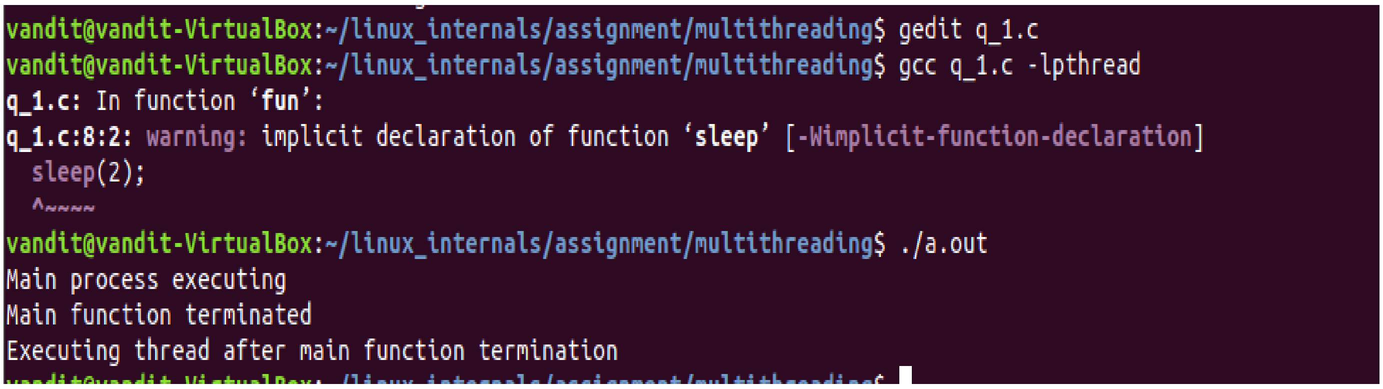
LINUX INTERNALS
MULTITHREADING ASSIGNMENT

Vandit Prajapati (150187)

Q-1) Write a pthread application where main task terminated but pending pthreads task still execute.

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

void *fun()
{
	printf("Executing thread after main function termination\n");
	sleep(2);
	return 0;
}

int main()
{
	//Creating a thread object
	pthread_t tid;

	//Creating thread
	pthread_create(&tid,0,fun,0);

	printf("Main process executing\n");

	printf("Main function terminated\n");

	//Pending thread executing
	pthread_exit(NULL);
	return 0;
}
```

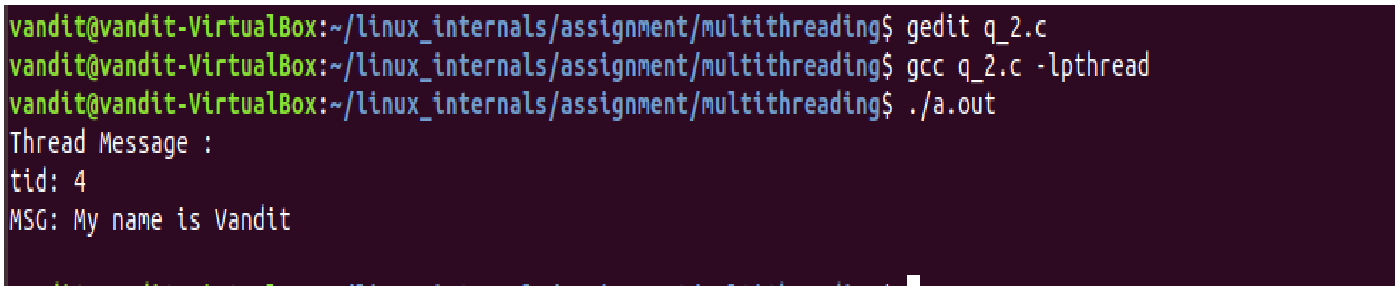


```
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ gedit q_1.c
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ gcc q_1.c -lpthread
q_1.c: In function ‘fun’:
q_1.c:8:2: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
  sleep(2);
  ^~~~~
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ ./a.out
Main process executing
Main function terminated
Executing thread after main function termination
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$
```

Q-2) Write a program where a structure of information passed to pthread task function, and display structure of information.

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

struct information
{
        int tid;
        char a[100];
};

void *fun(void *inf)
{
        struct information *i;
        i=(struct information *)inf;
        printf("Thread Message :\ntid: %d\nMSG: %s\n",i->tid,i->a);
}

int main()
{
        pthread_t t1,tid;
        int rc;
        struct information inf;

        inf.tid=4;
        strcpy(inf.a,"My name is Vandit\n");
```

```
        //Passing structure object as argument
        pthread_create(&t1,NULL,fun,(void *)&inf);

        pthread_exit(NULL);
        printf("Exit Main Thread \n");
}
```

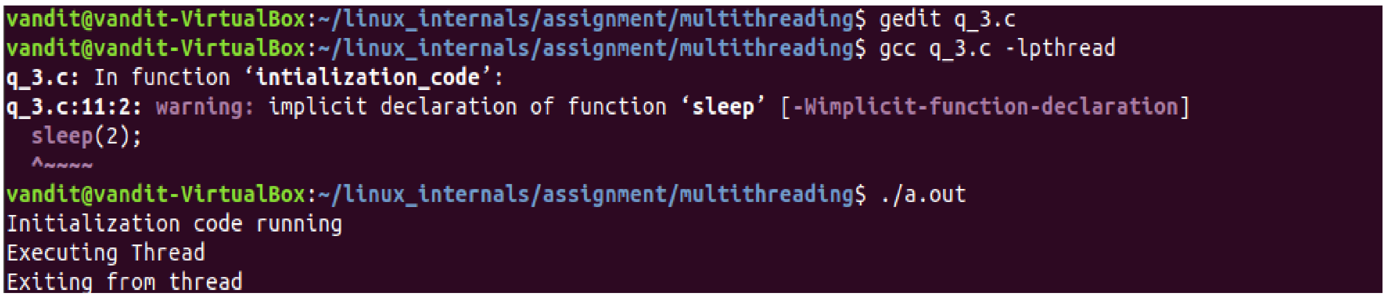


```
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ gedit q_2.c
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ gcc q_2.c -lpthread
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ ./a.out
Thread Message :
tid: 4
MSG: My name is Vandit
```

Q-3) Write a pthread program that implements simple initialization code.

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

//For running initialization code
pthread_once_t once=PTHREAD_ONCE_INIT;

//Initialization code
void *intialization_code()
{
        printf("Initialization code running\n");
        sleep(2);
}

//Thread Function
void *fun()
{
        pthread_once(&once,(void *)intialization_code);
        printf("Executing Thread\n");
        sleep(2);
        printf("Exiting from thread\n");
}
```

```
int main()
{
        pthread_t t;

        t=pthread_create(&t,NULL,fun,NULL);
        pthread_exit(NULL);
        printf("Exiting from main program\n");
}
```

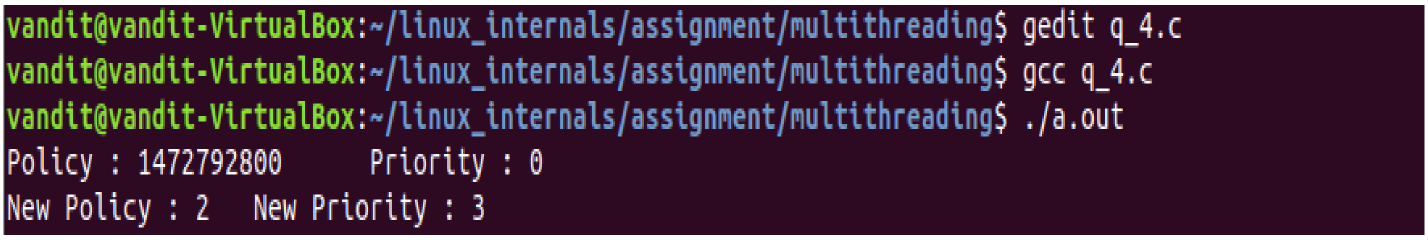


```
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ gedit q_3.c
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ gcc q_3.c -lpthread
q_3.c: In function ‘intialization_code’:
q_3.c:11:2: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
  sleep(2);
  ^~~~~
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ ./a.out
Initialization code running
Executing Thread
Exiting from thread
```

Q-4) Write a program, which get and set pthread scheduling policy and priority.

```
#include<stdio.h>
#include<pthread.h>
#include<sys/types.h>
#include<unistd.h>

void *fun()
{
        printf("Thread executing\n");
}

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

        //getting the scheduling policy and parameter
        ret=pthread_getschedparam(pthread_self(),&policy,&param);
```

```
        printf("Policy : %d\t Priority : %d\n",policy,param.sched_priority);

        //setting the Round Robin policy
        policy=SCHED_RR;
        //setting the priority as 3
        param.sched_priority=3;s
        //Setting the priority and policy
        ret=pthread_setschedparam(pthread_self(),policy,&param);

        //Getting the new values of priority and policy
        ret=pthread_getschedparam(pthread_self(),&policy,&param);
        printf("New Policy : %d\t New Priority : %d\
n",policy,param.sched_priority);

        return 0;
}
```

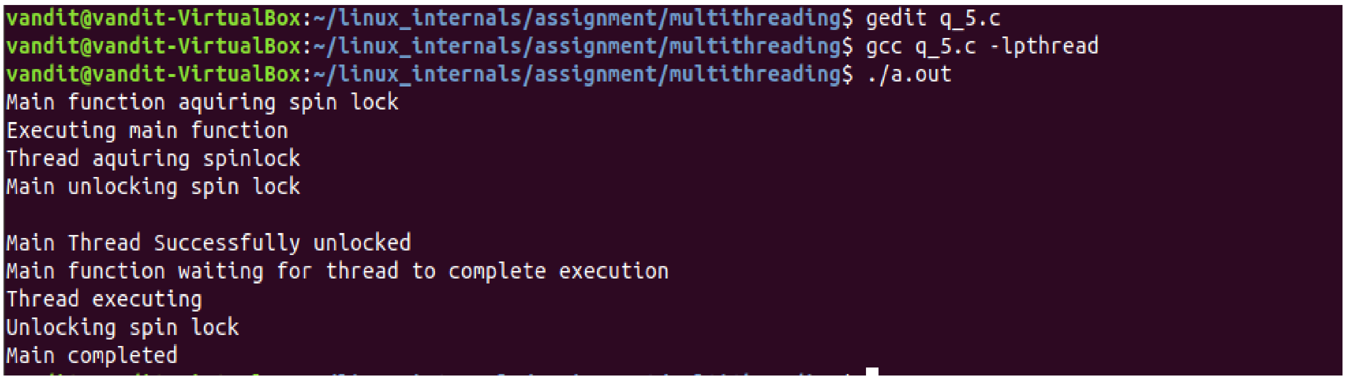


```
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ gedit q_4.c
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ gcc q_4.c
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ ./a.out
Policy : 1472792800      Priority : 0
New Policy : 2   New Priority : 3
```

Q-5) Write a program that implements threads synchronization using pthread spinlock techniques.

```
#include<stdio.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
#include<errno.h>
#include<bits/types.h>
#include<sys/types.h>

static pthread_spinlock_t spinlock;
volatile int slock;

void *spinlockthread(void *i)
{
	int rc;
	int count=0;
	printf("Thread aquiring spinlock\n");
	rc=pthread_spin_lock(&slock);
	printf("Thread executing\n");
	printf("Unlocking spin lock\n");
	rc=pthread_spin_unlock(&slock);
	return NULL;
}

int main()
{
	int rc=0;
	pthread_t thread;

	if(pthread_spin_init(&slock,PTHREAD_PROCESS_PRIVATE)!=0)
		perror("init");

	printf("Main function aquiring spin lock\n");
	rc=pthread_spin_lock(&slock);

	rc=pthread_create(&thread,NULL,spinlockthread,(int *)1);

	printf("Executing main function\n");
	sleep(5);

	printf("Main unlocking spin lock\n");
	rc=pthread_spin_unlock(&slock);

	if(rc==0)
		printf("\nMain Thread Successfully unlocked \n");
	else
		printf("\nMain Thread Successdully unlocked \n");


	printf("Main function waiting for thread to complete execution\n");
	rc= pthread_join(thread,NULL);
	printf("Main completed \n");
	return 0;
```

```
}
```



```
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ gedit q_5.c
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ gcc q_5.c -lpthread
vandit@vandit-VirtualBox:~/linux_internals/assignment/multithreading$ ./a.out
Main function aquiring spin lock
Executing main function
Thread aquiring spinlock
Main unlocking spin lock

Main Thread Successfully unlocked
Main function waiting for thread to complete execution
Thread executing
Unlocking spin lock
Main completed
```