#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

void \*avg\_func(void \*str);

void \*min\_func(void \*ptr);

void \*max\_func(void \*ptr);

double avg;

int min;

int max;

typedef struct datastruct

{

int size;

int \* values;

}datastruct;

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

{

printf("\n\nWelcome to paheeThredz, by Sean Staz\n\n");

while(argc <=1)

{

printf("Incorrect input. No arguments entered.\n");

printf("Please enter one or more inputs.\n");

exit(0);

}

int i = 0;

int copy[argc-1];

for(i; i < (argc -1); i++)

{

copy[i] = atoi(argv[i+1]);

}

pthread\_t thread1, thread2, thread3;

const char \*message1 = "This is Thread 1";

const char \*message2 = "This is Thread 2";

const char \*message3 = "This is Thread 3";

int t1, t2, t3;

printf("Running: %s\n\n", argv[0]);

datastruct ds = {argc - 1, copy};

/\* Create independent threads each of which will execute appropriate function\*/

t1 = pthread\_create(&thread1, NULL, (void \*) avg\_func, (void \*) &ds);

if(t1)

{

fprintf(stderr,"Error - pthread\_create() return code: %d\n", t1);

exit(EXIT\_FAILURE);

}

t2 = pthread\_create(&thread2, NULL, (void \*) min\_func, (void \*) &ds);

if(t2)

{

fprintf(stderr,"Error - pthread\_create() return code: %d\n",t2);

exit(EXIT\_FAILURE);

}

t3 = pthread\_create(&thread3, NULL, (void \*) max\_func, (void \*) &ds);

if(t3)

{

fprintf(stderr,"Error - pthread\_create() return code: %d\n", t3);

exit(EXIT\_FAILURE);

}

printf("pthread\_create() for Thread 1 returns: %d\n",t1);

printf("pthread\_create() for Thread 2 returns: %d\n",t2);

printf("pthread\_create() for Thread 3 returns: %d\n\n",t3);

/\* Wait till threads are complete before main continues. \*/

pthread\_join(thread1, NULL);

pthread\_join(thread2, NULL);

pthread\_join(thread3, NULL);

printf("The average: %g\n", avg);

printf("The minimum: %d\n", min);

printf("The maximum: %d\n", max);

exit(EXIT\_SUCCESS);

}

void \*avg\_func(void \*ptr)

{

datastruct \* copy;

copy = (datastruct \*) ptr;

int sz = copy->size;

int i;

for(i = 0; i < sz; i++)

{

avg += (copy->values[i]);

} //If I used double for avg it would have given 82.8571 which doesn't match the example output

avg = (int)(avg / sz); //Used cast to match example output given on instructions.

}

void \*min\_func(void \*ptr)

{

datastruct \* copy;

copy = (datastruct \*) ptr;

int sz = copy->size;

int i;

min = (copy->values[0]);

for(i = 1; i < sz; i++)

{

if(min > (copy->values[i]))

{

min = (copy->values[i]);

}

}

}

void \*max\_func(void \*ptr)

{

datastruct \* copy;

copy = (datastruct \*) ptr;

int sz = copy->size;

int i;

max = copy->values[0];

for(i = 1; i < sz; i++)

{

if(max < copy->values[i])

{

max = copy->values[i];

}

}

}

**Complexity of Multithreading:**

The complexity of multithreading is O(3sz).