Manuel Barajas

Part 1:

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

{

int i,sum=0;

if(argc < 3)

{

printf("Not enough numbers to add up!");

exit(1);

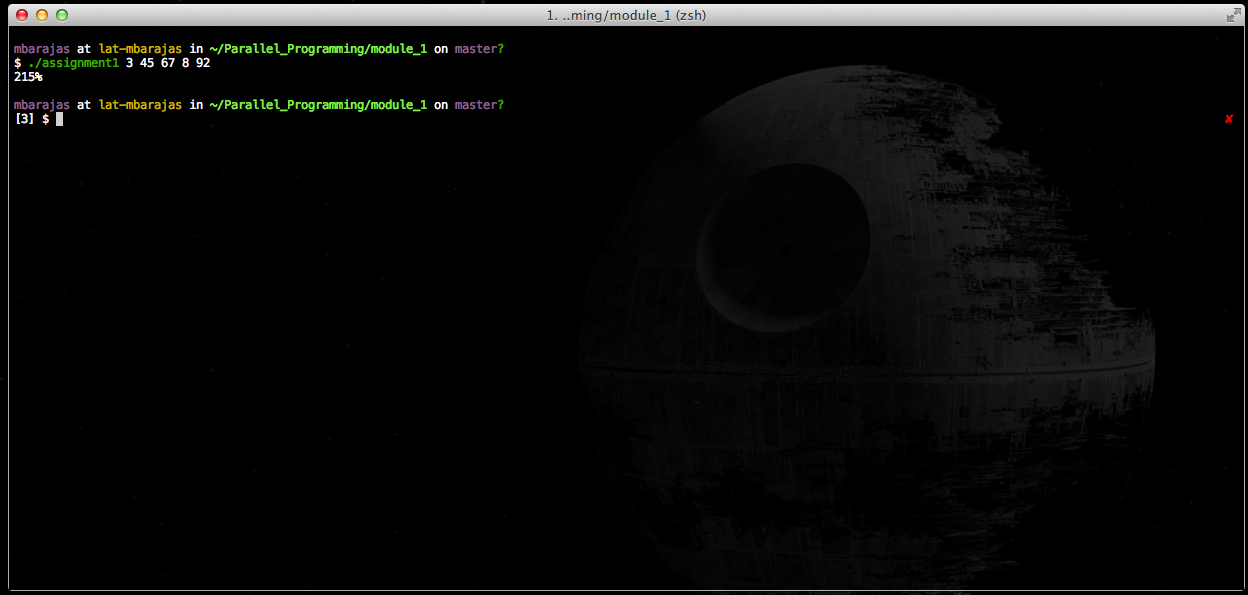
}

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

sum = sum + atoi(argv[i]);

printf("%d",sum);

}



Part 2:

#include <pthread.h>

#define THREADS 2

#define ARRAYSIZE 6

#define ITERATIONS ARRAYSIZE / THREADS

int sum=0;

int numArray[ARRAYSIZE];

pthread\_mutex\_t sum\_mutex;

void \*thread\_sum(void \*tid)

{

int start;

int \*mytid;

int end;

int mysum=0;

mytid = (int \*) tid;

start = (\*mytid \* ITERATIONS);

end = start + ITERATIONS;

int i;

for (i=start; i < end ; i++) {

mysum = mysum + numArray[i];

}

pthread\_mutex\_lock (&sum\_mutex);

sum = sum + mysum;

pthread\_mutex\_unlock (&sum\_mutex);

pthread\_exit(NULL);

}

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

{

int i;

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

numArray[i] = atoi(argv[i]);

}

pthread\_t threads[THREADS];

pthread\_attr\_t attr;

pthread\_mutex\_init(&sum\_mutex, NULL);

pthread\_attr\_init(&attr);

pthread\_attr\_setdetachstate(&attr, PTHREAD\_CREATE\_JOINABLE);

int tids[THREADS];

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

tids[i] = i;

pthread\_create(&threads[i], &attr, thread\_sum, (void \*) &tids[i]);

}

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

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

}

printf ("%d", sum);

pthread\_attr\_destroy(&attr);

pthread\_mutex\_destroy(&sum\_mutex);

pthread\_exit (NULL);

}

