***Process Synchronization Using Semaphore***

***Algorithm Using BUFFER\_SIZE – 1 Slots***

///Producer consumer problem with buffer\_size-1 slots filled

///Adnan Ismail Shah Muzavor

#include<stdio.h>

#include<stdlib.h>

///Global variables

int in=0,out=0,i;

void wait(int \*s)

{

///s -> Semaphore (Here buffer)

///cp -> Current process who is willing to access buffer

while(\*s<0)

{

printf("\nWaiting");

}

(\*s)--;

}

void signal(int \*s)

{

(\*s)++;

}

void Producer(int \*mutex,int \*empty,int \*full,int n,int buffer[])

{

/// => If so check if slots are empty to produce

wait(empty);

/// => Check if it can access the buffer

wait(mutex);

/// => Producer can produce only if buffer was empty

if((in+1)%n!=out)

{

/// => If reach till here produce in current slot

buffer[in]=1;

printf("\nProduced at slot number: %d => ",in);

in=(in+1)%n;

/// => Display buffer

for(i=0; i<n; i++) printf("%d ",buffer[i]);

printf("\n");

}

else

{

printf("\nBuffer is full,no item can be produced.");

}

/// => Free the buffer so that consumer can access it

signal(mutex);

/// => Increase the full by 1 as one slot has been occupied

signal(full);

}

void Consumer(int \*mutex,int \*empty,int \*full,int n,int buffer[])

{

/// => If so check if slots not empty, i.e full>0 indicating there is something to consume

wait(full);

/// => Check if it can access the buffer

wait(mutex);

/// => Consume only if had something to consume

if(in!=out)

{

/// => If reach till here consume from current slot

buffer[out]=0;

printf("\nConsumed at slot number: %d => ",out);

out=(out+1)%n;

for(i=0; i<n; i++) printf("%d ",buffer[i]);

printf("\n");

}

else

{

printf("\nBuffer is empty,no item to consume.");

}

/// => Free the buffer so that consumer can access it

signal(mutex);

/// => Since we are emptying one block, increment empty to indicate

signal(empty);

}

int main()

{

int i,op,n;

printf("\nEnter size of buffer: ");

scanf("%d",&n);

int buffer[n]; /// -> Buffer of size 5.

for(i=0; i<n; i++) buffer[i]=0;

int mutex=1; /// -> Indicates buffer is available.

int full=0; /// -> Initially zero slots are full in buffer.

int empty=n; /// -> Initially all slots(5) are empty.

while(1)

{

printf("\n-------------------------------");

int exit=0;

printf("\nEnter 1 -> to produce\nEnter 2 -> to consume\nEnter 3 to exit\n");

scanf("%d",&op);

switch(op)

{

case 1:

Producer(&mutex,&empty,&full,n,buffer);

break;

case 2:

Consumer(&mutex,&empty,&full,n,buffer);

break;

case 3:

exit=1;

break;

default:

printf("\nInvalid input");

}

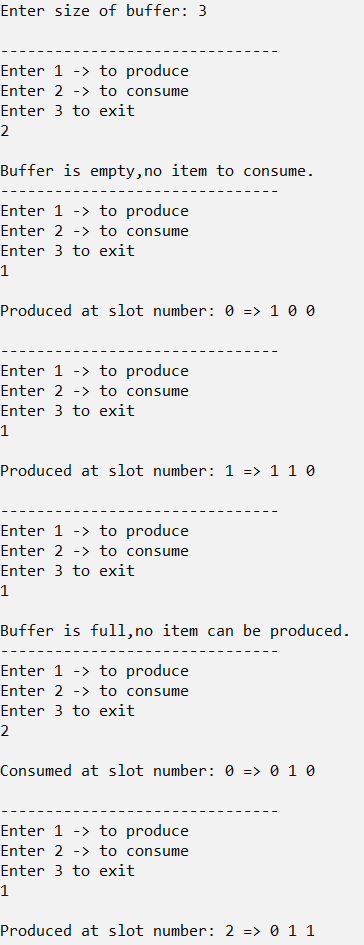
if(exit==1) break;

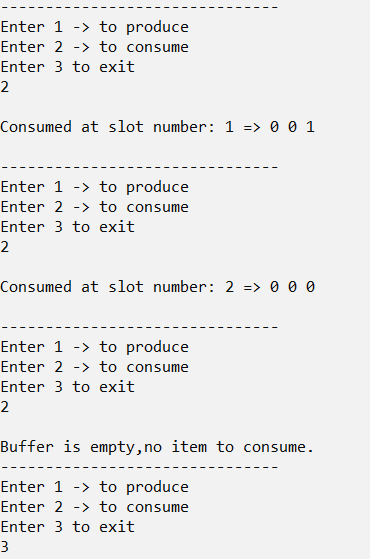
}

return 1;

}

***Output:***

******

******

***Algorithm Using BUFFER\_SIZE Slots***

///Producer consumer problem with buffer\_size slots filled

///Adnan Ismail Shah Muzavor

#include<stdio.h>

#include<stdlib.h>

///Global variables

int in=0,out=0,i,counter=0;

void wait(int \*s)

{

///s -> Semaphore (Here buffer)

///cp -> Current process who is willing to access buffer

while(\*s<0)

{

printf("\nWaiting");

}

(\*s)--;

}

void signal(int \*s)

{

(\*s)++;

}

void Producer(int \*mutex,int \*empty,int \*full,int n,int buffer[])

{

/// => If so check if slots are empty to produce

wait(&empty);

/// => Check if it can access the buffer

wait(&mutex);

/// => Producer can produce only if buffer was empty

if(counter!=n)

{

/// => If reach till here produce in current slot

buffer[in]=1;

printf("\nProduced at slot number: %d => ",in);

in=(in+1)%n;

counter++;

/// => Display buffer

for(i=0; i<n; i++) printf("%d ",buffer[i]);

printf("\n");

}

else

{

printf("\nBuffer is full,no item can be produced.");

}

/// => Free the buffer so that consumer can access it

signal(&mutex);

/// => Increase the full by 1 as one slot has been occupied

signal(&full);

}

void Consumer(int \*mutex,int \*empty,int \*full,int n,int buffer[])

{

/// => If so check if slots not empty, i.e full>0 indicating there is something to consume

wait(&full);

/// => Check if it can access the buffer

wait(&mutex);

/// => Consume only if had something to consume

if(counter!=0)

{

/// => If reach till here consume from current slot

buffer[out]=0;

printf("\nConsumed at slot number: %d => ",out);

out=(out+1)%n;

counter--;

for(i=0; i<n; i++) printf("%d ",buffer[i]);

printf("\n");

}

else

{

printf("\nBuffer is empty,no item to consume.");

}

/// => Free the buffer so that consumer can access it

signal(&mutex);

/// => Since we are emptying one block, increment empty to indicate

signal(&empty);

}

int main()

{

int i,op,n;

printf("\nEnter size of buffer: ");

scanf("%d",&n);

int buffer[n]; /// -> Buffer of size 5.

for(i=0; i<n; i++) buffer[i]=0;

int mutex=1; /// -> Indicates buffer is available.

int full=0; /// -> Initially zero slots are full in buffer.

int empty=n; /// -> Initially all slots(5) are empty.

while(1)

{

printf("\n-------------------------------");

int exit=0;

printf("\nEnter 1 -> to produce\nEnter 2 -> to consume\nEnter 3 -> to exit\n");

scanf("%d",&op);

switch(op)

{

case 1:

Producer(&mutex,&empty,&full,n,buffer);

break;

case 2:

Consumer(&mutex,&empty,&full,n,buffer);

break;

case 3:

exit=1;

break;

default:

printf("\nInvalid input");

}

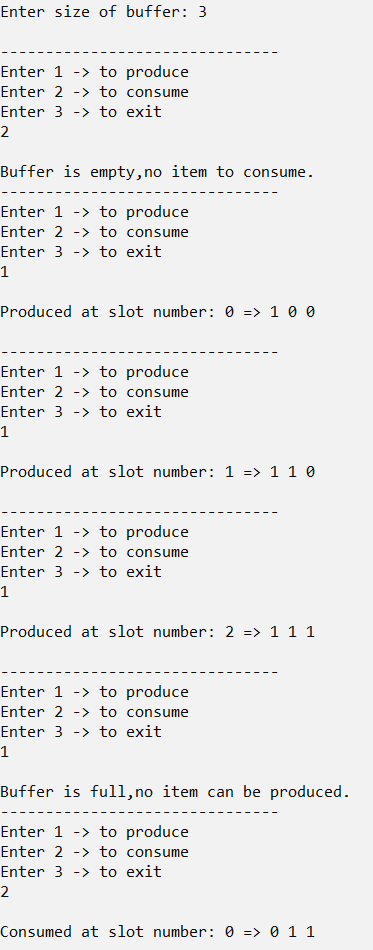
if(exit==1) break;

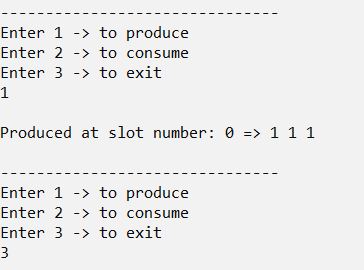
}

return 1;

}

***Output:***

******

******