Signals and Signal Handling

Write C program for each of the problem below. After executing each of the C program, you will need to capture the output in text file format.

Use the following file naming convention to complete your submission.

- C program file will be named as 201801nnn_signals_x.c
- Text file with output captured will be names as 201801nnn_signals_x.txt
- Where nnn is 3 digits of your student id and x is assignment problem number 1, 2, 3 etc.

Problem 1

This problem teaches how to use alarm signal (SIGALRM). Write a C program 201801nnn_signals_1.c to do the following:

- main() will wait for user to input the string value using scanf.
- main() will set the signal handler for SIGALRM signal which will be called when program has waited 10 seconds for user to enter the string.
- Implement alarm handler which will display the message to user that “it took too long to enter the string” and then exit the program with -1 exist status.
- If user enters string within 10 seconds, main() should exit with status 0 (success).

Problem 2

We learn about inter-process communication using signals and pipes in this problem. Consider the problem mentioned in the assignment on pipes as basis for pipe communication between parent and the child process. Write a C program 201801nnn_signals_2.c to do the following:

Consider you have two variables for questions and answers as below.

```
char* questions[] = {"Quit", "In which university do you study?", "Which course are you studying?","What is your area of interest?"};

char* answers[] = {"Quit", "DAIICT", "Systems Software", "Kernel Programming"};
```

- Parent process will accept input as question number from user. If the question number is between 0 to 3, the actual text of question will be sent from parent to child process using the following.

  ```
  write(fd1[WRITE], questions[que], strlen(questions[que])+1);
  ```

- Once parent sends the question to child, it will send SIGUSR1 signal to child and then goes to sleep.

- While parent process is accepting number and sending question to child, child process is just sleeping (doing nothing)

- Child will implement SIGUSR1 signal handler in which ,
  - It will read the question sent by parent using the following.

    ```
    bytesRead = read(fd1[READ], message, MSGLEN );
    ```

  - Find the index of the question from questions[] string array. It uses that index to get the correct answer from the answers[] string array

- Reply the answer to parent using the following.
```
write(fd2[WRITE], answers[que], strlen(answers[que])+1);
```

- Parent will implement SIGUSR2 signal handler in which
  - It will read the answer from child using the following.
```
bytesRead= read(fd2[READ], message, MSGLEN);
```

  - Accept the next question number from user and send the question text to child using pipe
```
write(fd1[WRITE], questions[que], strlen(questions[que])+1);
```

  - Then send SIGUSR1 again to child so child can respond with answer

- In SIGUSR1 handler, if child finds that the question number sent by parent process is equal to 0 (i.e. Quit) then it exits after closing all the open file handles. Parent upon reading question number equal to 0 from and after receiving "Quit" as the answer from child closes all open file handles and waits for child to exit using wait() system call to ensure that we don't create a zombie child process. Once parent comes out of wait call it also exits.

Problem 3

This exercise will teach you how to send signal from one process to another process or process group belonging to different process groups then the one sending signal .

- **Write C program 201801nnn_signals_3_sendsignal.c**
  - Accept from used process id to which you need to send signal to
  - Accept from user is the signal needs to send to process or process group (e.g. 1 indicates process and 2 indicates process group)
  - Using kill() system call send signal to either process or process group depending on the 2nd input above. (Hint kill(pid) will send signal to a specific process with pid as process id and kill(-pid) will send signal to all the processes in process group of process pid)

- **Write C program 201801nnn_signals_3_receivesignal_samepgrp.c**
  - Implement signal handler for SIGINT signal which just prints the message using the following.
```
printf("Process id %d received SIGINT signal", getpid());
```
  - Register signal handler for SIGINT using signal() system call .
  - Create child process and print the pid and process group id for both parent and child processes using getpid() and getpgid(0) systems calls. You will notice that getpgid(0) system call will print the same process group number as pid of parent which means parent and child below belong to the same process groups.

- **Write C program 201801nnn_signals_3_receivesignal_diffpgrp.c**
  - Implement signal handler for SIGINT signal which just prints the message using the following.
```
printf("Process id %d received SIGINT signal", getpid());
```
  - Register signal handler for SIGINT using signal() system call .
  - Create child process but this time set new process group of child as child's pid using setpgid(0).

- Now print the pid and process group id for both parent and child processes using getpid() and getpgid(0) systems calls. You will notice that getpgid(0) prints different process group numbers.

Execution 1:

- First execute 201801nnn_singals_3_receivesignal_samegrp executable which is waiting for signal to be received .
    - Now on different terminal window run executable for 201801nnn_signals_3_sendsignal .
        - First select process id of parent process running 201801nnn_signals_3_receivesignal_samegrp and choose option 2 for sending signal to process group.
        - You should get the message from signal handler for both parent and the child process and they should both terminate since they belong to the same process group. Check using ps command.

Execution 2:

- First execute 201801nnn_signals_3_receivesignal_samegrp executable which is waiting for signal to be received .
    - Now on different terminal window run executable for 201801nnn_signals_3_sendsignal .
        - First select process id of parent (or child) process running 201801nnn_signals_3_receivesignal_samegrp and choose option 1 for sending signal to individual process.
        - You should get the message from signal handler for parent (or child) process depending which process id you entered and only that process will terminate. Note that the other process child (or parent) is still running. Check using ps command and kill it.

Execution 3:

- First execute 201801nnn_signals_3_receivesignal_diffgrp executable which is waiting for signal to be received
    - Now on different terminal window run executable for 201801nnn_signals_3_sendsignal .
        - First select process id of parent process running 201801nnn_signals_3_receivesignal_diffgrp and choose option 2 for sending signal to process group.
        - Even though you selected process id of parent and sending signal to process group, child will not see that signal and will not terminate because it is in separate process group. Hence, you should get the message from signal handler for parent only and parent process will terminate. Note that the child process is still running. Check using ps command and kill it.