CN Lab 12-1-2022: Assignments: Signals, Semaphores, Shared Memory

1. Circular Signalling – Two way

Processes P1 , P2 , P3 , P4 will form a circle by knowing their pids of previous and next process of it, using a Message Q as explained in below steps. (steps must be followed as specified only.)

- P2 gets executed first, and it sends a message containing its pid as : p2pid with type value 2.
- P1 gets executed , it receives message and finds p2pid from message and sends SIGUSR1 to P2. P1 notes that its next (right side) process is P2 and its pid is p2pid.
- P2 catches the signal and **it finds out, who has sent that signal to it.** (that is P2 should find the pid of P1 after receiving signal from P1).(Hint : **sigaction()** .. can be used for this). Now P2 also notes that its previous(left) process is P1(i.e. p1pid).
- P3 gets executed, and it sends a message containing its pid as : p3pid with type value 3.
- P2 receives the message from msq and it notes that its next (right side) process is P3 and its pid is p3pid.
- **P2 sends a message containing P1's pid as : type = 1, content = p1pid.**
- P2 sends a signal SIGUSR1 to P3.
- P3 catches the signal and **it finds out, who has sent that signal to it.** (that is P3 should find the pid of P2 after receiving signal from P2).(Hint : **sigaction()** .. can be used for this)). Now P3 also notes that its previous(left) process is P2 (i.e. p2pid).
- P4 gets executed, and it sends a message containing its pid as : p4pid with type value 4.
  - (Note that in the message queue msq, there are two messages now, one is type=1, p1pid, another is type=4, p4pid)
- P3 receives the message from msq of type=4 , and it notes that its next (right side) process is P4 and its pid is p4pid.
- P3 sends a signal SIGUSR1 to P4.
- P4 catches the signal and **it finds out, who has sent that signal to it.** (that is P4 should find the pid of P3 after receiving signal from P3).(Hint : **sigaction()** .. can be used for this)). Now P4 also notes that its previous(left) process is P3 (i.e. p3pid).
- P4 receives a message from msq (there is one message left in msq) . P4 notes out from that message that its next (right side) process is P1, and sends a SIGUSR1 to P1 (ie. p1pid).
- P1 catches the signal and **it finds out, who has sent that signal to it.** (that is P1 should find the pid of P4 after receiving signal from P4).(Hint : **sigaction()** .. can be used for this)). Now P1 also notes that its previous(left) process is P4 (i.e. p4pid).
- Now the circular signalling should as : P1 signals to P2, after receiving that signal, P2 signals to P3, then P3 to P4, and P4 to P1.
- This circular signalling of P1 → P2 → P3 → P4 → P1 should happen **3 times**.
- Now Reverse circular signalling should be as : P1 signals P4, P4 to P3, P3 to P2, P2 to P1.
- This reverse circular signalling of P1 → P4 → P3 → P2 → P1 should happen **3 times**.
- All the processes are to be stopped.
  (You may use SIGUSR1 for circular signalling and SIGUSR2 for reverse circular signalling)

2. Group Signalling – one to All

Processes P1 , P2 , P3 , P4 will form a circle to know their pids of previous and next process of it, using a Message Q as explained in below steps. (steps must be followed as specified only.)

- P2 gets executed first, and it sends a message containing its pid as : p2pid with type value 2.
- P1 gets executed, it receives message(with type=2) and finds p2pid from message and sends SIGUSR1 to P2.
- P2 catches the signal and **it finds out, who has sent that signal to it.** (that is P2 should find the pid of P1 after receiving signal from P1).(Hint : **sigaction()** .. can be used for this).
- P3 gets executed, and it sends a message containing its pid as : p3pid with type value 3.
- P2 receives the message from msq (with type=3) and it notes that its next (right side) process is P3 and its pid is p3pid.
- **P2 sends two messages: first one containing P1's pid as : type = 1, content = p1pid, and second message containing P2's pid as : type=18, content=p2pid. (this type=18 messages are received by P1 , later and it forms a group id)**
- P2 sends a signal SIGUSR1 to P3.
- P3 catches the signal. **P3 sends a message as : type=18, content=p3pid. (this type=18 messages are received by P1 , later and it forms a group id)**

- P4 gets executed, and it sends a message containing its pid as : p4pid with type value 4.
    - (Note that in the message queue msq, there are four messages at the moment: type=1, p1pid, , type=18, p2pid, type=18 , p3pid, type=4, p4pid
- P3 receives the message(with type=4) from msq of type=4
- P3 sends a signal SIGUSR1 to P4.
- P4 catches the signal. . P4 sends a message as : type=18, content=p4pid. (this type=18 messages are received by P1 , later and it forms a group id)
- P4 receives a message from msq (with type=1) and sends a SIGUSR1 to P1 (ie. p1pid).
- P1 catches the signal. Now P1 forms grouppid of all pids. P1 receives all the pids from the msg (with type= 18) and forms into a group.
- P1 sends three messages to msq, with type=2,content grouppid, type=3 ,content grouppid, type=4, grouppid.
- All the Pi receives message of type=i and notes the grouppid.
- P1 sends a SIGUSR2 signal to the group (i.e. to P2,P3,P4) using killg().
- When a Pi process receives a SIGUSR2, then it also has to send SIGUSR2 to all other processes using grouppid.
  You can print any output in each of the Pi window, but the last lines of the output should be as follows:
  Each Pi should print in their window that from which Pj it has got the signal. That means there should be three SIGUSR2 signal catching outputs in each of Pi.

3. P1P2P3P4 – Synchronization-Semaphores

P1 creates semaphores S12=0 , S41=0.
P2 creates semaphores S12=0 , S23=0.
P3 creates semaphores S23=0 , S34=0.
P4 creates semaphores S34=0 , S41=0.

P1: P1 has to display " I am P1. Enter any character to sem-signal(S12)".
P1 reads a char , and displays " I am signalling semaphore signal of S12 " ;
P1 has to sem-signal(S12); It has to display " I am waiting for semaphore S41";
P1 has to sem-wait(S41);
Next P1 has to display " I got semaphore signalling from P4 ";
P1 has to continue.

P2: P2 has to display " I am P2. I am waiting for Semaphore S12 "
P2 has to sem-wait(S12);
P2 has to display " I got semaphore S12 signalling from P1 ";
Then it has to display " Enter any character to sem-signal(S23)".
P2 reads a char , and displays " I am signalling semaphore signal of S23 " ;
P2 has to sem-signal(S23);
P2 has to continue.

P3 : Same as above : change S23 , S34.
P4 : Same as above : change S34 , S41.

4. P1xP2y – Semaphores-Shared Memory

P1 , P2 are processes , x, y are shared memory, use semaphores S1=0, S2=0.
P1 initialises x=1 and y =1;
P1 has to continue as : { display " I am reading shm y " ; read(y) ; make x = y+1 ;
display " Enter any char to signal S1" ; cin>> ch; signal(S1);
display " I am waiting for S2" ; wait(s2); }
P2 has to continue as : { display " I am waiting for S1" ; wait(S1); read(x) ;
make y = x+1 ; display " Enter any char to signal S2" ; cin>> ch; signal(S2); }

Submission link :
https://forms.gle/kgBdu9EJXQEkU9ke6