Mawlana Bhashani Science and Technology University

Lab-Report

Lab Report No: 03

Lab Report Name: How to view threads of a process on Linux and thread program .

Course code: ICT-3110

Course title: Operating System Lab

Date of Performance: 13-09-2020

Date of Submission:

Submitted by

Name: Md Masud Rana

ID: IT-18002

3rd year 1st semester

Session: 2017-18

Dept. of ICT.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT,

MBSTU.

Experiment No: 03

Experiment Name: How to view threads of a process on Linux and thread program .

Theory : A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

A thread shares with its peer threads few information like code segment, data segment and open files. When one thread alters a code segment memory item, all other threads see that.

A thread is also called a lightweight process. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Types of Thread: Threads are implemented in following two ways –

User Level Threads − User managed threads.

Kernel Level Threads − Operating System managed threads acting on kernel, an operating system core.

Code-

Open ▾ | new.c ~/program | Save

```c
 #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<(0xFFFFFFFF);i++);

    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-

```
masud@masud-VirtualBox:~/program$ gcc new.c -lpthread
masud@masud-VirtualBox:~/program$ ./a.out

 Thread created successfully

 Thread created successfully

 Second thread processing

 First thread processing
masud@masud-VirtualBox:~/program$
```

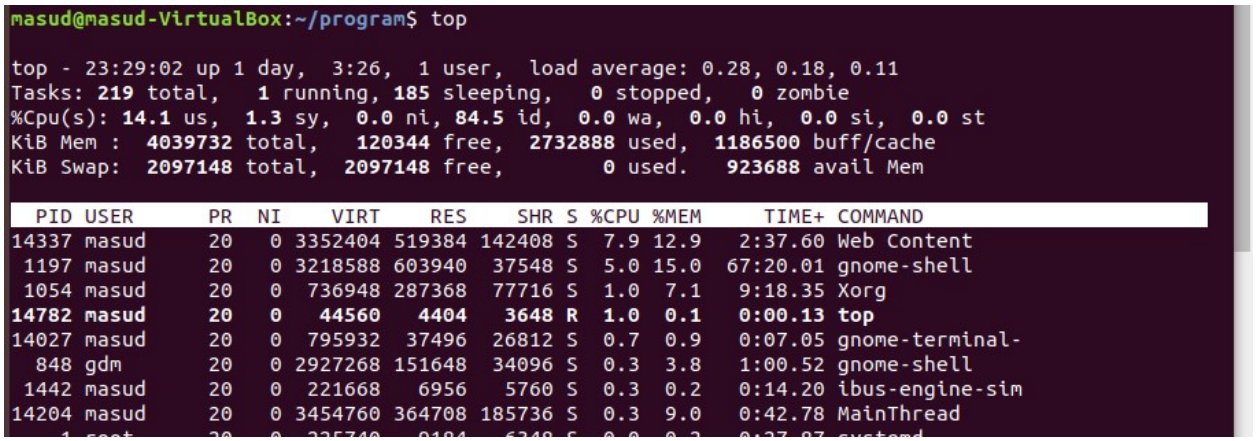
Thread in command line:

Here are several ways to show threads for a process on Linux.

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

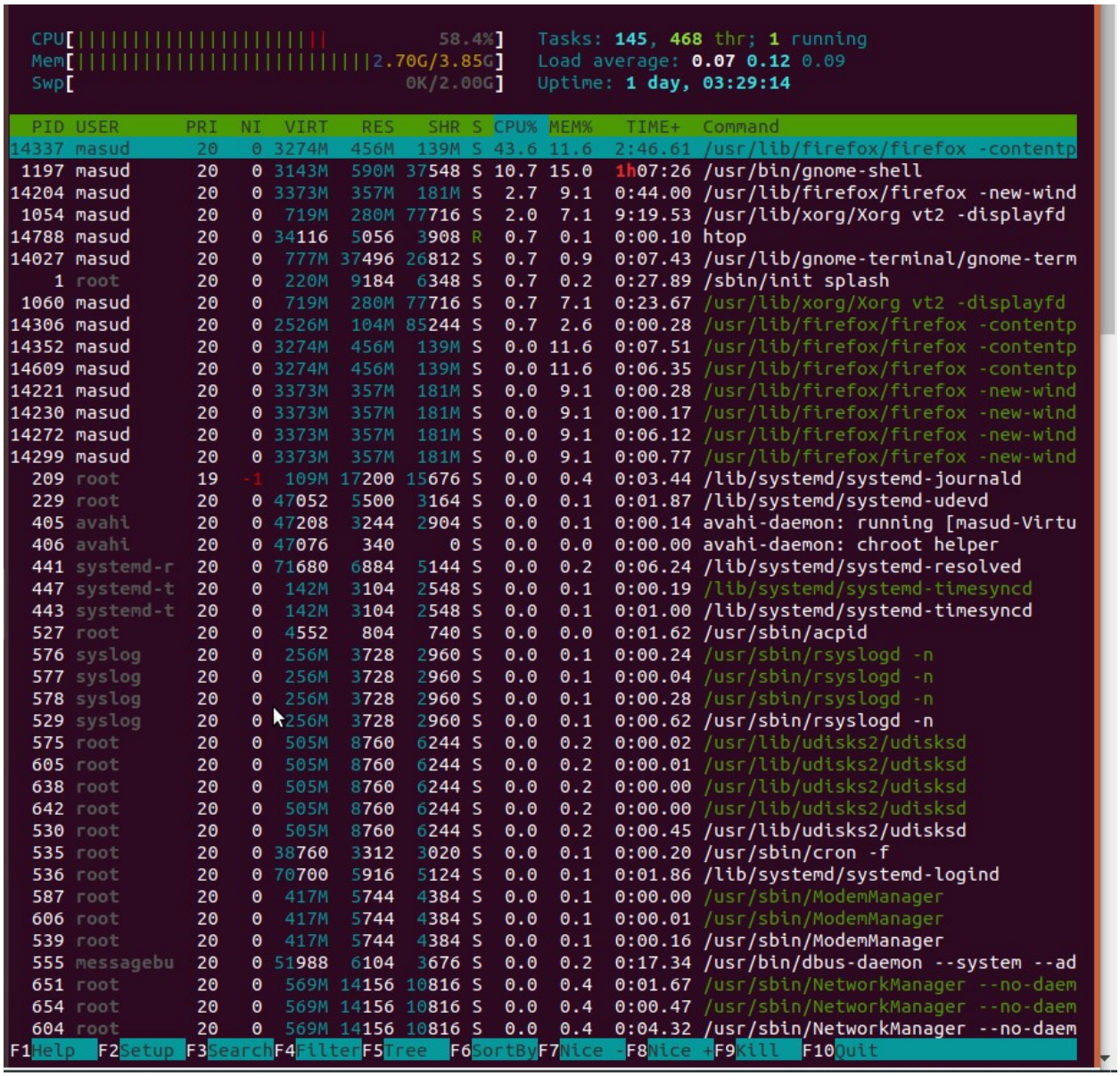
```
masud@masud-VirtualBox:~/program$ ps -T -p 1197
  PID  SPID TTY          TIME CMD
 1197  1197 tty2     01:06:55 gnome-shell
 1197  1199 tty2     00:00:00 gmain
 1197  1200 tty2     00:00:14 gdbus
 1197  1201 tty2     00:00:00 dconf worker
 1197  1210 tty2     00:00:00 threaded-ml
 1197  1211 tty2     00:00:00 JS Helper
 1197  1212 tty2     00:00:00 JS Helper
 1197  1213 tty2     00:00:00 JS Helper
 1197  1214 tty2     00:00:00 JS Helper
 1197  1215 tty2     00:00:00 JS Helper
```

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.



To restrict the top output to a particular process and check all threads running inside the process: then we use $ top -H –p <pid>.

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 learn how to view threads of process on linux.Here we use c language for creating threads.We also use command line show the threads for a process on linux.For command line we use ps ,top and Htop command.From this lab we also learn threads are implemented two ways such as user level threads and kernel level threads.