Ταυτόχρονος Προγραμματισμός

Εργασία 1,Ομάδα 15

Αγγέλης Μάριος-Κασιδάκης Θεόδωρος

ΑΕΜ:2406-2258

1.Fifo Pipe

```
//Read function
void *thread_read(void *ptr){
    while(index_write==0){}
    //Waiting  when pipe is empty
    while (close_flag==0 ){
        //Do not  overread,wait "write" to continue
        if(index_read==(index_write-1)){sched_yield();}
        else{
            //read a character from buffer
            index_read++;
            if(index_read==size){
                while(index_write==0){} //buffer is circular
                index_read=0;
            }
        }
    }
    if(close_flag==1){
        //Read all characters from buffer and return
    }
}
```

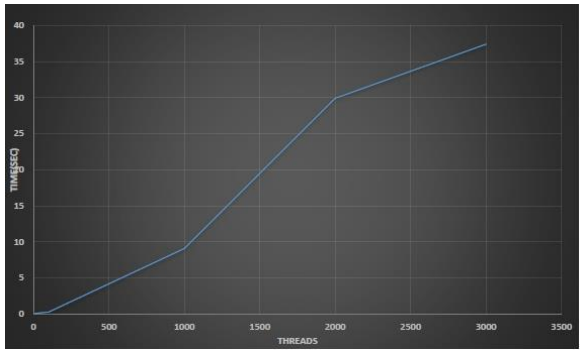
```
//Write function
void *thread_write( void *ptr ){
    while(1){
        //Do not overwrite,wait  "read" to continue
        if(index_write==(index_read-1){sched_yield();}
        else{
            //reading from a file
            if(EOF){close_flag=1;break;}
            pipe_write();
            index_write++;
            if(index_write==size){
                //buffer is circular
                while(index_read==0){}
                index_write=0;
            }
        }
    }
}
```

2.Prime numbers recognition

```
//master thread(main)
//create "size" threads and initialize each flag and each close to 0.
while(1){
    //reading a number from a file
    if (EOF){break;}
    while(1){   //Spinning until finding a free worker
        for(int i=0;i<size;i++){
            if(info[i].flag==0){
                /*Main found a free worker and
                gave him work*/
                info[i].prime=number;
                info[i].flag=1;//Worker has job now
            }
        }
    }
}
//Wait all threads to return and exit
```

```
//thread struct
struct T {
    Pthread_t id;
    int iret,flag,close,position;
    long int prime;
};
void *thread_func(void *arg){
    struct T *thread_struct=(struct T*)arg;
    while(1){
        while(thread's flag==0 and thread's close ==0){}
        if(thread's close==1){break;}
        primetest(info[thread_struct->position].prime);
        //do main's work and make your flag=0
        info[thread_struct->position].flag=0;
        //This thread is an available worker now
    }
}
```

//threads diagram



3.QuickSort

```
int quicksort(int *a,int left,int right)
    struct arguments args1,args2;
    args2.flag=1;
    args1.flag=1;
    // quicksort swapping  using
    //i and j variables
    if(left<i-1){        //left recurssion step
        //create a thread with flag=0
    }
    if(i+1<right){
        //create a thread with flag=0
   }
   //Wait left and right threads to return
   while(args1.flag==0 || args2.flag==0){}
   return(1);
}
```

```
struct arguments{      //thread struct
     int left,right,flag;
};
void *thread_func(void *ptr){
     //Call quicksort with ptr's left and right
     //with typecast((struct arguments *)ptr->..)
     //thread finished quicksort-ready to return
     ((struct arguments *)ptr)->flag=1;
     return NULL;
```