Linux Internals
Assignment 4
By: Pavan Hegde

Program 1
Write a pthread application where main task terminated but pending pthreads task still execute.

Code:

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

static void *threadfunc(void *arg)
{
        printf("Thread Executing after main() is terminated\n");
        char *s=(char *)arg;
        printf("%s\n",s);
        sleep(5);
        printf("Thread Executed after main() is terminated\n");
        return (void *)strlen(s);
        //return 0;
}

int main(int argc,char *argv[])
{
        pthread_t t1;
        void *res;
        int s;
        s=pthread_create(&t1,NULL,threadfunc,"Hello World");
        printf("Executing main() Function\n");
        sleep(3);
        //pthread_join(t1,&res);
```

```
	pthread_exit(NULL);
	//printf("Thread Executed after main() is terminated");
	exit(0);
	return 0;
}
```

Output:

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ gedit prog1.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ gcc -o prog1 prog1.c -lpthread
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ ./prog1
Executing main() Function
Thread Executing after main() is terminated
Hello World
Thread Executed after main() is terminated
```

Program 2

Write a program where a structure of information passed to pthread task function, and display structure of information.

Code:

```
#include<stdio.h>
#include<pthread.h>
#include<stdlib.h>
#include<string.h>
struct my_thread
{
	int thread_id;
	char msg[50];
};
void *printmsg(void *threadobj)
{
	pthread_t thread_ID;
	struct my_thread *t1;
```

```
        t1=(struct my_thread*) threadobj;
        thread_ID = pthread_self();
        printf("\n thread ID : %u",thread_ID);
        printf("\nThis is Thread no.-%d & Showing Thread message : %s \n",t1->thread_id,t1->msg);
}
int main()
{
        pthread_t thread2,thread3,thread4,thread_ID;
        //int rc;
        struct my_thread t2,t3,t4;
        t4.thread_id=4;
        strcpy(t4.msg,"....Fourth thread....\n");
        t2.thread_id=2;
      strcpy(t2.msg,".....Second thread....\n");
        t3.thread_id=3;
      strcpy(t3.msg,"....Third thread....\n");
        thread_ID=pthread_self();
        printf("\n Main thread (First Thread) ID : %u\n",thread_ID);
        pthread_create(&thread4,NULL,printmsg,(void *)&t4);
        pthread_create(&thread2,NULL,printmsg,(void *)&t2);
        pthread_create(&thread3,NULL,printmsg,(void *)&t3);
        //printf("\n EXIT THREAD\n");
        printf("\n");
        pthread_exit(NULL);
}
```

Output:

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ gedit prog2.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ gcc -o prog2 prog2.c -lpthread
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ ./prog2
```

Main thread (First Thread) ID : 1035106112

thread ID : 1018316544

This is Thread no.-3 & Showing Thread message : ....Third thread....

thread ID : 1026709248

This is Thread no.-2 & Showing Thread message : .....Second thread....

thread ID : 1035101952

This is Thread no.-4 & Showing Thread message : ....Fourth thread....

Program 3

Write a pthread program that implements simple initialization code.

Code:

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

pthread_once_t once= PTHREAD_ONCE_INIT;

void *myinit()
{
        printf("\n Init function\n");
        printf("\n Init function will run only once because of pthread_once \n");
}
```

```
void *mythread(void *i)
{
        printf("\n Geting in to thread : %d\n",(int *)i);
        pthread_once(&once,(void *)myinit);
        sleep(2);
        printf("\n Exiting from thread: %d\n",(int *)i);
}
int main()
{
        pthread_t thread,thread1,thread2;
        pthread_create(&thread,NULL,mythread,(void *)1);
        sleep(2);
        pthread_create(&thread1,NULL,mythread,(void *)2);
        sleep(2);
        pthread_create(&thread2,NULL,mythread,(void *)3);
        printf("\n Exiting for Main Thread\n");
        pthread_exit(NULL);
        return 0;
}
```

Output:

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ gedit prog3.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ gcc -o prog3 prog3.c -lpthread
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ ./prog3

 Geting in to thread : 1

 Init function

 Init function will run only once because of pthread_once
```

Geting in to thread : 2

Exiting from thread: 1

Exiting for Main Thread

Exiting from thread: 2

Geting in to thread : 3

Exiting from thread: 3

Program 4

write a program, which get and set pthread scheduling policy and priority.

Code:

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

int main()
{
        pthread_t tid;
        struct sched_param param;
        int priority,policy,result;

        //scheduling parameters of target thread
        result=pthread_getschedparam(pthread_self(),&policy,&param);
        printf("Getting policy & priority before setting it\n\n");
        if(result!=0)
                perror("error in getschedparam");
```

```
	printf("\n-----------------Main thread--------------\n policy : %d \t priority : %d\
n",policy,param.sched_priority);
	printf("Setting policy & priority \n\n");
	policy=SCHED_FIFO;
	param.sched_priority=15;
	result=pthread_setschedparam(pthread_self(),policy,&param);
	if(result!=0)
		perror("error in setschedparam");



	printf("Getting policy & priority after setting it\n\n");
	result=pthread_getschedparam(pthread_self(),&policy,&param);
	if(result!=0)
		perror("error in getschedparam");
	printf("\n-----------------Main thread--------------\n policy : %d \t priority : %d\
n",policy,param.sched_priority);


	return 0;
}
```

Output:

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ gedit
prog4.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ gcc -o
prog4 prog4.c -lpthread
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$
sudo ./prog4
Getting policy & priority before setting it



-----------------Main thread--------------
 policy : 0        priority : 0
Setting policy & priority
```

Getting policy & priority after setting it

----------Main thread----------

policy : 1 priority : 15

Program 5

Write a program that implements threads synchronization using pthread spinlock techniques.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<unistd.h>
#include<string.h>
#include<errno.h>
#include<bits/types.h>
#include<sys/types.h>
pthread_spinlock_t spinlock;
volatile int lock;

void *spinlockthread(void *i)
{
        int p=0;
        int count=0;
        printf("Entered in thread %d, implementing spin-lock \n",(int *)i);
        p=pthread_spin_lock(&lock);
        printf("Thread %d unlocked the spin-lock\n",(int *)i);
        return NULL;
}

int main()
{
```

```
	int p=0;
	pthread_t thread1,thread2;
	if(pthread_spin_init(&lock,PTHREAD_PROCESS_PRIVATE)!=0)
		perror("init");
	printf("main, implement spin-lock to thread\n");
	p=pthread_spin_lock(&lock);

	printf("Main, implementing the spin-lock thread\n");

	p=pthread_create(&thread1,NULL,spinlockthread,(int*)1);

	//printf("Main, wait a bit the spin lock\n");
	sleep(5);
	p=pthread_spin_unlock(&lock);

	if(p==0)
		printf("\n Main thread successfully unlocked\n");
	else
		printf("\n Main thread unsuccessfully unlocked\n");

	printf("I am in Main, waiting for the thread to end\n");
	p=pthread_join(thread1,NULL);
	p=pthread_spin_destroy(&lock);
	return 0;
}
```

Output:

```
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ gedit prog5.c
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ gcc -o prog5 prog5.c -lpthread
pavan@pavan-VirtualBox:~/Training/Linux_internals_tools/Assignment_4_Multithreading$ ./prog5
main, implement spin-lock to thread
Main, implementing the spin-lock thread
Entered in thread 1, implementing spin-lock
```

Main thread successfully unlocked
I am in Main, waiting for the thread to end
Thread 1 unlocked the spin-lock