OS

10 July 2020 11:44

os

OPERATING SYSTEM
Synchronization

What Is In This Chapter?

- This is about getting processes to coordinate with each other.
- How do processes work with resources that must be shared between them?
- How do we go about acquiring locks to protect regions of memory?
- How is synchronization really used?

OPERATING SYSTEM
Synchronization

Topics Covered

- Background
- The Critical-Section Problem
- Peterson's Solution
- Synchronization Hardware
- Semaphores
- Classic Problems of Synchronization
- Synchronization Examples
- Atomic Transactions

PROCESS SYNCHRONIZATION

The Producer Consumer Problem

A **producer** process "produces" information "consumed" by a **consumer** process.
Here are the variables needed to define the problem:

```
#define BUFFER_SIZE 10
typedef struct {
    DATA            data;
} item;
item   buffer[BUFFER_SIZE];
int    in = 0;              // Location of next input to buffer
int    out = 0;             // Location of next removal from buffer
int    counter = 0;         // Number of buffers currently full
```

Consider the code segments on the next page:

- Does it work?
- Are all buffers utilized?

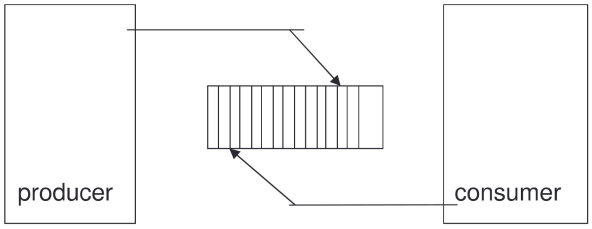
PROCESS SYNCHRONIZATION

A **producer** process "produces" information "consumed" by a **consumer** process.

PRODUCER

```
item   nextProduced;

while (TRUE) {
    while (counter == BUFFER_SIZE);
    buffer[in] = nextProduced;
    in = (in + 1) % BUFFER_SIZE;
    counter++;
}
```



The Producer Consumer Problem

```
#define BUFFER_SIZE 10
typedef struct {
    DATA        data;
} item;
item   buffer[BUFFER_SIZE];
int     in = 0;
int     out = 0;
int     counter = 0;
```

CONSUMER

```
item  nextConsumed;

while (TRUE) {
    while (counter == 0);
    nextConsumed = buffer[out];
    out = (out + 1) %
    BUFFER_SIZE;
    counter--;
}
```