First name, last name: Ehsan ALIDELBI ELWAN.

Here we can find the 3 algorithms used to handle the different scenarios of prod/cons problem

1. /\*        Version 1        \*/
2. // init( mutex: m=0 ; sem\_t: empty=size of buffer ,sem\_t: full=0  );
3. **void** producer (....) {
4. **for** (....) {
5. sem\_wait(empty);   p(m);
6. insert(buff, posW, msg);
7. v(m);
8. posW = ( posW + 1 ) % nbOfPositions
9. sem\_post(full);
10. }
11. }
12. **void** consumer (....) {
13. **for** (....) {
14. sem\_wait(full);   p(mutex);
15. extractMsg(....);
16. posR = ( posR + 1 ) % nbOfPositions;
17. v(mutex) ;  sem\_post(empty);
18. }
19. }
21. /\*        Version 2        \*/
22. // We can alter only the producers, if we manage to alternate the producers type correctly then they should be consumed in the right order.
23. // init( mutex: black\_Mutex=0 ; mutex: white\_Mutex=1 ; sem\_t: empty=size of buffer ,sem\_t: full=0  );
24. **void** producer (..., **int** type) {
25. **for** (....) {
26. **if** ( type == 0 ) { p( black\_Mutex ); } **else** { p( white\_Mutex ); }
27. sem\_wait(empty);   p(mutex);
28. insert(buff, posW, msg);
29. v(mutex);
30. posW = ( posW + 1 ) % nbOfPositions
31. sem\_post(full);
32. **if** ( type == 0 ) { v( white\_Mutex ); } **else** { v( black\_Mutex ); }
33. }
34. } // No changes in consumer algorithm.
36. /\*        Version 3        \*/
37. // init( mutex: m=0 ;  sem\_t: empty=size of buffer ,sem\_t: full=0  , sem\_t colors[nbColors] = 0 for each sem in colors[i]);
38. **void** producer (..., **int** type) {
39. **for** (....) {
40. sem\_wait(empty);
41. insert(buff, posW, msg);
42. posW = ( posW + 1 ) % nbOfPositions
43. sem\_post(full);
44. **if** ( posR + ( posW - 1 ) == 0 ) //then it was the first msg
45. { sem\_post(colors[type]); }
46. }
47. **void** consumer(..., **int** type) {
48. **for** (....) {
49. sem\_wait(colors[type]);
50. **if** (buffer[posR].type != type) { sem\_wait(colors[type]); }
51. sem\_wait(full);
52. p(m);
53. getMsg(..., posR);
54. sem\_post(empty);
55. **if** (buffer[posR].type != -1 ) //to make sure we have msg in the next position
56. { sem\_post(colors[buffer[posR].type]); }
57. v(m);
58. }
59. }

Code for version 1:

1. **void** \*producer(**void** \*arg) {
2. **int** i;
3. TypeMessage theMessage;
4. Parameters \*param = (Parameters \*)arg;
5. sleep(1); // to make sure that all consumers and producers have been
7. **for** (i = 0; i < NB\_TIMES\_PROD; i++) {
8. sem\_wait(empty);
9. sprintf(theMessage.info, "%s n°%d", "Hello", i);
10. theMessage.typeOfMessage = param->typeOfMessage;
11. theMessage.producerNumber = param->threadNumber;
13. pthread\_mutex\_lock(&mutex\_display);
14. printf("\tProducer %d : Message = (%d, %s, %d)\n", param->threadNumber,
15. theMessage.typeOfMessage, theMessage.info,
16. theMessage.producerNumber);
18. makePut(theMessage);
19. #ifdef TRACE
20. showBuffer();
21. #endif
22. pthread\_mutex\_unlock(&mutex\_display);
23. sem\_post(full);
24. }
25. **return** NULL;
26. }
28. /\*--------------------------------------------------\*/
29. **void** \*consumer(**void** \*arg) {
30. **int** i;
31. TypeMessage theMessage;
32. Parameters \*param = (Parameters \*)arg;
33. // sleep(1); // to make sure that all consumers and producers have been
35. **for** (i = 0; i < NB\_TIMES\_CONS; i++) {
36. sem\_wait(full);
37. pthread\_mutex\_lock(&mutex\_display);
38. makeGet(&theMessage);
39. printf("\t\tConso %d : Message = (%d, %s, %d)\n", param->threadNumber,
40. theMessage.typeOfMessage, theMessage.info,
41. theMessage.producerNumber);
42. #ifdef TRACE
43. showBuffer();
44. #endif
46. pthread\_mutex\_unlock(&mutex\_display);
47. sem\_post(empty);
48. }
49. **return** NULL;
50. }
52. **int** main(){
53. //after  initializeSharedVariables();
54. **if** ((empty = sem\_open("/empty", O\_CREAT, 0644, nbPositions)) == SEM\_FAILED || (full = sem\_open("/full", O\_CREAT, 0644, 0)) == SEM\_FAILED) {
55. perror("sem\_open");
56. exit(EXIT\_FAILURE);
57. }
58. }

Code for version 2:

1. **void** \*producer(**void** \*arg) {
2. TypeMessage theMessage;
3. Parameters \*param = (Parameters \*)arg;
4. sleep(1); // to make sure that all consumers and producers have been
6. **for** (**int** i = 0; i < NB\_TIMES\_PROD; i++) {
8. **if** (param->typeOfMessage == 0) {
9. pthread\_mutex\_lock(&mutex\_b);
10. } **else** {
11. pthread\_mutex\_lock(&mutex\_w);
12. }
13. sem\_wait(empty);
14. sprintf(theMessage.info, "%s n°%d", "Hello", i);
15. theMessage.typeOfMessage = param->typeOfMessage;
16. theMessage.producerNumber = param->threadNumber;
18. pthread\_mutex\_lock(&mutex\_display);
19. printf("\tProducer %d : Message = (%d, %s, %d)\n", param->threadNumber,
20. theMessage.typeOfMessage, theMessage.info,
21. theMessage.producerNumber);
23. makePut(theMessage);
24. #ifdef TRACE
25. showBuffer();
26. #endif
27. pthread\_mutex\_unlock(&mutex\_display);
28. sem\_post(full);
29. **if** (param->typeOfMessage == 0) {
30. pthread\_mutex\_unlock(&mutex\_w);
31. } **else** {
32. pthread\_mutex\_unlock(&mutex\_b);
33. }
34. }
35. **return** NULL;
36. }
38. /\*--------------------------------------------------\*/
40. /\* ----- No changes for consumer code ------\*/
41. **int** main() {
42. pthread\_mutex\_t mutex\_w = PTHREAD\_MUTEX\_INITIALIZER;
43. pthread\_mutex\_t mutex\_b = PTHREAD\_MUTEX\_INITIALIZER;
44. pthread\_mutex\_lock(&mutex\_w); // we assume that the first producer will always be of type black so we lock the white mutex
45. **if** ((empty = sem\_open("/empty", O\_CREAT, 0644, nbPositions)) == SEM\_FAILED || (full = sem\_open("/full", O\_CREAT, 0644, 0)) == SEM\_FAILED) {
46. perror("sem\_open");
47. exit(EXIT\_FAILURE);
48. }
49. }

Code for version 3: