上　海　交　通　大　学　试　卷（ B 卷）

（ 2012　至 2013　学年 第 1 学期 ）

班级号________________________　　学号______________ 姓名 ____________

课程名称　　计算机系统基础（2）______________________成绩 _____________

Problem 1: Concurrency (20points)

1.

2.

3.

4.

Problem 2: Address Translation (28points)

1　1)

2)

我承诺，我将严格遵守考试纪律。

承诺人：________

题号	1	2	3	4	5				
得分									
批阅人(流水阅卷教师签名处)									

2

1) [1] [2] [3] [4]

[5] [6] [7]

2) [1] [2] [3] [4]

[5]

Problem 3: Virtual Memory (30points)

1 1)

2)

2 [1] [2]

[3] [4] [5] [6]

[7] [8]

3 1)

2)

3)

Problem 4: Networking (22points)

1.

2.

3.

Problem 1: Concurrency (20 points)

```
void * do_it1(void * id)
{
    int fd = (*(int *)id);
    char c;
    read(fd, &c, 1);
    int a = n, b;
    n--;
    printf("%c%d",c, n);
    b = n;
    if(b == (a-1)) {
       printf("get here\n");
    }
    return NULL;
}
```

```
void * do_it2(void * id)
{
    n++;
    return NULL;
}
```

```
#include "csapp.h"
int n = 5;
int main_A(void)
{
    int fd;

    fd = open("foo.txt",
              O_RDONLY, 0);

    if(fork()==0){
        do_it1((void*)&fd);
        exit(0);
    }

    wait(NULL);
    do_it1((void*)&fd);

    exit(0);
}
```

```
#include "csapp.h"
int n = 5;
int main_B(void)
{
    int fd;
    pthread_t pid1,pid2,pid3;
    fd = open("foo.txt",
              O_RDONLY, 0);

    pthread_create(&pid1, NULL,
            do_it1, (void*)&fd);
    pthread_create(&pid2, NULL,
            do_it1, (void*)&fd);
    pthread_create(&pid3, NULL,
            do_it2, (void*)&fd);

    pthread_join(pid1,NULL);
    pthread_join(pid2,NULL);
    pthread_join(pid3,NULL);
    exit(0);
}
```

1) Suppose the file `foo.txt` consists of ASCII characters "**`exam_ics`**". What is the output of function **`main_A`** and **`main_B`**? (If there are multiple possibilities, give **two** of them). (4'*2 = 8')

2) In function **`main_B`**, if we want to make sure that "**`printf("get here\n");`**" should be executed every time, how can we modify the program? (NOTE: Modify the function **`main_B`**, **`do_it1`** and **`do_it2`** by **`SEMAPHORE`**). (4')

3) In function **`main_B`**, if we create multiple threads with function **`do_it1()`** and create multiple threads with function **`do_it2()`**, and we want to make sure that the value of **`n`** either be 4 or be 5, how can we modify the program? (NOTE: Modify the function **`main_B`**, **`do_it1`** and **`do_it2`** by using **`SEMAPHORE`**). (4’)

4) Consider the following execution flow that generate **`deadlock`**.

   Initially: **`a = 1, b = 1`**

```
Thread 1:    Thread 2:    Thread 3:
P(b);        P(a);          P(a);
P(a);        V(a);          P(b);
V(a);        P(b);          V(a);
V(b);        V(b);          V(b);
```

   Explain why this may cause deadlock. (4’)

Problem 2: Address Translation (28 points)

This problem concerns the way virtual addresses are translated into physical addresses. Below are the specifications of the system on which the translation occurs:

- ✧ The main memory is byte addressable.
- ✧ The memory accesses are to **1-byte** words (not 4-byte words).
- ✧ The system is configured with **256MB** of virtual memory.
- ✧ Physical addresses are **20 bits** wide (m = 20).
- ✧ Single level page table and the page size is **1KB**.
- ✧ The TLB is **2-way** set associative with **16** total entries.
- ✧ The L1 d-cache is physically addressed and **direct** mapped, with a **4-byte** line size and **16** total sets.

1. Warm-up Questions (2’ * 2 = 4’)

   1) How many bits are needed to represent the virtual address space?

   2) What is the total number of page table entries?

The contents of the TLB, first 16 entries of the page table and cache are given below. All numbers are in hexadecimal. According to the illustration and answer the questions. Please fill in the blanks.

Set	Tag	PPN	Valid	Tag	PPN	Valid
0	03	-	0	09	0D	1
1	00	-	0	07	02	1
2	03	2D	1	02	-	0
3	04	-	0	0A	-	0
4	0B	39	1	06	-	0
5	10	16	1	02	11	1
6	03	04	0	06	15	1
7	07	-	0	01	0D	1

TLB: 8 sets, 16 entries, 2-way set associative

VPN	PPN	Valid	VPN	PPN	Valid
00	7D	1	08	11	1
01	51	0	09	13	1
02	33	1	0A	17	0
03	-	0	0B	9	1
04	2	1	0C	-	0
05	-	0	0D	-	0
06	16	1	0E	2D	1
07	-	0	0F	0D	1

Page Table: Only the first 16 PTEs are shown

Index	Tag	Valid	Blk0	Blk1	Blk2	Blk3
0	19	1	09	EE	12	61
1	15	0	-	-	-	-
2	1B	1	60	17	18	19
3	32	1	43	6D	8F	9
4	21	0	-	-	-	-
5	0E	0	-	-	-	-
6	30	1	11	C2	DF	CD
7	16	1	11	C2	1A	DF
8	24	1	3A	00	51	89
9	2E	0	-	-	-	-
A	2E	1	93	16	DD	3B
B	9D	1	04	96	34	15
C	12	0	-	-	-	-
D	16	0	-	-	-	-
E	1B	0	-	-	-	-
F	14	1	7A	E2	52	22

Cache: 16 sets, 1-byte blocks, direct mapped

2. Address translation

1) Please fill in the following blanks. You may indicate a page fault by entering '--' for PPN and Physical Address (2'*7=14')

Parameter	Value
Virtual Address	0xE693
VPN	0x__[1]__
TLB Index:	0x__[2]__
TLB Tag:	0x__[3]__
TLB Hit?(Y/N)	__[4]__
Page Fault?(Y/N)	__[5]__
PPN	0x__[6]__
Physical Address	0x__[7]__

2) Please fill in the following blanks according. You may indicate a cache miss by entering '--' for Cache Byte returned (2' * 5 = 10')

Parameter	Value
Physical Address	0x059E
Byte offset	0x__[1]__
Cache Index:	0x__[2]__
Cache Tag:	0x__[3]__
Cache hit?(Y/N)	__[4]__
Cache byte returned	0x__[5]__

Problem 3: Virtual Memory (30 points)

This problem is based on a 32-bit Linux system, which has following characteristics especially:

✧ Virtual address is **32 bits** wide.

✧ Page table is organized as a **two-level** structure. `[31:22]` of virtual address is index of L1 page table entry, and `[21:12]` is index of L2 page table entry. The remaining bits represent **`VPO`**.

Consider the following program:

```
#include <unistd.h>
#include <sys/mman.h>
#include <stdio.h>

#define PAGE_SIZE 4 * 1024
int main (void)
{
   char *p1 = NULL;
   char *p2 = NULL;
   int i;
A:
   p1 = mmap(0, 32 * PAGE_SIZE, PROT_READ | PROT_WRITE,
             MAP_PRIVATE | MAP_ANON, 0, 0);
   p2 = mmap(0, 32 * PAGE_SIZE, PROT_READ | PROT_WRITE, __[1]__, 0, 0);

   for (i = 0; i != 32 * PAGE_SIZE; i++)
      p1[i] = 1;
B:
   fork();
C:
   for (i = 0; i != 32 * PAGE_SIZE; i++)
      p2[i] = p1[i];
D:
   return 0;
}
```

1. Warm-up questions (2' * 2 = 4').

   1) How many PTEs (Page Table Entries) are needed totally in the given virtual memory system (including L1 and L2 page table).

   2) Pointer p2 is expected to point to a shared array object which is initialized by zero-filled pages. Please fill in the blank in the program properly.

2. After calling mmap(), p1 is 0xb73e0000 and p2 is 0xb7400000 and the base address of L1 page table is 0xef400000, fill in the following blanks (2'*8 = 16')

    Address range of array pointed by p1: ____[1]____

    Address range of array pointed by p2: ____[2]____

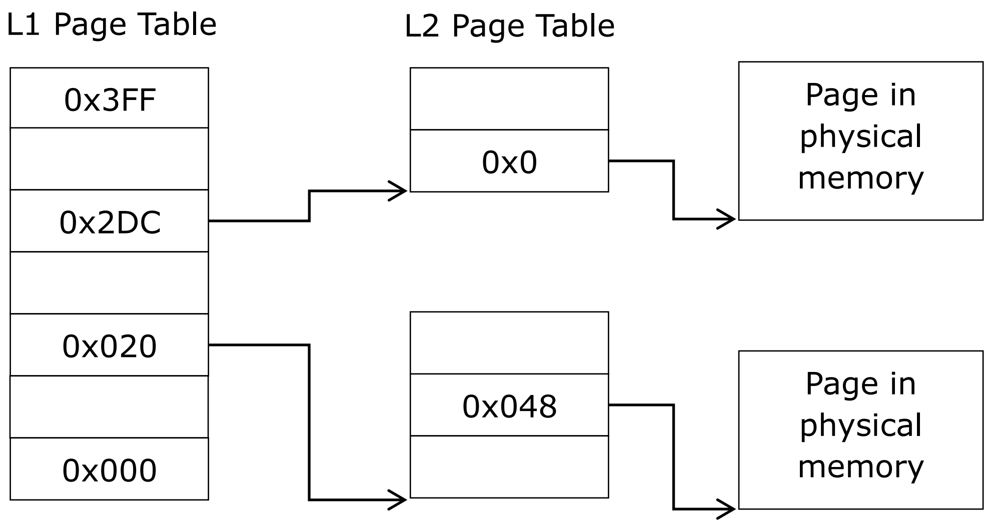
    Index of L1 PTE of p1: 0x___[3]____ Index of L2 PTE of p1: 0x___[4]___

    Index of L1 PTE of p2: 0x___[5]____ Index of L2 PTE of p2: 0x___[6]___

    Address of L1 PTE of p1: 0x___[7]____

    Address of L1 PTE of p2: 0x___[8]____

3. Assume the page table is like below when the program arrives at **label A**. The number within block is the **index** of page table. The white block without number means one or more empty page table entries. (10')



    1) How many page fault exceptions are raised by codes between **label A** and **B**. Please briefly explain your answer. (2' + 2' = 4')

    2) How many page fault exceptions are raised by codes between **label C** and **D**. Please briefly explain your answer. (2' + 2' = 4')

    3) According to virtual memory sketch in Linux, what is the content starting from the address specified by **0x020** in L1 page table and **0x048** in L2 page table? (2')

Problem 4: Networking (22 points)

Your task is to design a simple voting system including **1 server** and **several clients**. The server process serves at `2.3.4.5:8087` and handles requests **concurrently**. There are three objects being voted. Each object has a unique ID (starting from 0).

Communication protocol between the server and the clients is defined as follows:

1. A client can send **VOTE** for some object by sending the following message:
   `VOTE OBJECT_ID`
2. If a **VOTE** received is valid, the server returns the following message:
   `ACK K0 K1 K2`
   (Where **`'Ki'`** is the current number of votes of the object with **`ID=i`**)
3. If a vote received is invalid (when `OBJECT_ID` is invalid), the server returns an error message:
   `NEG`

Examples:

1. Client => Server: **`VOTE 2`**
   Server => Client: **`ACK 5 6 4`**
   Then Client prints: **`5 6 4`**
2. Client => Server: **`VOTE 4`**
   Server => Client: **`NEG`**
   Then Client prints: **`Error`**

We provide the **client side code** (You must first read it carefully before proceeding)

```
int main(void)
{
    char buf[100], head[100]; rio_t rio;
    int objectid, clientfd;

    clientfd = Open_clientfd("2.3.4.5", 8087);
    scanf("%d", &objectid);
    sprintf(buf, "VOTE %d\n", objectid);

    Rio_writen(clientfd, buf, strlen(buf));
    Rio_readinitb(&rio, clientfd);
    Rio_readlineb(&rio, buf, 100);

    sscanf(buf, "%s", head);
    if (!strcmp(head, "ACK"))
        printf("%s\n", buf + 4); /* 4 == strlen("ACK ") */
    else
        printf("Error\n");
    close(clientfd);
    return 0;
}
```

1. Please complete the **`'main'`** function and **`'process_thread'`** function in the **server** side. (6'+8')

```
void *process_request(void *argp);

int votes[3]; /* record the numbers of votes for the three objects */
sem_t mutex; /* to protect what? */

int main(void)
{
   struct sockaddr_in clientaddr; int clientlen, listenfd;
   int *argp; pthread_t tid;

   /* Fill your code here */

   return 0;
}
```

```
void *process_request(void *argp)
{
   char buf[100];
   rio_t rio;
   int clientfd;
   int objectid;

   /* Fill your code here */

   Return NULL;
}
```

2. Suppose a client is the first one to access the server, and the user input is 2. What is the output of the client? (4')
3. If we allow each client to vote for one object only once, please describe your solution. (no need to write codes) (4')