CSE 344 SYSTEM PROGRAMMING:

REPORT:

- Serdil Anıl Ünlü
- 1801042672

HOW I SOLVED THIS PROBLEM?

```
void suppliersemOperation(int index){

    struct sembuf waitpostSem = {index, 1, 0};

    if (semop(semid, &waitpostSem, 1) == -1)
    {
            perror("semop");
            exit(1);
    }

}
```

-I am using the semop function to do the post operation in the above function. With supplier I'm only doing post here once.

```
void consumersemOperation(){

    struct sembuf waitpostSem[2] = {{0, -1, 0},{1,-1,0}};

    if (semop(semid, waitpostSem, 2) == -1)
    {
            perror("semop");
            exit(1);
    }

}
```

-Since the consumer will consume 2 times, I used the system semaphore's wait feature 2 times in a row.

```c
void* supplier_print(){
    FILE *fp;

    fp = fopen(fileName,"r");

    if(fp == NULL){
        perror("cannot open file.\n");
    }

    int c;

    do
    {
        char buffer[26];
        timestamp(buffer);

        c = fgetc(fp);
        if(c == '\n' || c == EOF)break;
        printf("timestamp:%s ,Supplier: read from input a '%c'. Current amounts: %d x '1', %d x '2'.\n",buffer,c,semctl(semid

        switch (c)
        {
        case '1':
            suppliersemOperation(0);

        break;

        case '2':
            suppliersemOperation(1);
        break;

        default:
            break;
        }
        timestamp(buffer);

        printf("timestamp:%s ,Supplier: delivered a '%c'. Post-delivery amounts: %d x '1', %d x '2'.\n",buffer,c,semctl(semid
```

-In this function, I am reading the supplier's file, and according to the incoming numbers 1 and 2, I perform the post operation within the switch(case) structure. If a comes up, I call 0 in the function and if 2 comes up, I call 1. I also used this function to print something to the terminal. I also used this function to print something to the terminal.

```
void* consumer_print(void* arg){

    char buffer[26];
    int b = (intptr_t)arg;
    timestamp(buffer);

    for(int i=0;i<N;i++){

        printf("timestamp:%s ,Consumer-%d at iteration %d (waiting). Current amounts: %d x '1', %d x '2'.\n",buffer,b,i,semct

        consumersemOperation();
        timestamp(buffer);
        printf("timestamp:%s ,Consumer-%d at iteration %d (consumed). Post-consumption amounts: %d x '1', %d x '2'.\n",buffer

    }
    printf("timestamp:%s,Consumer-%d has left.\n",buffer,b);

    return NULL;

}
```

-I’m doing consumer’s print here, I’m using the ‘GETVALUE’ operation with semctl to print the value of the semaphore.

```
void timestamp(char buf[26]){

    time_t timer;
    struct tm* timestamp;

    timer = time(NULL);
    timestamp = localtime(&timer);

    strftime(buf, 26, "%Y-%m-%d %H:%M:%S", timestamp);
}
```

This function is to show the timestamp operation at the beginning of each line.

```c
struct sigaction sa;
memset(&sa,0,sizeof(sa));
sa.sa_handler=handler;
sigaction(SIGINT,&sa,NULL);
char *input = NULL;
char *output = NULL;
```

```c
if(sigusr1_count == 1){
    write(1, "SIGINT signal is caught, exiting gracefully...\n", 44);
    free(input);
    free(output);
    return -1;
}
```

I have defined a global function for the SIGINT operation, at the same time I have a value with a global value, then I perform the sigaction operation in the main and call the signal in between.

```c
if(setvbuf(stdout, NULL, _IONBF, 0)) {
    perror("failed to change the buffer of stdout\n");
    return EXIT_FAILURE;
}
```

Since all output will be written to STDOUT without buffering I use setvbuf function.

```c
if ((key = ftok("/tmp", 1)) == -1)
{
    perror("ftok");
    exit(1);
}
```

```c
if ((semid = semget(key, 2, IPC_CREAT|IPC_EXCL|0600)) == -1)
{
    perror("semget");
    exit(1);
}
```

I used the ftok and semget operations respectively to open the key in main.

```c
pthread_t array[C];

for (int i = 0; i < C; i++)
{
    pthread_create(&array[i],NULL,consumer_print,(void*)(intptr_t)i);
}
```

First of all, I install consumer threads in Main. I open the consumer thread in the for loop as much as the number of C.

```c
for (int i = 0; i < C; i++)
{
    pthread_join(array[i],NULL);
}
```

Also , for the consumer, I am performing the join operation in the for loop again.

```
pthread_t thread1_id;


pthread_create(&thread1_id,NULL,supplier_print,NULL);

pthread_detach(thread1_id);
```

I perform the detach operation after creating the supplier here.

```
semctl(semid, 0, IPC_RMID);
```

Finally, I call the semctl function to delete the semaphores.

HOW MY DESIGN DECISION?

Since I use 2 semaphores as a design, I did not work in a certain order between the consumer and the supplier. That's why they work in a messy way when printing to the terminal. Besides, I used a separate function to print the post and wait processes separately to the terminal.

WHICH REQIREMENTS I ACHIEVED AND WHICH I HAVE FAILED?

-My program is working perfectly, I have done all the given tasks without an problems.