Advanced Programming
COMS 3157

April 19, 2025

Patrick Shen

pts2125@columbia.edu

---

1. Q1? (2 marks)
   (a) What is a signal?
   (b) What is a signal handler?
2. Do the signal() and sigaction() methods pause the flow of code? (1 mark)
3. How would each of these signals be triggered? (4 marks)
   (a) SIGFPE
   (b) SIGINT
   (c) SIGTSTP
   (d) SIGCONT
4. Which two signals cannot be handled? (2 marks)
5. Briefly explain each argument in for sigaction(int signum, const struct sigaction *act, struct sigaction *oldact) (3 marks)
   (a) int signum
   (b) struct *act
6. What do each field for in the sigaction struct? (4 marks)
   (a) void (*sa_handler)(int);
   (b) void (*sa_sigaction)(int, siginfo_t *, void *);
   (c) sigset_t sa_mask;
   (d) int sa_flags;

```c
struct sigaction {
    void     (*sa_handler)(int);
    void     (*sa_sigaction)(int, siginfo_t *, void *);
    sigset_t   sa_mask;
    int        sa_flags;
    void     (*sa_restorer)(void); // obsolete, ignore
};
```

Listing 1: sigaction struct

7. Briefly explain what each function does for sa_mask in the sigaction struct (3 marks)
   (a) int sigemptyset(sigset_t *set)
   (b) int sigaddset(sigset_t *set, int signum)
   (c) int sigfillset(sigset_t *set)

8. What does the function call memset() do here? (1 mark)

```
1    struct sigaction act;
2
3    memset (&act, '\0', sizeof(act));
```

Listing 2: memset()

9. What does act = {0} do here? (1 mark)

```
1    struct sigaction act;
2
3    act = {0};
```

Listing 3: act

10. Suppose a SIGTERM signal comes in. What is the output? (1 mark)

```
1    static void hd1 (int sig, siginfo_t *siginfo, void *context)
2    {
3        printf("SIGTERM receieved.");
4    }
5
6    ....
7
8    struct sigaction act;
9
10   memset (&act, '\0', sizeof(act));
11
12   act.sa_sigaction = &hd1;
13   act.sa_flags = SA_SIGINFO;
14
15   if (sigaction(SIGTERM, &act, NULL) < 0)
16   {
17       perror("sigaction");
18       return 1;
19   }
```

Listing 4: simple example

11. What is the output of the code in these two scenarios? (2 marks)
   (a) Sending a SIGINT signal through Ctrl + C `immediately` after the code runs.

(b) Sending a SIGINT signal after 6 seconds.

```c
void sig_handler_2(int signum) {
    sleep(signum); // sleep for duration equal to signal number
    printf("%d\n", signum);
}

void B(void) {
    struct sigaction sa;
    memset(&sa, 0, sizeof(sa));          // Zero out the struct
    sa.sa_handler = sig_handler_2;       // Set handler function
    sigemptyset(&sa.sa_mask);            // Initialize signal mask to
   empty
    sigaddset(&sa.sa_mask, SIGINT);      // Block SIGINT during
   handler

    sigaction(SIGQUIT, &sa, NULL);       // Set handler for SIGQUIT

    kill(getpid(), SIGQUIT);             // Send SIGQUIT to self
    sleep(2);                            // Sleep after sending signal
    printf("%d\n", 200);
}

int main(void) {
    B();
    return 0;
}
```

Listing 5: pending signals

12. What is the output of the code? If a process was terminated, write "TERM". (1 mark)

```c
void C(void) {
    struct sigaction sa = {0}; /* memset struct to 0 */
    sa.sa_handler = sig_handler_1;
    pid_t pid = fork();
    if (pid == 0) {
        sigaction(SIGTERM, &sa, NULL);
        sleep(2);
        kill(getppid(), SIGTERM);
        kill(getpid(), SIGKILL);
    }
}

int main(void) {
    C();
    return 0;
}
```

Listing 6: signals and fork

13. What do the following keywords in C do? (2 marks)

(a) volatile

(b) sig_atomic_t

```
volatile sig_atomic_t signal_val = 0;
```

Listing 7: keywords

14. What does the raise(int iSig) function do? (1 mark)

15. What does the kill(pid_t iPid, int iSig) function do? (1 mark)

16. What is the output for each of these commands? The code is stored in a executable named "sleep". (2 marks)

(a) ./sleep 2 (Ctrl + C is not sent)

(b) ./sleep 5 (Ctrl + C is sent 4 seconds in)

```
    void catch_signal(int sig) {
        got_signal = 1;
    }

    int main(int argc, char *argv[]) {
        if (argc != 2) {
            fprintf(stderr, "Usage: %s <seconds>\n", argv[0]);
            return EXIT_FAILURE;
        }

        int max_snooze_secs = atoi(argv[1]);
        if (max_snooze_secs <= 0) {
            fprintf(stderr,
            "Error: Invalid number of seconds '%s' for max snooze
   time.\n",
            argv[1]);
            return EXIT_FAILURE;
        }

        struct sigaction action = {0};
        action.sa_handler = catch_signal;
        action.sa_flags = SA_RESTART;
        if (sigaction(SIGINT, &action, NULL) == -1) {
            perror("sigaction");
            return EXIT_FAILURE;
        }

        int count = 0;
        while (!got_signal && count < max_snooze_secs) {
            sleep(1);
            count++;
        }
        printf("Slept for %d of the %d seconds allowed.\n",
        count, max_snooze_secs);
```

```
34
35          return EXIT_SUCCESS;
36      }
```

Listing 10: sleep() example

17. Answer the following questions about the alarm(int time) function. (2 marks)
    (a) What does the alarm(int time) function do?
    (b) What happens if the time argument is set to 0?

18. If SIGALRM is not caught, what is the default behavior of SIGALRM? (1 mark)

19. What is the output of this code? (1 mark)

```
1  static void myHandler(int iSig)
2  {
3      printf("In myHandler with argument %d\n", iSig);
4      alarm(2); /* Set another alarm */
5  }
6
7  int main(void)
8  {
9      signal(SIGALARM, myHandler);
10     alarm(2); /* Set an alarm */
11     printf("Entering an infinite loop\n");
12     for (;;)
13         ;
14     return 0;
15 }
```

Listing 11: alarm() example

20. What is the output for each of these scenarios? (1 mark)
    (a) you enter the number '4' after 4 seconds
    (b) you enter the number '7' after 7 seconds

```
1  static void myHandler(int iSig)
2  {
3      printf("\nSorry. You took too long.\n");
4      exit(EXIT_RETURN);
5  }
6
7  itn main(void)
8  {
9      int i;
10     signal(SIGALRM, myHandler);
11     printf("Enter a number: ");
12     alarm(5);
13     scanf("%d", &i);
```

```
14      alarm(0);
15      printf("You entered the number %d.\n", i);
16      return 0;
17  }
```

Listing 12: time bomb example