Project Report:

Group members:

Sushmita Nikose
153050046
Achala Bhati
153050056

Project Title: Extending the Kernel API for creation and correct management of user-level threads.

Project Description:

Extending the kernel API for creation and correct management of user-level threads. Using this API, a user level program must be able to create, terminate one or more threads. Furthermore a user level program will be able to specify different scheduling policies for a group of threads. The API will also export basic primitives for safe inter-thread communication.

Approach :

For thread creation we have created a system call create, in which we are passing no of threads to create. In order to create each thread we use sys_clone function in our system call. Also we have passed the user level functions we created, to the system call, which will be executed by new created threads.

Then we have done thread scheduling using sched_set_scheduler, then we have passed scheduling policies and priority. We have executed threads on different scheduling policies and different priorities.

After that in order to do communication between threads we have use shared memory. We created shared memory using shmget() that is shared between the created threads. Different thread can read and write in this shared memory. So as all threads can try to read or write at the same time, for this purpose we need some synchonization method to provide safe communication between the threads. In order provide synchronization we have used semaphores, that will provide locks on conflicting operation.

Now at the end we make a system call to terminate a no of threads, this system call has parameters i.e. no of threads to terminate and a thread array that will keep track which threads are alive. Now in order to kill individual thread we use sys_kill method of the kernel library.

Design and Implementation :

Thread Creation : for thread creation we have implemented the system call sys_create.

*asmlinkage long sys_example(int sch,int no,void (*fn)(void),char ***child_stack,int flags,int shmid,int tid[])*

fn : We are passing the fn pointer of user space program , that new thread will execute and to execute thread we will require a stack , so before calling system call for the creation of the thread we will create our own execution stack, and we will pass it to system call. In top of stack first we have pushed the thread function so that thread can find function to execute.Also we will pass various flag that will tell resources shared between the processes, so we have passed following flag to create thread (Flags are used to differentiate between process and thread):

flags :
CLONE_SIGHAND: Shares the tables that identify the signal handlers and the blocked and pending signals.
CLONE_FS :Shares the table that identifies the root directory and the current working directory, as well as the value of the bitmask used to mask the initial file permissions of a new file (the so-called file umask).
CLONE_VM :Shares the memory descriptor and all Page Tables.
CLONE_FILES :Shares the table that identifies the open files.
CLONE_THREAD :Inserts the child into the same thread group of the parent, and forces the child to share the signal descriptor of the parent.

no : no of threads to create
In order to create thread in our system call we have used sys_clone method defined in kernel library.

sys_clone(flags,(child_stack+i)+4096-1,NULL,NULL,fn);*

In this we have passed stack that we have created in user space and also we have passed clone flags.

Thread Scheduling :
After thread creation we are giving it different scheduling policy and priority.

sched_setscheduler(tsk,sch,a)

tsk : task_struct of created thread.
Sch : variable that tell which scheduling policy to apply. It can take different values for different scheduling class.
SCHED_NORMAL : 0
SCHED_FIFO : 1
SCHED_RR : 2
a : It is pointer to structure sched_param, we use to provide different priorities :
a-> sched_priority=// can give priority

Thread Communication :
For thread communication we have used shared memory . We have created shared memory using shmget() function call. This call will return shmid(shared memory id). We created this memory area in parent process.

shmget(IPC_PRIVATE,SHM_SIZE,IPC_CREAT|IPC_EXCL|S_IRUSR|S_IWUSR);

IPC_PRIVATE : tells we have created new memory area to be shared.
SHM_SIZE : size of memory to be shared.
IPC_CREAT : Create a new segment. If this flag is not used, then **shmget**() will find the segment associated with *key* and check to see if the user has permission to access thesegment.
IPC_EXCL : This flag is used with **IPC_CREAT** to ensure that this call creates the segment. If the segment already exists, the call fails.
IRUSR|S_IWUSR : read and write permission.
Then we have used shmat(shmid,NULL,0) function to attach the shared memory to the threads , so that they can use it to communicate.

Synchronization : In order to provide safe communication we have implemented the semaphore , for this we have used

semget(250,1,IPC_CREAT|0600);

The **semget**() system call returns the System V semaphore set identifier associated with the argument *key*.

sem_lock(int sem_set_id)
sem_unlock(int sem_set_id)

these function is used to synchronize communication of the shared memory.

Thread Termination :
For exit we implement system call to exit

asmlinkage long sys_killing(int tid[],int n)

we have passed no of threads. Inside this system call we have used sys_kill in order to terminate the thread.

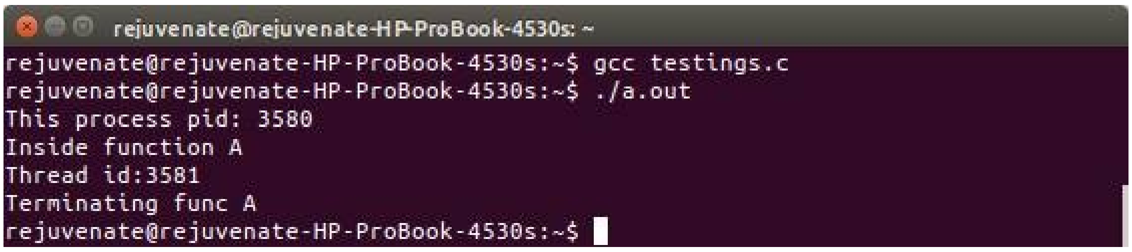
sys_kill(tid[i],SIGTERM)

In this we pass thread id of the thread that we want to terminate.
we will update the array that have no of threads alive in our system call.

Experiments :

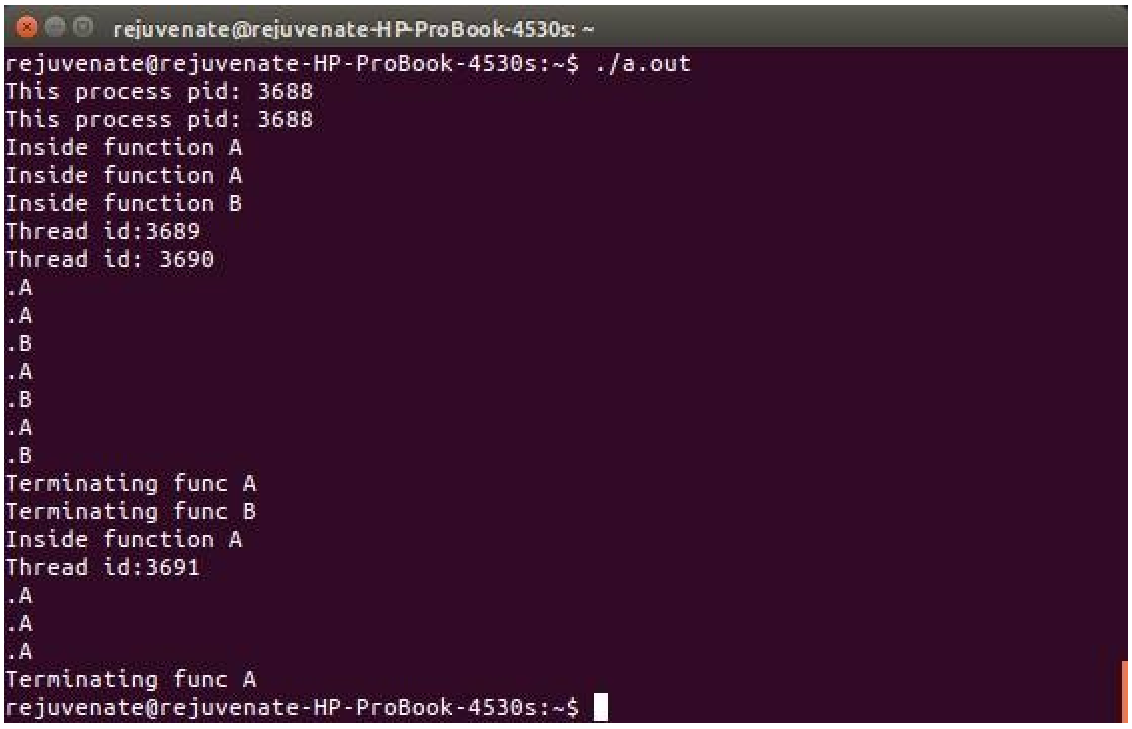
Thread Creation :

Single Thread Creation :



```
rejuvenate@rejuvenate-HP-ProBook-4530s:~$ gcc testings.c
rejuvenate@rejuvenate-HP-ProBook-4530s:~$ ./a.out
This process pid: 3580
Inside function A
Thread id:3581
Terminating func A
rejuvenate@rejuvenate-HP-ProBook-4530s:~$
```

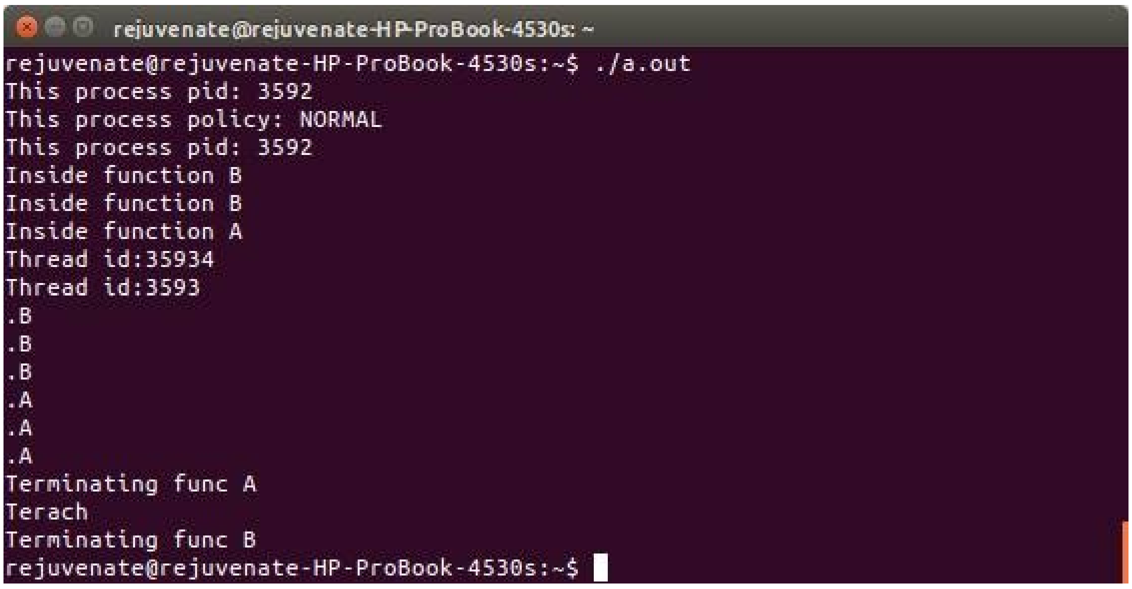
Multiple Thread Creation :



```
rejuvenate@rejuvenate-HP-ProBook-4530s:~$ ./a.out
This process pid: 3688
This process pid: 3688
Inside function A
Inside function A
Inside function B
Thread id:3689
Thread id: 3690
.A
.A
.B
.A
.B
.A
.B
Terminating func A
Terminating func B
Inside function A
Thread id:3691
.A
.A
.A
Terminating func A
rejuvenate@rejuvenate-HP-ProBook-4530s:~$
```

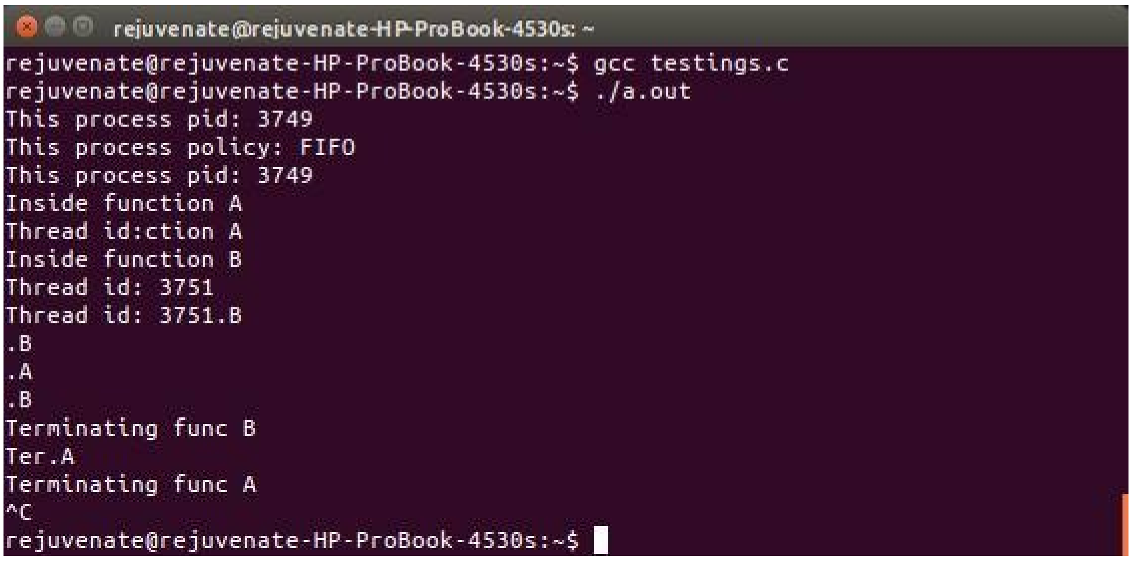
Thread Scheduling :

NORMAL :



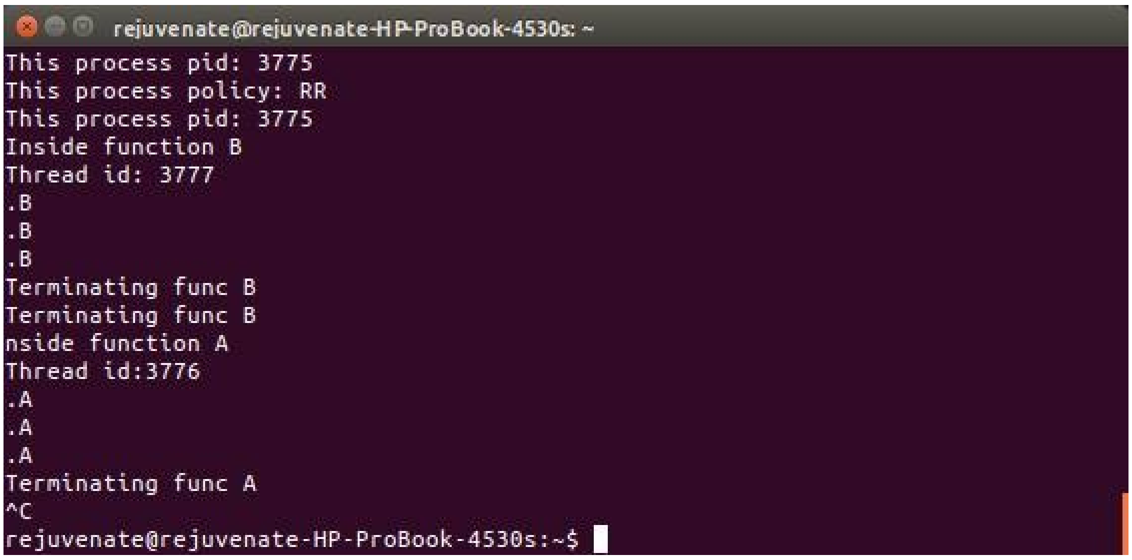
```
rejuvenate@rejuvenate-HP-ProBook-4530s:~$ ./a.out
This process pid: 3592
This process policy: NORMAL
This process pid: 3592
Inside function B
Inside function B
Inside function A
Thread id:35934
Thread id:3593
.B
.B
.B
.A
.A
.A
Terminating func A
Terach
Terminating func B
rejuvenate@rejuvenate-HP-ProBook-4530s:~$
```

FIFO :



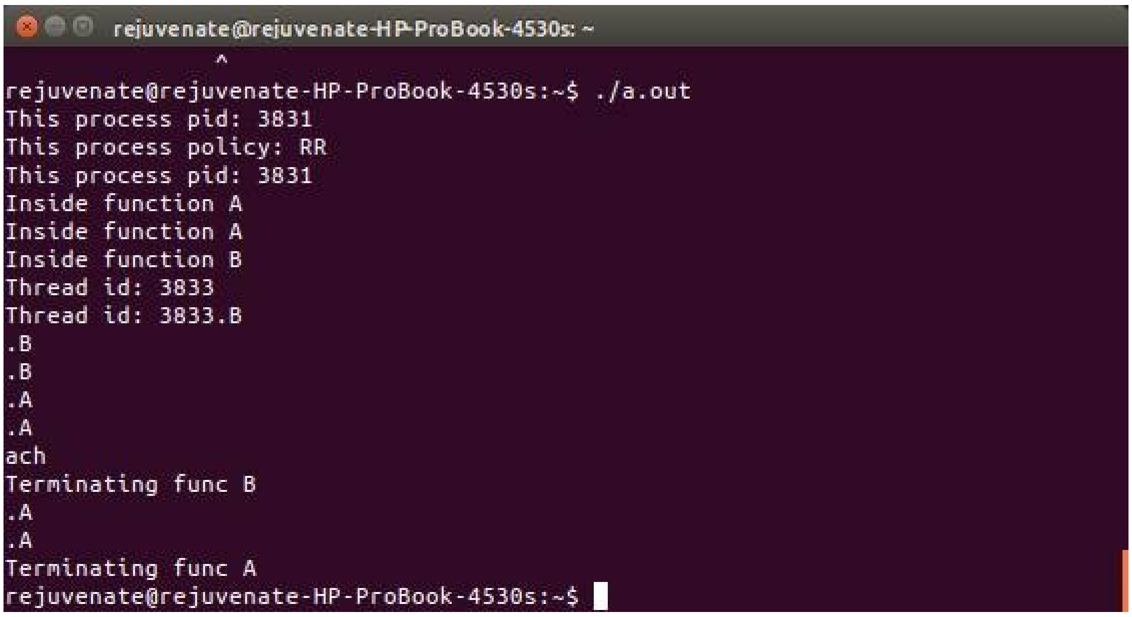
```
rejuvenate@rejuvenate-HP-ProBook-4530s:~$ gcc testings.c
rejuvenate@rejuvenate-HP-ProBook-4530s:~$ ./a.out
This process pid: 3749
This process policy: FIFO
This process pid: 3749
Inside function A
Thread id:ction A
Inside function B
Thread id: 3751
Thread id: 3751.B
.B
.A
.B
Terminating func B
Ter.A
Terminating func A
^C
rejuvenate@rejuvenate-HP-ProBook-4530s:~$
```

RR :



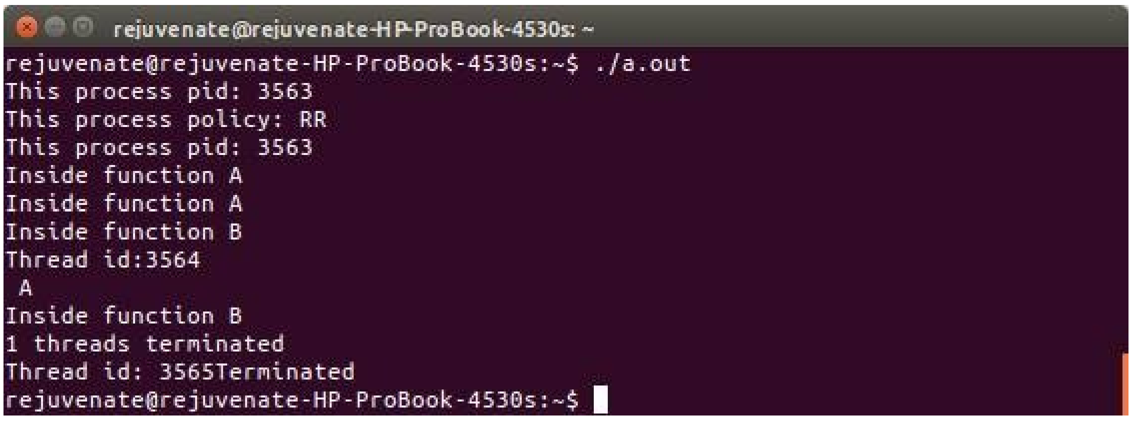
```
This process pid: 3775
This process policy: RR
This process pid: 3775
Inside function B
Thread id: 3777
.B
.B
.B
Terminating func B
Terminating func B
nside function A
Thread id:3776
.A
.A
.A
Terminating func A
^C
rejuvenate@rejuvenate-HP-ProBook-4530s:~$
```

Safe Thread Communication



```
               ^
rejuvenate@rejuvenate-HP-ProBook-4530s:~$ ./a.out
This process pid: 3831
This process policy: RR
This process pid: 3831
Inside function A
Inside function A
Inside function B
Thread id: 3833
Thread id: 3833.B
.B
.B
.A
.A
ach
Terminating func B
.A
.A
Terminating func A
rejuvenate@rejuvenate-HP-ProBook-4530s:~$
```

Thread Termination :



```
rejuvenate@rejuvenate-HP-ProBook-4530s:~$ ./a.out
This process pid: 3563
This process policy: RR
This process pid: 3563
Inside function A
Inside function A
Inside function B
Thread id:3564
 A
Inside function B
1 threads terminated
Thread id: 3565Terminated
rejuvenate@rejuvenate-HP-ProBook-4530s:~$
```