APES HW 3

CODE:

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <pthread.h>
#include <time.h>
#include <signal.h>
#include <sys/time.h>
#include <string.h>
#include <sys/syscall.h>

pthread_mutex_t pmutex;

typedef struct
{
      char * file_name;
      char * output;
      int child_number;
}my_thread;

struct node
{
      struct node * next;
      struct node * prev;
      char data;
};

FILE * mainptr;

void logger(FILE* file_pointer, char * message, char * thread_name,
char myChar, int ppid, int pid, long int tid)
{
      my_thread * child_thread1 = malloc(sizeof(my_thread));
      fprintf(file_pointer, "Thread_Name: %s Message: %s\n PPID: %d
PID: %d TID: %ld ",
thread_name, message, ppid, pid, tid);
      time_t clock = time(NULL);
      fprintf(file_pointer, "Timestamp: %s", ctime(&clock));
      if(myChar != NULL)
      fprintf(file_pointer, "Character occured thrice: %c \n", myChar);
      fprintf(file_pointer, "\n");
      fclose(file_pointer);
}

void cpu_usage(int value)
{
      if(value == SIGVTALRM)
```

```
      {
            char *comm = "cat /proc/stat | head -n 2";
            FILE * command_ptr = popen(comm, "r");
            pthread_mutex_lock(&pmutex);
            mainptr = fopen("Output.txt", "a");
            logger(mainptr, "CPU Usage Report", "Second thread", NULL,
getppid(), getpid(), syscall(SYS_gettid));
            pthread_mutex_unlock(&pmutex);
            if(command_ptr)
            while(!feof(command_ptr))
            {
                  char temp1;
                  temp1 = fgetc(command_ptr);
                  mainptr = fopen("Output.txt", "a");
                  fprintf(mainptr, "%c", temp1);
                  fclose(mainptr);
            }
      }
      if(value == SIGUSR1 || value == SIGUSR2)
      {
            mainptr = fopen("Output.txt", "a");
            pthread_mutex_lock(&pmutex);
            logger(mainptr, "Signal handler recieved, exiting threads",
"Second thread", NULL, getppid(), getpid(), syscall(SYS_gettid));
            pthread_mutex_unlock(&pmutex);
            pthread_exit(NULL);
      }
}

struct node * data_process(struct node * head)
{
    struct node * temp = head;
    struct node * newInfo = (struct node *)malloc(sizeof(struct
node));
    int count = 0;
    char arr[10];
    char e,f;
    int i=0;
    mainptr = fopen("Output.txt", "a");
    pthread_mutex_lock(&pmutex);
    logger(mainptr, "Executing first child", "First thread", NULL,
getppid(), getpid(), syscall(SYS_gettid));
    pthread_mutex_unlock(&pmutex);
    for(int i = 65; i <= 96; i++)
    {
      char a = (char)i;
      char b = (char)i+32;
      while(temp -> next != NULL)
      {
         if(temp -> data == a || temp -> data == b)
         {
            count++;
```

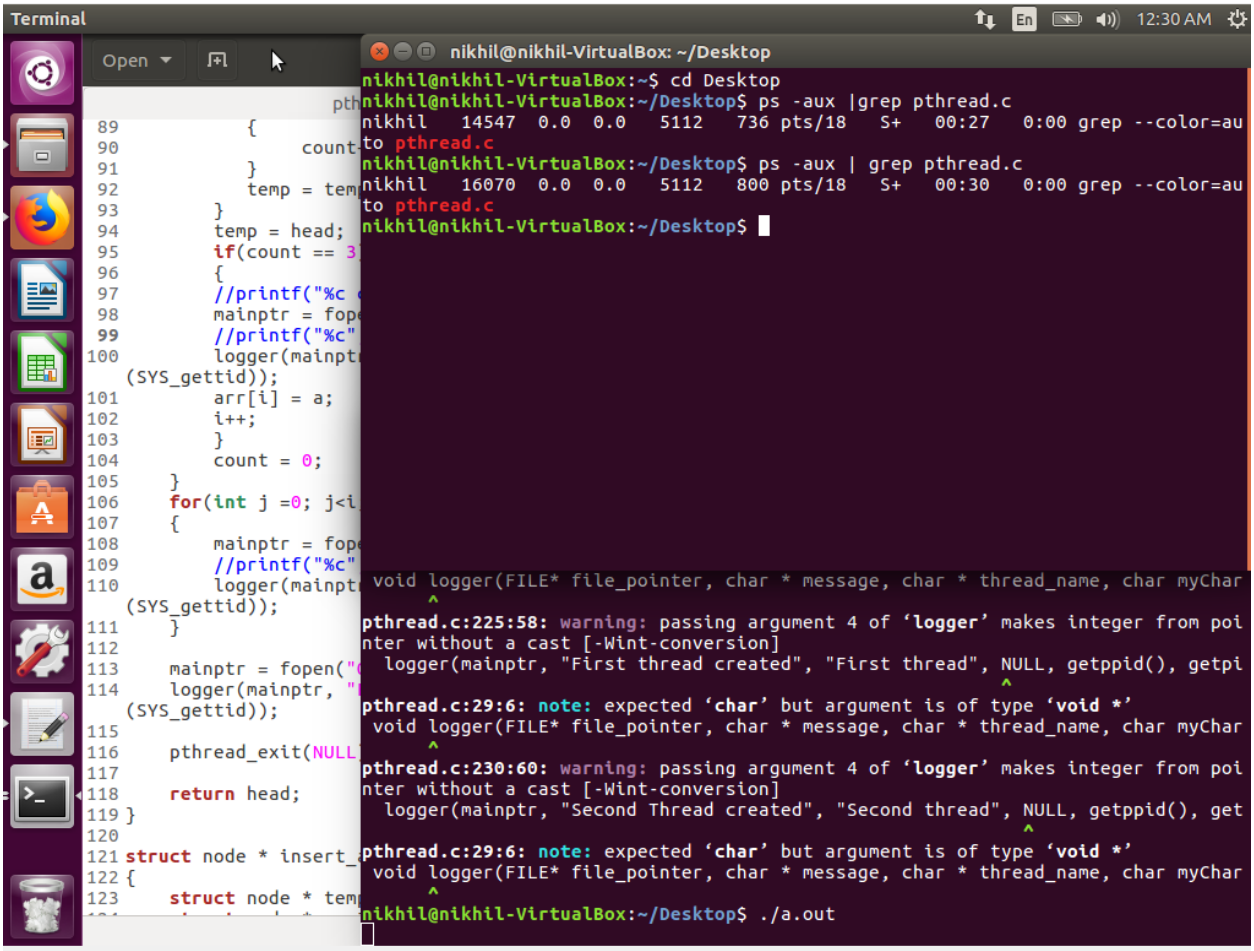
```
        }
        temp = temp -> next;
      }
      temp = head;
      if(count == 3)
      {
      mainptr = fopen("Output.txt", "a");
      pthread_mutex_lock(&pmutex);
      logger(mainptr, "Character occured thrice", "First thread", a,
getppid(), getpid(), syscall(SYS_gettid));
      pthread_mutex_unlock(&pmutex);
      arr[i] = a;
      i++;
      }
      count = 0;
    }
    for(int j =0; j<i; j++)
    {
      mainptr = fopen("Output.txt", "a");
      pthread_mutex_lock(&pmutex);
      logger(mainptr, "Character Processed", "First Thread", arr[j],
getppid(), getpid(), syscall(SYS_gettid));
      pthread_mutex_unlock(&pmutex);
    }

    mainptr = fopen("Output.txt", "a");
  pthread_mutex_lock(&pmutex);
    logger(mainptr, "Exiting first thread", "First Thread", NULL,
getppid(), getpid(), syscall(SYS_gettid));
  pthread_mutex_unlock(&pmutex);

    pthread_exit(NULL);

    return head;
}

struct node * insert_at_end(struct node * head, char x)
{
    struct node * temp = head;
    struct node * newInfo = (struct node *)malloc(sizeof(struct
node));
    newInfo -> data = x;
    newInfo -> next = NULL;
    if(head == NULL)
    {
      head = newInfo;
      return head;
    }
    else
    {
      while(temp -> next != NULL)
      {
```

```
            temp = temp -> next;
        }

        newInfo -> prev = temp;
        temp -> next = newInfo;
        newInfo -> next = NULL;
        return head;
      }
}

void * perform_task(void * my_data)
{
        my_thread * thread_data = (my_thread *)my_data;
        if(thread_data -> child_number == 1)
        {
                FILE * myfile;
                myfile = fopen(thread_data -> file_name, "r");

                FILE * first_child_fp;
                first_child_fp = fopen("Output.txt", "a" );

                if(myfile == NULL)
                {
                        printf("Error reading the file");
                }
                if(first_child_fp == NULL)
                {
                        printf("Error reading the file");
                }

                char temp;
                struct node * head = (struct node *)malloc(sizeof(struct
node));
                while(!feof(myfile))
                {
                        temp = fgetc(myfile);
                        head = insert_at_end(head, temp);
                }
                data_process(head);
        }

        else if(thread_data -> child_number == 2)
        {
                struct sigaction my_action;
                struct itimerval my_timer;

                memset (&my_action, 0, sizeof (my_action));
                my_action.sa_handler = &cpu_usage;

                my_timer.it_interval.tv_sec = 0;
                my_timer.it_interval.tv_usec = 100000;
```

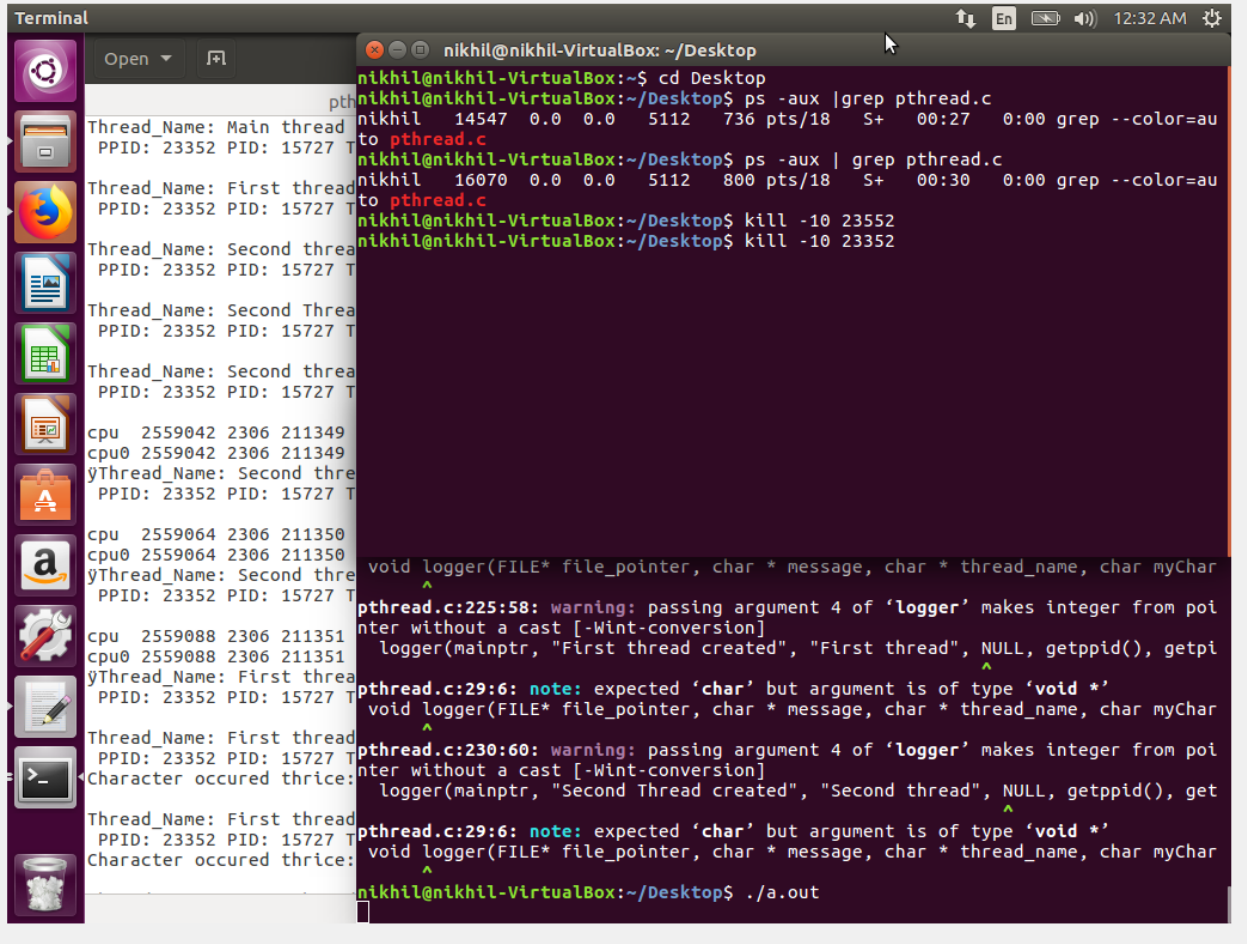
```
            my_timer.it_value.tv_sec = 0;
            my_timer.it_value.tv_usec = 100000;

            sigaction (SIGVTALRM, &my_action, NULL);
            sigaction (SIGUSR1, &my_action, NULL);
            sigaction (SIGUSR2, &my_action, NULL);

            setitimer (ITIMER_VIRTUAL, &my_timer, NULL);
            mainptr = fopen("Output.txt", "a");
            pthread_mutex_lock(&pmutex);
            logger(mainptr, "Executing second thread", "Second Thread",
NULL, getppid(), getpid(), syscall(SYS_gettid));
            pthread_mutex_unlock(&pmutex);

            while (1);
      }

      return NULL;
}

int main()
{
      pthread_t thread1, thread2;
      my_thread * child_thread1 = malloc(sizeof(my_thread));
      my_thread * child_thread2 = malloc(sizeof(my_thread));
      FILE * fileptr;
      fileptr = fopen("Valentinesday.txt", "r");

      child_thread1 -> file_name = "Valentinesday.txt";
      child_thread1 -> output = "Output.txt";
      child_thread1 -> child_number = 1;

      child_thread2 -> file_name = "Valentinesday.txt";
      child_thread2 -> child_number = 2;
      child_thread2 -> output = "Output.txt";

      mainptr = fopen(child_thread1 -> output, "w");
      pthread_mutex_lock(&pmutex);
      logger(mainptr, "Main thread started", "Main thread", NULL,
getppid(), getpid(), syscall(SYS_gettid));
      pthread_mutex_unlock(&pmutex);

      pthread_create(&thread1, NULL, perform_task, (void
*)child_thread1);
      mainptr = fopen(child_thread1 -> output, "a");
      pthread_mutex_lock(&pmutex);
      logger(mainptr, "First thread created", "First thread", NULL,
getppid(), getpid(), syscall(SYS_gettid));
      pthread_mutex_unlock(&pmutex);

      pthread_create(&thread2, NULL, perform_task, (void
*)child_thread2);
```

```
	mainptr = fopen(child_thread1 -> output, "a");
	pthread_mutex_lock(&pmutex);
	logger(mainptr, "Second Thread created", "Second thread", NULL,
getppid(), getpid(), syscall(SYS_gettid));
	pthread_mutex_unlock(&pmutex);

	pthread_join(thread1, NULL);
	pthread_join(thread2, NULL);
}
```

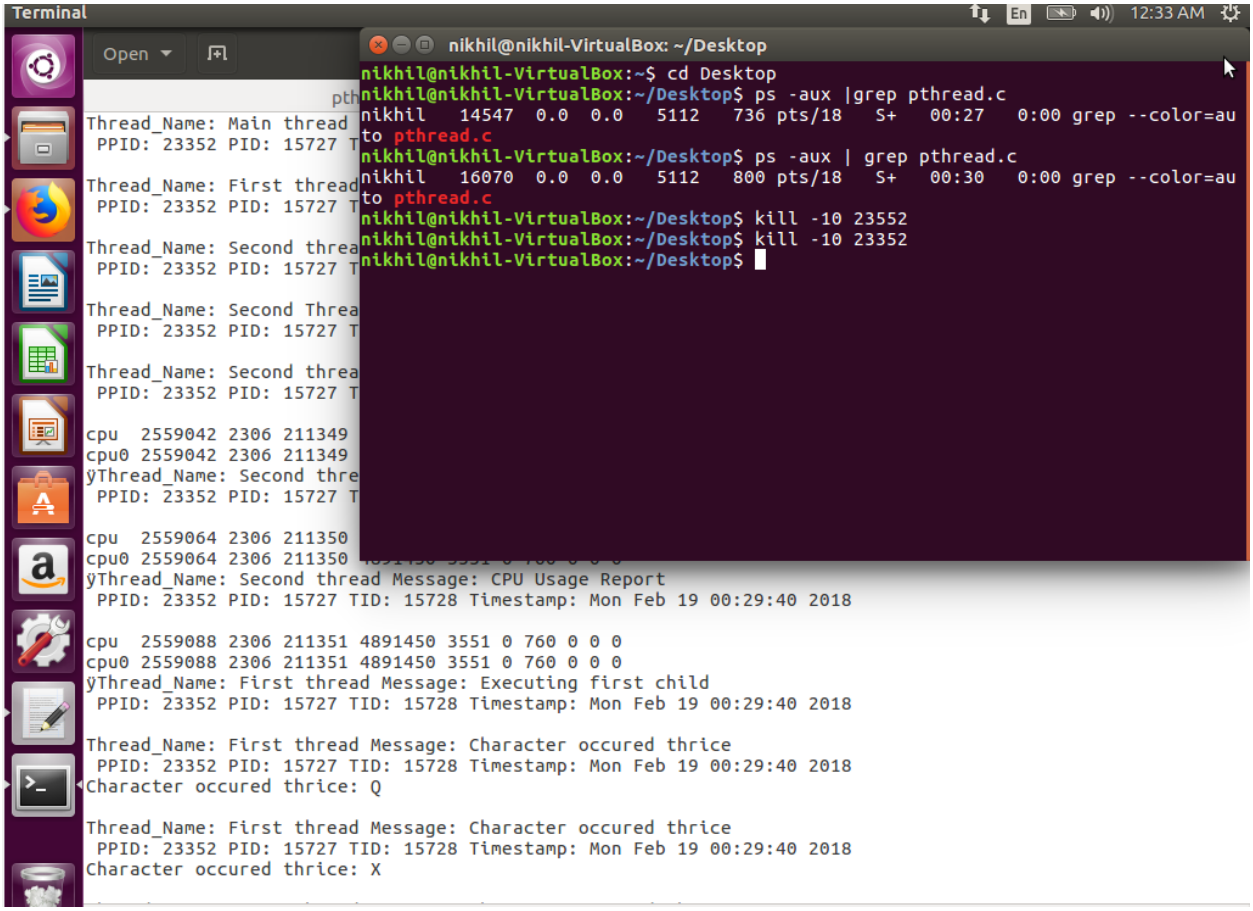
Snapshots:



The above screenshot shows running thread



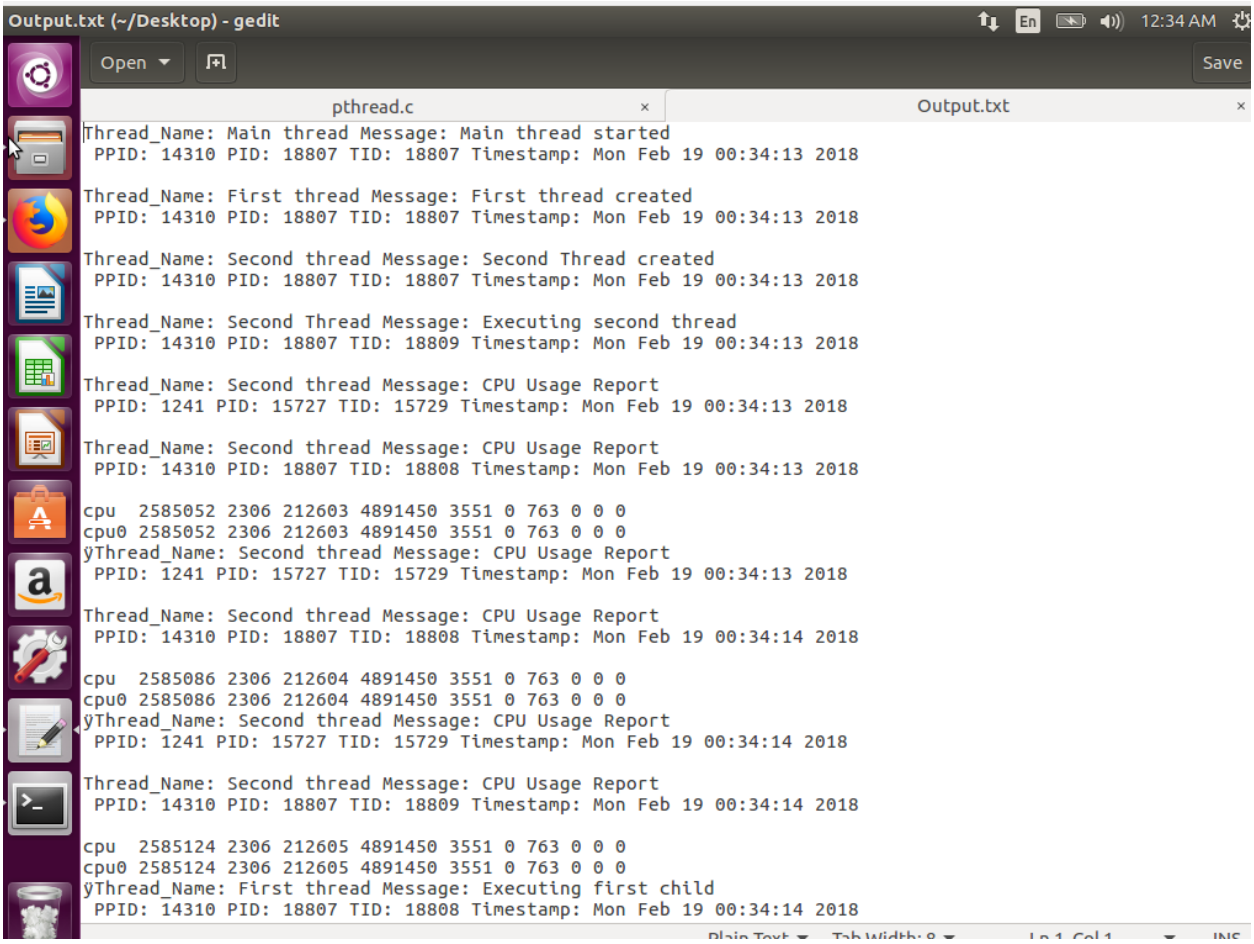
This is the command to kill the thread using SIGUSR1 handler. 10 is enum number for SIGUSR1 and 23352 is parent process id.

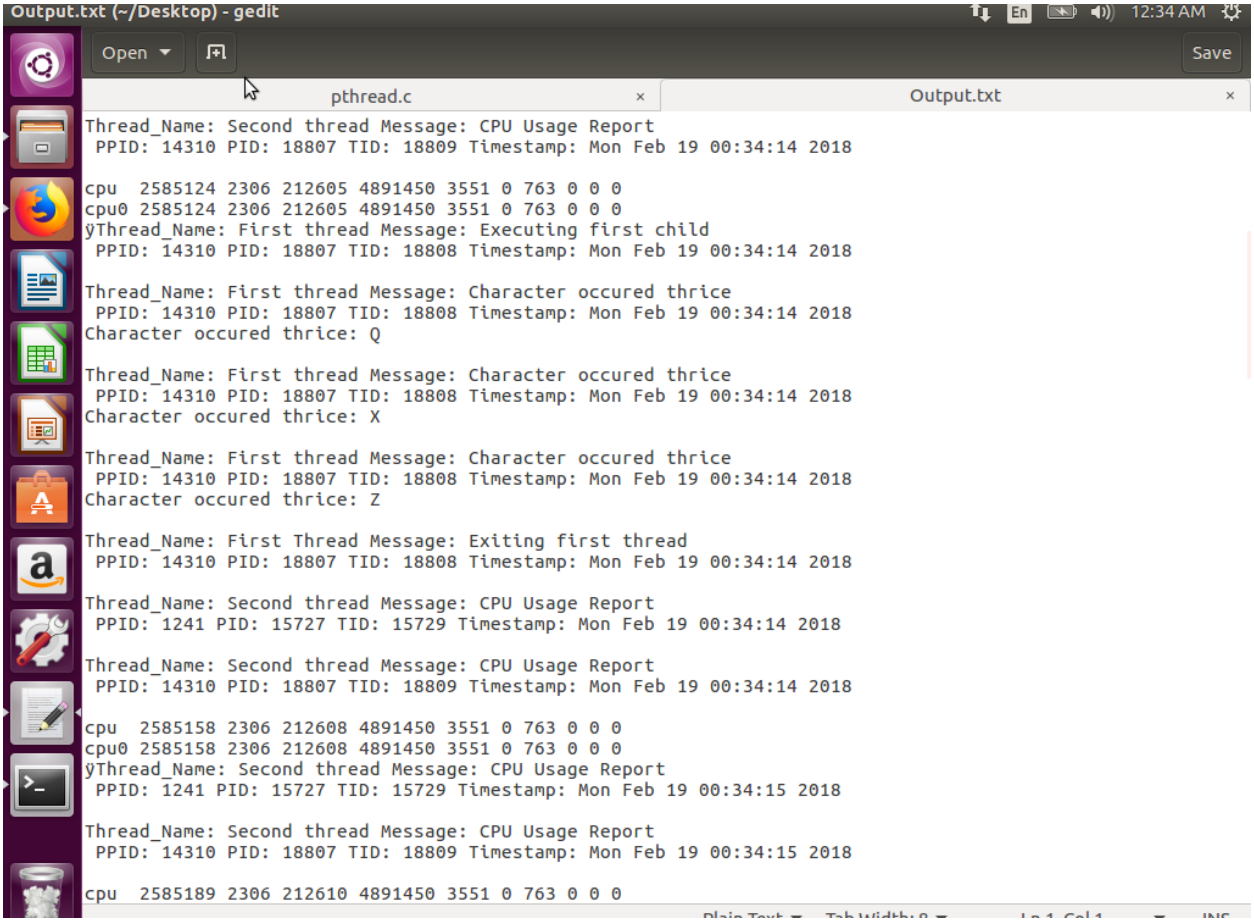


You can see the command is killed with other terminal shutting down

LOG File:

Images of log file are as below:


Output.txt (~/Desktop) - gedit
12:34 AM
Open
Save
pthread.c
Output.txt
Thread_Name: Main thread Message: Main thread started
PPID: 14310 PID: 18807 TID: 18807 Timestamp: Mon Feb 19 00:34:13 2018
Thread_Name: First thread Message: First thread created
PPID: 14310 PID: 18807 TID: 18807 Timestamp: Mon Feb 19 00:34:13 2018
Thread_Name: Second thread Message: Second Thread created
PPID: 14310 PID: 18807 TID: 18807 Timestamp: Mon Feb 19 00:34:13 2018
Thread_Name: Second Thread Message: Executing second thread
PPID: 14310 PID: 18807 TID: 18809 Timestamp: Mon Feb 19 00:34:13 2018
Thread_Name: Second thread Message: CPU Usage Report
PPID: 1241 PID: 15727 TID: 15729 Timestamp: Mon Feb 19 00:34:13 2018
Thread_Name: Second thread Message: CPU Usage Report
PPID: 14310 PID: 18807 TID: 18808 Timestamp: Mon Feb 19 00:34:13 2018
cpu 2585052 2306 212603 4891450 3551 0 763 0 0 0
cpu0 2585052 2306 212603 4891450 3551 0 763 0 0 0
ÿThread_Name: Second thread Message: CPU Usage Report
PPID: 1241 PID: 15727 TID: 15729 Timestamp: Mon Feb 19 00:34:13 2018
Thread_Name: Second thread Message: CPU Usage Report
PPID: 14310 PID: 18807 TID: 18808 Timestamp: Mon Feb 19 00:34:14 2018
cpu 2585086 2306 212604 4891450 3551 0 763 0 0 0
cpu0 2585086 2306 212604 4891450 3551 0 763 0 0 0
ÿThread_Name: Second thread Message: CPU Usage Report
PPID: 1241 PID: 15727 TID: 15729 Timestamp: Mon Feb 19 00:34:14 2018
Thread_Name: Second thread Message: CPU Usage Report
PPID: 14310 PID: 18807 TID: 18809 Timestamp: Mon Feb 19 00:34:14 2018
cpu 2585124 2306 212605 4891450 3551 0 763 0 0 0
cpu0 2585124 2306 212605 4891450 3551 0 763 0 0 0
ÿThread_Name: First thread Message: Executing first child
PPID: 14310 PID: 18807 TID: 18808 Timestamp: Mon Feb 19 00:34:14 2018


Output.txt (~/Desktop) - gedit
12:34 AM
Open
Save
pthread.c
Output.txt
Thread_Name: Second thread Message: CPU Usage Report
PPID: 14310 PID: 18807 TID: 18809 Timestamp: Mon Feb 19 00:34:14 2018
cpu 2585124 2306 212605 4891450 3551 0 763 0 0 0
cpu0 2585124 2306 212605 4891450 3551 0 763 0 0 0
ÿThread_Name: First thread Message: Executing first child
PPID: 14310 PID: 18807 TID: 18808 Timestamp: Mon Feb 19 00:34:14 2018
Thread_Name: First thread Message: Character occured thrice
PPID: 14310 PID: 18807 TID: 18808 Timestamp: Mon Feb 19 00:34:14 2018
Character occured thrice: Q
Thread_Name: First thread Message: Character occured thrice
PPID: 14310 PID: 18807 TID: 18808 Timestamp: Mon Feb 19 00:34:14 2018
Character occured thrice: X
Thread_Name: First thread Message: Character occured thrice
PPID: 14310 PID: 18807 TID: 18808 Timestamp: Mon Feb 19 00:34:14 2018
Character occured thrice: Z
Thread_Name: First Thread Message: Exiting first thread
PPID: 14310 PID: 18807 TID: 18808 Timestamp: Mon Feb 19 00:34:14 2018
Thread_Name: Second thread Message: CPU Usage Report
PPID: 1241 PID: 15727 TID: 15729 Timestamp: Mon Feb 19 00:34:14 2018
Thread_Name: Second thread Message: CPU Usage Report
PPID: 14310 PID: 18807 TID: 18809 Timestamp: Mon Feb 19 00:34:14 2018
cpu 2585158 2306 212608 4891450 3551 0 763 0 0 0
cpu0 2585158 2306 212608 4891450 3551 0 763 0 0 0
ÿThread_Name: Second thread Message: CPU Usage Report
PPID: 1241 PID: 15727 TID: 15729 Timestamp: Mon Feb 19 00:34:15 2018
Thread_Name: Second thread Message: CPU Usage Report
PPID: 14310 PID: 18807 TID: 18809 Timestamp: Mon Feb 19 00:34:15 2018
cpu 2585189 2306 212610 4891450 3551 0 763 0 0 0