Coding Project 2   CS 6233
Name: Wei Gu   ID: N14490190

Part 1

This program is currently a single threaded process. Your objective is to make this a multi-threaded program such that the reading of files is done concurrently.

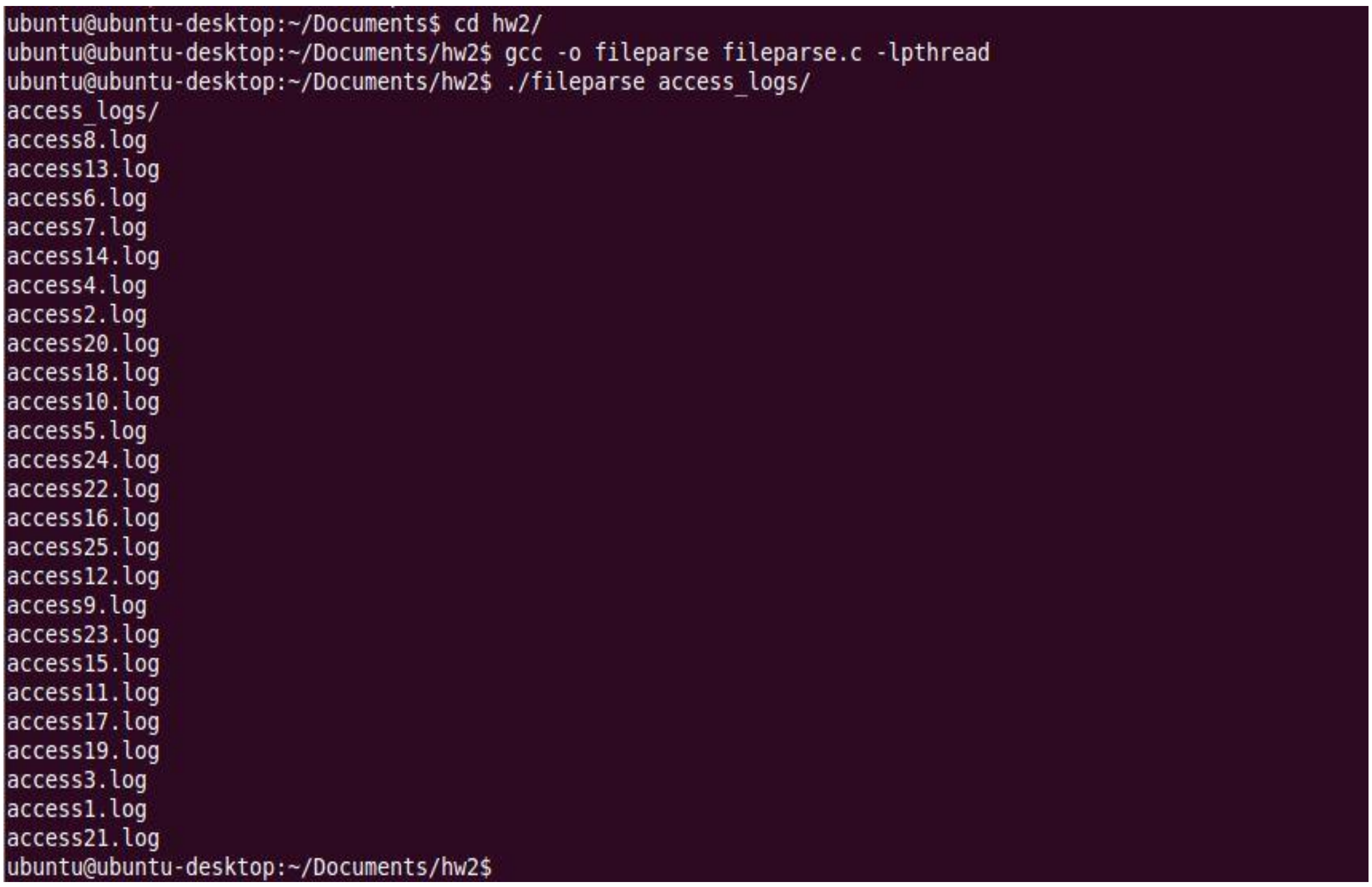
To do this, you’ ll need to use the pthreads library as discussed in class. Be careful to ensure that all threads have completed running before exiting and that you have destroyed any objects you created.

I use 2 thread to finish this task, code as below(in part1.zip):

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <dirent.h>
#include <pthread.h>

int file_flag[25]={0};//set flag for 25 files
pthread_mutex_t mutex;

int check_file(char filename[])
{
  pthread_mutex_lock(&mutex);//add lock
  char buf[2];
  buf[0]=filename[6];
  if(filename[7]!='.')
     {buf[1]=filename[7];}
  int tmp=atoi(buf);
  if(file_flag[tmp-1]==0)
  {
    file_flag[tmp-1]=1;
    pthread_mutex_unlock(&mutex);//unlock
    return 1;
  }
  else
  {
   pthread_mutex_unlock(&mutex);//unlock
   return 0;
  }
};

void *open_dir_read_file(void* arg)
{
  char* open_add=(char *)arg;
  DIR *dir; //directory stream
    FILE *file; //file stream
    struct dirent *ent; // directory entry structure
  char *line = NULL;    // pointer to
  size_t len = 1000;    //the length of bytes getline will allocate
  size_t read;

    char full_filename[256];    //will hold the entire file name to read
        // try to open the directory given by the argument
    if ((dir = opendir (open_add)) != NULL)
    {
        /* print all the files and directories within directory */
        while ((ent = readdir (dir)) != NULL)
        {
                //printf ("%s\n", ent->d_name);
            // Check if the list is a regular file
            if(ent->d_type == DT_REG)
            {
                // Create the absolute path of the filename
                snprintf(full_filename, sizeof full_filename, "./%s%s\0", open_add, ent->d_name);

                // open the file
        if(check_file(ent->d_name)==1)//check if this file was read and use mutex
         {
             printf("%s\n", ent->d_name);
             FILE* file = fopen(full_filename, "r");
              // file was not able to be open
              if (file != NULL)
               {
                 // Print out each line in the file
                  while ((read = getline(&line, &len, file)) != -1)
                  {
                      //printf("Retrieved line of length %d:\n", read);
                      //printf("%s", line);
                       }
                   fclose(file);
                 }
          }
            }
      }
        // Close the directory structure
        closedir (dir);
    }
    else
    {
        /* could not open directory */
        perror ("");
        //return -1;
    }
};

int main(int argc, char *argv[])
{
    // check the arguments
    if(argc < 2)
    {
        printf("Not enough arguments supplied\n");
        return -1;
    }
    if(argc > 2)
    {
        printf("Too many arguments supplied\n");
        return -1;
    }
        printf("%s\n",argv[1]);
        pthread_mutex_init(&mutex,NULL);
        int ret_1,ret_2;
        pthread_t tid_1,tid_2;
        ret_1=pthread_create(&tid_1,NULL,open_dir_read_file,(void *)argv[1]);
        ret_2=pthread_create(&tid_2,NULL,open_dir_read_file,(void *)argv[1]);
        pthread_join(tid_1,NULL);
        pthread_join(tid_2,NULL);
        pthread_exit(NULL);
        return 0;
}
```

input instruction:

```
gcc  -o fileparse fileparse.c -lpthread
./fileparse access_logs/
```



```
ubuntu@ubuntu-desktop:~/Documents$ cd hw2/
ubuntu@ubuntu-desktop:~/Documents/hw2$ gcc -o fileparse fileparse.c -lpthread
ubuntu@ubuntu-desktop:~/Documents/hw2$ ./fileparse access_logs/
access_logs/
access8.log
access13.log
access6.log
access7.log
access14.log
access4.log
access2.log
access20.log
access18.log
access10.log
access5.log
access24.log
access22.log
access16.log
access25.log
access12.log
access9.log
access23.log
access15.log
access11.log
access17.log
access19.log
access3.log
access1.log
access21.log
ubuntu@ubuntu-desktop:~/Documents/hw2$
```

Part 2

The current fileparse.c loops through logs and displays the line and the size of the line. The objective of the second part of this exercise is to get a count of the unique number of IP addresses in the log file. Note: the IP address is the first set of numbers on each line and it indicates the IP of the person that visited that page.

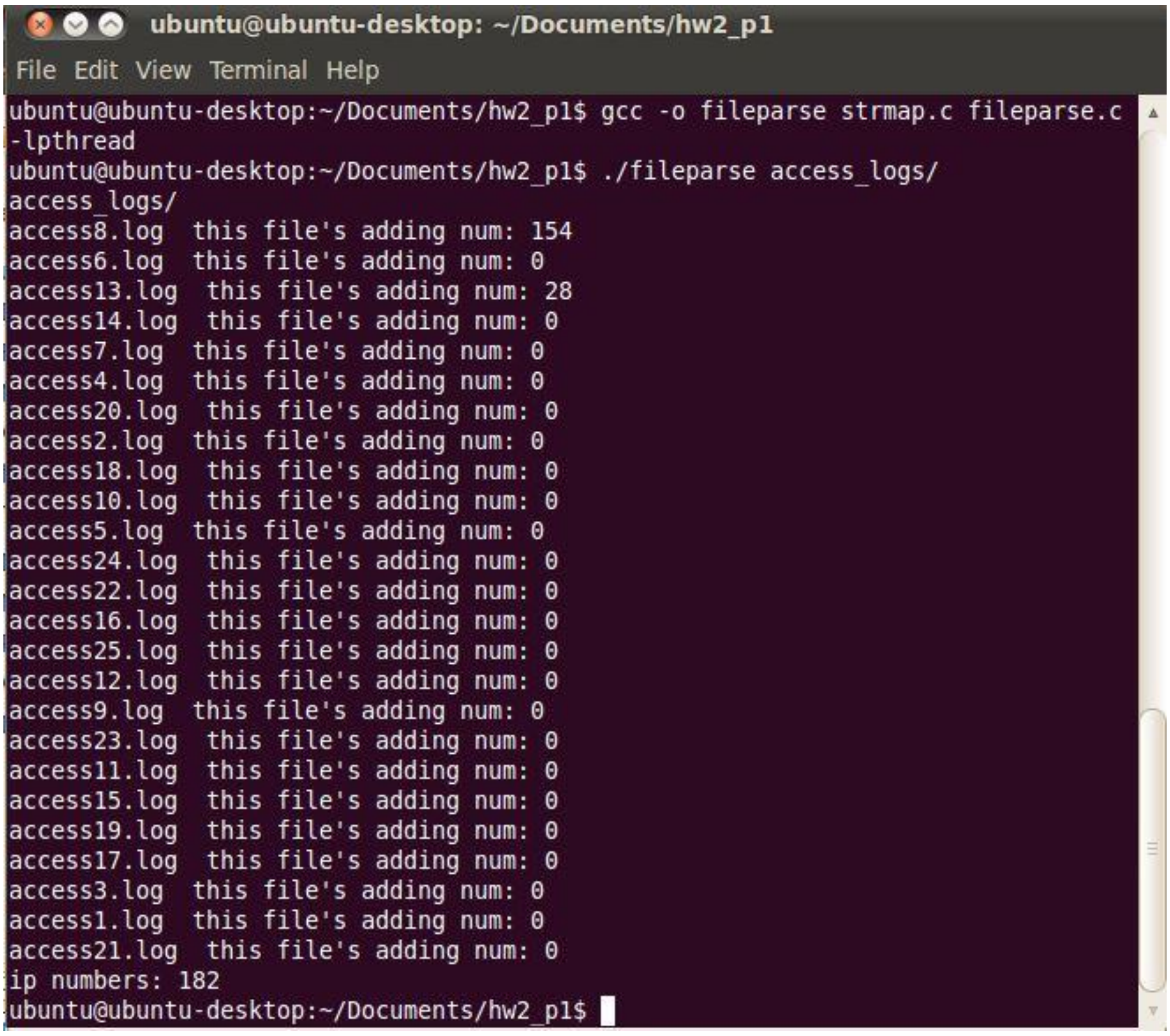
You can take any approach (in C - no calls to outside programs) you would like to do this but keep in mind that it must be free of race conditions. I would recommend using synchronization primitives provided by the pthreads library but other techniques such as using thread local storage would also be acceptable.
Make some changes to part1, and use string map to record ip. code as below(in part2.zip):

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <dirent.h>
#include <pthread.h>
#include "strmap.h"//use string map

int file_flag[25]={0};//set flag for 25 files
pthread_mutex_t mutex;
pthread_mutex_t map_lock;
int ip_num;
StrMap *map;

int check_file(char filename[])
{
  pthread_mutex_lock(&mutex);//add lock
  char buf[2];
  buf[0]=filename[6];
  if(filename[7]!='.')
     {buf[1]=filename[7];}
  int tmp=atoi(buf);
  if(file_flag[tmp-1]==0)
  {
    file_flag[tmp-1]=1;
    pthread_mutex_unlock(&mutex);//unlock
    return 1;
  }
  else
  {
   pthread_mutex_unlock(&mutex);//unlock
   return 0;
  }
};

void *open_dir_read_file(void* arg)
{
        char* open_add=(char *)arg;
        DIR *dir;   //directory stream
        FILE *file; //file stream
        struct dirent *ent; // directory entry structure
        char *line = NULL;  // pointer to
        size_t len = 1000;  //the length of bytes getline will allocate
        size_t read;
        long lines=0;
        char* ip=NULL;
          char full_filename[256];  //will hold the entire file name to read
        // try to open the directory given by the argument
    if ((dir = opendir (open_add)) != NULL)
    {
        /* print all the files and directories within directory */
        while ((ent = readdir (dir)) != NULL)
        {
                //printf ("%s\n", ent->d_name);
            // Check if the list is a regular file
            if(ent->d_type == DT_REG)
            {
                // Create the absolute path of the filename
                snprintf(full_filename, sizeof full_filename, "./%s%s\0", open_add, ent->d_name);

                // open the file
                              if(check_file(ent->d_name)==1)//check if this file was read and use mutex
                                {
                                    int tmp_num=0;

                                    FILE* file = fopen(full_filename, "r");
                                   // file was not able to be open
                                   if (file != NULL)
                                   {
                                  // Print out each line in the file
                                        while ((read = getline(&line, &len, file)) != -1)

                                       {
                                            lines++;
                                        ip = strtok (line, " ");
                                        pthread_mutex_lock(&map_lock); // lock the mutex associated
with minimum_value and update the variable as required
                                        int flag = sm_exists(map, ip); // sm_get
                                        if (flag == 0)
                                                 { // Not found, add ip into map
                                                   sm_put(map, ip, "");
                                                   ip_num++; // Increase the global ip num
                                                 tmp_num++;
                                                   }
                                        pthread_mutex_unlock(&map_lock); // unlock the mutex
                                           }
                                      printf("%s  this file's adding num: %d\n",
ent->d_name,tmp_num);
```

```
                                          fclose(file);
                                      }

                                  }
            }
      }
        // Close the directory structure
        closedir (dir);
    }
    else
    {
        /* could not open directory */
        perror ("");
        //return -1;
    }
};



int main(int argc, char *argv[])
{
    // check the arguments
    if(argc < 2)
    {
        printf("Not enough arguments supplied\n");
        return -1;
    }

    if(argc > 2)
    {
        printf("Too many arguments supplied\n");
        return -1;
    }
        map = sm_new(300);
        printf("%s\n",argv[1]);
        pthread_mutex_init(&mutex,NULL);
        pthread_mutex_init(&map_lock, NULL);
        int ret_1,ret_2;
        pthread_t tid_1,tid_2;

        ret_1=pthread_create(&tid_1,NULL,open_dir_read_file,(void *)argv[1]);
        ret_2=pthread_create(&tid_2,NULL,open_dir_read_file,(void *)argv[1]);

        pthread_join(tid_1,NULL);
        pthread_join(tid_2,NULL);
        printf("ip numbers: %d\n",ip_num);
        pthread_exit(NULL);
        return 0;
}
```

input instruction:

```
gcc -o fileparse strmap.c fileparse.c -lpthread
./fileparse access_logs/
```



ubuntu@ubuntu-desktop: ~/Documents/hw2_p1
File Edit View Terminal Help

```
ubuntu@ubuntu-desktop:~/Documents/hw2_p1$ gcc -o fileparse strmap.c fileparse.c
-lpthread
ubuntu@ubuntu-desktop:~/Documents/hw2_p1$ ./fileparse access_logs/
access_logs/
access8.log  this file's adding num: 154
access6.log  this file's adding num: 0
access13.log  this file's adding num: 28
access14.log  this file's adding num: 0
access7.log  this file's adding num: 0
access4.log  this file's adding num: 0
access20.log  this file's adding num: 0
access2.log  this file's adding num: 0
access18.log  this file's adding num: 0
access10.log  this file's adding num: 0
access5.log  this file's adding num: 0
access24.log  this file's adding num: 0
access22.log  this file's adding num: 0
access16.log  this file's adding num: 0
access25.log  this file's adding num: 0
access12.log  this file's adding num: 0
access9.log  this file's adding num: 0
access23.log  this file's adding num: 0
access11.log  this file's adding num: 0
access15.log  this file's adding num: 0
access19.log  this file's adding num: 0
access17.log  this file's adding num: 0
access3.log  this file's adding num: 0
access1.log  this file's adding num: 0
access21.log  this file's adding num: 0
ip numbers: 182
ubuntu@ubuntu-desktop:~/Documents/hw2_p1$
```