**Linux Internals**
**Assignment 3**
**By: Pavan Hegde**

**Program 1**
**Write a multithreading program, where threads runs same shared golbal variable of the process between them.**

**Code:**

```
#include<pthread.h>
#include<stdio.h>

pthread_t thid1,thid2;
int var;

void *thread1(void *arg)
{
	printf("created thread 1 is executing\n");
	//printf("Assign value to global variable\n");
	scanf("%d",&var);
	printf("Assigning value to global variable var= %d\n",var);
	return NULL;
}

void *thread2(void *arg)
{
	printf("created thread 2 is executing\n");
	//printf("Assign value to global variable\n");
	scanf("%d",&var);
	printf("Assigning value to global variable var= %d\n",var);
	return NULL;
}

int main(void)
{
	int ret=pthread_create(&thid1,NULL,thread1,NULL);
	int ret1=pthread_create(&thid2,NULL,thread2,NULL);
	if(ret)
		printf("thread 1 is not created\n");
	else if(ret1)
		printf("thread 2 is not createdn");
	else
	{
		printf("threads are created\n");

		pthread_join(thid2,NULL);
		pthread_join(thid1,NULL);
	}
	return 0;
}
```

**Output:**

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ gedit
prog1.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ gcc -o
prog1 prog1.c -lpthread
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$
./prog1
threads are created
created thread 2 is executing
created thread 1 is executing
5
Assigning value to global variable var= 5
7
Assigning value to global variable var= 7
```

**Program 2**
**Write a program where thread cancel itself.(use pthread_cancel())**

**Code:**

```c
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<unistd.h>

pthread_t thread;

void *threadfunc(void *threadid)
{
	printf("\nHello World\n");
	while(1);
}

int main()
{
	pthread_t thread;
	int t=0;
	printf("creating thread\n");
	pthread_create(&thread,NULL,threadfunc,NULL);
	printf("\n thread id : %lu",thread);
	sleep(5);
	t=pthread_cancel(thread);
	if(t==0)
		printf("\n cancel thread\n");
	printf("\n thread id : %lu\n",thread);
	return 0;
}
```

**Output:**

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ gedit prog2.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ gcc -o prog2 prog2.c -lpthread
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ ./prog2
creating thread

 thread id : 140537623365376
Hello World

 cancel thread

 thread id : 140537623365376
```

**Program 3**
**Write a program that changes the default properties of newly created posix threads.(ex: to change default pthread stack size )**

**Code:**

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<unistd.h>
#include<string.h>

void *proc(void *param)
{
	sleep(2);
	return 0;
}

int main()
{
	pthread_attr_t Attr;
	pthread_attr_t Attr1;
	pthread_t id1;
	pthread_t id2;
	void *stk;
	size_t siz;
	int err;
	size_t my_stksize=300;
	void *my_stack;
	pthread_attr_init(&Attr);
	pthread_attr_init(&Attr1);
	pthread_create(&id2,&Attr1,proc,NULL);

	pthread_attr_getstack(&Attr1,&stk,&siz);

	printf("Default: Addr=%08x Default size=%d\n",stk,siz);
	my_stack=(void *)malloc(my_stksize);

	//printf("Malloc check: Addr=%08x Default size=%d\n",my_stack,my_stksize);

	pthread_attr_setstack(&Attr,my_stack,my_stksize);

	pthread_create(&id1,&Attr,proc,NULL);
	pthread_attr_getstack(&Attr,&stk,&siz);
	printf("newly defined stack : Addr=%08x and Size=%d\n",my_stack,my_stksize);
	sleep(3);
	return 0;
}
```

**Output:**

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ gedit
prog3.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ gcc -o
prog3 prog3.c -lpthread
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$
./prog3
Default: Addr=00000000 Default size=0
newly defined stack : Addr=ea8467d0 and Size=300
```

**Program 4**
**Write a program where pthread task displays the thread id and also prints the calling process pid.**

**Code:**

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>

pthread_t tid;

static void *threadfunc(void *arg)
{
	pid_t pid;
	pthread_t tid;
	pid=getpid();
	tid=pthread_self();
	printf("pid : %u\n tid : %u\n",(unsigned int)pid,(unsigned int)tid);
	return 0;
}

int main(void)
```

```
{
        int err;
        err=pthread_create(&tid,NULL,threadfunc,NULL);
        if(err!=0)
                printf("can't create thread : %d\n", strerror(err));
        while(1);
        pthread_join(tid,NULL);
        exit(0);
        return 0;
}
```

**Output:**

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ gedit prog4.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ gcc -o prog4 prog4.c -lpthread
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ ./prog4
pid : 3445
 tid : 1538909952
```

**Program 5**
**Write a program that implements threads synchronization using mutex techniques.**

**Code:**

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<unistd.h>
#include<string.h>
#include<semaphore.h>

int sharedvar=5;

pthread_mutex_t my_mutex;

void *thread_inc(void *arg)
{
        pthread_mutex_lock(&my_mutex);
        sharedvar++;
        printf("after increment = %d\n",sharedvar);
        pthread_mutex_unlock(&my_mutex);
}

void *thread_dec(void *arg)
{
        pthread_mutex_lock(&my_mutex);
        sharedvar--;
        printf("after decrement = %d\n",sharedvar);
        pthread_mutex_unlock(&my_mutex);
}
int main()
{
	pthread_t thread1,thread2;
	pthread_mutex_init(&my_mutex,NULL);
	//static int x=10;

	pthread_create(&thread1,NULL,thread_inc,NULL);
	pthread_create(&thread2,NULL,thread_dec,NULL);

	pthread_join(thread1,NULL);
	pthread_join(thread2,NULL);

	printf("sharedvar = %d\n",sharedvar);
	return 0;
}
```

**Output:**

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ gedit
prog5.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$ gcc -o
prog5 prog5.c -lpthread
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_3_Multithreading$
./prog5
after decrement = 100
after increment = 101
sharedvar = 101
```