**ASSIGNMENT 2**

**Q1. A system with two dual-core processors has four processors available for scheduling. A CPU-sensitive application is running on this system. All the input is performed at program setup, when a single file must be opened. Similarly, all output is performed just before the program terminates, when the program results must be returned in a single file between start up and termination, the program is entirely CPU-bound. Your task is to improve the performance of this application by multithreading it. The application runs on the system that uses the one-to–one threading model (Each user thread maps to a kernel thread).**

* **How many threads will you create to perform input and output? Explain.**
* **How many threads will you create for the CPU-intensive portion of the program? Explain.**

In this case, threads are going to perform both input and output operations with concurrency. There should be as many threads as there are blocking system calls, as the threads will be spent blocking. Since, we need to avoid creation of unnecessary threads; so, there should be one thread for input and one for output. Since we have the advantage of four processors, we will have four threads for the CPU intensive portion. If we use any greater than 4, the program won’t run.

**Q2. Consider the following code segment:**

* **pid\_t pid;**

**pid=fork();**

**if(pid==0){**

**fork();**

**thread\_create( . . . . . . . );**

**}**

**fork();**

* **How many unique processes are created?**
* **How many unique threads are created?**

In this case, there are two unique threads and six unique processes.

**Q3. The program shown below uses Pthreads API what will be the output from the program at LINE C and LINE P?**

* **#include<pthread.h>**

**#include<stdio.h>**

**int value=0;**

**void \*runner(void \*param); //the thread**

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

**pid\_t pid;**

**pthread\_t tid;**

**pthrad\_attr\_t attr;**

**pid=fork();**

**if(pid==0){**

**pthread\_attr\_init(&attr);**

**pthread\_create(&tid,&attr,runner,NULL);**

**pthread\_join(tid,NULL);**

**printf(“CHILD: value = %d\n”); //LINE C**

**}**

**else if(pid>0){**

**wait(NULL);**

**printf(“PARENT : value = %d\n”); //LINE P**

**}**

**}**

**void \*runner(void \*param){**

**value=5;**

**pthread\_exit(0);**

**}**

At line C, the output should be ‘5’ and in line P, it should be ‘0’.

**Q4. Write a multithreaded program that calculates various statistical values for a list of numbers. This program will be passed a series of numbers on then command line and will then create three separate worker threads. One thread will determine the average of the numbers, the second will determine the maximum value, and the third will determine the minimum value.**

* #include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<pthread.h>

#include<sys/wait.h>

Int m,\*a;

void \*min(void \*par){

int i,minim;

minim=a[0];

for(i=1;i<m,i++){

if(a[i]<=minim){

minim=a[i];

}

}

printf(“The minimum value is: %d\n”,minim);

}

void \*max(void \*par){

int i,ma;

ma=a[0];

for(i=1;i<m,i++){

if(a[i]<=ma){

ma=a[i];

}

}

printf(“The maximum value is: %d\n”,ma);

}

void \*avg(void \*par){

int i,sum;

float av;

for(i=1;i<m,i++){

sum+=a[i];

}

av=sum/m;

printf(“The average value is: %d\n”,av);

}

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

int i;

pid\_t pid;

pthread\_t tid[3];

pthrad\_attr\_t attr[3];

pid=fork();

if(pid==0){

for(i=0;i<3,i++){

pthread\_attr\_init(&attr[i]);

}

pthread\_create(&tid[0],&attr[0],min,NULL);

pthread\_create(&tid[1],&attr[1],max,NULL);

pthread\_create(&tid[2],&attr[2],avg,NULL);

for(i=0;i<3,i++){

pthread\_join(tid[i],NULL);

}

printf(“Child process has been executed…\n”);

}

else if(pid>0){

wait(NULL);

printf(“Parent process has been executed…\n”);

}

return(0);

}