Operating Systems Assignment3Report

10/29/2016

Jinhao Chen

Project Objective:

The purpose of this project is to give us an opportunity to experiment with process synchronization mechanisms. In this assignment, a circular buffer with 15 positions is to be used to communicate information between two threads. In this project, the producer thread will read characters, one by one from a file and place it in the buffer and continue to do that until the EOF marker is reached, and consumer will read the character one by one and print it to the screen.

Overview:

In this project, we already got some code such as header, how to create variable and main function. The only thing we need to create is producer and consumer.

```
void *producer(){
    //Prepare for file Input
    char newChar;
    FILE* fp;
    fp= fopen("mytest.dat", "r");
    while(1){
        sem_wait(&buffer_empty);
        sem_wait(&mutex);
        //Loop continuously until the end of the file
        if(fscanf(fp,"%c",&newChar) != EOF){
            buffer[i] = newChar;
            i = (i + 1)%BUFFERSIZE;
        }
        else{
        //Places an asterisk in the buffer when the end of file is reached
                    buffer[i] = '*';
             //release the buffer
            sem_post(&mutex);
            sem_post(&buffer_full);
            break;
        }
        sem_post(&mutex);
        sem_post(&buffer_full);
    }
    fclose(fp);
}
```

In the previous picture, this is producer function. At the beginning if there are no empty slots and another thread uses the buffer, wait. Then get each character from the file until it reaches the end and place *. After this, it releases the buffer. The consumer is similar to the producer. The only difference is adding a sleep function to make a delay, and print the characters to the screen, and count the characters.

```
//CONSUMER FUNCTION
void *consumer(){
    while(1){
        sleep(1);                   //delay for consumer
        sem_wait(&buffer_full);
        sem_wait(&mutex);
        fflush(stdout);             //flushes out
        //Loops continuously until an asterisk is found in the buffer
        if(buffer[j] !='*'){
            printf("%c", buffer[j]);
            j = (j + 1)%BUFFERSIZE;
            if(buffer[j] != '\n')
                        count = count + 1;
                          //New lines not considered as a character
        }
        else{
            sem_post(&mutex);
            sem_post(&buffer_empty);
            break;
        }
        sem_post(&mutex);
        sem_post(&buffer_empty);
    }
}
```

Conclusion:

I tested the program several times, and the following images are the results on Linux. In my opinion, it seems that the program works correctly, when it prints the characters, it prints the characters one by one same as the requirement. After printing all the character, it gives the parent counter value.

```
[jinhao@login6 os]$ gcc os3.c -lpthread -lrt
[jinhao@login6 os]$ ./a.out
fsdafsadfasd
dsfsdafasd
dfsa
abcd
addsfgh
gfdhjkeyk
bvnvbnyhkyf
gbvnc
vcn
vcn
nbmv


------------------------------------------------
From parent counter  =  72
------------------------------------------------
                              End of simulation
[jinhao@login6 os]$
```