**POOJA GOEL**

**22210029**

**Assignment 1**

**CS 612 - Computer Systems**

**Question:**

**Word-count:**

**Using Lists to Count Words:**

Step 1:

Defining PINTOS\_LIST in word\_count\_l.c file

#define PINTOS\_LIST

Step2: complete functions in word\_count\_l.c file

void init\_words(word\_count\_list\_t\* wclist) { /\* TODO \*/

list\_init (wclist);

}

size\_t len\_words(word\_count\_list\_t\* wclist) {

  /\* TODO \*/

return list\_size(wclist);

}

word\_count\_t\* find\_word(word\_count\_list\_t\* wclist, char\* word) {

  /\* TODO \*/

   struct list\_elem \*a;

   for(a= list\_begin(wclist); a!=list\_end(wclist);a=list\_next(a))

   {

    struct word\_count\* b=list\_entry(a,struct word\_count,elem);

    if(strcmp(b->word,word)==0)

    return(b);

   };

  return NULL;

}

word\_count\_t\* add\_word(word\_count\_list\_t\* wclist, char\* word) {

  /\* TODO \*/

 word\_count\_t \*a;

a=find\_word(wclist,word);

if(a!=NULL)

{

  a->count=a->count+1;

  return a;

}

else{

  struct word\_count \*new =(struct word\_count \*)malloc(sizeof(struct word\_count));

  new->count=1;

  new->word=word;

  list\_push\_back(wclist, &(new->elem));

  return new;

}

  return NULL;

}

void fprint\_words(word\_count\_list\_t\* wclist, FILE\* outfile) { /\* TODO \*/

struct list\_elem \*a;

for(a= list\_begin(wclist); a!=list\_end(wclist);a=list\_next(a))

   {

    struct word\_count\* b=list\_entry(a,struct word\_count,elem);

    fprintf(outfile,"%6d %6s\n", b->count,b->word);

}

}

static bool less\_list(const struct list\_elem\* ewc1, const struct list\_elem\* ewc2, void\* aux) {

  /\* TODO \*/

struct word\_count\* a=list\_entry(ewc1,struct word\_count,elem);

struct word\_count\* b=list\_entry(ewc2,struct word\_count,elem);

if(less\_count(a,b)|| (!less\_count(a,b)&& !less\_count(b,a)&& less\_word(a,b)))

//if(a->count<b->count)

return true;

  return false;

}

void wordcount\_sort(word\_count\_list\_t\* wclist,bool less(const word\_count\_t\*, const word\_count\_t\*)) {

list\_sort(&(wclist->lst), less\_list, less);

list\_sort(wclist, less\_list, less);

}

Step3:

Compiling the code:

root@DESKTOP-TN1I17B:/mnt/f/Semester 1/Assignment1/assignment1/word-count# make

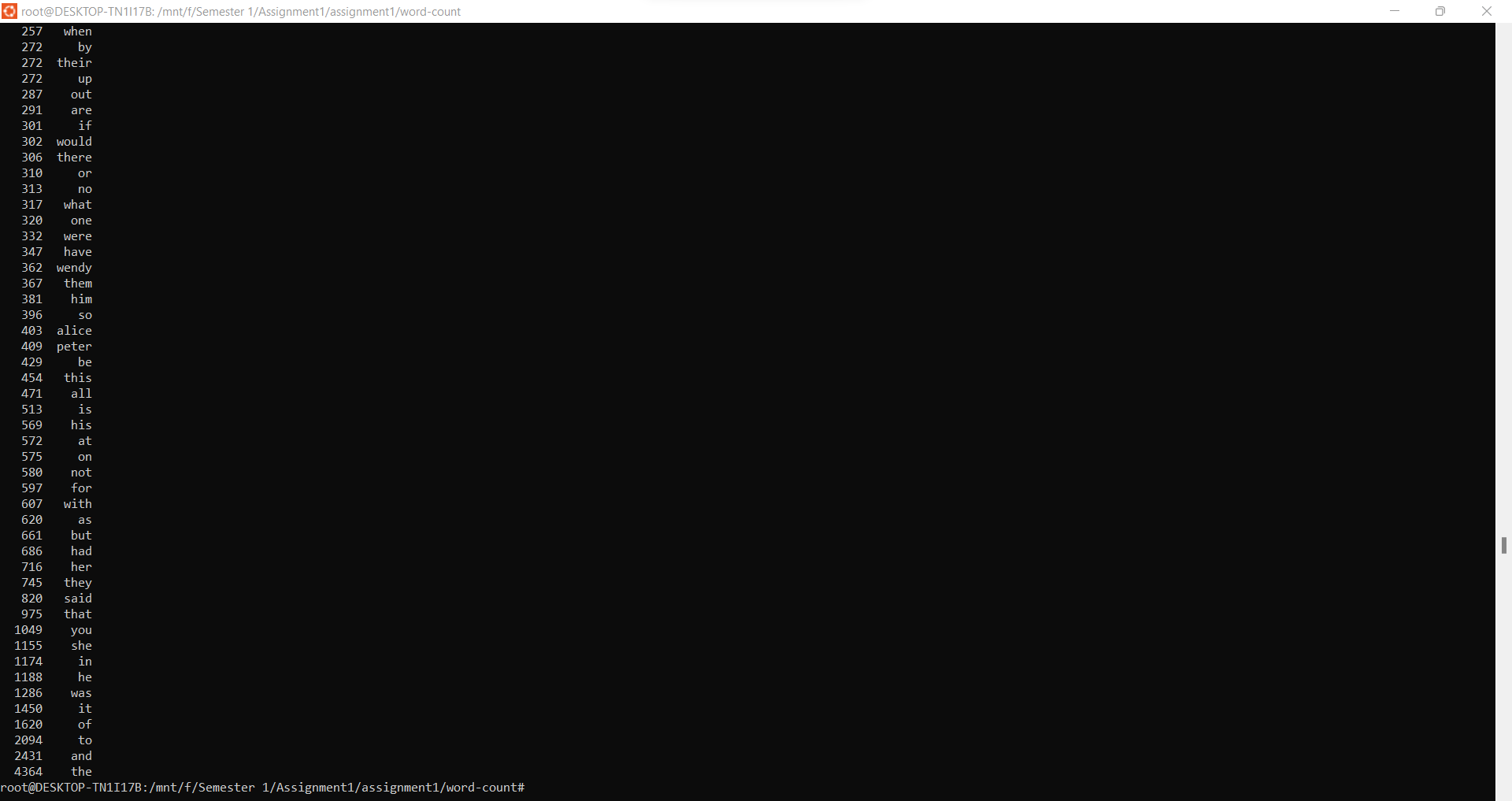
gcc -g3 -Wall -std=gnu99 -c word\_count\_multithread.c -o word\_count\_multithread.o

gcc -pthread word\_count\_multithread.o word\_helpers.o -o word\_count\_m

root@DESKTOP-TN1I17B:/mnt/f/Semester 1/Assignment1/assignment1/word-count# ./lwords gutenberg/alice.txt gutenberg/peter.txt



Count list is displaying is shown below:



**Using Multiple Threads to Count Words**

**Creating a new c file :- word\_count\_multithread.c**

**File uses pthreads and creates multiple threads to read files.**

/\*this file is creating multiple threads to read file and updating shared word\_count\_list\_t\* w\*/

#define PINTOS\_LIST

#define PTHREADS

#include<pthread.h>

#include "word\_count.h"

#include "word\_helpers.h"

#include "list.c"

#include "debug.c"

#include "word\_count\_l.c"

word\_count\_list\_t\* w; // defining global word\_count\_list\_t

pthread\_t thread[10]; //creating thread array globally

void\* threadWordCount(void\* arg)

{

    FILE\* f = fopen((char \*) arg, "r");

    count\_words(w, f);

    fclose(f);

    return NULL;

}

int main(int argc, char \*argv[]){

    w = (word\_count\_list\_t\*)malloc(sizeof(word\_count\_list\_t));

    init\_words(w);

    pthread\_mutex\_init(&w->lock, NULL);  // initializing mutex

    for(int i=0; i<argc; ++i){

      pthread\_create(&thread[i], NULL, threadWordCount, argv[i]);

    }

    for(int i=0; i<argc; ++i){

      pthread\_join(thread[i], NULL);

    }

    wordcount\_sort(w,NULL); // sorting updated w list

    fprint\_words(w, stdout);

    exit(0);

}

**Step:**

**Updating all functions in word\_count\_l.c**

#define PINTOS\_LIST

#ifndef PINTOS\_LIST

//#error "PINTOS\_LIST must be #define'd when compiling word\_count\_l.c"

//#define PINTOS\_LIST // defining PINTOS\_LIST for the compilation

#define PINTOS\_LIST

//#define PTHREADS

#endif

#include "word\_count.h"

#include "list.h"

#include "word\_helpers.h"

void init\_words(word\_count\_list\_t\* wclist) { /\* TODO \*/

#ifdef PINTOS\_LIST

#ifdef PTHREADS

   list\_init (&wclist->lst);

#else

list\_init (wclist);

#endif

#endif

}

size\_t len\_words(word\_count\_list\_t\* wclist) {

  /\* TODO \*/

#ifdef PINTOS\_LIST

#ifdef PTHREADS

  return list\_size(&(wclist->lst));

#else

return list\_size(wclist);

#endif

#endif

}

word\_count\_t\* find\_word(word\_count\_list\_t\* wclist, char\* word) {

  /\* TODO \*/

   #ifdef PINTOS\_LIST

   #ifdef PTHREADS

   struct list\_elem \*a;

   for(a= list\_begin(&(wclist->lst)); a!=list\_end(&(wclist->lst));a=list\_next(a))

   {

    struct word\_count\* b=list\_entry(a,struct word\_count,elem);

    if(strcmp(b->word,word)==0)

    return(b);

   };

   return NULL;

  #else

   struct list\_elem \*a;

   for(a= list\_begin(wclist); a!=list\_end(wclist);a=list\_next(a))

   {

    struct word\_count\* b=list\_entry(a,struct word\_count,elem);

    if(strcmp(b->word,word)==0)

    return(b);

   };

  return NULL;

  #endif

  #endif

}

word\_count\_t\* add\_word(word\_count\_list\_t\* wclist, char\* word) {

  /\* TODO \*/

 word\_count\_t \*a;

  #ifdef PINTOS\_LIST

  #ifdef PTHREADS

  a=find\_word(wclist,word);

if(a!=NULL)

{ pthread\_mutex\_lock(&(wclist->lock));

  a->count=a->count+1;

  pthread\_mutex\_unlock(&(wclist->lock));

  return a;

}

else{

  struct word\_count \*new =(struct word\_count \*)malloc(sizeof(struct word\_count));

  new->count=1;

  new->word=word;

  pthread\_mutex\_lock(&(wclist->lock));

  list\_push\_back(&(wclist->lst),&(new->elem));

  pthread\_mutex\_unlock(&(wclist->lock));

  return new;

}

  return NULL;

#else

a=find\_word(wclist,word);

if(a!=NULL)

{

  a->count=a->count+1;

  return a;

}

else{

  struct word\_count \*new =(struct word\_count \*)malloc(sizeof(struct word\_count));

  new->count=1;

  new->word=word;

  list\_push\_back(wclist, &(new->elem));

  return new;

}

  return NULL;

  #endif

  #endif

}

void fprint\_words(word\_count\_list\_t\* wclist, FILE\* outfile) { /\* TODO \*/

struct list\_elem \*a;

#ifdef PINTOS\_LIST

#ifdef PTHREADS

   for(a= list\_begin(&(wclist->lst)); a!=list\_end(&(wclist->lst));a=list\_next(a))

   {

    struct word\_count\* b=list\_entry(a,struct word\_count,elem);

    fprintf(outfile,"%6d %20s\n", b->count,b->word);

}

#else

for(a= list\_begin(wclist); a!=list\_end(wclist);a=list\_next(a))

   {

    struct word\_count\* b=list\_entry(a,struct word\_count,elem);

    fprintf(outfile,"%6d %6s\n", b->count,b->word);

}

#endif

#endif

}

static bool less\_list(const struct list\_elem\* ewc1, const struct list\_elem\* ewc2, void\* aux) {

  /\* TODO \*/

struct word\_count\* a=list\_entry(ewc1,struct word\_count,elem);

struct word\_count\* b=list\_entry(ewc2,struct word\_count,elem);

if(less\_count(a,b)|| (!less\_count(a,b)&& !less\_count(b,a)&& less\_word(a,b)))

//if(a->count<b->count)

return true;

  return false;

}

void wordcount\_sort(word\_count\_list\_t\* wclist,bool less(const word\_count\_t\*, const word\_count\_t\*)) {

#ifdef PINTOS\_LIST

#ifdef PTHREADS

list\_sort(&(wclist->lst), less\_list, less);

#else

list\_sort(wclist, less\_list, less);

#endif

#endif

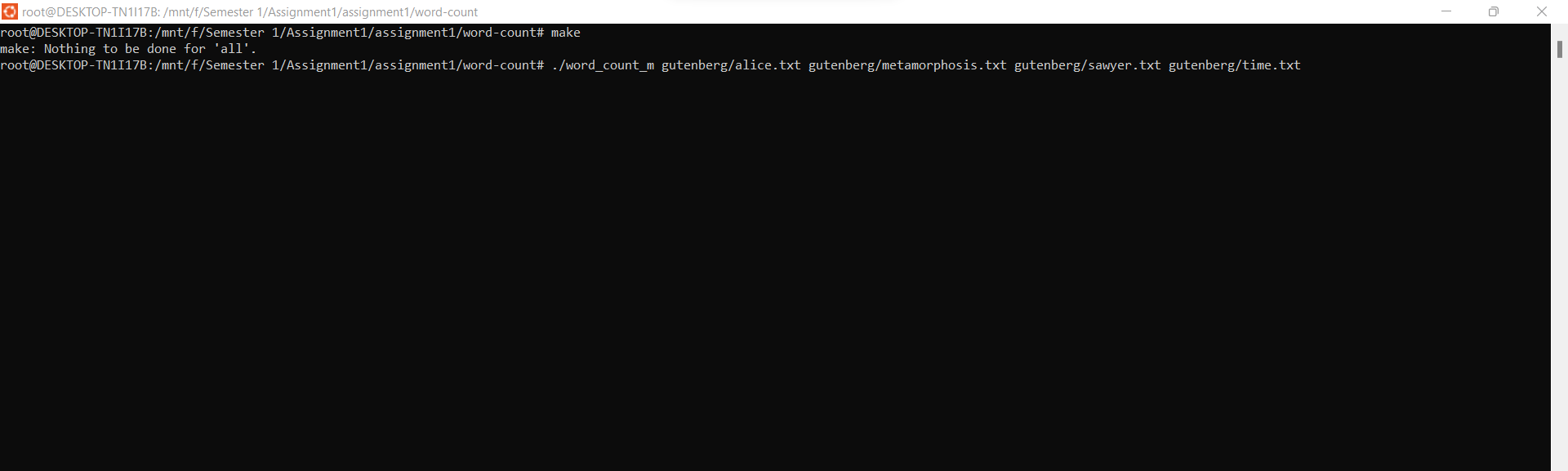
}

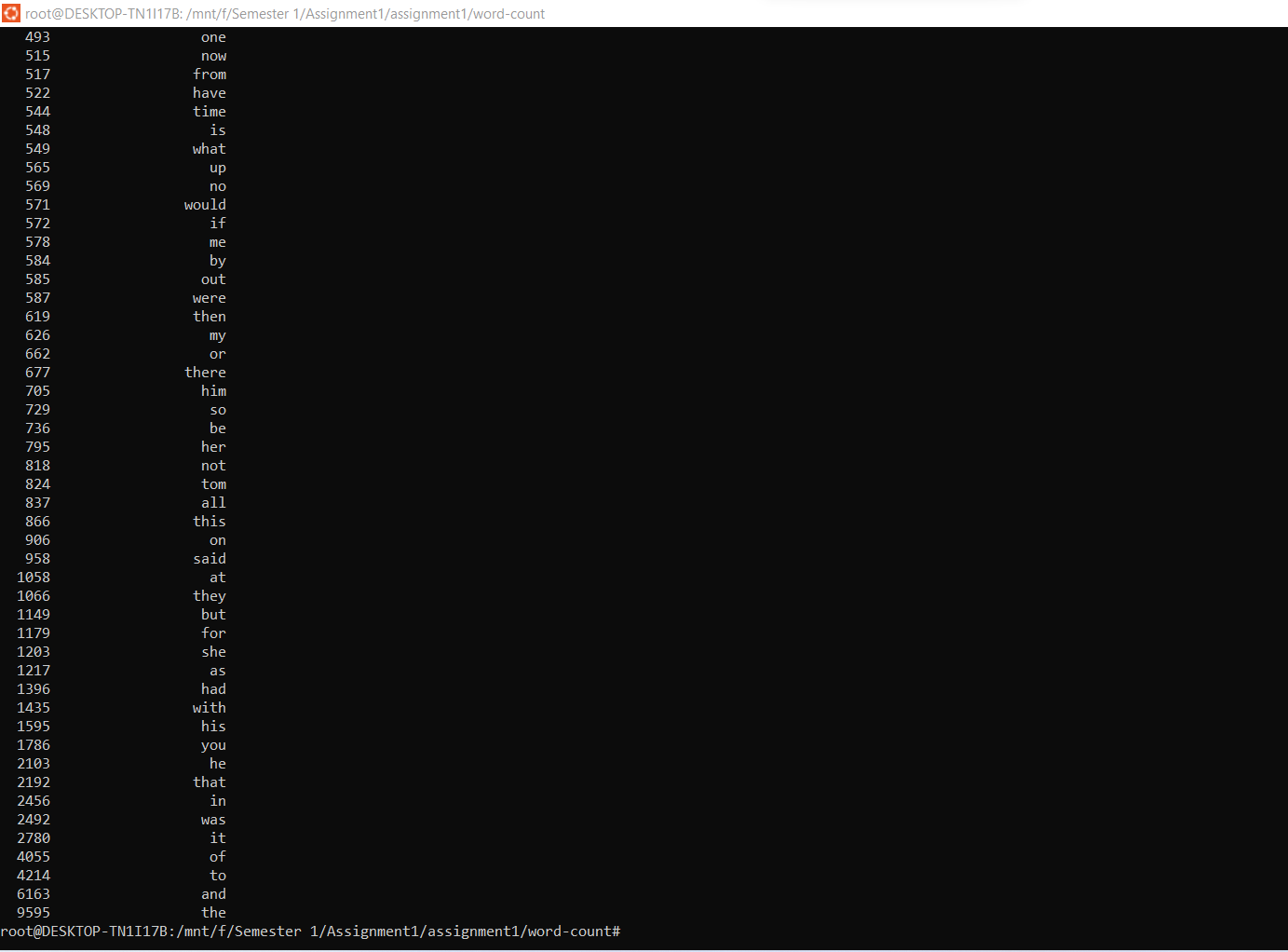
**Step:**

**Perform changes in make file accordingly**

**Step:**

**Compiling file**

****

****

**Question:**

**Shell**

int cmd\_cwd(struct tokens\* tokens); //defining cmd\_cwd function

int cmd\_cd(struct tokens\* tokens);// defining cmd\_cwd function

{cmd\_cwd, "cwd", "prints the current working directory to standard output,"},

    {cmd\_cd, "cd", "takes one argument, a directory path, and changes the current working directory to that directory."}

int cmd\_cd(unused struct tokens\* tokens)

{

  char\*filename; //defining charater pointer

  filename=tokens\_get\_token(tokens,1);

  if(filename==NULL)

  {

fprintf(stderr,"please specify a directory.\n");

  }

else if(chdir(filename)!=0)

fprintf(stderr,"%s not found.\n",filename);

return 1;

}

int cmd\_cwd(unused struct tokens\* tokens) // cmd\_cwd function uses getcwd function to read

{

  char cwd[PATH\_MAX];

   if (getcwd(cwd, sizeof(cwd)) != NULL) {

       printf("Current working directory: %s\n", cwd);

   } else {

       perror("getcwd() error");

       return 1;

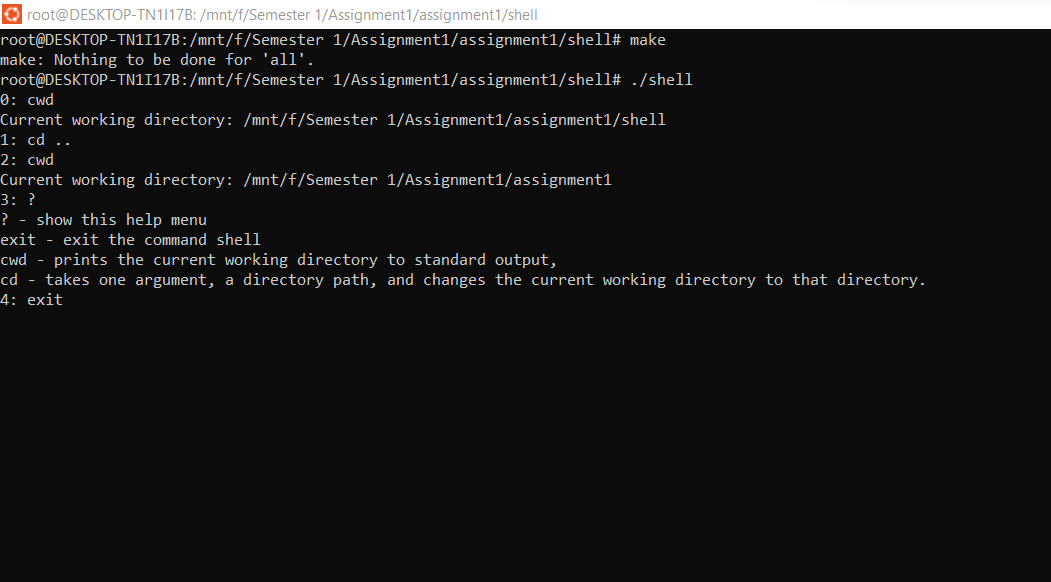
   }

   return 0;

}

**Step:**

**Compiling**

****