# ICS EXE 3

March 17, 2021

## 1 Process

Assume we have the following code:

```
1  int num = 3;
2  int main(void)
3  {
4    for (int i = 0; i < 2; ++i) {
5      pid_t pid = fork();
6
7      if (pid == 0) {
8        num++;
9        printf("child with i: %d, num: %d\n", i, num);
10     } else {
11       waitpid(pid, NULL, 0);
12       printf("parent with i: %d, num: %d\n", i, num);
13     }
14
15     num <<= 1;
16   }
17   return 0;
18 }
```

### 1.1

How many times will **fork()** be called in this program? How many times will **printf()** be called in this program?

3 6

### 1.2

If **waitpid()** in line 11 is removed, will the times that **fork()** is called change?

No

### 1.3

Please give a possible output of the program.

child with i: 0, num: 4
child with i: 1, num: 9
parent with i: 1, num: 8
parent with i: 0, num: 3
child with i: 1, num: 7
parent with i: 1, num: 6
It is the only possible output.

## 2 Signal1

Assume we have the following code:

```
1  void sig_han(int sig)
2  {
3    printf("signal handled\n");
4  }
5
6  int main()
7  {
8    sigset_t set ;
9    int i ;
10
11   signal(SIGKILL, sig_han);
12   signal(SIGINT, sig_han);
13   sigemptyset(&set);
14   sigaddset(&set, SIGINT);
15   sigprocmask(SIG_BLOCK, &set, NULL);
16
17   for (i = 0; i < 3; i++) {
18     printf("send signal\n");
19     kill(getpid() , SIGINT);
20   }
21
22   sigprocmask(SIG_UNBLOCK, &set, NULL);
23   return 0;
24 }
```

### 2.1

When run, the call at line 11 fails. Why? And what is the return value of this call? (You can refer to **Section 8.5** and the **man** page **signal(2)**)

"SIGKILL can be neither caught nor ignored." (Section 8.5, Figure 8.26). signal(SIGKILL, sig_han) returns SIG_ERR.

## 2.2

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

The output is:
send signal
send signal
send signal
signal handled

The pending signal is not delivered to the process when the signal is blocked. So the control flow of the program will not change to the signal handler until the signal is unblocked. The signal will not be delivered multiple times because pending signals are not queued. "the existence of a pending signal merely indicates that at least one signal has arrived" (Section 8.5.5, Correct Signal Handling).

# 3 Signal2

Assume we have the following code

```
1  void handler(int sig) {
2    static int beeps = 0;
3    printf("BEEP\n");
4    if (beeps < 4) {
5      beeps += 1;
6      fork();
7      /* next SIGALRM will be delivered in 1s */
8      alarm(1);
9    } else {
10     printf("BOOM!\n");
11     exit(0);
12   }
13 }
14 int main() {
15   /* install SIGALRM handler */
16   signal(SIGALRM, handler);
17
18   /* next SIGALRM will be delivered in 1s */
19   alarm(1);
20
21   /* signal handler returns control here each time */
22   while (1);
23
24   exit(0);
25 }
```

## 3.1

How many seconds will this program remain approximately?

5 seconds

## 3.2

How many **BEEPs** and **BOOMs** will be printed if you run the above program?

1+2+4+8+16=31 BEEPs, 16 BOOMs