UNIVERSITY OF
THESSALY

Ταυτόχρονος Προγραμματισμός

Εργασία 4,Ομάδα 15

Αγγέλης Μάριος-Κασιδάκης Θεόδωρος
AEM:2406-2258

4.1 Coroutines

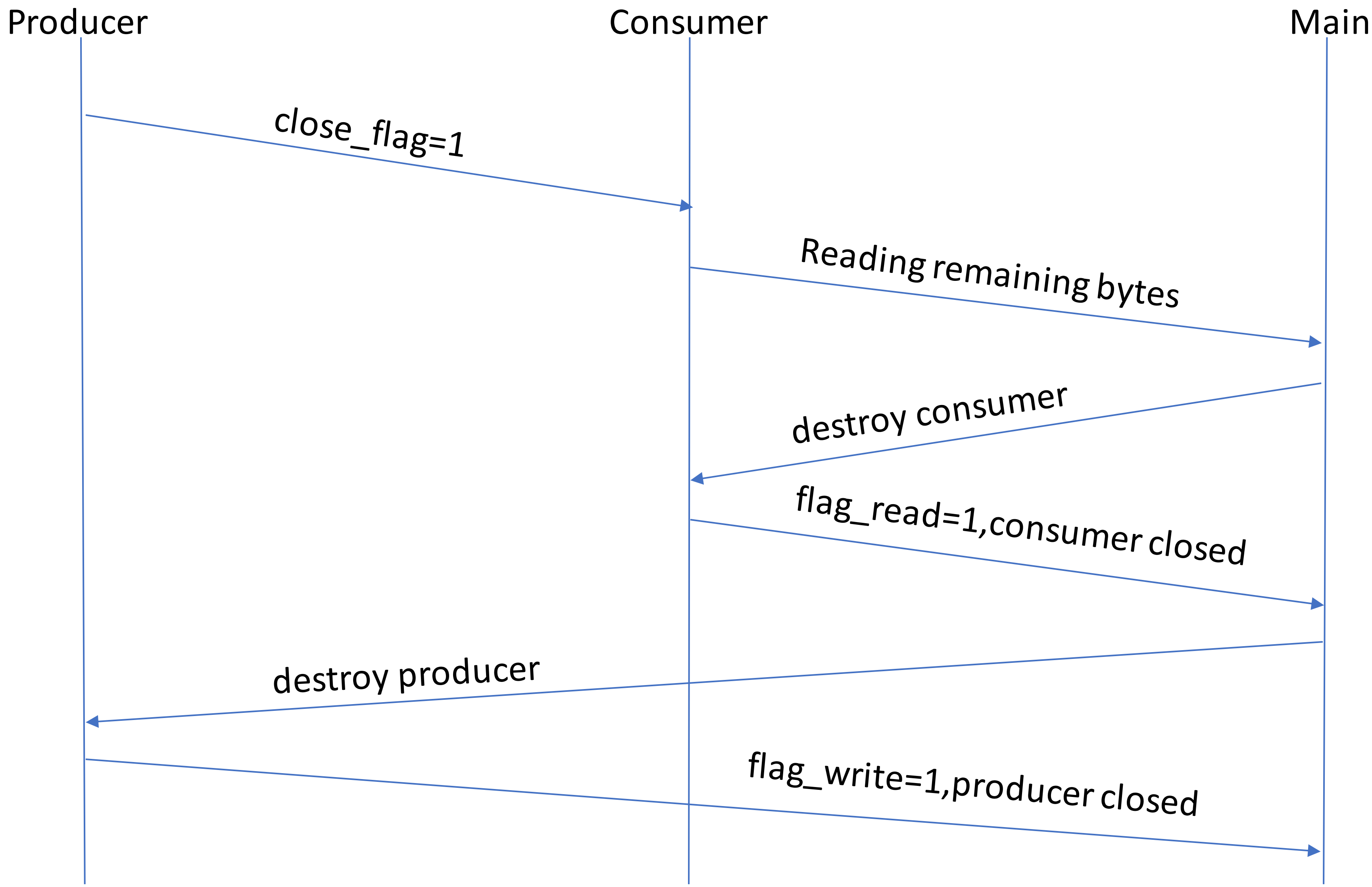
```
void producer(co_t *mycot,ucontext_t *tocontext ){
    while(1){
        //Reading from file
        if(EOF){            /*Eof,write is clossing*/
            close_flag=1;
            mycot->destination_context=*tocontext;
            mycoroutines_switchto(mycot);
        }
        /*Producer is writing inside buffer*/
        pipe_write(byte);
        index_write++;
        /*Buffer is full,switch to consumer*/
        if(index_write==size){
            mycot->destination_context=*tocontext;
            mycoroutines_switchto(mycot);
        }
    }
}
```

```
void consumer(co_t *mycot,ucontext_t *tocontext){
    while(1){
        if(close_flag==1){
            //Read remaining bytes
        }
        else{
            //Consumer is reading
            pipe_read(&read_byte);
            nwrite=write(fd,&read_byte,1);
            index_read++;
            if(index_read==size){
                index_read=0;
                mycot->destination_context=*tocontext;
                mycoroutines_switchto(mycot);
            }
        }
    }
```

Coroutines struct

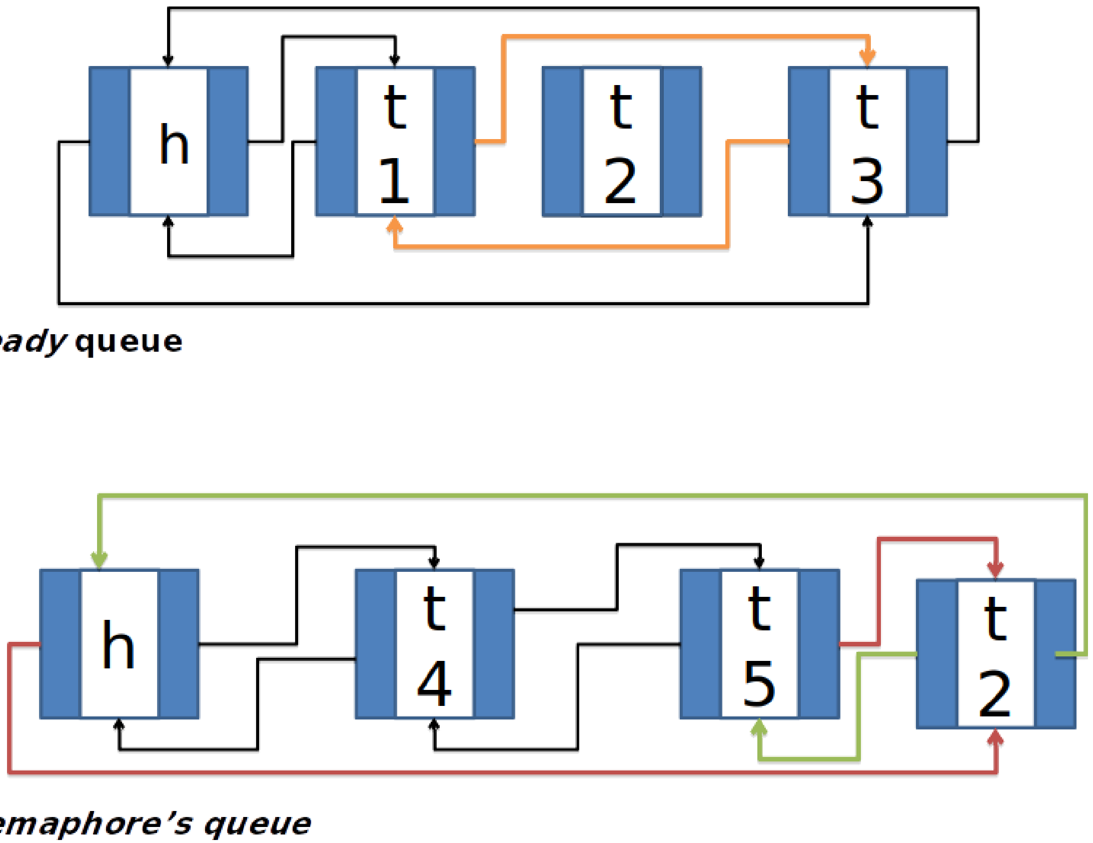
```
typedef struct coroutine {
        ucontext_t source_context;
        ucontext_t destination_context;
        char mystack[SIGSTKSZ];
}co_t;
```

Destroy coroutines FSM

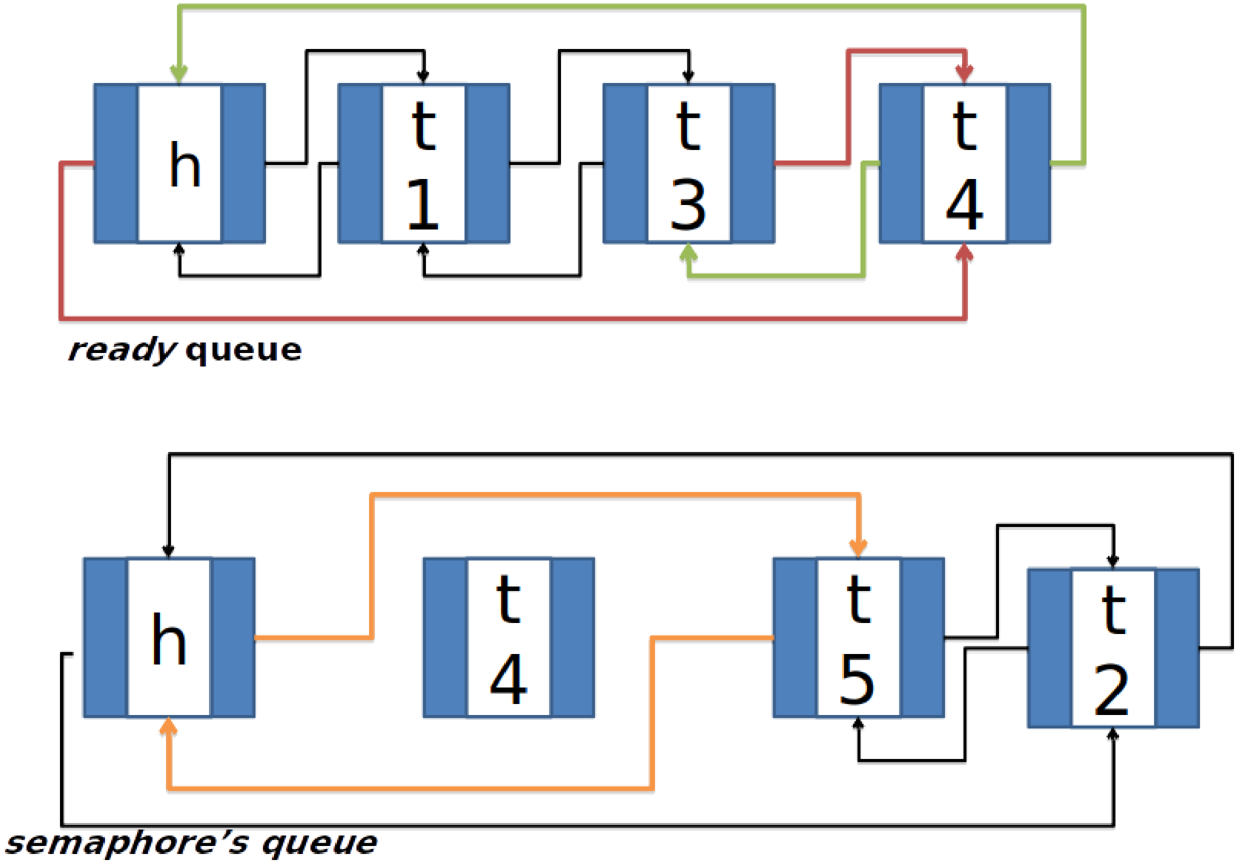


4.2 Ready list implementation

thread 2 (t2) executes down(&sem)



Some thread executes up(&sem)



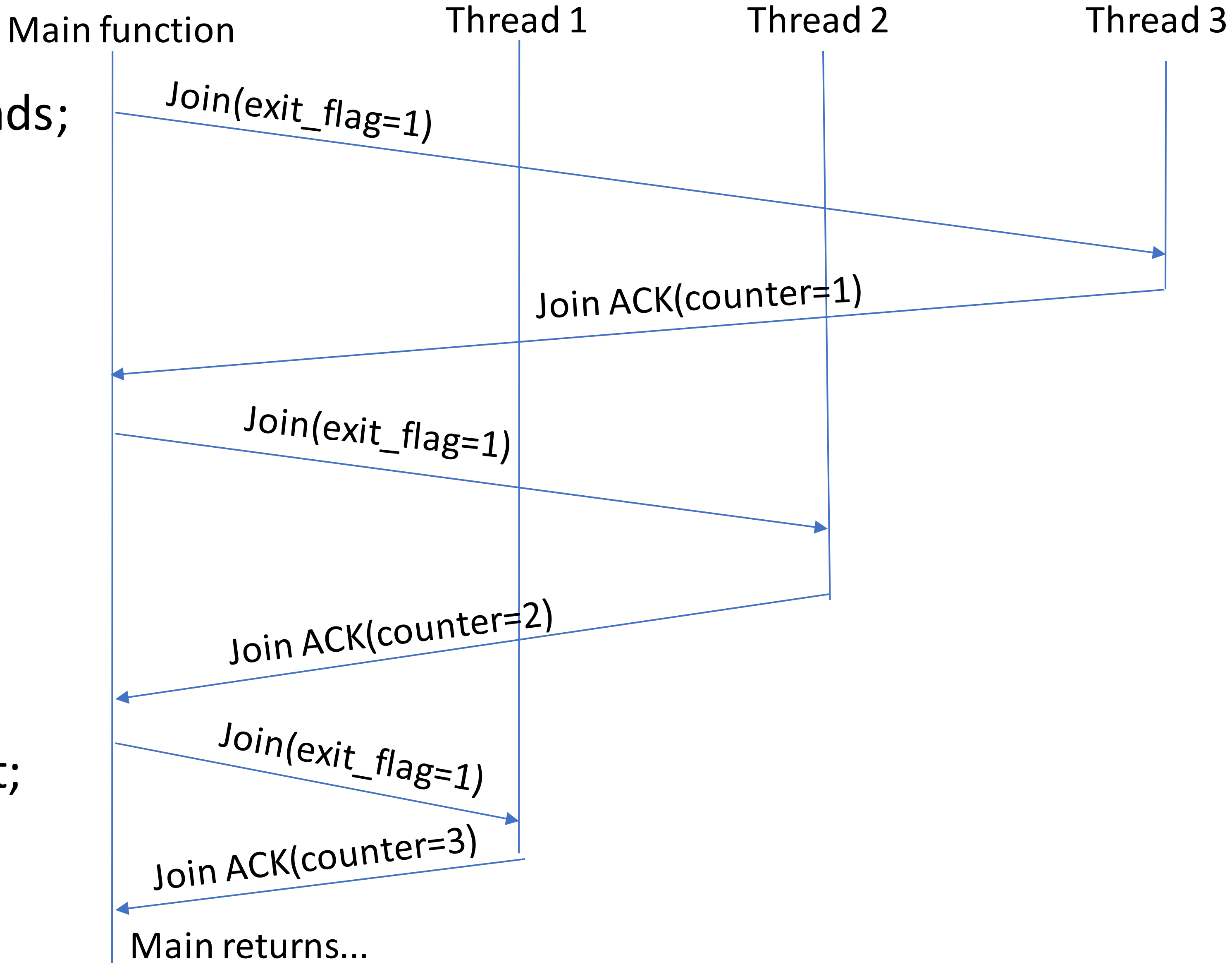
Semaphores struct

```
typedef struct semaphore{
        int sem_counter;
        int num_of_blocked_threads;
        thr_t *head;
}sem;
```

Thread struct

```
typedef struct node {
        int position;
        struct node *next;
        struct node *prev;
        int exit_flag;
        ucontext_t source_context;
        char mystack[SIGSTKSZ];
        int spin_sem;
}thr_t;
```

Join threads FSM


Main function
Thread 1
Thread 2
Thread 3
Join(exit_flag=1)
Join ACK(counter=1)
Join(exit_flag=1)
Join ACK(counter=2)
Join(exit_flag=1)
Join ACK(counter=3)
Main returns...