Introduction to Computer Systems
2016 Spring Middle Examination

Name_____ Student No._____ Score_____

Problem 1: (12 points)

1.

2.

3.

Problem 2: (12 points)

1. [1] **[2]**

2. [1] **[2]**

3.

Problem 3: (18 points)

1 **[1]** **[2]** **[3]** **[4]**

[5] **[6]** **[7]** **[8]**

[9] **[10]** **[11]** **[12]**

2.

3.

Problem 4: (18 points)

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

[5] [6] [7] [8]

2. [1]

[2]

[3]

[4]

[5]

Problem 5: (7 points)

1

2 [1]

[2]

Problem 6: (15 points)

1

2.

3.

Problem 7: (18 points)

1.

2.

3.

4

Problem 1: Process (12 points)

```
1.  char *ch;
2.
3.  int main() {
4.      ch = (char *) malloc(1);
5.      *ch = 'A';
6.
7.      if (Fork() == 0) {
8.          *ch = 'B'; printf("%c\n", *ch);
9.
10.         if (Fork() == 0) {
11.             printf("I'm process C\n");
12.         } else
13.             exit(0);
14.     } else {
15.         while (waitpid(-1, NULL, 0) > 0);
16.         char *argv[] = {"/bin/echo", ch, 0};
17.         Execve(argv[0], &argv[0], 0);
18.     }
19.
20.     free(ch);
21.     return 0;
22. }
```

Note: **/bin/echo** is an executable file that will print its arguments on the screen. No error occurs in the execution.

1. Please write down the output of the **child** and **parent** processes of the **fork** in line 7. (4')
2. Is the output **deterministic**? Please explain why. (4')
3. Please point out whether there is a memory leakage or double free problem about variable **ch** in each process? why? (4')

Problem 2: IO (12 points)

```
1.  int main() {
2.      int fd1, fd2, fd3;
3.      char c;
4.      fd1 = open("foo.txt", O_RDWR, 0);
5.      fd2 = open("bar.txt", O_RDWR, 0);
6.      fd3 = open("baz.txt", O_RDWR | O_CREAT | O_TRUNC,
7.                             S_IRUSR | S_IWUSR);
8.      dup2(fd2, fd1);
9.      if (Fork() == 0) {
10.         read(fd1, &c, 1); printf("%c", c);
11.         // close(fd1);
12.         fd1 = open("foo.txt", O_RDWR, 0);
13.         read(fd1, &c, 1); printf("%c\n", c);
14.     } else {
15.         wait(NULL);
16.         dup2(fd1, fd2);
17.         read(fd1, &c, 1);
18.         write(fd3, &c, 1);
19.         read(fd2, &c, 1);
20.         write(fd3, &c, 1);
21.     }
22.     close(fd1); close(fd2); close(fd3);
23. }
```

Note: Initially, **foo.txt** contains "**SJTU**"; **bar.txt** contains **"12345"**; **baz.txt** does not exist. No error occurs in the execution.

1. Please write down the output on the **screen** and the content of **baz.txt** when the program runs normally. (4')

   screen: __[1]__ **baz.txt**: __[2]__

2. Please write down the **refcount** of the file table for **foo.txt** and **bar.txt** in **line 10**. (4') NOTE: write 0 if the file table has already been freed.

   foo.txt:__[3]__ **bar.txt**:__[4]__

3. What is the value of **fd1** in **line 12**? (2') If **line 11** is uncommented, what is the value of **fd1** in **line 12**. (2')

Problem 3: Symbol (18 points)

The following program consists of two modules: **main** and **white**. Their corresponding source code files are shown below. (All the process of linking runs on an x86 machine)

/* main.c */

```
#include <stdio.h>

extern char *names[];

static int id;

int white(int n);

void main(void) {
    id = 103;
    char *str =
        names[white(id)];
    printf("%s %d\n",
            str, id);
}
```

/* white.c */

```
char *names[] = {"Chunxi",
   "Xuecai", "Dongma", "Xiaochun"};

int id = 102;

int white(int n) {
    int res = 0;
    switch(n) {
      case 100:
        res = 1; break;
      case 103:
        res =  2; break;
      case 104:
        res =  3; break;
      default:
        res =  0;
    }
    id = 233;
    return res;
}
```

1. For symbols that are defined and referenced in **main.o** and **white.o**, please complete the symbol tables. The format of them are same as ones in **section 7.5** of your ICS book. (1'x12=12')

Module	Name	Type	Bind	Value(Hex)	Size	Ndx
main.o	id	[1]	[2]	00000000	[3]	4
	main	[4]	[5]	00000000	88	[6]
	white	[7]	GLOBAL	00000000	[8]	[9]
white.o	id	OBJECT	[10]	00000010	[11]	[12]

2. Please explain why the **Value** of **id** in **white.o** is **0x00000010**. (2')
3. Please write down the output of **main.c**. (4')

Problem 4: Relocation (18 points)

The following program consists of two source files: **main.c** and **white.c**. The relocatable object files are also listed. (All the process of linking runs on an x86 machine)

```
/* main.c */
extern char *names[];
static int id;
char **str =
           &names[2];

int white(int n);

void main(void) {
  id = 103;
  white(id);
}
```

```
/* main.o */
.text
00000000 <main>:
... # omit operations on stack
14: c7 05 00 00 00  movl $0x67,0x0
    00 67
1b: 00 00 00
1e: a1 00 00 00 00   mov  0x0,%eax
23: 50               push %eax
24: e8 fc ff ff ff  call 25 <main+0x25>
...

.data:
00000000 <str>:
 0: 08 00 00 00
```

```
/* white.c */
char *names[] =
       {"Chunxi",
        "Xuecai",
        "Dongma",
        "Xiaochun"};
int id = 102;

int white(int n) {
  int res = 0;
  switch(n) {
  case 100:
    res = 1; break;
  case 103:
    res =  2; break;
  case 104:
    res =  3; break;
  case 106:
    return res;
  case 110:
    return white(0x68);
  default:
    res =  0;
  }
```

```
/* white.o */
.text
00000000 <white>:
... # "n" is stored in %eax at first
 7: 83 e8 64         sub  $0x64,%eax
 a: 83 f8 0a         cmp  $0xa,%eax
 d: 77 24            ja   33 <white+0x33>
 f: ff 24 85 00 00  jmp  *0x0(,%eax,4)
    00 00
16: b8 02 00 00 00   mov  $0x2,%eax
1b: eb 22            jmp  3f <white+0x3f>
...# omit some cases
27: 6a 68            push $0x68
29: e8 fc ff ff ff  call 2a <white+0x2a>
2e: eb 1d            jmp  50 <white+0x50>
...
3a: b8 01 00 00 00  mov  $0x1,%eax
3f: c7 05 00 00 00  movl $0xe9,0x0
    00 e9
46: 00 00 00
49: eb 05            jmp  50 <white+0x50>
... # ret
```

`  id = 233;`
`  return res;`
`}`
	`.rodata:`
	`00: 3a000000 33000000 33000000 16000000`
	`10: 1d000000 33000000 4b000000 33000000`
	`20: 33000000 33000000 24000000`

1. Fill in the relocation entries of **main.o** and **white.o** respectively. (1'*8=8')

Relocation entries of **main.o**

Section	Offset	Type	Symbol Name
.text	00000016	R_386_32	[1]
.text	00000025	[2]	[3]
.data	00000000	R_386_32	[4]

Relocation entries of **white.o**

Section	Offset	Type	Symbol Name
.text	00000012	R_386_32	[5]
.text	0000002a	[6]	white
.text	00000041	R_386_32	[7]
.rodata	00000000	[8]	white

2. After relocation and the program is built, what changes will happen to the underlined instructions/data according to a part of the symbol table and partial comparison of relocation tables given below? (2'*5=10')

Name	Section	Type	Value
white	.text	FUNC	080483db
main	.text	FUNC	0804842f
.rodata	.rodata	NOTYPE	080484f0
id	.data	OBJECT	0804a018
names	.data	OBJECT	0804a01c
id	.bss	OBJECT	0804a034

A comparison of relocation table of **main.o**

Section	Before relocation	After Relocation
.text	14: c7 05 <u>00 00 00 00</u> 67 00 00 00	[1]
.text	24: e8 <u>fc ff ff ff</u>	[2]
.data	0: <u>08 00 00 00</u>	[3]

A comparison of relocation table of **white.o**

Section	Before relocation	After relocation
.text	3f: c7 05 <u>00 00 00 00</u> e9	[4]
.rodata	00: <u>3a 00 00 00</u>	[5]

Problem 5: ELF (7 points)

There are two segment header tables for two different **executable ELF files**. You can reference **Figure 7-12** of your ICS book. (All the process of linking runs on an x86 machine)

```
# Read-only code segment
LOAD off 0x00000000 vaddr 0x08048000 paddr 0x08048000
align 2**12 filesz 0x000005a4 memsz 0x000005a4 flags r-x

# Read/write data segment
LOAD off  0x000005a4 vaddr ___[1]___ paddr ___[1]___
align 2**12 filesz 0x00000120 memsz 0x00000128 flags rw-
```

```
# Read-only code segment
LOAD off 0x00000000 vaddr 0x08048000 paddr 0x08048000
align 2**12 filesz 0x00004598 memsz 0x00004598 flags r-x

# Read/write data segment
LOAD off 0x00004f08 vaddr ___[2]___ paddr ___[2]___
align 2**12 filesz 0x00000628 memsz 0x00000ad8 flags rw-
```

1. In the read/write data segment, please explain why **`memsz`** is usually larger than **`filesz`**. (3')
2. Please fill blanks in the segment header tables. (2'*2=4')
   NOTE:
   1) The alignment requirement should be considered both in memory and ELF files. The loader should copy segments in files into memory in unit of page (4KB).
   2) Different segments should be assigned into different pages in memory.

Problem 6: Signal (14 points)

Reading the following code and answer the questions.

```
#include "csapp.h"
int count = 0;
void handler_usr1(int sig) { count += 1; }
void handler_usr2(int sig) { count -= 1; }
int main(void) {
    pid_t pid;
    int status;

    Signal(SIGUSR1, handler_usr1);
    Signal(SIGUSR2, handler_usr2);

    if ((pid = Fork()) == 0) {
        printf("1");
        Kill(getppid(), SIGUSR1);
    } else {
        printf("2");
        Kill(pid, SIGUSR2);
    }

    if (pid == 0){
        if ((pid = Fork()) == 0){
            printf("3");
            Kill(getppid(), SIGUSR1);
        } else {
            printf("4");
            Kill(pid, SIGUSR2);
        }
    }
    if (pid > 0) {
        while(Waitpid(-1,&status,0) > 0) {
            if (!WIFEXITED(status))
                printf("child terminated abnormally!\n");
        }
    }
    printf("\n");
    // printf("Count: %d \n", count);
}
```

Consider the origin process is father process. The process created by the first Fork() is child process. The process created by the second Fork() is grandchild process.

Suppose that

I. **`printf()`** function will not be interrupted by signals and immediately outputs the results.

II. No error occurs.

III. 'kill' command will always successfully send signal to the program and the program doesn't ignore any signals

1. Please draw the output dependency diagram for the program. (You only need to focus on printf("1"), printf("2"), printf("3"), printf("4"), printf("\n")) (5')
2. Please write down all the possible outputs of the underlined "**`// printf("Count: %d \n", count);`**" when it is uncommented. (you should pay attention to the relative order in each possible output) (6')
3. Base on question 2. Assume that the pids of the processes are 7543(father), 7544(child), 7545(grandchild). Please list one possible combination of three 'kill' commands to let the program output:

   Count: -4

   Count: -1

   Count: 2

   (you should also pay attention to their relative order) (3')

Problem 7: Networking (18 points)

Your task is to design a simple system to get TA's contact information. This system consists of one server and several clients. The hostname of the server is **`ipads.se.sjtu.edu.cn`**, and the port number is **`6666`**.

Suppose you have some questions about one of the labs, and you want to contact the TA who is responsible for this lab. Firstly, you need to send a request to get the lab lists containing the information of each lab's lead person. Then you will know the TA's name for this lab. Finally, you need to send a request again using the TA's name to query for the contact information.

The detailed **communication protocol** between the server and the clients is defined as follows:

1. A client sends the **LABLIST** request to get the information of labs:

   "LABLIST\n"

2. The server receives the **LABLIST** request and returns the following messages containing lab information: (**NOTICE:** there is a space between the lab's name and TA's name, and each pair is followed by '\n', finally **END LABLIST** identifies the end of the messages)

   "lab1 dzy" '\n' "lab2 ssj" '\n' "lab3 yqq" '\n' "lab4 yyy" '\n' ... "lab10 dzy" '\n' "END LABLIST" '\n'

3. If the name of the lab provided by the user doesn't appear in the lab list, the client prints an error message and exits:

   "Invalid labname!\n"

4. If the name of the lab is found in the list, a client will send the **INFO** request to get the contact information:

   "INFO yyy\n"

5. The server receives the **INFO** request and returns the contact information:

   "yaoyouyang666@gmail.com\n"

Examples:

1] Command: ./queryInfo lab4
Client => Server, sends request: "LABLIST\n"
Server => Client, sends response: "lab1 dzy" '\n' "lab2 ssj" '\n' "lab3 yqq" '\n' "lab4 yyy" '\n' "END LABLIST" '\n'

```
   Client => Server, sends request: "INFO yyy\n"
   Server => Client, sends response: "yaoyouyang666@gmail.com\n"
   Client prints "yaoyouyang666@gmail.com\n"
2] Command: ./queryInfo lab007
   Client => Server, sends request: "LABLIST\n"
   Server => Client, sends response: "lab1 dzy" '\n' "lab2 ssj"
   '\n' "lab3 yqq" '\n' "lab4 yyy" '\n' "END LABLIST" '\n'
   Client prints "Invalid labname!\n"
```

Please complete the following **client-side code**.

Hint: you can use **`sprintf`** to construct the requests. Also you can use **`sscanf`** to parse the messages from the server.

```
#include "csapp.h"

/* the program is executed by ./queryInfo <labname> */
int main(int argc, char **argv) {
   char *labname = argv[1];
   char send_buf[1024], recv_buf[1024];
   memset(send_buf, 0, 1024);
   memset(recv_buf, 0, 1024);

   /* connect to <hostname>:<port> and send out the request */
   [1] // Write your code here (3')

   /* read the lablist information from server */
   /* get the TA's name */
   [2] // Write your code here (9')

   /* send INFO request */
   [3] // Write your code here (3')

   /* finally get the contact information
      and print to the screen */
   [4] // Write your code here (3')

   return 0;
}
```