İŞLETİM SİSTEMLERİ PROJE RAPORU

Projemiz çerçevesinde geliştirdiğimiz cevapları geliştirirken ağırlıklı olarak ders kapsamında sunulan pdflerden faydalandık. Kodlarının görsellerini ve test sürelerinin tablolarını sunarak başlıyoruz. Program kodları ve test tabloları sonrasında bu üç part'ın kendi aralarındaki kıyaslamaları da tablo olarak sunulacaktır.

Part 1 Kısmının Kodları:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <string.h>
#include <time.h>

char interfilenames[5][16] = {"interfile1.txt","interfile2.txt","interfile3.txt","interfile4.txt","interfile5.txt"};

int main(int argc,char* argv[])
{
    int K;
    sscanf(argv[1],"%d",&K);
    int N;
    sscanf(argv[2],"%d",&N);

    if(N<=0 || N>5)
    {
    printf("Hatali N Girisi\n");
        return 0;
    }
    if(K < 0)
    {
        printf("Hatali K Girisi\n");
        return 0;
    }
    if((argc-3) != N)
    {
        printf("Eksik Dosya Sayisi\n");
        return 0;
    }


    int length = 1,length2,a_length = 3,a_in = 0,fd,i,j,k,key;
    char* buffer = (char*) malloc(sizeof(char));
    char* fnumber = (char*) malloc(12 * sizeof(char));
    int *array = (int*) malloc(a_length * sizeof(int));
    char cbuffer[12];

    clock_t start = clock(),stop;
    for(i = 0;i < N;i++)
    {
        if(fork() == 0)
        {
            fd = open(argv[i + 3],O_RDWR);
            if( fd < 0)
            {
                printf("%d.Dosya Bulunamadi\n",(i + 1));
                exit(0);
            }

            while(length != 0)
            {
                length = read(fd,buffer,sizeof(char));
                if(a_in == a_length - 2)
                {
                    a_length += 10;
                    array = realloc(array,a_length * sizeof(int));
                }
```

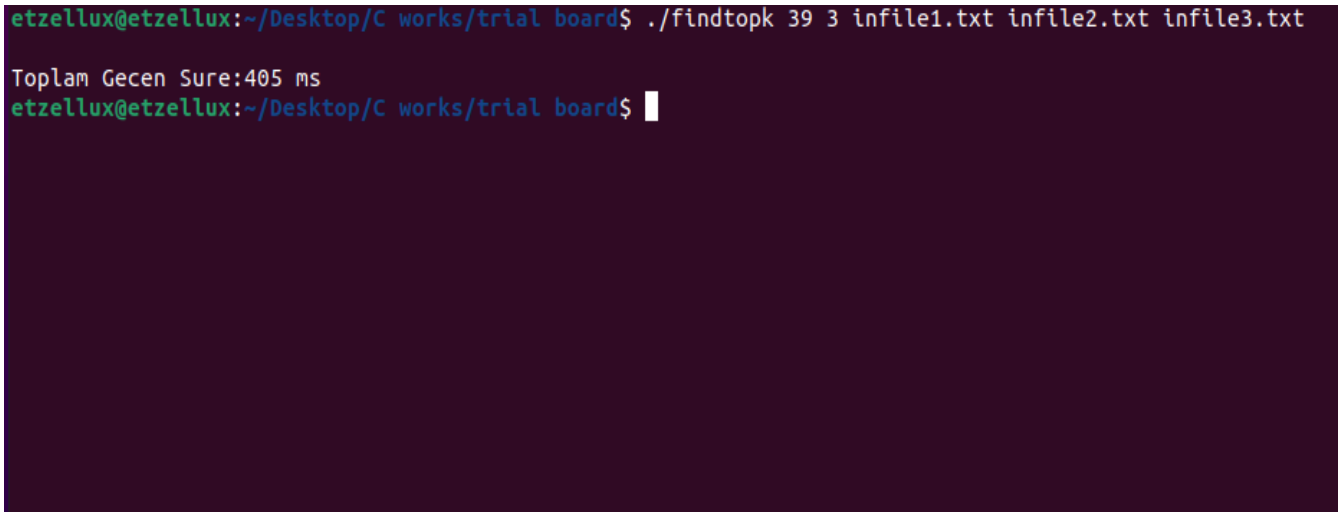
```
            }
            if(strcmp(buffer," ") != 0)
            {
                strcat(fnumber,buffer);
            }
            else
            {
                sscanf(fnumber,"%d",array + a_in);
                strcpy(fnumber,"\0");
                a_in++;
            }
        }
        for(j = 0;j < a_in;j++)
        {
            key = array[j];
            k = j - 1;

            while(k >= 0 && key < array[k])
            {
                array[k + 1] = array[k];
                k--;
            }
            array[k + 1] = key;
        }

        close(fd);
        fd = open(interfilenames[i],O_CREAT | O_EXCL | O_RDWR,S_IRWXU);

        length2 = sprintf(cbuffer,"%d",array[K]);
        strcat(cbuffer," ");
        write(fd,cbuffer,length2 + 1);

        close(fd);

        free(buffer);
        free(fnumber);
        free(array);

        exit(0);
    }
}
while( wait(NULL) > 0);

if(1)
{

    for(i = 0; i < N;i++)
    {
        fd = open(interfilenames[i],O_RDWR);

        length = 1;

        while(length != 0)
        {
            length = read(fd,buffer,sizeof(char));
            if(a_in == a_length - 2)
            {
                a_length += 1;
                array = realloc(array,a_length * sizeof(int));
            }
            if(strcmp(buffer," ") != 0)
```

```
        while(length != 0)
        {
            length = read(fd,buffer,sizeof(char));
            if(a_in == a_length - 2)
            {
                a_length += 1;
                array = realloc(array,a_length * sizeof(int));
            }
            if(strcmp(buffer," ") != 0)
            {
                strcat(fnumber,buffer);
            }
            else
            {
                sscanf(fnumber,"%d",array + a_in);
                strcpy(fnumber,"\0");
                a_in++;
                break;
            }
        }
        close(fd);
        remove(interfilenames[i]);
    }

    for(j = 0;j < a_in;j++)
    {
        key = array[j];
        k = j - 1;

        while(k >= 0 && key < array[k])
        {
            array[k + 1] = array[k];
            k--;
        }
        array[k + 1] = key;
    }

    fd = open("output.txt",O_CREAT | O_EXCL | O_RDWR,S_IRWXU);

    for(j = 0;j < a_in;j++)
    {
        strcpy(cbuffer,"\0");
        length2 = sprintf(cbuffer,"%d",array[j]);
        strcat(cbuffer,"\n");
        write(fd,cbuffer,length2 + 1);
    }

    close(fd);
    stop = clock();

    free(buffer);
    free(fnumber);
    free(array);

    printf("\nToplam Gecen Sure:%ld ms\n",stop-start);

}

return 0;
```

Part 1 ‘ in kodlaması görüldüğü gibidir. Bu kısma verilen 100, 1000 ve 10000 sayı içeren, içerisinde 0-100 arası sayılar doldurduğumuz text dosyalarını çalıştırma, işleme ve sonuca ulaştırma süreleri ise ***saniye*** cinsinden tablomuzda görülmektedir:

N Değeri / Dosyadaki Değer Sayısı	1	2	3	4	5
100	0,12	0,187	0,223	0,272	0,414
1000	0,17	0,202	0,276	0,362	0,468
10K	0,168	0,254	0,287	0,374	0,480

Part 1 kısmı test edilirken k değeri sabit 50 olarak alınmış olup sonuçlar buna göre elde edilmiştir. Part 1 kısmına ait örnek çıktı ise şöyledir:



Part 2 Kısmının Kodları:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <errno.h>
#include <time.h>


double time_spent = 0.0;


//******* FUNCTIONS  *******

int receive_data();
void send_data(int msg_id);
void sort_and_write(int* arr, int size);


struct mesg_buffer {
    long mesg_type;
    char mesg_text[1];
} message;



int top_k,PROCESS_COUNT;        //PROCESS_COUNT = N DEGERI
char filenames[6][15];

int main(int argc,char* argv[])
{
    int files = argc-3;
    PROCESS_COUNT = atoi(argv[2]);
    if(files > PROCESS_COUNT+1){
        printf("\n\n!!!  Girdiginiz Dokuman Sayisi Fazla !!!\n\n");
        exit(0);
    }else if(files < PROCESS_COUNT+1){
        printf("\n\n!!!  Girdiginiz Dokuman Sayisi Eksik !!!\n\n");
        exit(0);
    }
    top_k = atoi(argv[1]);
    printf("N = %d\n",PROCESS_COUNT);
    printf("k = %d\n",top_k);
    printf("Dosyalar :  ");
    strcat(filenames[6],argv[argc-1]);
    for(int a = 0;a<files-1;a++){
        strcat(filenames[a],argv[a+3]);
        printf("%s  ",filenames[a]);

    }
    printf("\nOUTPUT FILE : %s\n\n",filenames[6]);


    int length = 1,a_length = 10,a_in = 0,fd,i,j,k,key;
    char* buffer = (char*) malloc(sizeof(char));
    char* fnumber = (char*) malloc(12 * sizeof(char));
    int *array = (int*) malloc(a_length * sizeof(int));
```

```
clock_t begin = clock();

for(i = 0;i < PROCESS_COUNT;i++)
{
    if(fork() == 0)
    {

        key_t key;
        int msgid;

        // ftok to generate unique key
        key = ftok("progfile", 65);

        // msgget creates a message queue
        // and returns identifier
        msgid = msgget(key, 0666 | IPC_CREAT);
        message.mesg_type = 1;


        fd = open(filenames[i],O_RDWR);
        while(length != 0)
        {
            length = read(fd,buffer,sizeof(char));
            if(a_in == a_length - 2)
            {
                a_length += 10;

            }
            if(strcmp(buffer,"\n") != 0)
            {
                strcat(fnumber,buffer);
            }
            else
            {
                sscanf(fnumber,"%d",array + a_in);
                strcpy(fnumber,"\0");
                a_in++;
            }
        }
        for(j = 0;j < a_in;j++)
        {
            key = array[j];
            k = j - 1;

            while(k >= 0 && key < array[k])
            {
                array[k + 1] = array[k];
                k--;
            }
            array[k + 1] = key;
        }

        wait(NULL);
        sprintf(message.mesg_text,"%d",array[top_k]);

        if(strcmp(message.mesg_text,"0")){
            send_data(msgid);
        }
```

```
            free(buffer);
            free(fnumber);
            free(array);
            close(fd);

            exit(0);
        }
    }

    sleep(3);
    wait(NULL);
    printf("\nPARENT STARTED\n\n");
    receive_data();
    clock_t end = clock();
    // calculate elapsed time by finding difference (end - begin) and
        // dividing the difference by CLOCKS_PER_SEC to convert to seconds
        time_spent += (double)(end - begin) / CLOCKS_PER_SEC;

        printf("\nTime elpased is %f seconds\n", time_spent);


    return 0;
}


void send_data(int msgid){

            // msgsnd to send message
            msgsnd(msgid, &message, sizeof(message), 0);

            // display the message
            printf("Data send is : %s \n", message.mesg_text);

}

int receive_data()
{

    int *parent_array = (int*) malloc(PROCESS_COUNT*sizeof(int));
    key_t key;
    int msgid;

    // ftok to generate unique key
    key = ftok("progfile", 65);

    // msgget creates a message queue
    // and returns identifier
    msgid = msgget(key, 0666 | IPC_CREAT);

    printf("%d Tane data alÄ±nacak\n",PROCESS_COUNT);


//*********** PARENT PROCESS SAYILARI DIZIYE KAYDEDIYOR ************

    for(int i = 0; i<PROCESS_COUNT;i++){
        msgrcv(msgid, &message, sizeof(message), 1, 0);
        // msgrcv to receive message
        // display the message
        printf("Data Received is : %s \n",message.mesg_text);
        parent_array[i] = atoi(message.mesg_text);
    }
```

```
        parent_array[i] = atoi(message.mesg_text);
    }


    printf("Kuyruk destroy ediliyor\n");
    msgctl(msgid, IPC_RMID, NULL);

    sort_and_write(parent_array,PROCESS_COUNT);

    free(parent_array);
    return 0;
}

void sort_and_write(int* arr, int size){


 //*********SELECTION SORT****************

    int startScan, minIndex;
    int minElem;
    for (startScan = 0; startScan < (size - 1); startScan++) {
        minIndex = startScan;
        minElem = arr[startScan];
        for (int index = startScan + 1; index < size; index++) {
            if (arr[index] < minElem) {
                minElem = arr[index];
                minIndex = index;
            }
        }
        arr[minIndex] = arr[startScan];
        arr[startScan] = minElem;
    }
    printf("\nDizi Siralandi\t");
    int outft;
    outft = open(filenames[6],O_CREAT | O_TRUNC | O_RDWR , 0666);
    char buf[12];
    char cbuffer[12];
    int ret =  -1,length2;

    if(outft == -1)
    {
        printf("Failed to create\n");
    }
    else{
        printf("\nCreate output file success\n\n");
        for(int k=0;k<size;k++) {

            strcpy(cbuffer,"\0");
            length2 = sprintf(cbuffer,"%d",arr[k]);
            strcat(cbuffer,"\n");
            ret = write(outft,cbuffer,length2 + 1);

                if (ret < 0)                                //check for erroneous condition
                    printf("ret is %d, errno %d \n", ret,errno);

    }

    printf("\n");
}
}
```

Part 2 ‘ in kodlaması görüldüğü gibidir. Bu kısma verilen 100, 1000 ve 10000 sayı içeren, içerisinde 0-100 arası sayılar doldurduğumuz text dosyalarını çalıştırma, işleme ve sonuca ulaştırma süreleri ise ***saniye*** cinsinden tablomuzda görülmektedir:

N Değeri / Dosyadaki Değer Sayısı	1	2	3	4	5
100	0,000357	0.000414	0.000445	0.000512	0.000534
1000	0,000344	0,000347	0,00041	0,0000463	0,0000543
10K	0.000393	0.000363	0.000457	0.000433	0,000535

Part 2 kısmı test edilirken k değeri 25, 250, 2222 olarak alınmış olup sonuçlar buna göre elde edilmiştir. Part 2 kısmına ait örnek çıktı ise şöyledir:

```
root@kali:~/Desktop/findtopk# ./findtopk_mqueue.out 250 5 1000-1.txt 1000-2.txt 1000-3.txt 1000-4.t
xt 1000-5.txt 1000-output.txt
N = 5
k = 250
Dosyalar :  1000-1.txt  1000-2.txt  1000-3.txt  1000-4.txt  1000-5.txt
OUTPUT FILE : 1000-output.txt

Data send is : 26
Data send is : 26
Data send is : 24
Data send is : 25
Data send is : 25

PARENT STARTED

5 Tane data alınacak
Data Received is : 26
Data Received is : 26
Data Received is : 24
Data Received is : 25
Data Received is : 25
Kuyruk destroy ediliyor

Dizi Sıralandı
Create output file success



Time elpased is 0.000543 seconds
```

Part 3 Kısmının Kodları:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <pthread.h>
#include <time.h>

int k_sayilari[5];

struct t_args {
    char* dosya;
    int indis;
    int k;
};

void *sirala(void *argumanlar){

    int length = 1,a_length = 10,a_in = 0,fd,i,j,k,key;
    char* buffer = (char*) malloc(sizeof(char));
    char* fnumber = (char*) malloc(12 * sizeof(char));
    int *array = (int*) malloc(a_length * sizeof(int));

    fd = open(((struct t_args*)argumanlar)->dosya,O_RDWR);
    while(length != 0)
    {
    length = read(fd,buffer,sizeof(char));
    if(a_in == a_length - 2)
        {
        a_length += 10;
        array = realloc(array,a_length * sizeof(int));
    }
    if(strcmp(buffer,"\n") != 0)
    {
        strcat(fnumber,buffer);
    }
    else
    {
            sscanf(fnumber,"%d",array + a_in);
        strcpy(fnumber,"\0");
        a_in++;
    }
   }

   for(j = 0;j < a_in;j++)
   {
    key = array[j];
    k = j - 1;

    while(k >= 0 && key < array[k])
        {
        array[k + 1] = array[k];
        k--;
    }
    array[k + 1] = key;
   }

   printf("K. SAYI : %d\n",array[((struct t_args*)argumanlar)->k]);
   k_sayilari[((struct t_args*)argumanlar)->indis] = array[((struct t_args*)argumanlar)->k];
   close(fd);
   pthread_exit(NULL);
```

```
        {
        array[k + 1] = array[k];
        k--;
     }
     array[k + 1] = key;
   }

   printf("K. SAYI : %d\n",array[((struct t_args*)argumanlar)->k]);
   k_sayilari[((struct t_args*)argumanlar)->indis] = array[((struct t_args*)argumanlar)->k];
   close(fd);
   pthread_exit(NULL);
}

int main(int argc, char *argv[]){

   if(argc < 4){printf("Yeterli parametre girmediniz!\n");exit(1);}

   int i;
   int k = atoi(argv[1]);
   int n = atoi(argv[2]);

   pthread_t threads[n];
   struct t_args *argumanlar = (struct t_args *)malloc(sizeof(struct t_args));

   clock_t timer = clock();

   for(i = 0;i < n;i++){

      printf("Gonderilen dosya : %s\n",argv[i+3]);
      argumanlar->dosya = argv[i+3];
      argumanlar->indis = i;
      argumanlar->k = k;
      pthread_create(&threads[i],NULL,sirala,(void *)argumanlar);
      pthread_join(threads[i], NULL);
   }

   timer = clock() - timer;
   printf("Threadler bitti! - Gecen zaman : %f\n",((double)timer)/CLOCKS_PER_SEC);

   int fd = open(argv[n+3], O_WRONLY | O_CREAT | O_TRUNC, 0644);
  if (fd < 0)
  {
     perror("r1");
     exit(1);
  }

  char cbuffer[12];
  int lenght;

  for(i = 0;i < n;i++){

     strcpy(cbuffer,"\0");
     lenght = sprintf(cbuffer,"%d",k_sayilari[i]);
     strcat(cbuffer,"\n");
     write(fd,cbuffer,lenght+1);

  }

close(fd);

}
```

Part 3 ‘ in kodlaması görüldüğü gibidir. Bu kısma verilen 100, 1000 ve 10000 sayı içeren, içerisinde 0-100 arası sayılar doldurduğumuz text dosyalarını çalıştırma, işleme ve sonuca ulaştırma süreleri ise ***saniye*** cinsinden tablomuzda görülmektedir:

N Değeri / Dosyadaki Değer Sayısı	1	2	3	4	5
100	0.001407	0.001520	0.004118	0.004449	0.004717
1000	0.007827	0.012528	0.022334	0.025794	0.032842
10K	0.144218	0.317573	0.452676	0.642773	0.705386

Part 3 kısmı test edilirken k değerleri 3794, 4080, 5567 olarak alınmış olup sonuçlar buna göre elde edilmiştir. Part 3 kısmına ait örnek çıktı ise şöyledir:

```
root@ofbahar:~/Masaüstü/Codes/OS/test# ./findtopk_thread 4000 1 1.txt out.txt
Gonderilen dosya : 1.txt
K. SAYI : 4080
Threadler bitti! - Gecen zaman : 0.144218
root@ofbahar:~/Masaüstü/Codes/OS/test# ./findtopk_thread 4000 2 1.txt 2.txt out.txt
Gonderilen dosya : 1.txt
K. SAYI : 4080
Gonderilen dosya : 2.txt
K. SAYI : 3989
Threadler bitti! - Gecen zaman : 0.317573
root@ofbahar:~/Masaüstü/Codes/OS/test# ./findtopk_thread 4000 3 1.txt 2.txt 3.txt out.txt
Gonderilen dosya : 1.txt
K. SAYI : 4080
Gonderilen dosya : 2.txt
K. SAYI : 3989
Gonderilen dosya : 3.txt
K. SAYI : 4114
Threadler bitti! - Gecen zaman : 0.452676
root@ofbahar:~/Masaüstü/Codes/OS/test# ./findtopk_thread 4000 4 1.txt 2.txt 3.txt 4.txt out.txt
Gonderilen dosya : 1.txt
K. SAYI : 4080
Gonderilen dosya : 2.txt
K. SAYI : 3989
Gonderilen dosya : 3.txt
K. SAYI : 4114
Gonderilen dosya : 4.txt
K. SAYI : 4045
Threadler bitti! - Gecen zaman : 0.642773
root@ofbahar:~/Masaüstü/Codes/OS/test# ./findtopk_thread 4000 5 1.txt 2.txt 3.txt 4.txt 5.txt out.txt
Gonderilen dosya : 1.txt
K. SAYI : 4080
Gonderilen dosya : 2.txt
K. SAYI : 3989
Gonderilen dosya : 3.txt
K. SAYI : 4114
Gonderilen dosya : 4.txt
K. SAYI : 4045
Gonderilen dosya : 5.txt
K. SAYI : 3963
Threadler bitti! - Gecen zaman : 0.705386
```

Son olarak K ’yı 50 alarak tüm Partlardaki işlemleri tek bir bilgisayar üzerinde test ederek ortalama süreyi ***saniye*** cinsinden hesapladık. Bunların tabloları ise şekildeki gibidir:

Ortalama Süre / N=1 İçin / Değer Sayısı	Part 1 Ortalama Süre	Part 2 Ortalama Süre	Part 3 Ortalama Süre
100	0.12	0,0012	0.0418
1000	0,17	0.0135	0.342
10K	0,168	0.0318	0.452

Ortalama Süre / N=2 İçin / Değer Sayısı	Part 1 Ortalama Süre	Part 2 Ortalama Süre	Part 3 Ortalama Süre
100	0,187	0,0287	0.0517
1000	0.202	0.0312	0.396
10K	0.254	0.0367	0.492

Ortalama Süre / N=3 İçin / Değer Sayısı	Part 1 Ortalama Süre	Part 2 Ortalama Süre	Part 3 Ortalama Süre
100	0,223	0.0320	0.0628
1000	0,276	0.0428	0.434
10K	0,287	0.0413	0.502

Ortalama Süre / N=4 İçin / Değer Sayısı	Part 1 Ortalama Süre	Part 2 Ortalama Süre	Part 3 Ortalama Süre
100	0,272	0.0410	0.0711
1000	0,362	0.0578	0.517
10K	0,374	0.0673	0.607

Ortalama Süre / N=5 İçin / Değer Sayısı	Part 1 Ortalama Süre	Part 2 Ortalama Süre	Part 3 Ortalama Süre
100	0,414	0.0520	0.0835
1000	0,468	0.0623	0.634
10K	0,480	0.0773	0.702