CS330 Assignment 3

Nishi Metha
Maurya Jadav
Mohd Umam
Shubham Kumar

November 15, 2022

Condition variable

Defined struct cond_t which includes a name and sleeplock as variables. Used this struct for implementing condition variables. Implemented cond_init, cond_wait, cond_signal and cond_broadcast functions. Cond_wait used condsleep function written in proc.c, while cond_broadcast used a predefined function wakeup and cond_signal used an edited verion wakeupone of the same wakeup function. Spinlock in wakeup is converted to sleeplock so that process sleeps while it has to wait.

Semaphore

Similarly a struct semaphore is defined that contains an integer variable to store the semaphore value, a condition variable and sleeplock. Semaphore.c contains sem_init, sem_wait and sem_post. Sem_init initialises condition variable by cond_init, and initialises sleeplock using initsleeplock functions. The semaphore value is initialised by input parameter. Sem_wait allows semaphore number of processes to run and while next processes waits until sem_post is called by any of the process. Sem_post increments the alue by 1 and wakes up a sleeping process using $cond_signal$.

System calls

1 Barrier

We have used an array of condition variable to implement Barrier. The barrier array is an integer array which stores -1 when not initialised or freed, and number of processes called after initialised.

1.1 barrier_alloc

Initialises the first free barrier and initialises its value 0 from -1, initialise corresponding condition variable and returns the id of the barrier that is allocated.

1.2 barrier

Uses a global lock for printing so that no other process can print when 1 process is printing may be that process be of different barrier id. Uses the sleep lock of condition variable so that a process of same barrier waits until all the process of the same barrier enters first.

1.3 barrier_free

Sets the value of freed barrier again to -1.

2 Condition Producer Consumer

Created buffer_elem struct having 2 condition variables insert and delete. A sleeplock, an integer storing value and a flag named full.

2.1 buffer_cond_init

Initialises tail and head to 0.also initialises inser, delete, print sleeplock that is common for all processes. Initialises the values of the buffer_elem.

2.2 cond_produce

Producer produces if the buffer_elem that the tail points to is empty, and while producing uses inserlock of that particular buffer_elem.

2.3 cond_consume

Similar process as of producer but vice versa.

3 Semaphore Producer Consumer

Same function as of contion producer and consumer using emaphores.

3.1 buffer_sem_init

Initialises nextp, nextc the producer and consumer pointers to 0. Initialises 4 semaphoers namely - sem_prod=0, sem_cons=0, sem_empty=size of buffer, sem_full=0

3.2 sem_produce

Produces if the buffer element is empty using sem_prod semaphore. Every time nextp is incremented cyclically. checking empty is done by semaphore and posted full when produced.

3.3 sem_consume

Produces if the buffer element is empty using sem_prod semaphore. Method similar to producer just viceversa.

Observation

Semaphore takes more time to implement as the number increases.