



LAB – 3

Operating Systems



GroupID - 6

- ◆ Abhijeet Sonkar IIB2019009
- ◆ Aditya Aggarwal IIT2019210
- ◆ Ambika Singh IIB2019017
- ◆ Avneesh Kumar IIB2019010
- ◆ Divy Agrawal IIT2019211



Questions Assigned

6

Implement blocking semaphore using the `pause()` and `kill()` system calls (to block and unblock a process, as demonstrated in my example programs / videos). Now, use your solution to ensure mutually exclusive access to a shared variable by two cooperating processes. Demonstrate the correctness of your solution by applying it to solve the race condition problem that I have created in my sample programs “`producer3.c`” and “`consumer3.c`”.



Table of Contents

1

Basic Concepts

2

Code Explanation

3

Demonstration



1

Basic Concepts



What is Semaphores?

- Proposed by Dijkstra in 1965
- Technique to manage concurrent processes by using integer value, called semaphore.
- Used to solve critical section problem.
- Achieve process synchronization in multi process communication

Binary Semaphore

- Also called mutex lock
- Can have only two values 0 and 1
- Implemented to solve critical section problem in multiple processes.
- Mutual Exclusion is possible.

Counting Semaphore

- Its value can lie in any unrestricted domain
- No mutual exclusion
- More than one process can enter critical section.



More on Binary Semaphore

- Used to control access to single resource.
- Used to enforce mutual exclusion for a critical section.
- Semaphore is created with an initial value of 1 (no task is executing critical section)
- While entering critical section, a task must try to acquire semaphore to prevent other task from entering the critical section.
- While exiting from critical section, the task must release the semaphore to allow another task to execute the critical section.

How it is different from blocking binary semaphore?

- In blocking binary semaphore, when process P1 is executing the critical section, and process P2 tries to acquire semaphore, then P2 will not be able to acquire semaphore and P2 will be stored in a queue. P2 goes to blocked state and does not wakes up until a signal is received.
- P1 on exiting from the critical section, releases the semaphore and wakes up one of the process from the queue.
- It guarantees mutually exclusive access to critical section.



Pseudo code for blocking binary semaphore

```
int sem = 1;    // initial value of shared variable
```

```
Down(sem){  
    //this comparison in if condition is non-atomic  
    while(sem<=0){  
        Store this process onto Queue  
        pause()  
    }  
    //this is also non-atomic operation  
    sem--;  
}
```

```
Up(sem){  
    sem++;    //this is also non-atomic operation  
    Pop one process from the queue and wakeup it  
}
```


What is race condition?

- A race condition occurs when two or more threads or cooperating processes can access shared data and they try to change it at the same time. Because the thread/process scheduling algorithm can swap between threads/processes at any time, we don't know the order in which the threads or the cooperating processes will attempt to access the shared data. And this may lead to different results every time pertaining to the sequence of execution of the cooperating processes or threads. .
- It can be avoided by ensuring mutual exclusion of processes or threads accessing the shared variable at the same time. Also use of many synchronization algorithms or locks can prevent race conditions.

What is atomic operation?

- Atomic implies indivisibility and irreducibility, so Atomic operation are those operations that finish their tasks as a whole i.e. no other interruption is allowed before its completion i.e., an atomic operation must be either performed entirely or not performed at all.
- it is an operation that will always be executed without any other process being able to read or change state that is read or changed during the operation. It is effectively executed as a single step.



2

Code Explanation

Lets briefly discuss the code



```
typedef struct
{
    int x;
    int f;
    int prid;
    int cnid;
} mydata;
int *s;
int id;
void my_handler() {}
```

```
int *s;
```

- shared variable which will be used to implement semaphore and implement mutual exclusion of the two cooperating processes.

```
int id;
```

- it is a global variable declared to store the id of the shared memory associated with the key for shared variable s.



```
void down()
{
    while (!__sync_val_compare_and_swap(s, 1, 0))
    {
        pause();
    }
}
```

`void down()`

function is invoked by the process trying to enter the critical section to acquire the semaphore so that it can gain access to the critical section.

`pause();`

- Causes calling process/thread to sleep until a signal is delivered.
- Signaling leads to the process getting unblocked and starting its execution again.

`while (!__sync_val_compare_and_swap(s, 1, 0))`

- non-busy waiting
- if no process has yet acquired the semaphore and the critical section then invoking process can acquire it by decrementing its value from 1 to 0 and enter critical section.
- if critical section is already acquired, then invoking process gets blocked and waits until next signal.
- decrementing semaphore is not atomic, So we swap the previous value with 0 using atomic swap function `__sync_val_compare_and_swap(s, 1, 0)`.
- function return previously stored value, which is used to check if any process has acquired critical section or not.



```
void up()
{
    __sync_val_compare_and_swap(s, 0, 1);
    kill(data->cnid, SIGUSR1);
}
```

`void up()` function is invoked by the process exiting the critical section to release previously acquired semaphore, so that other process can execute critical section

`__sync_val_compare_and_swap(s, 0, 1);`

- To release the access of critical section, we need to increment the semaphore from 0 to 1
- But incrementing is not an atomic operation, so we will use above function to swap semaphore value atomically.

`kill(data->cnid, SIGUSR1);`

- Send signal to process whose process_id (`data->cnid`) has been specified.

```

void create_shared_memory()
{
    key_t keys = ftok("buffer.txt", 18);
    if (keys < 0)
    {
        perror("error0:");
        exit(1);
    }
    id = shmget(keys, sizeof(int), IPC_CREAT | 0666);
    if (id < 0)
    {
        perror("error1:");
        exit(1);
    }
    s = (int *)shmat(id, NULL, 0);
    if (s == (void *)-1)
    {
        perror("error2:");
        exit(1);
    }
    (*s) = 1;
}

```

```
ftok("buffer.txt", 18);
```

Used to create a unique key associated with shared memory.

```
if (keys < 0){...}
```

Checks if the `keys` was generated successfully or not. If not print error and exit from program.

```
id = shmget(keys, sizeof(int), IPC_CREAT | 0666);
```

returns the identifier of the shared memory segment associated with the value of the argument key.

IPC_CREAT to create a new segment.

0666 gives all users on the system to access the shared memory and access to read and write.

```
if (id < 0){...}
```

Checks if the `id` was generated successfully or not. If not print error and exit from program.

```
s = (int *)shmat(id, NULL, 0);
```

accepts a shared memory ID, `id`, and attaches the indicated shared memory to the program's address space. The returned value is a pointer of type `(void *)` to the attached shared memory.

```
if (s == (void *)-1){...}
```

Used to check if shared memory segment was attached successfully or not to the the address space of calling process. Return -1 in case of error, prints error and terminates program.

```
(*s) = 1;
```

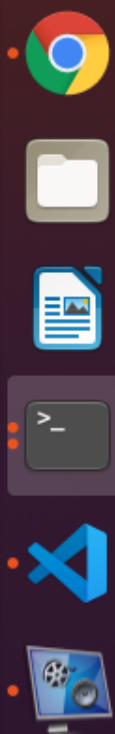
Initializes the value of semaphore to 1.



3

Demonstration

Lets have a look at the screenshots



aditya@aditya: ~/Desktop/os/OS_Group_6/codes



adio Code



```
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./producer3
Producer Program Started...
...The value of n is: 100000...
Producer: Final Value of data->x = 0
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./producer3
Producer Program Started...
...The value of n is: 100000...
Producer: Final Value of data->x = 0
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./producer3
Producer Program Started...
...The value of n is: 100000...
Producer: Final Value of data->x = 0
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./producer3
Producer Program Started...
...The value of n is: 100000...
Producer: Final Value of data->x = 0
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./producer3
Producer Program Started...
...The value of n is: 100000...
Producer: Final Value of data->x = 0
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./producer3
Producer Program Started...
...The value of n is: 100000...
```

```
68     n = 100000;
69
70     create_shared_memory();
71     pause();
72
73     for(i = 1; i<=n; i++){
74         down();
75         (data->x)--;
76         up(data);
77     }
78     data->f = 1;
79 }
80
```



> OUTLINE



aditya@aditya: ~/Desktop/os/OS_Group_6/codes



```
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ gcc producer3.c -o producer3
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ gcc consumer3.c -o consumer3
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$
```



Radio Code

60

 $\otimes 0 \triangle 0$

Radio Code




```
aditya@aditya:~/Desktop/os/OS_Group_6/codes$
```

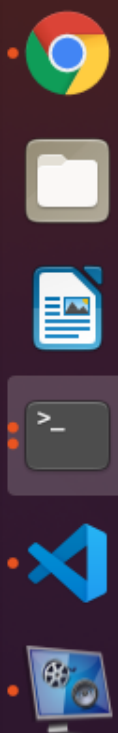


Radio Code

```
...The value of n is: 10000000...
```

> OUTLINE

Ln 28, Col 36 Tab Size: 4 UTF-8 LF C Linux  



aditya@aditya: ~/Desktop/os/OS_Group_6/codes



adio Code



...The value of n is: 1000000...

Producer: Final Value of data->x = 0

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./producer3

Producer Program Started...

...The value of n is: 1000000...

Producer: Final Value of data->x = 0

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./producer3

Producer Program Started...

...The value of n is: 1000000...

Producer: Final Value of data->x = 0

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./producer3

Producer Program Started...

...The value of n is: 1000000...

Producer: Final Value of data->x = 0

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./producer3

Producer Program Started...

...The value of n is: 1000000...

Producer: Final Value of data->x = 0

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./producer3

Producer Program Started...

...The value of n is: 1000000...

Producer: Final Value of data->x = 0

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$

40 perror("error2:");

41 exit(1);

42 }

43 }

44 int main(){

45 printf("Consumer Program Sta

46 signal(SIGUSR1,my_handler);

47 int status, i, n;

48 int shmid;

49 mydata * data;

50 key_t key;

51 key = ftok("buffer.txt", 15)

52 if(key<0){

53 perror("error2:");



> OUTLINE



aditya@aditya: ~/Desktop/os/OS_Group_6/codes



aditya@aditya:~/Desktop/os/OS_Group_6/codes\$ gcc consumer3.c -o consumer3

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$ gcc producer3.c -o producer3

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./consumer3

Consumer Program Started...

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./consumer3

Consumer Program Started...

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./consumer3

Consumer Program Started...

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./consumer3

Consumer Program Started...

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./consumer3

Consumer Program Started...

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./consumer3

Consumer Program Started...

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./consumer3

Consumer Program Started...

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./consumer3

Consumer Program Started...

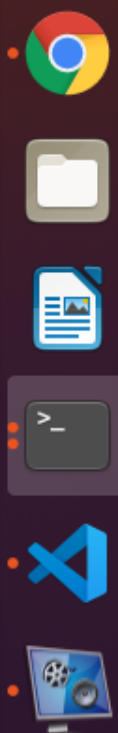
aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./consumer3

Consumer Program Started...

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$./consumer3

Consumer Program Started...

aditya@aditya:~/Desktop/os/OS_Group_6/codes\$



aditya@aditya: ~/Desktop/os/OS_Group_6/codes



udio Code



```
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./producer3
Producer Program Started...
...The value of n is: 10000000...
Producer: Final Value of data->x = 0
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./producer3
Producer Program Started...
...The value of n is: 10000000...
Producer: Final Value of data->x = 0
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./producer3
Producer Program Started...
...The value of n is: 10000000...
Producer: Final Value of data->x = 0
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./producer3
Producer Program Started...
...The value of n is: 10000000...
Producer: Final Value of data->x = 0
aditya@aditya:~/Desktop/os/OS_Group_6/codes$
```



aditya@aditya: ~/Desktop/os/OS_Group_6/codes



```
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ gcc producer3.c -o producer3
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ gcc consumer3.c -o consumer3
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$ ./consumer3
Consumer Program Started...
aditya@aditya:~/Desktop/os/OS_Group_6/codes$
```

```
49 mydata = data;
50 key_t key;
51 key = ftok("buffer.txt", 17);
52 if(key<0){
53     perror("error0:");
54     exit(1);
55 }
56 shmid = shmget(key, sizeof(m
57 if(shmid<0){
58     perror("error1:");
59     exit(1);
60 }
61 data = (mydata *) shmat(shmi
62 if(data == (void *) -1){
```



> OUTLINE

vokoscreen 2.5.0

Icons: Monitor, Microphone, Music, Gear, Webcam, Info

☒ Fullscreen ☐ Window ☐ Area

Display 1: 1366 x 768

☐ Magnification ...

☐ Showkey

☐ Showclick ...

Countdown 0

Buttons: Start, Stop, Pause, Play, Send, Info

00:00:00 25 0 F:1366x768 libx264 mkv Pulse 25

```
71 printf("...The value of n is: %d...\n",n);
72 create_shared_memory(data);
73 kill(data->cnid,SIGUSR1);
74
75 for(i=1; i<=n; i++){
76     down();
77     (data->x)++;
78     up(data);
79 }
80
81 while(data->f == 0){
82     sleep(1);
83 }
84 printf("Producer: Final Value of data->x = %d\n", data->x);
85
86 shmctl(shmid, IPC_RMID, NULL);
87 shmctl(id, IPC_RMID, NULL);
88
89
```

There is an available update.

Download Update

Later

Release Notes

> OUTLINE

THANK
YOU

