Homework 09

1. *FORK AND EXECVE*. Read the C program and answer the questions below. NOTE: /bin/echo is an executable file that will print its arguments on the screen.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include  <sys/types.h>  #include
<sys/wait.h> char *ch;

int main()
{ ch = malloc(1); *ch = ’A’;

          if (fork() == 0) { *ch = ’B’; printf("%c\n",
                    *ch);

                    if (fork() == 0) {
                              printf("C\n");
                    }
                    else {
                              exit(0);
                    }
          }
          else { while (waitpid(-1, NULL, WUNTRACED) > 0); char *my_argv[] =
                    {"/bin/echo", ch, 0}; execve(my_argv[0], my_argv, 0);
          }

          free(ch); return 0;
}
```

   (a) What is the possible output of this program? Is the output deterministic? Please explain why.

   (b) Is there any memory leakage or double free issue for the variable ch in each process? Please explain why.

2. *BLOCKINGANDUNBLOCKINGSIGNALS*. Linux provides an explicit mechanism for blocking signals. Read the C program and answer the questions below.

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

```
5
6           void sig_han(int sig)
7           {
8           printf("signal handled\n");
9           }
10
11            int main()
12            {
13            sigset_t set;
14            int i;
15
16            signal(SIGKILL, sig_han);
17            signal(SIGINT, sig_han);
18            sigemptyset(&set);
19            sigaddset(&set, SIGINT);
20            sigprocmask(SIG_BLOCK, &set, NULL);
21
22                        for (i = 0; i < 3; i++) {
23                        printf("send signal\n");
24                        kill(getpid(), SIGINT);
25                        }
26
27            sigprocmask(SIG_UNBLOCK, &set, NULL);
28            return 0;
29            }
```

(a) When run, the call at line 16 fails. Why? And what is the return value of this call and what value would errno be set to? (You can run the program or refer to the man page signal(2)).

(b) What is the output of this program? Please explain your answer.

3. (Optional) *SIGNAL HANDLER INSIDE*. For a user-defined signal handler (suppose a x86-64 machine runnnig Linux),

   (a) does it run in user mode or kernel mode (that is, in non-privileged mode or privileged mode)?

   (b) what stack does it use, does it share the same stack with your normal functions? Use GDB to stop inside a signal handler and check the stack address.

   (c) where does the signal handler function return to, what is the next instruction after the signal handler function retq? Use GDB to stop inside a signal handler and step instruction-by-instruction to check where is the handler function returns to (You may find GDB's si, ni, disassemble commands useful).