

Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 11

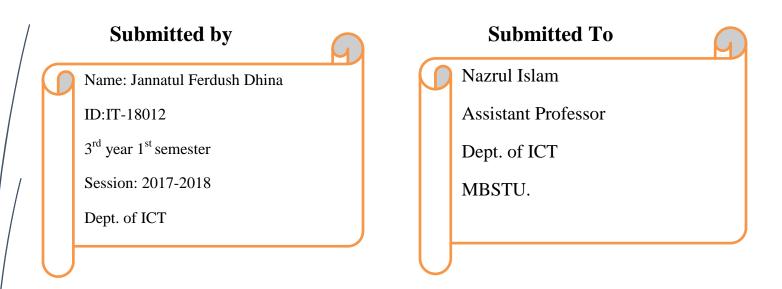
Course code: ICT-3110

Course title: Operating System Lab

Date of Performance: 23-09-2020

Date of Submission:

Reference: Tazneen Akter (The mother board of my laptop is suddenly damaged and for that, I wasn't able to do my lab (2,3,5,6). So I took her help).



Experiment no: 03

Experiment Name: Threads on Operating System.

Theory: Thread is a single sequence stream within a process. Threads have same properties as of the process so they are called as light weight processes. Threads are executed one after another but gives the illusion as if they are executing in parallel. Each thread has different states. Each thread has

- 1. A program counter
- 2. A register set
- 3. A stack space

Types of Threads:

User Level thread (ULT) -

Is implemented in the user level library, they are not created using the system calls. Thread switching does not need to call OS and to cause interrupt to Kernel. Kernel doesn't know about the user level thread and manages them as if they were single-threaded processes.

Kernel Level Thread (KLT) –

Kernel knows and manages the threads. Instead of thread table in each process, the kernel itself has thread table (a master one) that keeps track of all the threads in the system. In addition kernel also maintains the traditional process table to keep track of the processes. OS kernel provides system call to create and manage threads.

Working Process:

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

#include<pthread.h>

#include<stdlib.h>

#include<unistd.h>

```
pthread_t tid[2];
void* doSomeThing(void *arg)
unsigned long i = 0;
pthread_t id = pthread_self();
if(pthread_equal(id,tid[0]))
printf("\n First thread processing\n");
}
else
printf("\n Second thread processing\n");
for(i=0; i<(0xFFFFFFF);i++);</pre>
return NULL;
int main(void)
int i = 0;
int err;
while(i < 2)
err = pthread create(&(tid[i]), NULL, &doSomeThing, NULL);
if (err != 0)
printf("\ncan't create thread :[%s]", strerror(err));
else
printf("\n Thread created successfully\n");
i++;
}
sleep(5);
return 0;
```

Output:

```
Terminal

File Edit View Search Terminal Help

tazneen@tazneen-HP-Laptop-14-bs0xx:~$ cd Tazneen
tazneen@tazneen-HP-Laptop-14-bs0xx:~/Tazneen$ gcc thread.c -lpthread
tazneen@tazneen-HP-Laptop-14-bs0xx:~/Tazneen$ ./a.out

Thread created successfully

First thread processing

Thread created successfully

Second thread processing
tazneen@tazneen-HP-Laptop-14-bs0xx:~/Tazneen$
```

Thread in command line:

1.ps-

In ps command, "-T" option enables thread views. The following command list all threads created by a process with <pid>

```
File Edit View Search Terminal Help
tazneen@tazneen-HP-Laptop-14-bs0xx:~$ ps
PID TTY TIME CMD
3478 pts/0 00:00:00 bash
3479 pts/0 00:00:00 ps
tazneen@tazneen-HP-Laptop-14-bs0xx:~$
```

2.top-

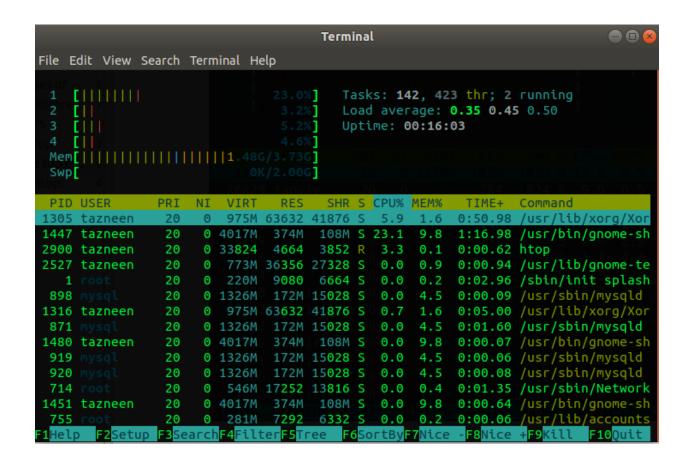
The top command can show a real-time view of individual threads. To enable thread views in the top output, invoke top with "-H" option. This will list all Linux threads.

											reminat
File E	dit View	Search	h Ter	minal Hel	p						
top -	11:35:54	up 2	28 mi	in, 1 us	ser, lo	oad aver	ag	e: 0.4	13, 0.	63, 0.59	
										<pre>0 zombie</pre>	
											si, 0.0 st
KiB Me										241140 but	
KiB S	wap: 209	7148	tota	al, 209 7	7148 Tre	ee,		0 use	ed. 2	005900 av	all Mem
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
	tazneen	20	0	4113744	465728	111540	S	24.7	11.9	2:56.81	gnome-shell
1305	tazneen	20	0	997312	64088	42332		5.3	1.6	1:50.23	
3409	tazneen	20	0	44544	4168	3516	R	1.0	0.1	0:00.18	
1	root	20	0	225456	9080	6664	S	0.3	0.2	0:04.43	systemd
11	root	20	0	0	0	0	I	0.3	0.0	0:01.98	rcu_sched
24	root	20	0	0	0	Mem 0	S	0.3	0.0	0:00.04	ksoftirqd/2
111	root	20	0	0	0	0	I	0.3	0.0	0:02.82	kworker/u8:2-ev
3205	root	20	0	0	0	0	Ι	0.3	0.0	0:00.40	kworker/1:1-eve
2	root	20	0	0	0	0		0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	Ι	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	Ι	0.0	0.0		rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-kb
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
10	root	20	0	0	0	0	S	0.0	0.0	0:00.06	ksoftirqd/0
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.01	migration/0
13	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
16	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/1
17	root	rt	0	0	0	0	S	0.0	0.0		migration/1
18	root	20	0	0	0	0	S	0.0	0.0		ksoftirqd/1
20	root	0	-20	0	0	0	Ι	0.0	0.0	0:00.00	kworker/1:0H-kb
21	root	20	0	0	0	0	S	0.0	0.0		cpuhp/2
22	root	-51	0	0	0	0		0.0	0.0		idle_inject/2
23	root	rt	0	0	0	0	S	0.0	0.0	0:00.12	migration/2
26	root	0	- 20	0	0	0	Ι	0.0	0.0		kworker/2:0H-kb
27	root	20	0	0	0	0		0.0	0.0		cpuhp/3
28	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/3

To restrict the top output to a particular process and check all threads running inside the process: then we use \$ top -H -p

3.htop-

A more user-friendly way to view threads per process is via htop, an ncurses-based interactive process viewer. This program allows you to monitor individual threads in tree views.



Discussion: In this lab we have implemented Threads on operating system using c language. By solving this problem we learn about thread in command line .