Creation Of Threads

B.Ajay
21110103

Aim:

To Create Threads Using the pthreads library in C

Algorithm:

1.Create a Runner Function that the threads run

2. Create Threads using pthread_create and pass the required parameters

3.Incase multiple threads are used use mutex lock for synchronisation

4. Multiple threads can be created by using thread_id in a array

5. Once the thread runs other threads are joined using pthread_join

6. Destroy mutex lock and clean up after thread finished execution

Description:

1.pthread_create - create a new thread

Syntax:

```
#include <pthread.h>

int pthread_create(pthread_t *restrict thread,
const pthread_attr_t *restrict attr,
void *(*start_routine)(void *),
void *restrict arg);
```

Description:

The **pthread_create**() function starts a new thread in the calling

```
process.  The new thread starts execution by invoking
start_routine(); arg is passed as the sole argument of
start_routine().
```

2.pthread_join:

Syntax:

```
pthread_join - join with a terminated thread

#include <pthread.h>


int pthread_join(pthread_t thread, void **retval);
```

Description:

The **pthread_join**() function waits for the thread specified by

thread to terminate. If that thread has already terminated, then **pthread_join**() returns immediately. The thread specified by *thread* must be joinable.

3. pthread_attr

Syntax:

#include <pthread.h>

```
int pthread_attr_init(pthread_attr_t *attr);
int pthread_attr_destroy(pthread_attr_t *attr);
```

Description:

The **pthread_attr_init**() function initializes the thread attributes object pointed to by *attr* with default attribute values. After this call, individual attributes of the object can be set using various related functions (listed under SEE ALSO), and then the object can be used in one or more pthread_create(3) calls that create threads.

4.pthread_exit:

Syntax:

#include <pthread.h>

```
noreturn void pthread_exit(void *retval);
```

Description:

The **pthread_exit**() function terminates the calling thread and returns a value via *retval* that (if the thread is joinable) is available to another thread in the same process that calls pthread_join(3).

5.pthread_mutex_:

Syntax:

#include <pthread.h>

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex)
```

Description:

The mutex object referenced by *mutex* shall be locked by a call to *pthread_mutex_lock*() that returns zero or **[EOWNERDEAD]**. If the mutex is already locked by another thread, the calling thread shall block until the mutex becomes available. This operation shall return with the mutex object referenced by *mutex* in the locked state with the calling thread as its owner.

Creation Of Threads

B.Ajay
21110103

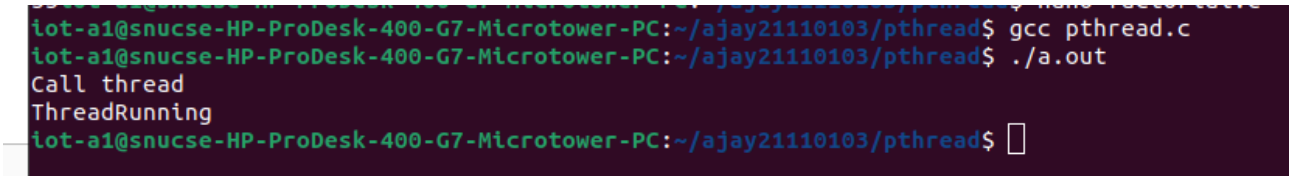
Code:

pthread.c

```
#include <stdio.h>
#include <pthread.h>
void * runner(void *arg){
	printf("ThreadRunning\n");
	return NULL;
}

int main(){
	pthread_t t1;
	printf("Call thread\n");
	pthread_create(&t1,NULL,runner,NULL);
	pthread_join(t1,NULL);
	return 0;
}
```

Output:



Summation Using Mutex Locks:

Summation.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
int s;
#define N 5
pthread_t n[N];
pthread_mutex_t msum;
int a[10];
int sum=0;
void* summation(void* param){
	long tid=(long)param;
	int s,e,tsum;
	s=(int)tid*2;
	e=s+2;
	for(int i=s;i<e;i++){
		tsum+=a[i];
	}
	pthread_mutex_lock(&msum);
```

```
        sum+=tsum;
        pthread_mutex_unlock(&msum);
        pthread_exit(0);

}
void main(){
        //a={1,2,3,4,5,6,7,8,9,10};
        for(int i=1;i<=10;i++){
                a[i-1]=i;
        }
        pthread_attr_t attr;
        pthread_mutex_init(&msum,NULL);
        pthread_attr_init(&attr);
        pthread_attr_setdetachstate(&attr,PTHREAD_CREATE_JOINABLE);
        for(int i=0;i<N;i++){
                pthread_create(&n[i],&attr,summation,(void*)i);
        }
        pthread_attr_destroy(&attr);
        for(int i=0;i<N;i++){
                pthread_join(n[i],NULL);
        }
        printf("%d",sum);
        pthread_mutex_destroy(&msum);
        pthread_exit(0);

}
```

Output:

```
iot-a1@snucse-HP-ProDesk-400-G7-Microtower-PC:~/ajay21110103/pthread$ gcc -w summation.c
iot-a1@snucse-HP-ProDesk-400-G7-Microtower-PC:~/ajay21110103/pthread$ ./a.out
55iot-a1@snucse-HP-ProDesk-400-G7-Microtower-PC:~/ajay21110103/pthread$
```

Result:

Thus Threads were created using pthreads library and their usecases observed.