

# Experiment Report Four

Course Name: Operating System

Experiment Name: Producer and Consumer Problem

Student Name: 李添豪

Student ID: 5140309349

Class Number: F1403015

## Experiment Environment

Operating System: Ubuntu LTS 16.04 (based on VMware)

Kernel Version: 4.4.6

## Experiment Target

Using semaphore to solve producer and consumer problem.

## Experiment Design

1. Initial buffer as 0
2. Creating two threading as producer and consumer
3. Every time buffer changes print the current value and who made this change.

## Experiment Code

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<pthread.h>

#include<semaphore.h>

#include<malloc.h>

#include <unistd.h>

#define BUFFER\_SIZE 3

#define Producer\_NUM 4

#define Consumer\_NUM 4

void **\*** p\_create**();**

void **\*** c\_create**();**

void **\*** p\_sub**(**void **\***num**);** //producer routine

void **\*** c\_sub**(**void **\***num**);** //consumer routine

int buffer**=**0**;**

pthread\_mutex\_t mutex**;**

sem\_t empty**,**full**;**

int main**()**

**{**

printf**(**"IN/OUT\t\tC/P\t\tBuffer\n"**);**

fflush**(**stdout**);**

pthread\_t c\_create\_pid**,**p\_create\_pid**;**

pthread\_mutex\_init**(&**mutex**,NULL);**

sem\_init**(&**empty**,**0**,**BUFFER\_SIZE**);**

sem\_init**(&**full**,**0**,**0**);**

// create two threads to create consumer and producer individually

pthread\_create**(&**p\_create\_pid**,NULL,**p\_create**,NULL);**

pthread\_create**(&**c\_create\_pid**,NULL,**c\_create**,NULL);**

pthread\_join**(**p\_create\_pid**,NULL);**

pthread\_join**(**c\_create\_pid**,NULL);**

**return** 0**;**

**}**

void **\*** p\_create**()**

**{**

pthread\_t **\***producer**;**

producer**=(**pthread\_t **\*)**calloc**(**Producer\_NUM**,sizeof(**pthread\_t**));**

**for(**int i**=**0**;**i**<**Producer\_NUM**;++**i**)**

**{**

sleep**(**rand**()%**3**);** //create randomly

pthread\_create**(&**producer**[**i**],NULL,**p\_sub**,&**i**);**

**}**

**for(**int i**=**0**;**i**<**Producer\_NUM**;++**i**)** pthread\_join**(**producer**[**i**],NULL);**

pthread\_exit**(NULL);**

**}**

void **\*** c\_create**()**

**{**

pthread\_t **\***consumer**;**

consumer**=(**pthread\_t **\*)**calloc**(**Consumer\_NUM**,sizeof(**pthread\_t**));**

**for(**int i**=**0**;**i**<**Consumer\_NUM**;++**i**)**

**{**

sleep**(**rand**()%**3**);** //create randomly

pthread\_create**(&**consumer**[**i**],NULL,**c\_sub**,&**i**);**

**}**

**for(**int i**=**0**;**i**<**Consumer\_NUM**;++**i**)** pthread\_join**(**consumer**[**i**],NULL);**

pthread\_exit**(NULL);**

**}**

void **\*** p\_sub**(**void **\***num**)**

**{**

int ID**=\*(**int **\*)** num**;**

**for(**int i**=**0**;**i**<**3**;++**i**)**

**{**

sleep**(**rand**()%**4**);**

//Check

sem\_wait**(&**empty**);**

pthread\_mutex\_lock**(&**mutex**);**

//Critical region

buffer**+=**1**;**

printf**(**"IN\t\tP%d\t\t%d\n"**,**ID**,**buffer**);**

fflush**(**stdout**);**

//Quit

pthread\_mutex\_unlock**(&**mutex**);**

sem\_post**(&**full**);**

**}**

pthread\_exit**(NULL);**

**}**

void **\*** c\_sub**(**void **\***num**)**

**{**

int ID**=\*(**int **\*)** num**;**

**for(**int i**=**0**;**i**<**3**;++**i**)**

**{**

sleep**(**rand**()%**4**);**

//Check

sem\_wait**(&**full**);**

pthread\_mutex\_lock**(&**mutex**);**

//Critical region

buffer**-=**1**;**

printf**(**"OUT\t\tC%d\t\t%d\n"**,**ID**,**buffer**);**

fflush**(**stdout**);**

//Quit

pthread\_mutex\_unlock**(&**mutex**);**

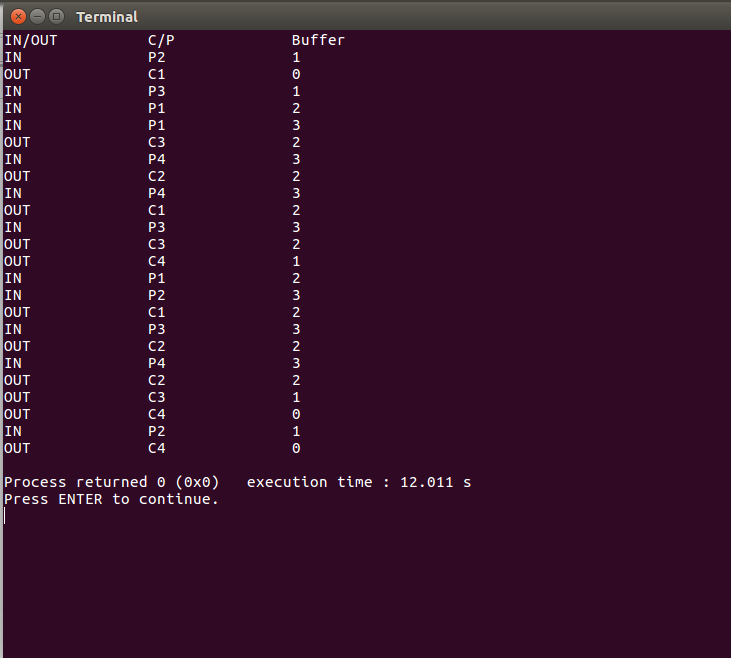
sem\_post**(&**empty**);**

**}**

pthread\_exit**(NULL);**

**}**

**Test:**

****