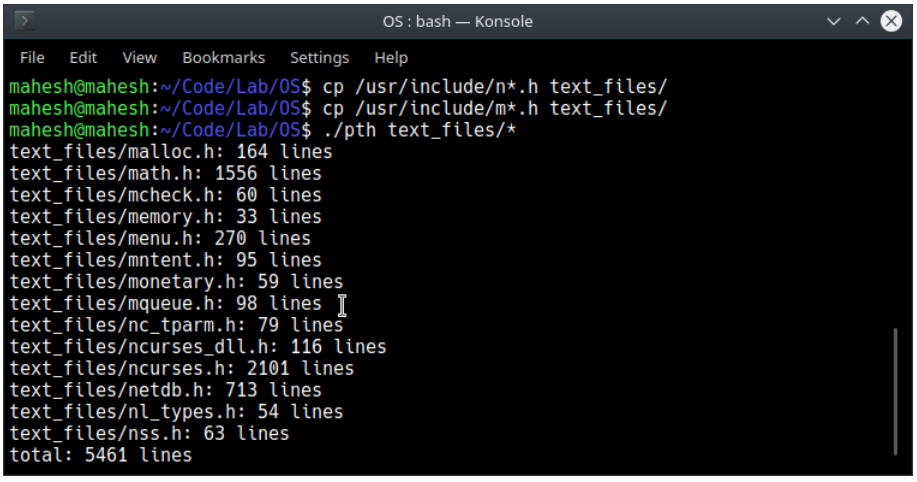
Program 3.

Write a program to demonstrate usage of pthread library.

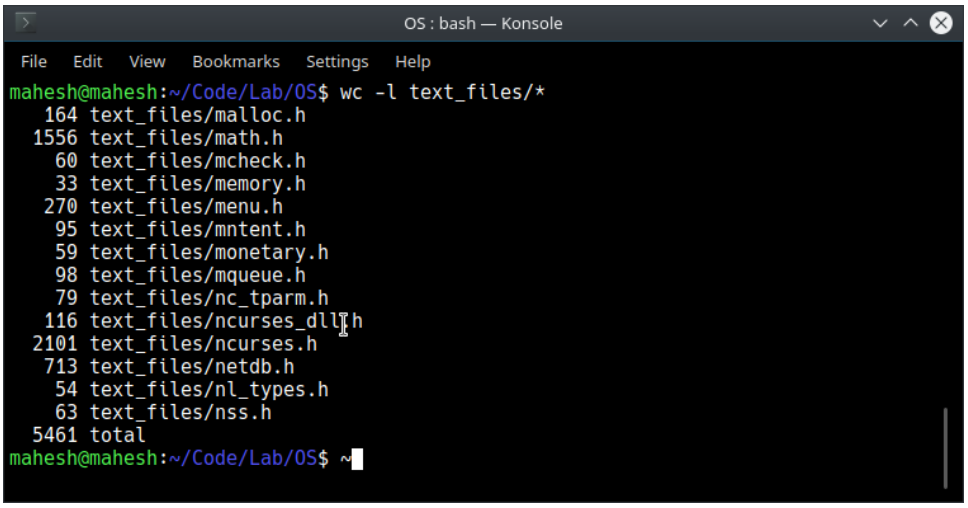
```
#include<pthread.h>
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<unistd.h>

// This program takes n files as input from arguments
// Print number of lines in each file and then total of lines

// Count line breaks
// n_breaks + 1 lines will be there, unless we encounter EOF in
beginning

struct input {
     char *filename;
     int *slot; // write the answer in an integer array
}; // for use by linecount function

int count_breaks(char *buf, ssize_t sz) {
     ssize_t i = 0;
     int count = 0;
     for(i=0;i<sz;i++) {
          if (buf[i] == '\n') count++;
     }
     return count;
}

void *linecount(void *inp) {
     struct input *in = (struct input *)inp;
     char buf[256];
     ssize_t rd;
     int fd = open(in->filename, O_RDONLY);
     if (fd < 0) {
          printf("Cannot open %s\n", in->filename);
          *(in->slot) = -1;
          return NULL;
     }
     int count = 0;
     while((rd = read(fd,buf,256)) > 0) {
          count += count_breaks(buf,rd);
     }
     *(in->slot) = count;
     return NULL;
}

struct input *create_input(char *filename, int *slot) {
     struct input *inp = malloc(sizeof(struct input));
     inp->filename = filename;
     inp->slot = slot;
```

```
     return inp;
}

int main(int argc, char **argv) {
     pthread_t *threads = malloc((argc-1)*sizeof(pthread_t));
     int *results = malloc((argc-1)*sizeof(int));
     pthread_attr_t attr;
     pthread_attr_init(&attr);
     int i, total = 0;

     if (argc < 2) {
          fprintf(stderr, "Needs at least 1 argument\n");
          return 1;
     }

     for(i = 1; i < argc; i++) {
          struct input *in = create_input(argv[i], &results[i-1]);
          // an intentional memory leak, will clear on exiting
anyway;
          pthread_create(&threads[i-1], &attr, linecount, in);
     }

     for(i=1; i < argc; i++) {
          int jstatus = pthread_join(threads[i-1], NULL);
          if (jstatus != 0) {
               fprintf(stderr, "cannot join thread for file %s",
argv[i]);
               continue;
          }
          if(results[i-1] >= 0) {
               printf("%s: %d lines\n", argv[i], results[i-1]);
               total += results[i-1];
          }
     }

     printf("total: %d lines\n", total);
     free(threads);
     free(results);
     return 0;
}
```

Output:



For comparison below is the output of unix wc -l



Case when a file does not exist