FUTURE INSTITUTE OF ENGINEERING AND MANAGEMENT

CC – 294

UNDER

MAKAUT, WB

<PRODUCER AND CONSUMER PROBLEM>

CONTINUOUS ASSESSMENT#2

<OPERATING SYSTEM>

<BCAC302>

PRESENTED BY

<KRISHNAYAN BHADRA>

<29401222026>

<BCA>

<3rd SEM>

Academic Session: 2023-24

Report on the producer consumer problem

Table of contents

Introduction

The Producer-Consumer problem is a classical multi-process synchronization problem, that is we are trying to achieve synchronization between more than one process.

Description

There is one producer in the producer consumer problem producer is producing some items whereas there is one Consumer that is consuming the items produced by the producer. The same memory buffer is shared by both producers which is of fixed size.

The task of the producer is to produce the item put it into the memory buffer, and again start producing items. Whereas the task of the consumer is to consume the item from the memory buffer

Types

i) The producer should produce data only when the buffer is not full. In case it is found that the buffer is full, the producer is not allowed to store any data into the memory buffer.

(ii) Data can only be consumed by the consumer if and only if the memory buffer is not empty. In case it is found that the buffer is empty the consumer is not allowed to use any data from the memory buffer.

(iii) Accessing memory buffer should not be allowed to producer and consumer at the same time.

Producer Code

```
int count = 0;
void producer (void)
{ int item P;
  while (1)
{ produce_item(item P)
  while (count == n);
  buffer [in] = item P;
  in = (in + 1) mod n
  count = count + 1;
}
}
```

```
Load Rp, m[count]
increment Rp
Store m[count], Rp
```

```
int count;
void consumer (void)
{ int item C;
  while (1)
{ while (count == 0);
item C = buffer [out];
out = (out+1) mod n;
count = count - 1;
}
}
```

1) "in" used in a producer code represent the next empty buffer
2) "out" used in consumer code represent first filled buffer.
3) count is further divided into 3 lines code represented in the block in both the producer and consumer code represented in the block in both the producer and consumer code.

Explanation of producer code

- Rp is a register which keeps the value of m[count]
- Rp is incremented value of Rp is stored back to ~~buffer~~ m[count]
- Rp is incremented (As element has been added to buffer)

Explanation of consumer code

-- Re is a register which keeps the value of m[count]
-- Re is decremented (As element has been removed out of buffer)
-- the decremented value of re is stored back to m[count]

Buffer