哪些是正确的？并简要说明原因

A．在 client-server 模型中，server 通常使用监听套接字 listenfd 和多个 client 同时通信

B．在 client-server 模型中，套接字是一种文件标识符

C．准确地说，IP 地址是用于标识主机的 adapter (network interface card)，并非主机

D．Web 是一种互联网协议

E．域名和 IP 地址是一一对应的

F．Internet 是一种 internet

答：

得分

第八题（10 分）

有三个线程 PA、PB、PC 协作工作以解决文件打印问题：PA 将记录从磁盘读入内存缓冲区 Buff1，每执行一次读一个记录；PB 将缓冲区 Buff1 的内容复制到缓冲区 Buff2，每执行一次复制一个记录； PC 将缓冲区 Buff2 的内容打印出来，每执行一次打印一个记录。缓冲区 Buff1 可以放 4 个记录；缓冲区 Buff2 可以放 8 个记录。请用信号量及 P、V 操作实现上述三个线程以保证文件的正确打印。

```
PA() {
    while(1)  {
            ①
        从磁盘读入一个记录
            ②
        将记录放入 Buff1
            ③
    }
}

PB() {
    while(1)  {
            ④
        从 Buff1 中取出一个记录
            ⑤
        将记录放入 Buff2
            ⑥
    }
}

PC() {
    while(1)  {
            ⑦
        从 Buff2 中取出一个记录
            ⑧
        打印
    }
}
```

1．请设计若干信号量，给出每一个信号量的作用和初值。

2．请将信号量上对应的 PV 操作填写在代码中适当位置。注意：每一标号处可以不填入语句（请标记成 X），或填入一条或多条语句。

标号	对应的操作
①	
②	
②	
③	
④	
⑤	
⑥	
⑧	

北京大学信息科学技术学院考试试卷

考试科目：计算机系统导论 **姓名：**_____ **学号：**_____

考试时间： 2017 年 1 月 2 日 **小班教师：**_____

题号	一	二	三	四	五	六	七	八	**总分**
分数									
阅卷人									

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题和答题纸，共 22 页。

装订线内

不要答题

得分

第四题（10 分）

在 x86_64 环境下，考虑如下 4 个文件（main.c, value.c, f1.c, f2.c）：

```
/* main.c */
#include <stdio.h>

extern void f_void_void();
extern int f_int_void();
void *f;

int main()
{
    int a = 1, b, c;

    f = (void *)f_void_void;

    ((void (*)(int))f)(a);
    b = ((int (*)())f)();
    printf("b = %d\n", b);

    f = (void *)f_int_void;

    ((void (*)())f)(a);
    c = ((int (*)(int))f)(a);
printf("c = %d\n", c);

    return 0;
}

/* value.c */
int BIG;

/* f1.c */
#include <stdio.h>

extern int BIG;
int small = 1;

void f_void_void() {
    small += 1;
    BIG += 1;
    printf("small = %d, BIG = %d\n", small, BIG);
}
```

```c
/* f2.c*/
#include <stdio.h>

extern int BIG;
static int small;

int f_int_void() {
   small += 1;
   BIG += 1;
   printf("small = %d, BIG = %d\n", small, BIG);
   return small + 1;
}
```

使用命令

`gcc -o main main.c f1.c f2.c value.c`

编译这四个文件。

使用

`./main`

运行编译好的程序。

1. 对于程序中的相应符号，请给出它的属性（局部或全局，强符号或弱符号），不确定的请画 X。

源文件	符号名	局部或全局？	强符号或弱符号？
main.c	f		
value.c	BIG		
f1.c	small		
f2.c	small		

2. 请补全程序运行的输出，不确定的请画 X。

```
small = _____, BIG = _____
small = _____, BIG = _____
b = _____
small = _____, BIG = _____
small = _____, BIG = _____
c = _____
```

得分

第五题（10 分）

Part I

请阅读以下程序，然后回答问题。假设程序中的函数调用都可以正确执行，并默认 printf 执行完会调用 fflush。

```
int main() {
    int cnt=1;
    int pid_1,pid_2;
    pid_1=fork();
    if(pid_1==0)    {
        pid_2=fork();
        if(pid_2!=0)    {
            wait(pid_2,NULL,0);
            printf("B");
        }
        printf("F");
        exit(0);
    }
    else {
        A
        B
        wait(pid_1,NULL,0);
        pid_2=fork();
        if(pid_2==0)    {
            printf("D");
            cnt-=1;
        }
        if(cnt==0)
             printf("E");
        else
            printf("G");
        exit(0);
    }
}
```

（1）如果程序中的 A、B 位置的代码为空，列出所有可能的输出结果：

（2）如果程序中的 A、B 位置的代码为：

A: printf("C");

B: exit(0);

列出所有可能的输出结果：

Part II

请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行，且每条语句都是原子动作）：

```
pid_t pid;
int even = 0;
int counter1 = 0;
int counter2 = 1;
void handler1(int sig) {
    if (even % 2 == 0) {
        printf("%d\n", counter1);
        counter1 = ____A_____ ;
        } else {
            printf("%d\n", counter2);
            counter2 = ____B____ ;
        }
        even = even+___C_____ ;
    }
void handler2(int sig) {
        if (_____D_____) {
        counter1 = even*even;
        } else {
        counter2 = even*even;
        }
}
int main() {
    signal(SIGUSR1, handler1);
    signal(SIGUSR2, handler2);
```

```
if ((pid = fork()) == 0) {
    while (1) {};
}
    while (even < 30) {
        kill(pid,  E  );
        sleep(1);
        kill(pid,  F  );
        sleep(1);
        even = even+  G  ;
    }
    kill(pid, SIGKILL);
    exit(0);
}
```

（1）完成程序，使得程序在输出的数字为以下 Q 队列的前 30 项，Q 队列定义如下：

$$Q_0 = 0,\ Q_1 = 1,\ Q_{n+2} = \begin{cases} Q_n + 1, & \text{n\%2} = 0 \\ Q_n \times 2, & \text{n\%2} \neq 0 \end{cases} \quad (n = 0,1,2,3,\ldots)$$

（注：若某个位置中的程序内容，对本次程序执行结果没有影响，请在相应位置填写“无关”）

A: _____

B: _____

C: _____

D: _____

E: _____

F: _____

G: _____

得分

第六题（12 分）

程序 i16.c 如下：

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <sys/wait.h>
#include "csapp.h"

int main(int argc, char **argv)
{
    int fd1, fd2;
    char *cs;
    struct stat stat;
    int sz, i;

    fd1 = Open(argv[1], O_RDWR, S_IRUSR | S_IWUSR);
    Fstat(fd1, &stat);
    sz = stat.st_size;
    fd2 = Open(argv[2], O_RDWR | O_TRUNC | O_CREAT, \
               S_IRUSR | S_IWUSR);

    cs = (char *)Mmap(NULL, sz, PROT_READ | PROT_WRITE, \
            _____X_____, fd1, 0);
    if (Fork()) {
        Wait(&i);
        Write(fd2, cs, sz);
    } else {
        for (i = 0; i < sz; i++) {
            if (islower(cs[i]))
                cs[i] = (char) toupper(cs[i]);
            else if (isupper(cs[i]))
                cs[i] = (char) tolower(cs[i]);
        }
        _____Y_____;
    }
    Munmap(cs, sz);
    Close(fd1);
    Close(fd2);
}
```

1、假定 t1 内容为 ICSexam，根据 X/Y 处的内容，判断执行./i16 t1 t2 之后的结果：（每小题均只有一个正确答案）

1.1、X 为 MAP_PRIVATE，Y 为空时，结果为（　　）

A. t1 为 ICSexam，t2 为 ICSexam

B. t1 为 ICSexam，t2 为 icsEXAM

C. t1 为 icsEXAM，t2 为 icsEXAM

D. t1 为 icsEXAM，t2 为 ICSexam

1.2、X 为 MAP_SHARED 时，Y 为空时，结果为（　　）

A. t1 为 ICSexam，t2 为 ICSexam

B. t1 为 ICSexam，t2 为 icsEXAM

C. t1 为 icsEXAM，t2 为 icsEXAM

D. t1 为 icsEXAM，t2 为 ICSexam

1.3、X 处的内容为 MAP_PRIVATE，Y 为 Write(fd2, cs, sz)时，t2 的结果为（　　）

A. t2 为 ICSexam

B. t2 为 icsEXAM

C. t2 为 icsEXAMICSexam

D. t2 为 icsEXAMicsEXAM

1.4、X 处的内容为 MAP_SHARED，Y 为 Write(fd2, cs, sz)时，t2 的结果为（　　）

A. t2 为 ICSexam

B. t2 为 icsEXAM

C. t2 为 icsEXAMICSexam

D. t2 为 icsEXAMicsEXAM

2、X 为 MAP_SHARED，当子进程运行到 Y 处时，内存映射的内容如下：（只保留了/proc/pid/maps 中的三部分内容：分别为 address/perms/pathname）

```
00400000-00405000 r-xp         /home/tao/i16/i16
00604000-00605000 r--p         /home/tao/i16/i16
00605000-00606000 __A___       /home/tao/i16/i16
7f5df1e61000-7f5df2020000 r-xp  /lib/x86_64-linux-gnu/libc-2.23.so
7f5df2020000-7f5df2220000 ---p  /lib/x86_64-linux-gnu/libc-2.23.so
7f5df2220000-7f5df2224000 r--p  /lib/x86_64-linux-gnu/libc-2.23.so
7f5df2224000-7f5df2226000 rw-p  /lib/x86_64-linux-gnu/libc-2.23.so
7f5df2226000-7f5df222a000 rw-p
7f5df222a000-7f5df2250000 r-xp  /lib/x86_64-linux-gnu/____B_____
7f5df243f000-7f5df2442000 rw-p
7f5df244c000-7f5df244d000 rw-s  _______C_____
7f5df244d000-7f5df244f000 rw-p
7f5df244f000-7f5df2450000 r--p  /lib/x86_64-linux-gnu/____B_____
7f5df2450000-7f5df2451000 rw-p  /lib/x86_64-linux-gnu/____B_____
7f5df2451000-7f5df2452000 rw-p
7ffd4e35f000-7ffd4e380000 rw-p  [____D_____]
7ffd4e3b0000-7ffd4e3b2000 r--p  [vvar]
7ffd4e3b2000-7ffd4e3b4000 r-xp  [vdso]
ffffffffff600000-ffffffffff601000 r-xp  [vsyscall]
```

根据程序执行情况，填充空白处：

A：_____

B：_____

C：_____

D：_____

3、X 为 MAP_SHARED，当子进程运行到 Y 处时：（每小题均只有一个正确答案）

3.1、关于页表的描述，正确的是：（　　）

A．由于 i16 本身的代码和数据只占用了低 32 位空间，所以这部分空间只需使用 2 级页表

B．所有虚拟地址空间都是 48 位地址空间，都需要使用完整的 4 级页表

C．当页表项无效时（P 位为 0），MMU 不会使用到其它位的内容

D．除读写权限外，需要在页表项中为 COW 机制提供专门的支持

3.2、如果子进程在 Y 处访问 7f5df2224 这一页，可能发生以下异常：（　）

A. Page Fault 或 General Protection Fault

B. General Protection Fault 或 Segmentation Fault

C. Page Fault 或 Segmentation Fault

D. Page Fault 或 General Protection Fault 或 Segmentation Fault

3.3、假设 TLB 有 64 个表项，4 路组相联，当访问 7f5df2224 这一页时，所对应的 TLBT 应为：（　）

A. 0111 1111 0101 1101 1111 0010 0010 0010

B. 0111 1111 0101 1101 1111 0010 0010 0010 01

C. 1111 0101 1101 1111 0010 0010 0010 0100

D. 11 1111 0101 1101 1111 0010 0010 0010 0100

3.4、当发生 TLB 命中时，则意味着：（　）

A. 相应的页表项，在 L1 Cache 里

B. 要访问的数据内容，在 L1 Cache 里

C. 如果要访问的数据内容已在 L1 Cache 中，则该 Cache Line 的权限位中应具有相应的读写执行权限

D. 以上都不对

得分

第七题（12 分）

下面是一个基于进程的并发 echo server 的主要代码(省略了与本题无关的部分)，每段代码都注释了相应的功能，请补充空缺的代码（**可能是 0 行或多行**)。请确保任何由你打开的文件描述符都被显式地关闭。

```
int open_listenfd(char *port) {
    struct addrinfo hints, *listp, *p;
    int listenfd, optval = 1;

    /* Get a list of potential server addresses */
    memset(&hints, 0, sizeof(struct addrinfo));
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_flags = AI_PASSIVE | AI_ADDRCONFIG;
    hints.ai_flags |= AI_NUMERICSERV;
    Getaddrinfo(NULL, port, &hints, &listp);

    /* Walk the list for one that we can bind to */
    for (p = listp; p; p = p->ai_next) {
        /* Create a socket descriptor. If failed, try the next */
        if ((listenfd = socket(p->ai_family,
            p->ai_socktype, p->ai_protocol)) < 0)
            (1)
        /* Eliminates error from bind */
        Setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR,
            (const void *)&optval, sizeof(int));
        /* Bind the descriptor to the address */
        if (bind(listenfd, p->ai_addr, p->ai_addrlen) == 0)
            (2) else /* Success, use the current one */
            (3)      /* Bind failed, try the next */
    }
    (4)
    if (!p) return -1; /* No address worked */

    /* Make it a listening socket*/
    if (listen(listenfd, LISTENQ) < 0) {
        (5)
        return -1;
    }
    return listenfd;
}
```

```
void echo(int fd); /* service a client via fd */
void handler(int sig) {
    while (waitpid(-1, 0, WNOHANG) > 0);
    return;
}
int main(int argc, char **argv) {
    int listenfd, connfd;
    socklen_t clientlen;
    struct sockaddr_storage clientaddr;

    Signal(    (10)    , handler);
    listenfd = Open_listenfd(argv[1]);
    while (1) {
        clientlen = sizeof(struct sockaddr_storage)
        __(6)__ = Accept(__(7)__, (SA *)&clientaddr, &clientlen);
        if (Fork() == 0) {
            /* Child services client and close fd */
            __(8)__
            exit(0); /* Child exits */
        }
        __(9)__
    }
}
```

答：

(1) _____

(2) _____

(3) _____

(4) _____

(5) _____

(6) _____

(7) _____

(8) _____

(9) _____

(10) _____

得分

第八题（12 分）

1.请阅读下列代码 badcnt.c：

```
/* Global shared variable */
volatile long cnt = 0; /* Counter */
int main(int argc, char **argv)
{
    long niters;
    pthread_t tid1, tid2;
    niters = atoi(argv[1]);
    Pthread_create(&tid1, NULL, thread, &niters);
    Pthread_create(&tid2, NULL, thread, &niters);
    Pthread_join(tid1, NULL);
    Pthread_join(tid2, NULL);
    /* Check result */
    if (cnt != (2 * niters))
        printf("BOOM! cnt=%ld\n", cnt);
    else
        printf("OK cnt=%ld\n", cnt);
    exit(0);
}

/* Thread routine */
void *thread(void *vargp)
{
    long i, niters = *((long *)vargp);

    for (i = 0; i < niters; i++)
        cnt++;

    return NULL;
}
```

运行后可能产生如下结果：

```
linux> ./badcnt 10000
OK cnt=20000
```

或者：

```
linux> ./badcnt 10000
BOOM! cnt=13051
linux>
```

为什么会产生不同的结果？请分析产生不同结果的原因。

2.某辆公交车的司机和售票员为保证乘客的安全，需要密切配合、协调工作。司机和售票员的工作流程如下所示。请编写程序，用 P、V 操作来实现司机与售票员之间的同步。

司机进程：

```
while(1) {
    ①
    启动车辆;
    ②
    正常行驶;
    ③
    到站停车;
    ④
}
```

售票员进程：

```
while(1) {
    ⑤
    关门;
    ⑥
    报站名或维持秩序;
    ⑦
    到站开门;
    ⑧
}
```

（1）请设计若干信号量，给出每一个信号量的作用和初值。

（2）请将信号量对应的 PV 操作填写在代码中适当位置。

注：每一标号处可以不填入语句（请标记成 X），或填入一条或多条语句。

标号	对应的操作
①	
②	
③	
④	
⑤	
⑥	
⑦	
⑧	

北京大学信息科学技术学院考试试卷

考试科目：计算机系统导论 **姓名：**_____ **学号：**_____

考试时间：2019 年 1 月 7 日 **小班号：**____**小班教师：**_____

题号	一	二	三	四	五	六	七	八	总分
分数									
阅卷人									

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题和答题纸，共　　页。

得分

第三题（10 分）

本题基于下列 m.c 及 foo.c 文件所编译生成的 m.o 和 foo.o，编译过程未加优化选项。

```
//m.c
void foo();
int buf[2] = {1, 2};
int main(){
    foo();
    return 0;
}
```

```
//foo.c
extern int buf[];
int *bufp0 = &buf[0];
int *bufp1;
void foo(){
static int count = 0;
    int temp;
bufp1 = &buf[1];
temp = *bufp0;
*bufp0 = *bufp1;
*bufp1 = temp;
count++;
}
```

对于每个 foo.o 中定义和引用的符号，请用“是”或“否”指出它是否在模块 foo.o 的.symtab 节中有符号表条目。如果存在条目，则请指出定义该符号的模块（foo.o 或 m.o）、符号类型（局部、全局或外部）以及它在模块中所处的节；如果不存在条目，则请将该行后继空白处标记为“/”。

第一问每行 1 分，该行全部答对才给分数，节名如果漏了“.”可以算对。

用英文回答的，如果正确也可以给分。

符号	.symtab 条目?	符号类型	定义符号的模块	节
bufp0	是	全局	foo.o	.data
buf				
bufp1				
foo				
temp				
cnt				

下图左边给出了 m.o 和 foo.o 的反汇编文件，右边给出了采用某个配置链接成可执行程序后再反汇编出来的文件。根据答题需要，其中的信息略有删减。

```
0000000000.....<main>:
55                      push   %rbp
```

```
0000000000000fe8 <main>:
 fe8:  55                      push   %rbp
```

```
48 89 e5                mov     %rsp,%rbp
b8 00 00 00 00      mov     $0x0,%eax
e8 00 00 00 00      callq   e <main+0xe>          ①
b8 00 00 00 00      mov     $0x0,%eax
5d                      pop     %rbp
c3                      req
```

```
0000000000.....<foo>:
55                          push   %rbp
48 89 e5                mov     %rsp,%rbp
48 c7 05 00 00 00 00 00 00 00 00
                        movq            $0x0,0x0(%rip)
   ③②
48 8b 05 00 00 00 00 mov     0x0(%rip),%rax      ④
   8b 00                    mov     (%rax),%eax
89 45 fc                mov     %eax,-0x4(%rbp)
48 8b 05 00 00 00 00 mov     0x0(%rip),%rax      ⑤
48 8b 15 00 00 00 00 mov     0x0(%rip),%rdx      ⑥
8b 12                       mov     (%rdx),%edx
89 10                       mov     %edx,(%rax)
48 8b 05 00 00 00 00 mov     0x0(%rip),%rax      ⑦
8b 55 fc                mov     -0x4(%rbp),%edx
89 10                       mov     %edx,(%rax)
8b  05  00  00  00  00      mov     0x0(%rip),%eax
⑧
83 c0 01                add     $0x1,%eax
89  05  00  00  00  00      mov     %eax,0x0(%rip)
⑨
90                              nop
5d                              pop     %rbp
c3                              retq
```

```
 fe9:  48 89 e5              mov    %rsp,%rbp
 fec:  b8 00 00 00 00      mov    $0x0,%eax
 ff1:  e8 ____①____        callq   1000 <foo>
 ff6:  b8 00 00 00 00      mov    $0x0,%eax
 ffb:  5d                    pop    %rbp
 ffc:  c3                    retq
   ......略去部分和答题无关的信息......
0000000000001000 <foo>:
 1000:  55                    push   %rbp
 1001:  48 89 e5              mov    %rsp,%rbp
 1004:  48 c7 05 ?? ?? ?? ?? ?? ?? ?? ??
                              movq    ②  , ③
(%rip)
 100f:  48 8b
05 ?? ?? ?? ??  mov     0x????(%rip),%rax
 1016:  8b 00                 mov    (%rax),%eax
 1018:  89 45 fc              mov    %eax,-0x4(%rbp)
 101b:  48 8b 05 ?? ?? ?? ?? mov    ⑤ (%rip),%rax
 1022:  48 8b 15 ?? ?? ?? ??  mov    0x????(%rip),%rdx
 1029:  8b 12                 mov    (%rdx),%edx
 102b:  89 10                 mov    %edx,(%rax)
 102d:  48 8b 05 ?? ?? ?? ?? mov    0x????(%rip),%rax
 1034:  8b 55 fc              mov    -0x4(%rbp),%edx
 1037:  89 10                 mov    %edx,(%rax)
 1039:  8b 05 ?? ?? ?? ??     mov
0x????(%rip),%eax
 103f:  83 c0 01              add    $0x1,%eax
 1042:  89 05 ?? ?? ?? ??    mov    %eax, ⑨ (%rip)
 1048:  90                     nop
 1049:  5d                     pop    %rbp
 104a:  c3                     retq
   ......略去部分和答题无关的信息......
0000000000002330 <buf>:
   ......略去部分和答题无关的信息......
0000000000002338 <bufp0>:
   ......略去部分和答题无关的信息......
0000000000003024 <count.1837>:
   ......略去部分和答题无关的信息......
0000000000003028 <bufp1>:
```

在上图中对所涉及到的重定位条目进行用数字①至⑨进行了标记，请根据下表中所提供的重定位条目信息，计算相应的重定位引用值并填写下表。

编号	重定位条目信息	应填入的重定位引用值
①	r.offset = 0xa　　r.symbol = 本题不提供 r.type = R_X86_64_PC32　　r.addend = -4	
②	r.offset = 0xb　　r.symbol = buf r.type = R_X86_64_32　　r.addend = +4	
③	r.offset = 0x7　　r.symbol = bufp1	

	r.type = R_X86_64_PC32 r.addend = -8	
⑤	r.offset = 0x1e r.symbol = bufp0 r.type = R_X86_64_PC32 r.addend = -4	
⑨	r.offset = 0x44 r.symbol = 本题不提供 r.type = R_X86_64_PC32 r.addend = -4	

得分

第四题（10 分）

Bob 是一名刚刚学完异常的同学，他希望通过配合 kill 和 signal 的使用，能让两个进程向同一个文件中交替地打印出字符。可惜他的 tshlab 做得不过关，导致他写的这个程序有各种 BUG。你能帮帮他吗？

```
 1  #include "csapp.h"
 2  #define MAXN 6
 3  int parentPID = 0;
 4  int childPID = 0;
 5  int count = 1;
 6  int fd1 = 1;
 7  void handler1() {
 8      if (count > MAXN)
 9          return;
10      for (int i = 0; i < count; i++)
11          write(fd1, "+", 1);
12          [          X          ]
13      kill(parentPID, SIGUSR2);
14  }
15  void handler2() {
16      if (count > MAXN)
17          return;
18      for (int i = 0; i < count; i++)
19          write(fd1, "-", 1);
20          [          Y          ]
21      kill(childPID, SIGUSR1);
22  }
23
24  int main() {
25      signal(SIGUSR1, handler1);
26      signal(SIGUSR2, handler2);
27      parentPID = getpid();
28      childPID = fork();
29      fd1 = open("file.txt",  O_RDWR);
```

```
30      if (childPID) {
31          [ Z ]
32          kill(childPID, SIGUSR1);
33      }
34      exit(0);
35  }
```

注意：假设程序能在任意时刻被系统打断、调度，并且调度的时间切片大小是不确定的，可以足够地长。在每次程序执行前，file.txt 是一个已经存在的**空文件**。

Part A.（1 分） 此时，X 处语句和 Y 处语句都是 count++;，Z 处语句是空语句。Alice 测试该代码，发现有时 file.txt 中没有任何输出！请解释原因。（提示：考虑 28 行语句 fork 以后，下一次被调度的进程，并从这个角度回答本题。不需要给出解决方案）

Part B.（6 分）Bob 根据 Alice 的反馈，在某两行之间加了若干代码，修复了 Part A 的问题。当 X 处代码和 Y 处代码都是 count++;、Z 处为空时，Bob 期望 file.txt 中的输出是：

```
+-+++-+++++-++++++++++++++++++++++++++++++
```

可 Alice 测评 Bob 的程序的时候，却发现有时 Bob 的程序在 file.txt 中的输出是：

```
+-+--+---+----+------
```

而与此同时，终端上出现了如下的输出：

```
+
```

Bob 找不到自己的代码的 BUG，只好向 Alice 求助。Alice 帮他做了如下分析：

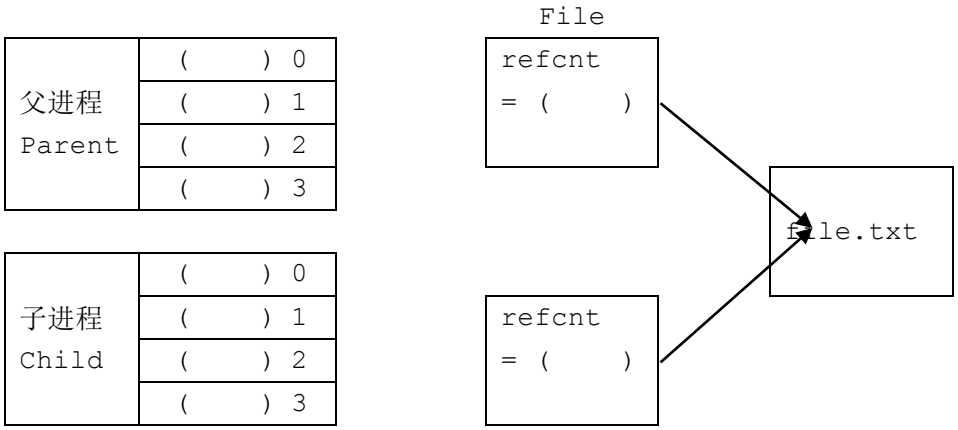
分析 1. 当程序第一次**在终端**上输出+的瞬间，请完成下表。要求：

(1) 在“描述符表”一栏中，用“√”勾选该进程当前 fd1 的值。

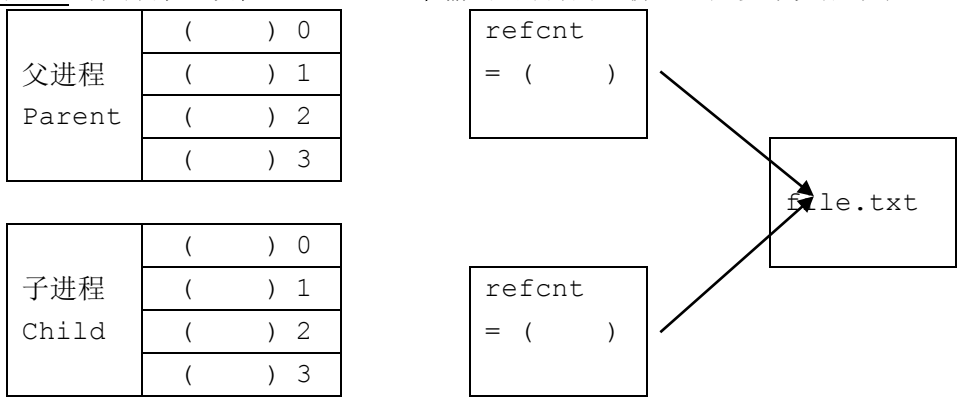
(2) 在“打开文件表”一栏中，填写该项的 refcnt（即，被引用多少次）。如果某一项不存在，请在括号中写“0”（并忽略其指向 v-node 表的箭头）。

(3) 画出“描述符表”到“打开文件表”的表项指向关系。不需要画关于标准输入/标准输出/标准错误的箭头。评分时不对箭头评分，**请务必保证前两步的解答与箭头的连接情况匹配。**

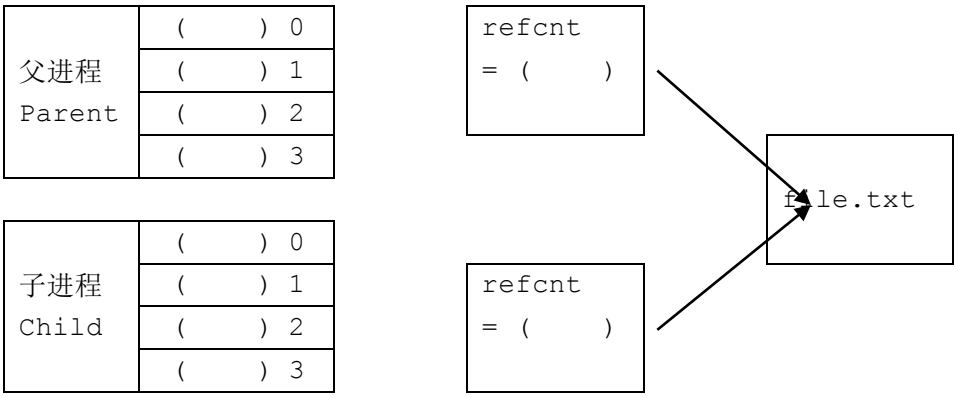
描述符表 Descriptor	打开文件表 Open	v-node 表 V-node



分析 2. 当程序第一次**在 file.txt 中**输出+的瞬间，仿照上题要求完成下表：



分析 3. 如果要产生 Bob 预期的输出，三级表的关系应当是什么？仿照上题要求完成下表：



Part C. (2 分) Bob 很高兴，他知道 Part B 的代码是怎么错的了！不过 Alice 仍然想考考 Bob。对于 Part B 的错误代码，如果终端上输出的是+++，那么

file.txt 中的内容是什么？请在下框中写出答案。

Part D. (1 分) Bob 修复了 Part B 的问题，使得代码能够产生预期的输出。现在，Bob 又希望自己的代码最终输出的是+--+++----++++------，为此，他对 X、Y、Z 处做了如下的修改。X、Y 处语句已做如下填写，请帮助 Bob 补上 Z 处语句。

X 处填写为：`count += 2;`
Y 处填写为：`count += 2;`
Z 处填写为：

得分

第五题（12 分）

1. (2 分，每空一分，考察多级页表和单级页表的设计思想的理解)

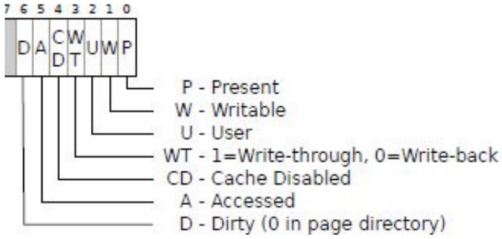
在进行地址翻译的过程中，操作系统需要借助页表(Page Table)的帮助。考虑一个 32 位的系统,页大小是 4KB,页表项(Page Table Entry)大小是 4 字节(Byte)，如果不使用多级页表，常驻内存的页表一共需要_____页。

考虑下图已经显示的物理内存分配情况，在二级页表的情况下，已经显示的区域的页表需要占据_____页。

VP0	已分配页
…	
VP1023	
VP1024	
…	
VP2047	
Gap	未分配页
1023 unallocated pages	
VP10239	已分配页
VP10240	
…	
VP11263	

2. (6 分，每错一空扣一分，扣完为止。考察地址翻译过程)

IA32 体系采用小端法和二级页表。其中两级页表大小相同，页大小均为 4KB，结构也相同。TLB 采用直接映射。TLB 和页表每一项的后 7 位含义如下图所示。为简便起见，假设 TLB 和页表每一项的后 8~12 位都是 0 且不会被改变。注意后 7 位值为“27”则表示可读写。



当系统运行到某一时刻时，TLB 内容如下：

索引	TLB 标记	内容	有效位
0	0x04013	0x3312D027	1
1	0x01000	0x24833020	0
2	0x005AE	0x00055004	1
3	0x00402	0x24AEE020	0
4	0x0AA00	0x0005505C	0
5	0x0000A	0x29DEE000	1
6	0x1AE82	0x00A23027	1
7	0x28DFC	0x00023000	0

一级页表的基地址为 0x0C23B00，物理内存中的部分内容如下：

地址	内容	地址	内容	地址	内容	地址	内容
00023000	E0	00023001	BE	00023002	EF	00023003	BE
00023120	83	00023121	C8	00023122	FD	00023123	12
00023200	23	00023201	FD	00023202	BC	00023203	DE
00023320	33	00023321	29	00023322	E5	00023323	D2
0005545C	97	0005545D	C2	0005545E	7B	0005545F	45
00055464	97	00055465	D2	00055466	7B	00055467	45
0C23B020	27	0C23B021	EB	0C23B022	AE	0C23B023	24
0C23B040	27	0C23B041	40	0C23B042	DE	0C23B043	29
0C23B080	05	0C23B081	5D	0C23B082	05	0C23B083	00
2314D200	23	2314D201	12	2314D202	DC	2314D203	0F
2314D220	A9	2314D221	45	2314D222	13	2314D223	D2

29DE404C	27	29DE404D	42	29DE404E	BA	29DE404F	00
29DE4400	D0	29DE4401	5C	29DE4402	B4	29DE4403	2A

此刻，系统先后试图对两个已经缓存在 cache 中的内存地址进行写操作，请分析**完成写之后**系统的状态（写的地址和上面的内存地址无交集），完成下面的填空。若不需要某次访问或者缺少所需信息，请填“\”。

第一次向地址 0xD7416560 写入内容，TLB 索引为：_____，完成写之后该项 TLB 内容为：_____，

二级页表页表项地址为：_____，物理地址为：_____。

第二次向地址 0x0401369B 写入内容，

TLB 索引为：_____，完成写之后该项 TLB 内容为：_____

二级页表页表项地址为：_____，物理地址为：_____。

3. （2 分，考察对于虚拟内存独立地址空间的理解）

本学期的 fork bomb 作业中，大家曾用 fork 逼近系统的进程数量上限。下面有一个类似的程序，请仔细阅读程序并填空。

```
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
#define N 4
int main() {
        volatile int pid, cnt = 1;
        for (int i = 0; i < N; i++) {
                if ((pid = fork()) > 0) {
                        cnt++;
        }
        while (wait(NULL) > 0);
        return 0;
}
```

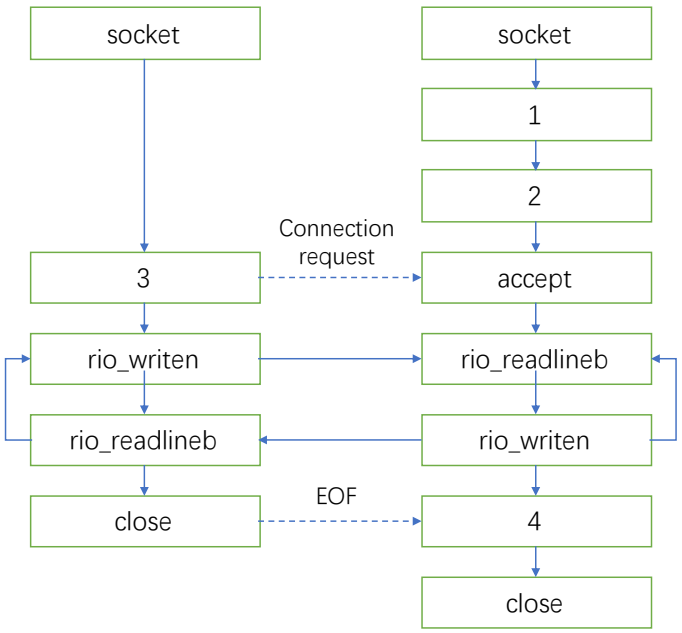
得分

第六题（10 分）

1. 请在以下的表格中填入相应的内容，每个空格仅需填一项。其中第 1 列可填选项包括：网络层、传输层、应用层；第 2 列可填选项 IP、UDP、TCP、HTTP；第 3 列可填选项包括：是、否。
（说明：面向连接的协议保障数据按照发送时的顺序被接收）

协议层次	协议名称	是面向连接的吗？
网络层		
	HTTP	
	TCP	

2. 下图描述了客户端与服务器套接字连接和通信的过程，请在空格处补充相关步骤。



得分

第七题（10 分）

给定如下程序：

```
#include <stdio.h>
#include <pthread.h>
int i = 0;
int j = 0;
void *do_stuff1(void * arg __attribute__((unused))) {
  int a;
  for (a = 0; a < 1000; a++)
   {i++; j++;}
  return NULL;
}
void *do_stuff2(void * arg __attribute__((unused))) {
  int a;
  for (a = 0; a < 1000; a++)
   {j++; i++;}
  return NULL;
}
int main() {
  pthread_t tid1, tid2;
  pthread_create(&tid1, NULL, do_stuff1, NULL);
  pthread_create(&tid2, NULL, do_stuff2, NULL);
  pthread_join(tid1, NULL);
  pthread_join(tid2, NULL);
  printf("%d, %d\n", i, j);
  return 0;
}
```

1. （2 分）用以下元素的编号写出 i++编译后的汇编代码。
   (a) mov　(b) add　(c) 0x601040　(d) %eax (e) $0x1
   比如，回答(a) (e) (d)表示 mov $0x1, %eax

2. （4 分）请回答该程序是否有可能输出如下结果，并简述原因。
   (a) 2000, 2000
   (b) 1500, 1500
   (c) 1000, 1000
   (d) 2, 2

3. （2 分）卜廷江同学在学习了信号量之后，决定要让程序能稳定输出 2000, 2000，于是对程序进行了如下改写。

```
1.  #include <stdio.h>
2.  #include <pthread.h>
3.  int i = 0;
4.  int j = 0;
5.  sem_t si;
6.  sem_t sj;
7.  void *do_stuff1(void * arg __attribute__((unused))) {
8.  int a;
9.    for (a = 0; a < 1000; a++) {
10. P(&si);
11. P(&sj);
12. i++;
13. j++;
14. V(&si);
15. V(&sj);
16. }
17. return NULL;
18. }
19. void *do_stuff2(void * arg __attribute__((unused))) {
20. int a;
21. for (a = 0; a < 1000; a++) {
22. P(&sj);
23. P(&si);
24. j++;
25. i++;
26. V(&si);
27. V(&sj);
28. }
```

```
29. return NULL;
30. }
31. int main() {
32. pthread_t tid1, tid2;
33. Sem_init(&si, 0, 1);
34. Sem_init(&sj, 0, 1);
35. pthread_create(&tid1, NULL, do_stuff1, NULL);
36. pthread_create(&tid2, NULL, do_stuff2, NULL);
37. pthread_join(tid1, NULL);
38. pthread_join(tid2, NULL);
39. printf("%d\n", i);
40. return 0;
41. }
```

请问卜廷江同学的程序有什么潜在问题？为什么？

4. （2 分）如果对卜廷江同学的程序改动一处数字来消除上述问题，同时仍然保证输出结果稳定为 2000, 2000，应该如何改动？
回答：将_____（填行号）行的_____（填数字）改为_____（填数字）。

得分

第八题（10 分，每空 1 分）

题目假设：软硬件系统为 64 位 Linux 操作系统，页大小为 4KB；栈的分配空间最大为 8MB；题目中的 echoserveri/echoserverp/echoservert 分别为原书中的迭代 echo 服务器、基于进程的并发 echo 服务器、基于线程的并发 echo 服务器；当客户端连入服务器时，假设客户端只进行了连接，没有进行后续的读写操作。

启动 echoserveri，查看内存占用情况如下：（输出结果中删除了部分没有用到的列）

```
linux$ pmap -X `pidof echoserveri`
18859:   ./echoserveri 15213
Address Perm   Offset Device Size  Rss Pss Anonymous Mapping
00400000 r-xp 00000000  fc:02   20   20  20         0 echoserveri
00604000 r--p 00004000  fc:02    4    4   4         4 echoserveri
00605000 rw-p 00005000  fc:02    4    4   4         4 echoserveri
00a9a000 rw-p 00000000  00:00  132    8   8         8      (1)
7f56da7af000 r-xp 00000000  fc:00 1792 1220  11         0 libc-2.23.so
7f56da96f000 ---p 001c0000  fc:00 2048    0   0         0 libc-2.23.so
7f56dab6f000 r--p 001c0000  fc:00   16   16  16        16 libc-2.23.so
7f56dab73000 rw-p 001c4000  fc:00    8    8   8         8 libc-2.23.so
7f56dab75000 rw-p 00000000  00:00   16   16  16        16
7f56dab79000 r-xp 00000000  fc:00   96   92   1         0 libpthread-2.23.so
7f56dab91000 ---p 00018000  fc:00 2044    0   0         0 libpthread-2.23.so
7f56dad90000 r--p 00017000  fc:00    4    4   4         4 libpthread-2.23.so
7f56dad91000 rw-p 00018000  fc:00    4    4   4         4 libpthread-2.23.so
7f56dad92000 rw-p 00000000  00:00   16    4   4         4
7f56dad96000 r-xp 00000000  fc:00  152  152   1         0      (2)
7f56dafac000 rw-p 00000000  00:00   16   16  16        16
7f56dafbb000 r--p 00025000  fc:00    4    4   4         4      (2)
7f56dafbc000 rw-p 00026000  fc:00    4    4   4         4      (2)
7f56dafbd000 rw-p 00000000  00:00    4    4   4         4
7ffd382f8000  (3)  00000000   (4)   132   24  24        24 [stack]
7ffd38364000 r--p 00000000  00:00   12    0   0         0 [vvar]
```

```
7ffd38367000 r-xp 00000000  00:00    8    4   0          0 [vdso]
ffffffffff600000 r-xp 00000000  00:00    4    0   0          0 [vsyscall]
==== ==== === =========
6540 1608 153       120 KB
```

填空：

1、_____

2、_____

3、_____

4、_____

在上面的表格中，Size 一列指的是 **VSS (Virtual Set Size)**，表示一个进程可访问的总的地址空间的大小（the total accessible address space of a process）；**RSS (Resident Set Size)** 表示一个进程实际驻留在 RAM 中的空间大小（the total memory actually held in RAM for a process），其中包括了该进程所用到的所有共享空间(如共享库或私有 COW 空间);**PSS (Proportional Set Size)** 与 RSS 的区别在于共享空间的尺寸，当某个共享空间为 30 页时，如果有三个进程共享该空间，则每个进程的该共享空间的 PSS 只为 10 页，而每个进程的该共享空间的 RSS 仍为 30 页。根据上述定义，可以做出如下判断：VSS >= RSS >= PSS。

可以根据 echoserveri 的相关数值，对 echoserverp 和 echoservert 的运行情况进行估计：（单选题）

5、使用 echoserverp 作为服务器端，当十个客户端连入 echo 服务器时，总的 VSS 值可能为：

A. 约 80MB
B. 约 16MB
C. 约 6MB
D. 约 2MB

6、使用 echoserverp 作为服务器端，当十个客户端连入 echo 服务器时，总的 RSS 值可能为：

A. 约 80MB
B. 约 16MB

C. 约 6MB

D. 约 2MB

7、使用 echoserverp 作为服务器端，当十个客户端连入 echo 服务器时，总的 PSS 值可能为：

A. 约 16MB

B. 约 6MB

C. 约 2MB

D. 约 500KB

8、使用 echoservert 作为服务器端，当十个客户端连入 echo 服务器时，总的 VSS 值可能为：

A. 约 80MB

B. 约 16MB

C. 约 6MB

D. 约 2MB

9、使用 echoservert 作为服务器端，当十个客户端连入 echo 服务器时，总的 RSS 值可能为：

A. 约 80MB

B. 约 16MB

C. 约 6MB

D. 约 2MB

10、使用 echoservert 作为服务器端，当十个客户端连入 echo 服务器时，总的 PSS 值可能为：

A. 比 echoserverp 作为服务器时的总 PSS 值大

B. 比 echoserverp 作为服务器时的总 PSS 值小

装订线内

不要答题

北京大学信息科学技术学院考试试卷

考试科目：计算机系统导论 **姓名：**_____ **学号：**_____

考试时间：2019 年 12 月 30 日**小班号：**__**小班教师：**____

题号	一	二	三	四	五	六	七	**总分**
分数								
阅卷人								

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题和答题纸，共 16 页。

得分

第三题（10 分）

本题基于下列 m.c 及 foo.c 文件所编译生成的 m.o 和 foo.o，编译和运行在 x86-64/Linux 下完成，编译过程未加优化选项。

```
//m.c
void myswap();
char buf[2] = {1, 2};
void *bufp0;
void *bufp1;
char temp;
int main(){
......//略去题目无关代码
myswap(temp);
......//略去题目无关代码
   return 0;
}
```

```
//foo.c
extern int buf[];
char *bufp0;
......//略去题目无关代码
char *bufp1 = &buf[1];
void myswap(char temp){
static int count = 0;
......//略去题目无关代码
bufp0 = &buf[0];
temp = *bufp0;
......//略去题目无关代码
*bufp0 = *bufp1;
*bufp1 = temp;
......//略去题目无关代码
count++;
}
```

（1） 对于每个 foo.o 中定义和引用的符号，请用“是”或“否”指出它是否在模块 foo.o 的.symtab 节中有符号表条目。如果存在条目，则请指出在模块 foo.o 中的符号类型（局部、全局或外部）、它在 foo.o 模块中所处的节（.data、.text、.bss 或 COMMON）以及其强弱信息（强、弱、非强非弱）；如果不存在条目，则请将该行后继空白处标记为“/”（请将下表画到答题纸上）。

符号	.symtab 条目?	符号类型	节	强弱
bufp1	是	全局	.data	强
buf				
bufp0				
temp				
count				

（2） 我们使用 REF(myswap.m)->DEF(myswap.foo)表示链接器将把模块 m 中对符号 myswap 的任意引用与模块 foo 中对 swap 的定义关联起来。对于下面的示例，用这种方法来说明链接器如何解析每个模块对多重定义符号的引用。如果有链接时错误，请标记“错误”。如果链接器从定义中任意选择一个，请标记“未知”。

a) REF(bufp0.m) ->DEF(_____)

b) REF(bufp1.m) ->DEF(_____)

（3） 下图左边给出了 m.o 和 foo.o 的反汇编文件，右边给出了采用某个配置链接成可执行程序后再反汇编出来的文件。根据答题需要，其中的信息略有删减。

```
0000000000......<main>:
55                  push   %rbp
48 89 e5            mov    %rsp,%rbp
48 83 ec 10         sub    $0x10,%rsp
......
b8 00 00 00 00      mov    $0x0,%eax
e8 00 00 00 00       callq  1a <main+0x1a>
①
......
c9                  leaveq
c3                  req
```

```
    0000000000......<myswap>:
    55                  push   %rbp
    48 89 e5            mov    %rsp,%rbp
    ......
    48 c7 05 00 00 00 00 00 00 00 00

    movq   $0x0,0x0(%rip)        ③②
48 8b 05 00 00 00 00 mov    0x0(%rip),%rax
④
0f b6 00             movzbl (%rax),%eax
88 45 fc             mov    %al,-0x4(%rbp)
......
48 8b 05 00 00 00 00 mov    0x0(%rip),%rax
⑤
48 8b 15 00 00 00 00 mov    0x0(%rip),%rdx
⑥
0f b6 12             movzbl (%rdx),%edx
88 10                mov    %dl,(%rax)
48 8b 05 00 00 00 00 mov    0x0(%rip),%rax
⑦
0f b6 55 fc          movzbl -0x4(%rbp),%edx
88 10                mov    %dl,(%rax)
......
8b 05 00 00 00 00    mov    0x0(%rip),%eax
⑧
83 c0 01             add    $0x1,%eax
89 05 00 00 00 00    mov    %eax,0x0(%rip)
⑨
90                   nop
5d                   pop    %rbp
c3                   retq
```

```
0000000000000f28 <main>:
f28:  55                  push   %rbp
f29:  48 89 e5            mov    %rsp,%rbp
f2c:  48 83 ec 10         sub    $0x10,%rsp
......略去部分和答题无关的信息......
f38:  b8 00 00 00 00      mov    $0x0,%eax
f3d:   e8 _____①_____      callq  1000
<myswap>
......略去部分和答题无关的信息......
ffd:  c9                   leaveq
ffe:  c3                   retq
......略去部分和答题无关的信息......
0000000000001000 <myswap>:
1000:  55                  push   %rbp
1001:  48 89 e5            mov    %rsp,%rbp
......略去部分和答题无关的信息......
100d:  48 c7 05 ?? ?? ?? ?? ?? ?? ?? ??
movq    ②  ,0x????(%rip)
1018:                        48            8b
05 ?? ?? ?? ?? mov    0x????(%rip),%rax
101f:  0f b6 00             movzbl (%rax),%eax
1018:  88 45 fc                mov    %eax,-
0x4(%rbp)
......略去部分和答题无关的信息......
1035:     48   8b   05   ??   ??   ??   ??   mov
0x????(%rip),%rax
103c:   48  8b  15  ??  ??  ??  ??  mov      ⑥
(%rip),%rdx
1043:   0f  b6  12                        movzbl
(%rdx),%edx
1046:  88 10                mov    %dl,(%rax)
1048:     48   8b   05   ??   ??   ??   ??   mov
0x????(%rip),%rax
104f:   0f  b6  55  fc                movzbl -
0x4(%rbp),%edx
1053:  88 10                mov    %dl,(%rax)
......略去部分和答题无关的信息......
1057:    8b  05  ??  ??  ??  ??           mov
0x????(%rip),%eax
105d:  83 c0 01             add    $0x1,%eax
1060:  89 05 ?? ?? ?? ??    mov    %eax,  ⑨
(%rip)
1066:  90                   nop
1067:  5d                   pop    %rbp
1068:  c3                   retq
......略去部分和答题无关的信息......
000000000000a000 <buf>:
......略去部分和答题无关的信息......
000000000000a050 <bufp1>:
......略去部分和答题无关的信息......
000000000000cb24 <count.1837>:
......略去部分和答题无关的信息......
000000000000cb28 <bufp0>:
......略去部分和答题无关的信息......
```

在上图中对所涉及到的重定位条目进行用数字①至⑨进行了标记，请根据下表中所提供的重定位条目信息，计算相应的重定位引用值并填写下表。（请将下表的第 1 列和第 3 列画到答题纸上）

编号	重定位条目信息		应填入的重定位引用值
①	r.offset = 0x16 r.type = R_X86_64_PC32	r.symbol = 本题不提供 r.addend = -4	
②	r.offset = 0x14 r.type = R_X86_64_32	r.symbol = buf r.addend = +0	
⑥	r.offset = 0x3f r.type = R_X86_64_PC32	r.symbol = bufp1 r.addend = -4	
⑨	r.offset = 0x62 r.type = R_X86_64_PC32	r.symbol = 本题不提供 r.addend = -4	

得分

第四题（10 分）

分析以下C程序，其中f1.txt和f2.txt为已有用户有读写的文件，初始文件内容为空。

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <sys/types.h>
4.  #include <sys/stat.h>
5.  #include <fcntl.h>
6.  #include <unistd.h>
7.  #include <sys/types.h>
8.  #include <sys/wait.h>
9.
10. int main()
11. {
12.   int fd1,fd2,fd3,fd4;
13.   int pid;
14.   int c=1;
15.   fd1=open("./f1.txt",O_WRONLY,0);
16.   fd2=open("./f1.txt",O_WRONLY,0);
17.
18.   printf("fd1=%d,fd2=%d;\n",fd1,fd2);
19.
20.   write(fd1,"EECSPKU",7);
21.   write(fd2,"2019",4);
22.
23.   close(fd2);
24.
25.   fd3=open("./f2.txt",O_WRONLY,0);
26.   fd4=dup(fd3);
27.
28.   printf("fd3=%d,fd4=%d;\n",fd3,fd4);
29.
30.   pid=fork();
31.   if((pid==0)) {
32.     c--;
33.     write(fd3,"PKU",3);
34.     write(fd4,"ICS",3);
35.     printf("c= %d\n",c);
36.   }
37.   else {
38.     waitpid(-1,NULL,0);
39.     c++;
40.     write(fd3,"2019",4);
```

```
41.     close(fd3);
42.     close(fd4);
43.     printf("c= %d\n",c);
44.   }
45.
46.   if(c)
47.    write(fd1,"CS",2);
48.   c++;
49.   close(fd1);
50.   printf("c=%d\n",c);
51. }
```

当程序正确运行后，填写输出结果：

（1） 程序第 18 行：fd1=__①__，fd2=__②__；

（2） 程序第 28 行：fd3=__①__，fd4=__②__；

（3） 程序第35、43、50行输出c的值依次分别为：_____；

（4） 文件 f1.txt 中的内容为：_____；

（5） 文件 f2.txt 中的内容为：_____。

得分

第五题（10 分）

某 32 位机器有 24 位地址空间，采用二级页表，一级页表中有 64 个 PTE，二级页表中有 1024 个 PTE，一级页表 4KB 对齐，PTE 最高一位是有效位，没有 TLB 和 Cache。惠楚思程序员在该机器上执行了如下代码。

```
1. int* a=calloc(10000, sizeof(int));
2. int* b = a+5000;
3. fork();
4. *b=0x80C3F110;
```

（1） 假设编译后 b 的值保存在寄存器中，请问该机器页面大小为①__字节，VPN1 长度为②____，VPN2 长度为③____。

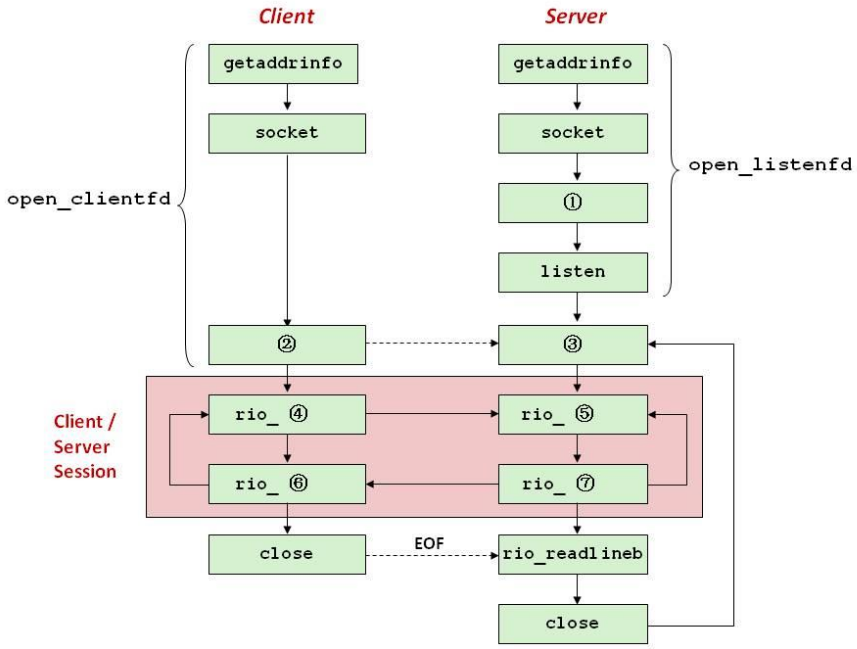
（2） 由于该执行结果不符合预期，惠楚思需要对程序执行过程进行还原。程序执行过程中硬件记录了部分内存访问记录，但访问记录并不完整。目前已知在父进程执行第四行的时候，进行了若干次物理内存的读写操作，其中第一次读内存操作从地址 0x67F0E8 读出 0x80AA32C4，另外有两次将 0x80C3F110 写入内存的操作，写入的地址分别是 0xC3F50D 和 0xAA3AD0，但写入顺序并不清楚。程序结束后变量 b 中保存的值是①_____。在解析“*b”的过程中，一级页表的起始地址为②_____；二级页表的起始地址为③_____；物理页面的起始地址为④_____。（地址均用 16 进制表示）

得分

第六题（10 分）

下图是一个基于 echo 服务器的 client-server 框架

（1） 请给图中的编号填写相应的函数名。



（2） 请补全下面 server 端 open_listenfd 函数中缺失的操作（Line 21, Line 26 和 Line 34）

```
Line 1:  int open_listenfd(char *port)
Line 2:  {
Line 3:     struct addrinfo hints, *listp, *p;
Line 4:     int listenfd, optval=1;
Line 5:     /* Get a list of potential server addresses */
Line 6:     memset(&hints, 0, sizeof(struct addrinfo));
Line 7:     hints.ai_socktype = SOCK_STREAM;
Line 8:     hints.ai_flags = AI_PASSIVE | AI_ADDRCONFIG;
Line 9:     hints.ai_flags |= AI_NUMERICSERV;
Line 10:    Getaddrinfo(NULL, port, &hints, &listp);
```

```
Line 11:
Line 12:    for (p = listp; p; p = p->ai_next) {
Line 13:       /* Create a socket descriptor */
Line 14:      if ((listenfd = socket(p->ai_family, p->ai_socktype,
Line 15:                             p->ai_protocol)) < 0)
Line 16:        continue;  /* Socket failed, try the next */
Line 17:       /* Eliminates "Address already in use" error from
bind */
Line 18:       Setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR,
Line 19:                (const void *)&optval , sizeof(int));
Line 20:
Line 21:       if (_⑧_(listenfd, p->ai_addr, p->ai_addrlen) == 0)
Line 22:         break;  /* Success */
Line 23:       Close(listenfd);
Line 24:    }
Line 25:    /* Clean up */
Line 26:    Freeaddrinfo(___⑨___);
Line 27:    if (!p) /* No address worked */
Line 28:       return -1;
Line 29:
Line 30:    if (listen(listenfd, LISTENQ) < 0) {
Line 31:       Close(listenfd);
Line 32:       return -1;
Line 33:    }
Line 34:    return __⑩___;
Line 35: }
```

得分

第七题（10 分）

解决第一类读者-写者问题的代码如下：

```
int readcnt;    /* Initially 0 */
sem_t mutex, w; /* Both initially 1 */
void reader(void)
{
(1)    while (1) {
(2)      P(&mutex);
(3)      readcnt++;
(4)      if (readcnt == 1) /* First in */
(5)        P(&w);
(6)      V(&mutex);
(7)         /* Reading happens here, need 7 time unit */
(8)      P(&mutex);
(9)      readcnt--;
(10)     if (readcnt == 0) /* Last out */
(11)        V(&w);
(12)     V(&mutex);
       }
}

void writer(void)
{
(13)  while (1) {
(14)    P(&w);
(15)       /* Writing here, need 8 time unit */
(16)    V(&w);
     }
}
```

假设有 5 个读者或写者到来，到达时刻和所需的时间如下：

线程	到达时刻	读写时间
R1	0	7
W1	1	8
R2	2	7
W2	4	8
R3	5	7

注意，为简单起见，只考虑读写时间，忽略所有其他时间（包括代码中其他语句的执行时间、线程切换、调度等等），亦假设没有更多的读者或写者或其他线程。

（1）在时刻 3 时，R1、W1 和 R2 所处的位置，请标出对应的代码行号。此刻，readcnt 和 w 的值分别是多少？R1:_；W1:_；R2:_；readcnt:_；w:_。

（2）在时刻 6 时，W2 和 R3 所处的位置，请标出对应的行号。此刻，readcnt 和 w 的值分别是多少？W2:_；R3:_；readcnt:_；w:_。

（3）如果不使用 P(&mutex)和 V(&mutex)，程序执行会不会出错？如果出错，会出什么错？

装订线内

不要答题

北京大学信息科学技术学院考试试卷

考试科目：计算机系统导论 **姓名：**_____ **学号：**_____

考试时间：2021 年 1 月 11 日 **小班号：**_____

题号	一	二	三	四	五	六	**总分**
分数							
阅卷人							

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题，共 16 页。

第三题（10 分）

本题基于下列 m.c 及 swap.c 文件所编译生成的 m.o 和 swap.o，编译和运行在 Linux/X86-64 下使用 GCC 完成，编译过程未加优化选项。

```
//m.c
void myswap();
void func();
char buf[2] = {1, 2};
void *bufp0;
void *bufp1;
char temp;
int main(){
    func();
    ......//略去题目无关代码
    myswap(temp);
    ......//略去题目无关代码
    return 0;
}
```

```
//swap.c
extern int buf[];
char *bufp0;
......//略去题目无关代码
char *bufp1 = &buf[1];
void myswap(char temp){
    static int count = 0;
    ......//略去题目无关代码
    bufp0 = &buf[0];
    temp = *bufp0;
    ......//略去题目无关代码
    *bufp0 = *bufp1;
    *bufp1 = temp;
    ......//略去题目无关代码
    count++;
}
void func(){
    ......//略去题目无关代码
}
```

1）对于每个 swap.o 中定义和引用的符号，请用“是”或“否”指出它是否在模块 swap.o 的.symtab 节中存在符号表条目。如果存在条目，则请指出定义该符号的模块（swap.o 或 m.o）、符号类型（局部、全局或外部）以及该符号在所属的模块（“即该符号在该模块中被定义”）中所处的节；如果不存在条目，则请将该行后继空白处标记为“/”。

符号	.symtab 有条目?	符号类型	定义符号的模块	节
buf				
bufp0				
count				
func				
temp				

2）下图左边给出了 m.o 和 swap.o 的反汇编文件，右边给出了采用某个配置链接成可执行程序后再反汇编出来的文件。根据答题需要，其中的信息略有删减。

```
0000000000......<main>:
55                      push   %rbp
48 89 e5                mov    %rsp,%rbp
b8 00 00 00 00          mov    $0x0,%eax
e8 00 00 00 00          callq  e <main+0xe>     ①
......
0f b6 05 00 00 00 00    movzbl 0x0(%rip),%eax  ②
0f be c0                movsbl %al,%eax
89 c7                   mov    %eax,%edi
b8 00 00 00 00          mov    $0x0,%eax
e8 00 00 00 00          callq  26 <main+0x26>  ③
......
b8 00 00 00 00          mov    $0x0,%eax
5d                      pop    %rbp
c3                      retq
```

```
0000000000......<myswap>:
55                 push   %rbp
48 89 e5           mov    %rsp,%rbp
89 f8              mov    %edi,%eax
88 45 fc           mov    %al,-0x4(%rbp)
......
48 c7 05 00 00 00 00 00 00 00 00          ⑤④
                     movq   $0x0,0x0(%rip)
......
48 8b 05 00 00 00 00  mov    0x0(%rip),%rax   ⑥
0f b6 00               movzbl (%rax),%eax
88 45 fc               mov    %al,-0x4(%rbp)
......
48 8b 05 00 00 00 00 mov    0x0(%rip),%rax    ⑦
48 8b 15 00 00 00 00 mov    0x0(%rip),%rdx    ⑧
0f b6 12             movzbl (%rdx),%edx
88 10                mov    %dl,(%rax)
48 8b 05 00 00 00 00 mov    0x0(%rip),%rax    ⑨
0f b6 55 fc          movzbl -0x4(%rbp),%edx
88 10                mov    %dl,(%rax)
......
8b 05 00 00 00 00     mov    0x0(%rip),%eax   ⑩
83 c0 01              add    $0x1,%eax
89 05 00 00 00 00     mov    %eax,0x0(%rip)   ⑪
90                    nop
5d                    pop    %rbp
c3                    retq

0000000000......<func>:
......
```

```
0000000000001000 <main>:
1000: 55                  push  %rbp
1001: 48 89 e5           mov   %rsp,%rbp
1004: b8 00 00 00 00      mov   $0x0,%eax
1009: e8      ①           callq 302e <func>
......
1010:  0f  b6  05          ②              movzbl
0x??????(%rip),%eax
1017: 0f be c0            movsbl %al,%eax
101a: 89 c7               mov    %eax,%edi
101c: b8 00 00 00 00      mov    $0x0,%eax
1021: e8      ③           callq  10e4 <myswap>
......
10db: b8 00 00 00 00       mov    $0x0,%eax
10e0: 5d                   pop    %rbp
10e1: c3                   retq

00000000000010e4<myswap>:
10e4: 55                   push   %rbp
10e5: 48 89 e5             mov    %rsp,%rbp
10e8: 89 f8                mov    %edi,%eax
10ea: 88 45 fc             mov    %al,-0x4(%rbp)
......
10f4: 48 c7 05      ⑤          ④
                   movq    ④ , ⑤ (%rip)
......
1104: 48 8b 05      ⑥       mov   ⑥ (%rip),%rax
110b: 0f b6 00              movzbl (%rax),%eax
110e: 88 45 fc              mov    %al,-0x4(%rbp)
......
1229: 48 8b 05      ⑦       mov   ⑦ (%rip),%rax
1230: 48 8b 15      ⑧       mov   ⑧ (%rip),%rdx
1237: 0f b6 12              movzbl (%rdx),%edx
123a: 88 10                 mov    %dl,(%rax)
123c: 48 8b 05      ⑨       mov   ⑨ (%rip),%rax
1243: 0f b6 55 fc           movzbl -0x4(%rbp),%edx
1247: 88 10                 mov    %dl,(%rax)
......
124b: 8b 05        ⑩        mov    ⑩ (%rip),%eax
1251: 83 c0 01              add    $0x1,%eax
1254: 89 05        ⑪        mov    %eax, ⑪ (%rip)
125a: 90                    nop
125b: 5d                    pop    %rbp
125c: c3                    retq

000000000000302e<func>:
......
   ......略去部分和答题无关的信息......
000000000000a000 <buf>:
   ......略去部分和答题无关的信息......
000000000000a250 <bufp1>:
   ......略去部分和答题无关的信息......
0000000000dedd38 <count.1837>:
   ......略去部分和答题无关的信息......
0000000000dedd40 <bufp0>:
   ......略去部分和答题无关的信息......
```

在上图中对所涉及到的重定位条目进行用数字①至⑪进行了标记，请根据下表中所提供的重定位条目信息，计算相应的重定位引用值并填写下表。

编号	重定位条目信息	应填入的重定位引用值 一律填写 32 位 16 进制数
①	r.offset = 0x0a　　r.symbol = 本题不提供 r.type = R_X86_64_PC32　　r.addend = -4	
③	r.offset = 0x22　　r.symbol = 本题不提供 r.type = R_X86_64_PC32　　r.addend = -4	
④	r.offset = 0x17　　r.symbol = 本题不提供 r.type = R_X86_64_32　　r.addend = 0	
⑨	r.offset = 0x15b　　r.symbol = 本题不提供 r.type = R_X86_64_PC32　　r.addend = -4	
⑪	r.offset = 0x172　　r.symbol = 本题不提供 r.type = R_X86_64_PC32　　r.addend = -4	

第四题（10 分）

C 语言中的代码如下：

```
int counter = 0;

void handler(int sig) {
        counter++;
}

int main(int argc, char *argv[])
{
        int fd1, fd2, fd3;
        char c1, c2, c3;
        char *fname = argv[1];
        int parent;

        signal(SIGUSR1, handler);

        fd1 = open(fname, O_RDONLY);
        read(fd1, &c1, 1);
        parent = getpid();
        if (fork()) {
                wait();
                fd2 = open(fname, O_RDONLY);
        } else {
                kill(parent, SIGUSR1);
                fd2 = dup(fd1);
        }
        read(fd2, &c2, 1);

        parent = getpid();
        if (fork()) {
                wait();
                fd3 = open(fname, O_RDONLY);
        } else {
                kill(parent, SIGUSR1);
                fd3 = dup(fd2);
        }

        read(fd3, &c3, 1);
        printf("%c %c %c %d\n", c1, c2, c3, counter);
        return 0;
}
```

当 fname 文件内容为 abcde 时，这段代码的打印结果为____行，输出结果如下：（不考虑出错的情况）

第五题（10 分）

考虑一对 n*n 矩阵相乘问题：C=AB。矩阵乘法函数通常是用 3 个嵌套的循环来实现的，分别用索引 i、j 和 k 来标识。如果改变循环的次序，我们就能得到矩阵乘法的 6 个在功能上等价的版本，如下图所示：

`for (i = 0; i < n; i++)`
`  for (j = 0; j < n; j++) {`
`  sum = 0.0;`
`  for (k = 0; k < n; k++)`
`  sum += A[i][k]*B[k][j];`
`  C[i][j] += sum;`
`  }`
	`for (j = 0; j < n; j++)`
	`  for (i = 0; i < n; i++) {`
	`  sum = 0.0;`
	`  for (k = 0; k < n; k++)`
	`  sum += A[i][k]*B[k][j];`
	`  C[i][j] += sum;`
	`  }`

1）ijk 版本	**2）jik 版本**

`for (j = 0; j < n; j++)`
`  for (k = 0; k < n; k++) {`
`  r = B[k][j];`
`  for (i = 0; i < n; i++)`
`  C[i][j] += A[i][k]*r;`
`  }`
	`for (k = 0; k < n; k++)`
	`  for (j = 0; j < n; j++) {`
	`  r = B[k][j];`
	`  for (i = 0; i < n; i++)`
	`  C[i][j] += A[i][k]*r;`
	`  }`

3）jki 版本	**4）kji 版本**

`for (k = 0; k < n; k++)`
`  for (i = 0; i < n; i++) {`
`  r = A[i][k];`
`  for (j = 0; j < n; j++)`
`  C[i][j] += r*B[k][j];`
`  }`
	`for (i = 0; i < n; i++)`
	`  for (k = 0; k < n; k++) {`
	`  r = A[i][k];`
	`  for (j = 0; j < n; j++)`
	`  C[i][j] += r*B[k][j];`
	`  }`

5）kij 版本	**6）ikj 版本**

本题目具有以下假设：

- 每个数组都是一个 double 类型的 n*n 的数组，sizeof(double)==8
- 虚拟存储分页管理，页大小为 4K 字节，且 A 的起始地址为页对齐，ABC 连续存放
- **TLB 为全相联结构**
- **初始时 TLB 为空，数据全部在硬盘中**
- 一级 dTLB 有 64 表项，采用 LRU（Least-Recently-Used）置换策略
- 分析过程只考虑矩阵操作本身的数据空间，且 i/j/k/n/r/sum 等变量都存放在寄存器中
- 分析过程不考虑进程切换和异常处理程序的影响
- **不考虑对页表本身的访问**

n 的取值	**缺页次数**	**一级 dTLB 失效次数**					
		1-ijk	**2-jik**	**3-jki**	**4-kji**	**5-kij**	**6-ikj**
8							
16							
64							
512							

第六题（10 分）

1. 火锅洞洞主终于要请吃火锅啦！由于防疫规定，洞主决定分批邀请，Alice、Bob、……、Zach 这 26 位同学顺理成章地成为了第一批受到邀请的同学。**他们围坐在一张圆桌旁，按照首字母顺序排列**（即 Alice 的右边是 Bob，Bob 的右边是 Carol，……，Zach 的右边是 Alice）。吃饭之前，洞主想要验证他们的身份，他提出了这样的要求：
    1) 每位同学有一个**编号**，即**字母序号减 1**，也就是说，Alice 是 0 号，Bob 是 1 号，以此类推。
    2) 每位同学需要同时拿着**旁边两位同学**的手机去找洞主（**不必带上自己的手机**），按照洞主的要求在树洞里发信息以验证身份。
    3) **可以同时有多位同学**找洞主验证（也就是说，Alice 需要拿着 Zach 和 Bob 的手机找洞主，并且在同一时间，Bob 也可以拿着 Alice 和 Carol 的手机找洞主。但是，如果 Alice 还没同时拿到 Zach 和 Bob 的手机，就**只能坐在座位上等待**。）

所幸，在座有多位字母君学过 ICS，他们决定采用**信号量**解决问题。

1.1 Alice 给出了这样的代码，其中 thread 函数的参数指明了字母君的编号：

```
1.  #include "csapp.h"
2.  #define N 26
3.  #define UP(i) ((i + 1) % N)
4.  #define DOWN(i) ((i - 1 + N) % N)
5.  sem_t sem[N];
6.
7.  void *thread(void *i){
8.      int num = (int)i;
9.      P(&sem[UP(num)]);
10.     P(&sem[DOWN(num)]);
11.
12.     /* verify identity */
13.
14.     V(&sem[UP(num)]);
15.     V(&sem[DOWN(num)]);
16. }
17. int main(){
18.     for (int i = 0; i < N; i++)
19.         Sem_init(&sem[i], 0, 1);
20.     pthread_t tid[N];
21.     for (int i = 0; i < N; i++)
22.         Pthread_create(&tid[i], NULL, thread, (void *)i);
23.     Pthread_exit(0);
24. }
```

Bob 一看，皱起了眉头。请问这段代码可能会发生什么问题（用一个专用名词表述即可）？

1.2 Carol 把 thread 函数换成了下面这样：

```
1.  void *thread(void *i){
2.      int num = (int)i;
3.      for (int i = 0; i < N; i++)
4.          P(&sem[i]);
5.      /* verify identity */
6.      for (int i = 0; i < N; i++)
7.          V(&sem[i]);
8.  }
```

Dave 指出，这段代码固然正确，但是效率太低了，他认为只要把第 3、4 行的代码换成一个 P 操作，把第 6、7 行的代码换成一个 V 操作，就能实现同样的效果。请问 Dave 的方法是：

1.2.1 P 操作：P(&sem[_____]);

1.2.2 V 操作：V(&sem[_____]);

1.3 Eve 看了 Dave 的方案还是摇头，这样执行效率太低了，没能利用**同时可以有多位**同学找洞主验证的条件。借助“**互斥锁加锁顺序规则**”，他给出了自己的代码。你能把代码补全吗？

```
1.  void *thread(void *i)
2.  {
3.      int num = (int)i;
4.      if (UP(num) < DOWN(num)){
5.          P(&sem[__A__]);
6.          P(&sem[__B__]);
7.      }
8.      else{
9.          P(&sem[__C__]);
10.         P(&sem[__D__]);
11.     }
12.     /* verify identity */
13.     V(&sem[UP(num)]);
14.     V(&sem[DOWN(num)]);
15. }
```

A:_____ B:_____ C:_____ D:_____

以上 4 空均填序号：

① UP(num)

② DOWN(num)

③ num

2. Hungary Alice 发现自己没吃上第一场的火锅，一怒之下，给上面的 26 位字母君出了一道难题：**用二元信号量实现信号量**（也就是值可以是任意非负整数的信号量）。精通 ICS 的你自然认为这是小儿科，迅速补全了以下代码。

```
#include "csapp.h"
typedef struct{
    int value;
    sem_t mutex;
    sem_t zero;
} my_sem_t;
void my_sem_init(my_sem_t *sem, unsigned int value){
    sem->value = value;
    Sem_init(&sem->mutex, 0, __A__);
    Sem_init(&sem->zero, 0, __B__);
}
void my_P(my_sem_t *sem){
    P(&sem->mutex);
    sem->value--;
    if (sem->value __C__ 0)
    {
        __D__;
        __E__;
    }
    else
        V(&sem->mutex);
}
void my_V(my_sem_t *sem){
    P(&sem->mutex);
    sem->value++;
    if (sem->value __F__ 0)
    {
        V(&sem->mutex);
        V(&sem->zero);
    }
    else
        V(&sem->mutex);
}
```

A:_____ B:_____ C:_____

D:_____ E:_____ F:_____

D 和 E 请填序号:

① P(&sem->mutex)

② P(&sem->zero)

③ V(&sem->mutex)

④ V(&sem->zero)