NPTL TGI concept

POSIX thread specification specifies that threads created from a process should share same process id. Native LinuxThread does not comply with this specification. NPTL (Native POSIX Thread Library) does not comply with this specification but introduced Task group ID to make it close to the required specification.

See the following links to get to know more about POSIX threads.

http://linux.die.net/man/7/pthreads

https://computing.llnl.gov/tutorials/pthreads/

fork_pt.c

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

int main(void)
{
        int pid;

        printf("before fork\n\n");

        if((pid = fork()) < 0){
                printf("fork error\n");
                exit(-2);
        }else if (pid == 0){
          printf("TGID(%ul), PID(%d) : of the child process\n", getpid(), syscall(__NR_gettid));
        }else{
                printf("TGID(%d), PID(%d) : of the parent process\n", getpid(),
syscall(__NR_gettid));
                sleep(2);
        }

        printf("after fork\n\n");

        return 0;
}
```

pthread_pt.c

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

```
#include <stdlib.h>
#include <linux/unistd.h>

intptr_t t_function(void *data)
{
	int id;
	int i=0;
	pthread_t t_id;
	id = *((int *)data);
	printf("TGID(%d), PID(%d), pthread_self(%u) : Child\n", getpid(), syscall(__NR_gettid),
pthread_self());
	sleep(2);
	return (int)(intptr_t)(id*id);
}

int main(void)
{
	int pid, status;
	int a = 1;
	int b = 2;
	pthread_t p_thread[2];

	printf("before pthread_create\n\n");

	if((pid = pthread_create(&p_thread[0], NULL, t_function, (void*)&a)) < 0){
		perror("thread create error : ");
		exit(1);
	}
	if((pid = pthread_create(&p_thread[1], NULL, t_function, (void*)&b)) < 0){
		perror("thread create error : ");
		exit(2);
	}

	pthread_join(p_thread[0], (void **)&status);
	printf("pthread_join(%d)\n", status);

	pthread_join(p_thread[1], (void **)&status);
	printf("pthread_join(%d)\n", status);


	printf("TGID(%d), PID(%d) : Parent\n", getpid(), syscall(__NR_gettid));

	return 0;
}
```

clone_pt.c

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <linux/unistd.h>
#include <linux/sched.h>


int sub_func(void *arg)
{
        printf("TGID(%d), PID(%d) : Child\n", getpid(), syscall(__NR_gettid));
        return 0;
}

int main(void)
{
        int pid;
        int child_a_stack[4096], child_b_stack[4096];

        printf("before clone\n\n");
        printf("TGID(%d), PID(%d) : Parent\n", getpid(), syscall(__NR_gettid));

        clone (sub_func, (void *)(child_a_stack+4095), CLONE_CHILD_CLEARTID |
CLONE_CHILD_SETTID, NULL);
        //      clone (sub_func, (void *)(child_b_stack+4095), CLONE_VM |CLONE_THREAD|
CLONE_SIGHAND, NULL);
        clone (sub_func, (void *)(child_b_stack+4095), CLONE_VM |CLONE_THREAD |
CLONE_SIGHAND, NULL);

        sleep(1);

        printf("after clone\n\n");
        return 0;
}
```