Introduction to Computer Systems 2
2011 Fall Final Examination

Name____________ Student No.______________ Score__________

Problem 1: Address Translation (26 points)

Suppose we have such a memory system:

[1]. The memory is byte addressable.
[2]. Memory accesses are to 1-byte words (not 4-byte words).
[3]. Virtual addresses are **23** bits wide (n=23).
[4]. Physical addresses are **19** bits wide (m=19).
[5]. The page size is **128** bytes.
[6]. It has a two-level page table hierarchy and the following figure shows the formats of the virtual and physical addresses
[7]. A **2-way set associative** with total **8** entries TLB reside in this system for caching PPN only as shown in the following figure

22 15	14 7	6 0
VPN-1	VPN-2	VPO

Virtual Address

18 7	6 0
PPN	PPO

Physical address

The following tables show the snapshot of memory system, including TLB, Page Directory and Page Table. All values are in hexadecimal notation. Here we only give the first ten of the Page Directory Entry, and the first five of Page Table entries at physical address 0x2D and 0x1D.

TLB			
Index	**Tag**	**PPN**	**Valid**
0	17	2E	0
	0A	6D	1
1	0D	0E	0
	03	3A	0
2	0B	39	1
	03	04	0
3	2C	68	1
	10	83	1

Page Directory					
VPN-1	**PDE**	**Valid**	**VPN-1**	**PDE**	**Valid**
000	2D	1	005	02	0
001	11	0	006	1D	1
002	28	0	007	0B	1
003	0D	1	008	17	1
004	33	0	009	18	0

Page Table(0x2D)			Page Table(0x1D)		
VPN-2	**PTE**	**Valid**	**VPN-2**	**PTE**	**Valid**
000	16	0	000	2B	1
001	0C	0	001	2C	0
002	08	1	002	05	1
003	23	1	003	12	0
004	1A	1	004	34	0

1. Choose the proper words given to fill in the blank of the following description about address translation in this memory system: (10')

   Words: [**PD**(Page Directory), **PT**(Page Table), **TLB**, **VPN1**, **VPN2**]

When translate a virtual address to a physical address in this system, MMU first try finding PPN in __[1]__ by use 16 high-order bits of the virtual address. If not find, MMU will use __[2]__ to find page table address in __[3]__ and then use __[4]__ to find PPN (physical page number) in corresponding __[5]__.

2. According the given virtual address, please fill the following blanks (In hexadecimal notation). If the value is unknown or meaningless, enter "**–**" for them (1' * 16 = 16').

   1) Virtual address：**0x1404**

   A. Address translation:

Parameter	**Value**
TLB Index	__[1]__
TLB Tag	__[2]__
TLB Hit?（Y/N）	__[3]__
VPN-1	__[4]__
VPN-2	__[5]__
Page Fault?（Y/N）	__[6]__
PPN	__[7]__

Physical Address	__[8]__

2) Virtual address：**0x30101**

A. <u>Address translation</u>:

Parameter	**Value**
TLB Index	__[1]__
TLB Tag	__[2]__
TLB Hit?（Y/N）	__[3]__
VPN-1	__[4]__
VPN-2	__[5]__
Page Fault?（Y/N）	__[6]__
PPN	__[7]__
Physical Address	__[8]__

Problem 2: Networking (25 points)

In this problem, we look at a simple chat-robot system called ‘SILI’ which is based on C/S model. We first consider the client, then the worker thread on the server, then the main server program. For simplicity, we assume the service runs on 10.131.251.142, port 3011 and the strings being spell-checked never exceed MAXBUF in length. We skip some error checking code for concision.

Client main code:

```
#define DEFAULT_PORT 3011
#define DEFAULT_IP “10.131.251.142”

int main(int argc, char** argv)
{
    int fd, num;
    char buf[MAXBUF];
    struct sockaddr_in serveraddr;
    char *msg = argv[1];

    if (strlen(msg) >= MAXBUF-1)
        exit(1);
    strncpy(buf, msg , MAXBUF);
    fd = socket(AF_INET, SOCK_STREAM, 0);
    serveraddr.sin_family = AF_INET;

    struct in_addr server_address;
    inet_aton(__[1]__, &server_address);
    serveraddr.sin_addr.s_addr = htonl(server_address.s_addr);
    serveraddr.sin_port =  __[2]__;

    connect(fd, __[3]__, __[4]__);

    while (fgets(buf, MAXBUF, stdin) != NULL){
        if (strcmp(buf,”Bye”)==0)
            break;
        rio_writen(fd, buf, strlen(buf));
        rio_readn(fd, buf, MAXBUF-1);
        buf[num] = 0;
        printf("%s", buf);
    }

    exit(0);
}
```

Next, the worker thread for the server. It should be designed so the main server can spawn a separate thread for each request. We assume that “query(buf)” modifies **buf**, so that it contains the result and returns the number of characters in the resulting string.

Server worker code:

```
#define DEFAULT_PORT 3011

int bytes_total = 0;
void* query_thread(void *arg)
{
    int n, clientfd = *(int *)arg;
    char *buf = malloc(MAXBUF);
    n = __[5]__  // read data from client
    bytes_total += n;
    n = query(buf);
    __[6]__  // write back the results to client
    close(clientfd);
    return NULL;
}
```

Server main code:

```
int main(int argc, char **argv)
{
    int listenfd, *clientfd, clientlen, optval=1;
    struct sockaddr_in serveraddr, clientaddr;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR,
                (const void *)&optval, sizeof(int));

    serveraddr.sin_family = AF_INET;
    serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
    serveraddr.sin_port = htons(DEFAULT_PORT);
    __[7]__
    listen(listenfd, 5);
    clientlen = sizeof(clientaddr);
    while (1) {
        pthread_t thread;
        clientfd = malloc(sizeof(int));
        *clientfd = __[8]__
        pthread_create(&thread, NULL, query_thread, clientfd);
        pthread_detach(thread);
    }
}
```

1. Fill the above blanks with C code. (2’*8=16’)
2. The code in function “**`query_thread()`**” may bring a race condition, describe this situation (3’) and propose an answer to solve this problem. (3’)
3. Why we need “**`pthread_detach`**” in the main server function? (3’)

Problem 3: Signal (20 points)

Read the following program and answer the questions

```c
#include <stdio.h>
#include <fcntl.h>
#include <signal.h>
#include <unistd.h>
#include <wait.h>
#include <stdlib.h>

int fd[3];
char ch;
int count = 0;

void sigchld_hander(int sig) { count += 1; close(fd[1]); }
void sigusr1_hander(int sig) { count += 10; }
void sigusr2_hander(int sig) { count += 100; ch = 'd'; }

int main(int argc, char **argv)
{
    int pid;

    signal(SIGCHLD, sigchld_hander);
    signal(SIGUSR1, sigusr1_hander);
    signal(SIGUSR2, sigusr2_hander);

    fd[0] = open("in.txt", O_RDWR, 0);
    fd[1] = open("in.txt", O_RDWR, 0);

    dup2(1, 0); dup2(2, 1);
    ch = 'a'; write(STDIN_FILENO, &ch, 1);
    ch = 'b'; write(STDOUT_FILENO, &ch, 1);
    ch = 'c'; write(STDERR_FILENO, &ch, 1);

    if ((pid = fork()) == 0) {
        kill(getppid(), SIGUSR1);
        kill(getppid(), SIGUSR1);

        fd[2] = open("out.txt", O_RDWR, 0);
        read(fd[0], &ch, 1); write(fd[2], &ch, 1);
        dup2(fd[0], fd[2]);
        read(fd[0], &ch, 1); write(fd[2], &ch, 1);

41    } else {
42        kill(pid, SIGUSR2);
43
44        fd[2] = open("out.txt", O_RDWR, 0);
45        while (waitpid(-1, NULL, 0) > 0) ;
46        read(fd[0], &ch, 1); write(fd[0], &ch, 1);
47        read(fd[2], &ch, 1); write(fd[2], &ch, 1);
48    }
49
50    exit(0);
51 }
```

NOTE: Suppose each time before the program executes, the content in the file "in.txt" is "ABCDEFG" and the content in the file "out.txt" is "HIJKLMN". Assume all system calls are successful and each time the program executes and exits normally.

1) If we execute "**./prog > tmp.txt**", what will be printed on the Console and what will be written into the file "tmp.txt"? (If nothing is printed or written, please fill with "None".) (2'*2 = 4')

   Console: __**[1]**__ tmp.txt: __**[2]**__

2) What are the possible values of "**fd[2]**" when the program executes at the **line 44**? Please write out all of them. (4')

3) What are the possible values of the variable "**count**" in the parent process and the child process just before they exit? Please write out them. (3'*2 = 6')

   Parent: __**[1]**__ Child: __**[2]**__

4) What are the possible contents in the file "in.txt" and "out.txt" when the program finishes? Please write out 2 possible results in pair (in.txt, out.txt). (3'*2 = 6')

   Pair-1: __**[1]**__ Pair-2: __**[2]**__

Problem 4: Virtual Memory (29 points)

The program below runs on the Pentium/Linux Memory System discussed in section 10.7 of the CSAPP. It will allocate **writable** memory for two arrays of type **integer** at first, and the number of elements in arrays is **ARRAY_LEN**. The array with name "**sarr**" is allocated as **shared**, while "**larr**" is marked as **private**. After these arrays have been initialized, the process will **fork** a child process, and then both parent process and child process will try to modify the value of the arrays.

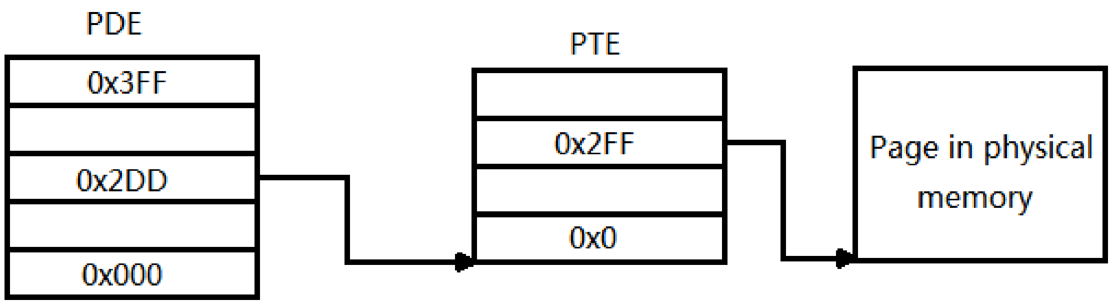
```
#include <unistd.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <stdio.h>

#define ARRAY_LEN 10240
int main(void)
{
    int *sarr, *larr;
    int i;
    pid_t pid;
A:
    sarr = mmap(-1,  __[1]__, __[2]__,  __[3]__, 0, 0);
    larr = mmap(-1,  __[4]__, __[5]__,  __[6]__, 0, 0);
    printf("0x%x,0x%x\n",(unsigned int)sarr,(unsigned int)larr);
    for(i=0; i<ARRAY_LEN; i++) {
        sarr[i]=i;   larr[i]=i;
    }
B:
    fork();
C:
    pid = getpid();
    for(i=0; i<ARRAY_LEN; i++) {
        sarr[i] = pid + i;
        larr[i] = pid + i;
    }
D:
}
```

NOTE: We assume that the parent process ends **BEFORE** the child process starts, thus these two processes have **NO** interleaving.

When the program arrives at label **A**, the page table is like below. The number within block is the index of PDE/PTE. The output of the function "`printf()`" is "`0xb77fa000,0xb77f0000`".

NOTE: The white block without number means one or more empty PDE/PTE, the white block with number means a filled PDE/PTE. Like in the page table, the white block above 0x2ff means that all PTEs in that page table that have index larger than 0x2ff is empty. **Level 1** page table entry is 32 bits.



1) Fill in the blank in the program. (1'*6=6')
2) How many page fault exceptions are triggered by the code between label **A** and label **B**? (4')
3) Draw a graph like the one given above to indicate the page table when the program reach label **B**. You can use a black block to represent multiple filled PDE/PTEs. (12')
4) During the execution between label C and label D in both processes, how many protection faults will be triggered due to copy-on-write? (3') explain the benefits of COW. (4') (NOTE: It is strongly recommended that you explain this with a simple example. You can give me the example in any form you like.)