A

Report on

Producer – Consumer Problem

By

Aashish Bhandari

12000405

K20VL

**Table of Contents**

**Chapter Particulars Page No.**

**Chapter 1. Introduction …………………………………………………………………………………………….. 3**

**Chapter 2. Technologies used …………………………………………………………………………………….4**

**Chapter 3. Website snapshots ……………………………………………………………………………………5-7**

**Chapter 4. Codes ……………………..…………………………………………………………………………………8-18**

**References………………………………………………………………………………………………………………………….19**

**Introduction**

This project gives the detailed working of the producer-consumer supply chain a about how problems occurs and what are those problems, what are the processes to solve them.

Producer Consumer Problem is a classical concurrency problem. In fact, it is one of the concurrency design patterns. It is also known as the bounded-buffer problem. The problem describes two processes, the producer, and the consumer, who share a common, fixed-size buffer used as a queue. The producer's job is to generate data and put it into the buffer. At the same time, the consumer consumes the data (i.e., by removing it from the buffer), one piece at a time. The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer. The solution for the producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer.

**Technologies Used**

1. **Html :**  All the skeleton of the webpage is written and maintained in html. All the contents of the page is written in html.
2. **CSS:** All the designs, colors and most of the animations are written in css classes. These css classes are attached with the html page and html page get its design.
3. **Javascript:** In this project javascript have been heavily used in the backend for making the background animation in the page layout. And these javascript has been served to the website using Content Delivery Network (CDN).
4. **Git:** This project is maintained and repaired with the help of the git version controller.
5. **VANTA.**js: This library has been used in the project for making the animation in the background of website.

**Websites-Snapshots**

This website is live at : “https://aashish365.github.io/producer-consumer/”

Text

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated with medium confidence

A screenshot of a video game

Description automatically generated

**Codes:**

1. **Html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link href="https://unpkg.com/aos@2.3.1/dist/aos.css" rel="stylesheet">

    <script src="https://unpkg.com/aos@2.3.1/dist/aos.js"></script>

    <link href="https://fonts.googleapis.com/css2?family=Teko:wght@400;500&display=swap" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r134/three.min.js"></script>

    <link rel="stylesheet" href="style.css" type="text/css">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.0/css/all.min.css"

        integrity="sha512-xh6O/CkQoPOWDdYTDqeRdPCVd1SpvCA9XXcUnZS2FmJNp1coAFzvtCN9BmamE+4aHK8yyUHUSCcJHgXloTyT2A=="

        crossorigin="anonymous" referrerpolicy="no-referrer" />

</head>

<body>

    <div class="mainContainer glow">

        <div class="section about" id="about">

            <div class="nav">

                <div class="container" id="container">

                    <div class="logo">

                        <div class="main\_topic">Producer-Consumer Problem</div>

                        <!-- <div class="mini\_topic"></div> -->

                    </div>

                    <ul>

                        <li><a href="#about">About</a></li>

                        <li><a href="#problem">Problem</a></li>

                        <li><a href="#solution">Solution</a></li>

                    </ul>

                </div>

            </div>

            <div class="content">

                <div class="intro">

                    <div data-aos-duration="1000" data-aos="fade-down-right" class="intro\_text">

                        <h1 class="intro\_text\_title">producer-consumer problem </h1>

                        <p class="intro\_text\_para">

                            Producer Consumer Problem is a classical concurrency problem. In fact it is one of the

                            concurrency design pattern. It is

                            also known as the bounded-buffer problem. The problem describes two processes, the producer

                            and

                            the consumer, who share a common, fixed-size buffer used as a

                            queue. The producer's job is to generate data and put it into the buffer. At the same time,

                            the

                            consumer consumes the data

                            (i.e., by removing it from the buffer), one piece at a time. The problem is to make sure

                            that

                            the producer won't try to add data into the buffer if it's full and that the consumer

                            won't try to remove data from an empty buffer. The solution for the producer is to either go

                            to

                            sleep or discard data if the buffer is full. The next time the consumer

                            removes an item from the buffer, it notifies the producer, who starts to fill the buffer

                            again.In the same way, the consumer can go to sleep if it finds the buffer to be empty. The

                            next

                            time the producer puts data

                            into the buffer, it wakes up the sleeping consumer.

                        </p>

                    </div>

                    <div data-aos-duration="1000" data-aos="fade-up-left" class="intro\_photo">

                    </div>

                </div>

            </div>

            <script src="https://cdn.jsdelivr.net/npm/vanta/dist/vanta.net.min.js"></script>

            <script>

                VANTA.NET('#about')

            </script>

        </div>

        <div class="section problem" id="problem">

            <div class="problem\_content">

                <div class="problem\_intro">

                    <div data-aos-duration="1000" data-aos="fade-up-right" class="problem\_intro\_photo"></div>

                    <div data-aos="fade-down-left" class="intro\_text">

                        <h1 class="intro\_text\_title">Problem</h1>

                        <p class="intro\_text\_para">

                            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.

                            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.

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

                        </p>

                    </div>

                </div>

            </div>

        </div>

        <div class="section solution" id="solution">

            <div class="problem\_content">

                <div class="problem\_intro">

                    <div data-aos="fade-down-right" class="intro\_text">

                        <h1 class="intro\_text\_title">Solution</h1>

                        <p class="intro\_text\_para">

                            The above problems of Producer and Consumer which occurred due to context switch and

                            producing inconsistent result can

                            be solved with the help of semaphores.

                            To solve the problem occurred above of race condition, we are going to use Binary Semaphore

                            and Counting Semaphore

                            <br>

                            <strong>Semaphore</strong> In Binary Semaphore, only two processes can compete to enter into

                            its

                            CRITICAL SECTION at any point in

                            time, apart from this the condition of mutual exclusion is also preserved.

                            <br>

                            <strong>Semaphore</strong> In counting semaphore, more than two processes can compete to

                            enter into

                            its CRITICAL SECTION at any

                            point of time apart from this the condition of mutual exclusion is also preserved.

                            <br>

                            <strong>Semaphore</strong>

                            A semaphore is an integer variable in S, that apart from initialization is

                            accessed by only two standard

                            atomic operations - wait and signal,

                        </p>

                    </div>

                    <div data-aos-duration="1000" data-aos="fade-up-left" class="solution\_photo"></div>

                </div>

            </div>

        </div>

    </div>

    <footer id="footer">

        <div data-aos="fade-up" class="foot\_container">

            <div class="col">

                <div class="copyright">&copy; <span id="date"></span> Aashish Bhandari.</div>

                <script>

                    document.getElementById("date").innerText = new Date().getFullYear()

                </script>

            </div>

            <div class="col">

                <h3>About</h3>

                <div class="copyright">

                    My name is Aashish Bhandari. <br />

                    I am a 3rd year BTech Student.

                </div>

            </div>

            <div class="col">

                <h3>Follow Me</h3>

                <div>

                    <a href="https://www.facebook.com/Aashish365"><i class="fa-brands fa-facebook"></i></a>

                    <a href="https://www.instagram.com/\_aashishbhandari\_/"><i class="fa-brands fa-instagram"></i></a>

                    <a href="https://www.linkedin.com/in/aashish-b-ab025b1a6/"><i class="fa-brands fa-linkedin"></i></a>

                    <a href="https://github.com/Aashish365"><i class="fa-brands fa-github"></i></a>

                </div>

            </div>

        </div>

        <script src="https://cdn.jsdelivr.net/npm/vanta/dist/vanta.dots.min.js"></script>

        <script>

            VANTA.DOTS('#footer');

            VANTA.DOTS("#nav");

        </script>

    </footer>

    </div>

    <script src="script.js"></script>

</body>

</html>

1. **CSS**

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    font-family: sans-serif;

}

html{

    scroll-behavior: smooth;

}

/\* .mainContainer{

    scroll-snap-type: y mandatory;

    overflow-y: scroll;

    height: 100vh;

} \*/

/\* .section{

    height: 100vh;

    padding-left: 100px;

    padding-right: 100px;

} \*/

#about{

    height: 100vh;

    overflow: hidden;

}

#problem{

    background-color: #4e2563;

    height: 100vh;

    overflow: hidden;

}

#solution{

overflow: hidden;

background-color: #362a51;

height: 100vh;

padding:190px;

}

.nav{

    position: fixed;

    width: 100%;

    z-index: 9999;

}

.container{

    padding: 5px;

    display: flex;

    align-items: center;

    justify-content: space-between;

    background-color: #f5f7f9;

}

.logo{

    font-weight: bold;

    text-transform: uppercase;

    font-size: 20px;

}

.logo:hover{

    cursor: pointer;

}

.main\_topic{

    color: rgb(57, 57, 57);

}

.mini\_topic{

    color:rgb(148, 67, 0);

}

ul{

    padding: 20px 20px;

    display: flex;

    align-items: center;

    list-style:none;

}

li{

    padding: 0px  20px;

    transition: all .2s;

}

li:hover, li:hover > a{

    cursor: pointer;

    transform: scale(1.15);

}

li:active{

    cursor: pointer;

    transform: scale(1);

}

a{

    text-decoration: none;

   color:rgb(57, 57, 57);

    font-size: 20px;

    animation: all .3s;

}

li,a:hover{

    color: rgb(22, 22, 22);

}

.content{

    padding-top:50px;

    display: flex;

    flex-direction: column;

    align-items: center;

}

.intro{

    display: flex;

    justify-content: center;

    padding-top: 50px;

}

.intro\_text{

    width: 50%;

}

.intro\_text\_title{

    color:rgb(215, 215, 220);

    padding: 30px 0px;

    font-size: 40px;

    text-transform: uppercase;

    text-shadow: 5px 5px rgba(219, 217, 217, 0.297);

}

.intro\_text\_para{

    font-family: sans-serif;

    color:rgb(215, 215, 220);

    font-size: 20px;

    line-height: 35px;

    text-align: justify;

}

.intro\_photo{

    margin: 10px 20px;

    width: 30%;

    background-image: url("./prod-cons.gif");

    background-position: center;

    background-size: contain;

    background-repeat: no-repeat;

}

#problem{

    overflow: hidden;

    padding-top: 100px;

   padding: 170px;

}

.problem\_intro{

    display: flex;

    justify-content: space-between;

}

.problem\_intro\_photo{

    margin: 10px 20px;

    width: 30%;

    background-image: url("./intro\_photo.jpg");

    background-position: center;

    background-size: contain;

    background-repeat: no-repeat;

}

.solution\_photo{

    /\* margin: 10px 20px; \*/

    width: 30%;

    background-image: url("./problems.png");

    background-position: center;

    background-size: contain;

    background-repeat: no-repeat;

}

#solution >.intro>.intro\_text>p{

    color: white;

}

/\* ///////////////////// \*/

/\* Footer \*/

footer{

overflow: hidden;

  color:#e8e3f7;

}

footer h3{

    padding:10px;

    font-weight: 600;

    font-size: 40px;

}

.foot\_container{

    padding: 50px;

    display: flex;

    align-items: center;

    justify-content: space-around;

}

.col{

    height: 30vh;

    display: flex;

    flex-direction: column;

    align-items: center;

    padding: 50px;

}

.col:not(:last-child){

    border-right: .2px solid rgb(165, 165, 192);

}

 i{

    color:white;

    padding: 10px;

 }

1. **JAVASCRIPT:**

AOS.init();

**References:**

1. <https://www.javatpoint.com/producer-consumer-problem-in-os>
2. <https://afteracademy.com/blog/the-producer-consumer-problem-in-operating-system>