Final Programming Test (10%)

CS5432 Advanced UNIX Programming, Instructor: Cheng-Hsin Hsu

Department of Computing Science, National Tsing Hua University, Taiwan

6:00 p.m., Dec. 28th – 6:00 a.m., Dec. 29th, 2023

- **Please use the C language for your answers (C++ is not allowed).**
- **You are allowed to search online for tips, but NOT allowed to use ChatGPT.**
- **You are not allowed to copy and paste source code from the Internet.**
- **Every output of your codes should follow the same format as the sample outputs.**
- **Please submit your source codes named *q1.c, q2.c, q3.c*, a single *Makefile* that can compile all your source codes simultaneously, and a *report* that describes your implementation and the way to run your code for each question to eeclass.**
- **You may assume that all the inputs are quite normal and do not need to handle exceptions.**
- **You should ensure that all your outputs are identical to the outputs in FreeBSD.**

1) (3%) Implement a single program to create three threads in the order of T1, T2, and T3. Then configure the threads properly so that `SIGINT`, `SIGTERM`, and `SIGUSR1` are exactly handled by T1, T2, and T3, respectively.

   Sample outputs

```
mao@MaosMBP:~/Desktop$ ./q1 &
[1] 3747
mao@MaosMBP:~/Desktop$ kill -USR1 %1
T3 handling SIGUSR1
mao@MaosMBP:~/Desktop$ kill -INT %1
T1 handling SIGINT
mao@MaosMBP:~/Desktop$ kill -TERM %1
T2 handling SIGTERM
```

2) (3%) Implement a function `sleep_us`, which is similar to sleep, but waits for a specified number of microseconds. You may build it upon `select` or `poll`. Instrument your code to measure whether your `sleep_us` returns in the precise time. Please record the time before and after your `sleep_us`, and print the elapsed (sleep) time in microseconds.

   Sample outputs

```
mao@MaosMBP:~/Desktop$ ./q2 1000000
Sleep time: 1000555 us
```

3) (4%) Use a single timer (either `alarm` or `setitimer`) to implement a set of functions that enables a process to set and clear any number of timers. Please print "*Alarm!*" when the alarm signal trigger. You don't have to implement extra code to parse arguments from the shell. We provide sample function prototypes below. Please use the code (in main function) to test your implementation, and include the results in the report.

```
void setAlarm(int sec){
  /* do something here ... */
}

void clearAlarm(){
  /* do something here ... */
}

int main (void) {
  // ...
  /* You should copy and paste these test cases to your implementation */
  setAlarm(2); //set 2 sec alarm at 0s, will finish at 2s after execution
  sleep(1);
  setAlarm(6); //set 6 sec alarm at 1s, will finish at 7s after execution
  sleep(1);
  setAlarm(3); //set 3 sec alarm at 2s, will finish at 5s after execution
  sleep(4);
  clearAlarm(); //clear all alarms at 6s after execution
  //....
```

```
}
```

Sample outputs

```
mao@MaosMBP:~/Desktop$ ./q3
Alarm!
```