Thread

Thread Creation

```c
#include <pthread.h>
#include<unistd.h>
#include<stdio.h>
int  i,j;
void *thread_function(void *arg);
int main(){
pthread_t a_thread; //thread declaration
pthread_create(&a_thread, NULL, thread_function, NULL); //thread
pthread_join(a_thread, NULL); //process wait for thread to finish
printf("Inside Main Program\n");
for(i=44;i<48;i++)
{
printf("%d\n",i);
}
}

void *thread_function(void *arg){
printf("inside thread\n");
for(j=0; j<5; j++){
printf("%d\n",j);
}
}
```

Display Thread ID and Pass the argument

```
  GNU nano 6.2
#include <pthread.h>
#include<unistd.h>
#include<stdio.h>
int  i,j;
void *thread_function(void *arg);
int main(){
pthread_t a_thread; //thread declaration
pthread_create(&a_thread, NULL, thread_function, "Thread 1"); //thread created
pthread_join(a_thread, NULL); //process wait for thread to finish
printf("Inside Main Program\n");

}

void *thread_function(void *arg){
printf("inside thread\n");

printf("Thread ID: %ld\t Thread_argument=%s\n",pthread_self(),(char*)arg);

}
```

Race Condition

Demonstration of Race Condition

Two threads trying to inc/dec counter value

```
  GNU nano 6.2
#include <pthread.h>
#include<unistd.h>
#include<stdio.h>
int  i,j;
void *fun1(void *arg);
void *fun2(void *arg);
int counter=1;
int main(){
pthread_t a_thread, b_thread; //thread declaration
pthread_create(&a_thread, NULL, fun1, NULL); //thread 1 created
pthread_create(&b_thread, NULL, fun2, NULL); //thread 2 created
pthread_join(a_thread, NULL); //process wait for thread to finish
pthread_join(b_thread, NULL);
printf("Inside Main Program\n");
printf("\n%d",counter);
}

void *fun1(void *arg){
int x=counter;
x++;
sleep(1);
counter=x;

}
void *fun2(void *arg){
int y=counter;
y--;
sleep(1);
counter=y;

}
```

Process Synchronization

Mutex to prevent Race Condition

```
  GNU nano 6.2
#include<pthread.h>
#include<stdio.h>
#include<unistd.h>
void *fun1();
void *fun2();
int count=1;
pthread_mutex_t l;
int main(){
pthread_mutex_init(&l, NULL);
pthread_t thread1, thread2;
pthread_create(&thread1,NULL, fun1, NULL);
pthread_create(&thread2,NULL, fun2, NULL);
pthread_join(thread1,NULL);
pthread_join(thread2,NULL);
pthread_join(thread2,NULL);
printf("Count: %d",count);
}

void *fun1(){
int x;
printf("Thread 1 acquiring lock\n");
pthread_mutex_lock(&l);
printf("Thread 1 acquired the lock\n");
x=count;
x++;
sleep(1);
count=x;
pthread_mutex_unlock(&l);
printf("Thread 1 released the lock\n");
}

void *fun2(){
int y;
printf("Thread 2 acquiring lock\n");
pthread_mutex_lock(&l);
printf("Thread 2 acquired the lock\n");
y=count;
y--;
sleep(1);
count=y;
pthread_mutex_unlock(&l);
printf("Thread 2 released the lock\n");
}
```