**Assignment II**

**Exercise 1 : helloworld.c**



Description

#include <stdio.h> // เรียกใช้งาน Library “Standard input/output”

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

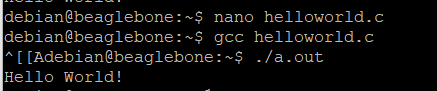
// เปิดใช้งานฟังก์ชันหลัก โดยที่กำหนดให้มี 2 argument โดยมี Argument Cout คือ argc ทำหนีที่นับจำนวน Argument ที่รับเข้ามา และ Vector Argument หรือ argv ที่จะทำหน้าที่เป็น Pointer

printf("Hello World!\n"); // แสดงคำว่า “Hello World!” ออกทางหน้าจอ

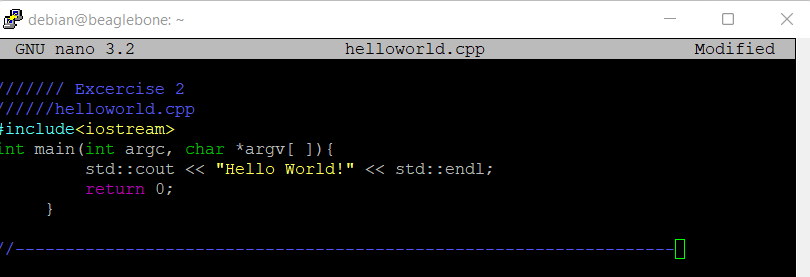
return 0; // จบการทำงาน

}

การจำลองการทำงาน



**Exercise 2 : helloworld.cpp**



Description

#include<iostream> // เรียกใช้งาน Library “iostream”

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

// เปิดใช้งานฟังก์ชันหลัก โดยที่กำหนดให้มี 2 argument โดยมี Argument Cout คือ argc ทำหนีที่นับจำนวน Argument ที่รับเข้ามา และ Vector Argument หรือ argv ที่จะทำหน้าที่เป็น Pointer

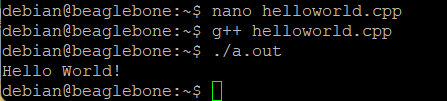
std::cout << "Hello World!" << std::endl;

// แสดงข้อความ Hello World! ออกสู่หน้าจอด้วยคำสั่ง std::cout และใช้คำสั่ง std::endl เพื่อจบบรรทัดและขึ้นบรรทัดใหม่ หากใช้คำสั่งพร้อมกับ cout ต้องมี << คั่นเสมอ

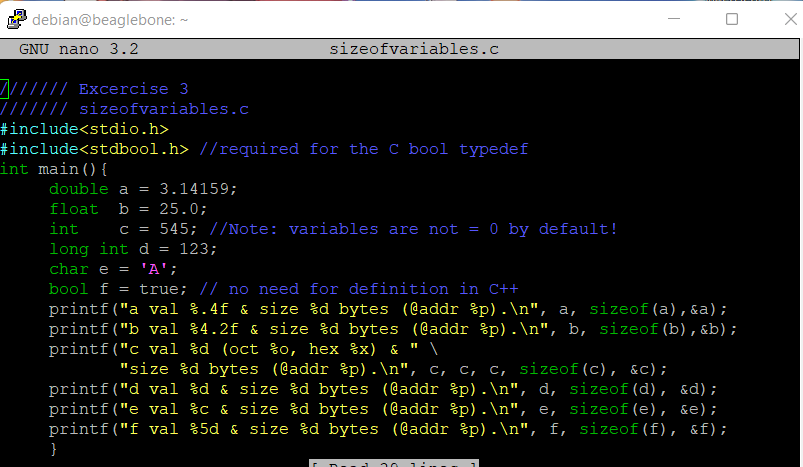
return 0; // จบการทำงาน

}

การจำลองการทำงาน



**Exercise 3 : sizeofvariables.c**

****

Description

#include<stdio.h> // เรียกใช้งาน Library “stdio.h”

#include<stdbool.h> //required for the C bool typedef // เรียกใช้งาน Library “stdbool.h”

int main(){

double a = 3.14159; // ประกาศตัวแปร a เป็นชนิด double (ตัวแปรที่เก็บค่าทศนิยมได้ แต่มีพื้นที่สำหรับเก็บข้อมูลเยอะกว่า float 2 เท่า) ที่รับเก็บ 3.14159

float b = 25.0; // ประกาศตัวแปร b เป็นชนิด float (ตัวแปรที่เก็บค่าทศนิยมได้) ที่เก็บค่า 25.0

int c = 545; //Note: variables are not = 0 by default! //ประกาศตัวแปร c เป็นชนิด Int ที่เก็บค่า 545

long int d = 123; //ประกาศตัวแปร d เป็นชนิด long Int (เก็บข้อมูลได้มากกว่า int 2 bytes) ที่เก็บค่า 123

char e = 'A'; //ประกาศตัวแปร e เป็นชนิด char ที่เก็บค่า ‘A’

bool f = true; // no need for definition in C++

printf("a val %.4f & size %d bytes (@addr %p).\n", a, sizeof(a),&a);

//แสดงค่าของ a ด้วยทศนิยม 4 ตำแหน่ง (%.4f) และ ขนาดของข้อมูลแสดงในรูปของจำนวนเต็ม (%d) และ Address ของ ตัวแปร a แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร a (%p) แล้วขึ้นบรรทัดใหม่ โดยที่นำค่ามาจาก a , sizeof(a) , address a

printf("b val %4.2f & size %d bytes (@addr %p).\n", b, sizeof(b),&b);

//แสดงค่าของ b ด้วยหลักจำนวน 4 หลักและทศนิยม 2 ตำแหน่ง (%4.2f) และ ขนาดของข้อมูลแสดงในรูปของจำนวนเต็ม (%d) และ Address ของ ตัวแปร d แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร d (%p) แล้วขึ้นบรรทัดใหม่ โดยที่นำค่ามาจาก b , sizeof(b) , address b (& : Address Operator)

printf("c val %d (oct %o, hex %x) & " \

"size %d bytes (@addr %p).\n", c, c, c, sizeof(c), &c);

//แสดงค่าของ c เป็นจำนวนเต็ม (%d) และ แสดงค่าของ c ในรูปของฐานแปด (%o) และฐานสิบหก (%x) เนื่องจากไม่ต้องการให้คำสั่งยาวเกินไปจึงใช้คำสั่ง “ \ ” เพื่อบอกโปรแกรมว่าคำสั่งยังไม่จบ , แสดงขนาดของตัวแปร c ในรูปของ integer (%d) และ Address ของ ตัวแปร c แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร c (%p) โดยที่นำค่ามาจาก c , sizeof(c) , address c (& : Address Operator)

printf("d val %d & size %d bytes (@addr %p).\n", d, sizeof(d), &d);

//แสดงค่าของ d เป็นจำนวนเต็ม %d และ ขนาดของข้อมูลแสดงในรูปของจำนวนเต็ม (%d) และ Address ของ ตัวแปร d แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร d (%p) แล้วขึ้นบรรทัดใหม่ โดยที่นำค่ามาจาก d , sizeof(d) , address d (& : Address Operator)

printf("e val %c & size %d bytes (@addr %p).\n", e, sizeof(e), &e);

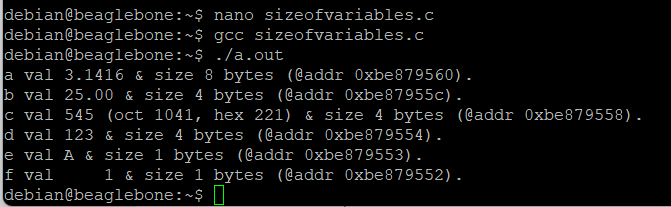
//แสดงค่าของ e เป็นตัวอักษร (char : %c) และ ขนาดของข้อมูลแสดงในรูปของจำนวนเต็ม (%d) และ Address ของ ตัวแปร e แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร e (%p) แล้วขึ้นบรรทัดใหม่ โดยที่นำค่ามาจาก e , sizeof(e) , address e (& : Address Operator)

printf("f val %5d & size %d bytes (@addr %p).\n", f, sizeof(f), &f);

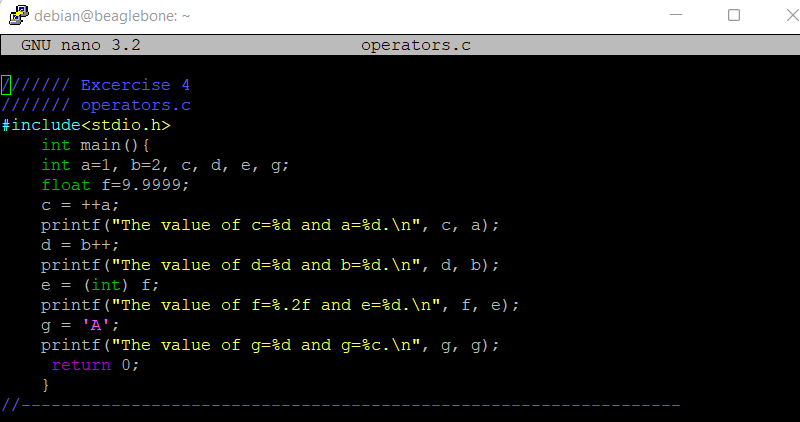
//แสดงค่าของ f เป็นจำนวนเต็ม 5 ตัว (%5d) หากไม่มีโปรแกรมจะทำการใส่ช่องว่างที่ข้างหน้าแทน และ ขนาดของข้อมูลแสดงในรูปของจำนวนเต็ม (%d) และ Address ของตัวแปร f แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร f (%p) แล้วขึ้นบรรทัดใหม่ โดยที่นำค่ามาจาก f , sizeof(f) , address f (& : Address Operator)

}

การจำลองการทำงาน

****

**Exercise 4 : Operators.c**

****

Description

#include<stdio.h> // เรียกใช้งาน Library “stdio.h”

int main(){ // เปิดใช้งาน main function

int a=1, b=2, c, d, e, g; //ประกาศตัวแปร a , b เป็นตัวแปรชนิด integer ที่เก็บค่า 1 และ 2 ตามลำดับ และ ตัวแปร c , d , e และ g เป็นตัวแปรชนิด integer

float f=9.9999; // ประกาศตัวแปร f เป็นตัวแปรชนิด float ที่เก็บค่า 9.9999

c = ++a; // ให้ตัวแปร c เก็บค่าจาก a+1

printf("The value of c=%d and a=%d.\n", c, a); // แสดง ค่าของ c และ a ในรูปของจำนวนเต็ม (integer : int : %d) แล้วขึ้นบรรทัดใหม่

d = b++; // ให้ตัวแปร d เก็บค่าจาก b แล้วบวกค่า b ไป 1

printf("The value of d=%d and b=%d.\n", d, b); // แสดง ค่าของ d และ b ในรูปของจำนวนเต็ม (integer : int : %d) แล้วขึ้นบรรทัดใหม่

e = (int) f; // ให้ตัวแปร e เก็บค่าของ f ในรูปของจำนวนเต็ม (integer : int)

printf("The value of f=%.2f and e=%d.\n", f, e); แสดง ค่าของ f ในรูปของทศนิยม 2 ตำแหน่ง (%.2f) และ e ในรูปของจำนวนเต็ม (%d) แล้วขึ้นบรรทัดใหม่

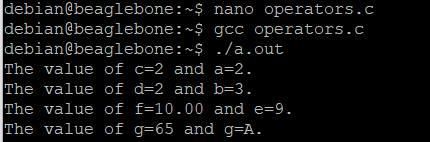
g = 'A'; // ให้ตัวแปร g รับค่าของ A ที่เป็นรหัส ASCII (นำค่า A ไปเทียบใน ASCII table แล้วแสดงในรูปฐานสิบ)

printf("The value of g=%d and g=%c.\n", g, g); // แสดง ค่าของ g ในรูปของจำนวนเต็ม (%d) และ ค่าของ g ในรูปของตัวอักษร (Character : Char : %c) แล้วขึ้นบรรทัดใหม่

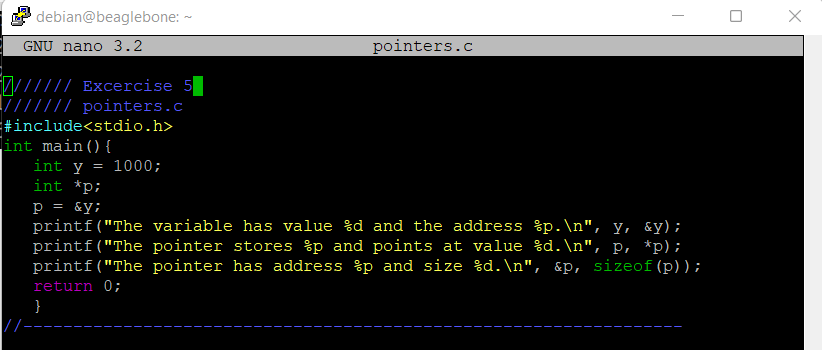
return 0; // จบการทำงาน

}

การจำลองการแสดงผล



**Exercise 5 : pointers.c**



Description

#include<stdio.h> //เรียกใช้ฟังก์ชัน “stdio.h”

int main(){ // เปิดใช้งาน main function

int y = 1000; // ประกาศตัวแปร y ชนิด integer ที่รับค่า 1000

int \*p; // ประกาศตัวแปร p เป็น Pointer

p = &y; // กำหนดให้ p ที่เป็น pointer ชี้ไปยัง Address ของตัวแปร y

printf("The variable has value %d and the address %p.\n", y, &y);

// แสดง ค่าของ yในรูปของจำนวนเต็ม (%d) และ Address ของตัวแปร y ด้วย pointer (%p) แล้วขึ้นบรรทัดใหม่

printf("The pointer stores %p and points at value %d.\n", p, \*p);

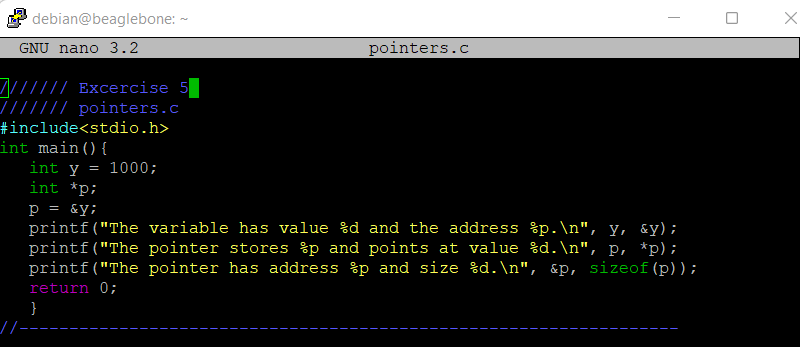
// แสดง address ที่ p (Pointer) ชี้อยู่ และ ค่าที่เก็บอยู่ใน address นั้นๆ แล้วขึ้นบรรทัดใหม่

printf("The pointer has address %p and size %d.\n", &p, sizeof(p));

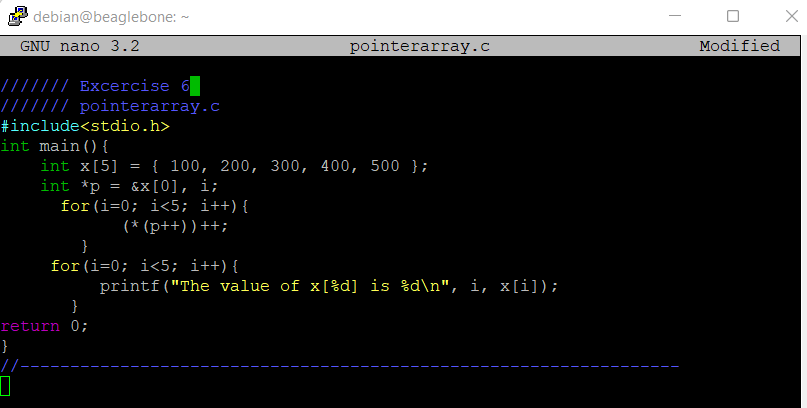
// แสดง address ที่ p (Pointer) ชี้อยู่ และขนาดของข้อมูล ณ ตำแหน่งนั้นๆ แล้วขึ้นบรรทัดใหม่

return 0; //จบการทำงาน

}

การจำลองการทำงาน

**Exercise 6 : pointerarray.c**



Description

#include<stdio.h> // เรียกใช้งาน Library “stdio.h”

int main(){ // เปิดใช้งาน main function

int x[5] = { 100, 200, 300, 400, 500 }; // ประกาศ Array x ชนิด Integer ที่เก็บค่า 100 , 200 , 300 , 400 และ 500 ตามลำดับ

int \*p = &x[0], i; // กำหนดให้ p เป็น Pointer ชี้ไปยังตำแหน่ง Address 0 และประกาศตัวแปร i เป็นชนิด Integer

for(i=0; i<5; i++){ // กำหนดให้โปรแกรมวนลูป 5 รอบ

(\*(p++))++; // เพิ่มค่า ณ ตำแหน่งที่ pointer p ชี้อยู่ (x[0]) ไป +1 และทำการเลื่อนตำแหน่งของ pointer p ไป +1

}

for(i=0; i<5; i++){ // กำหนดให้โปรแกรมวนลูป 5 รอบ

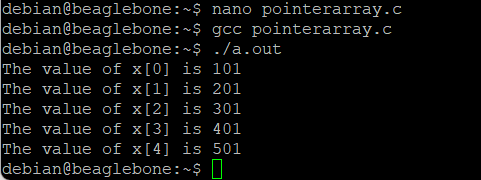
printf("The value of x[%d] is %d\n", i, x[i]); //แสดง ค่าของ Array x ในรูปของจำนวนเต็ม (%d) แล้วขึ้นบรรทัดใหม่

}

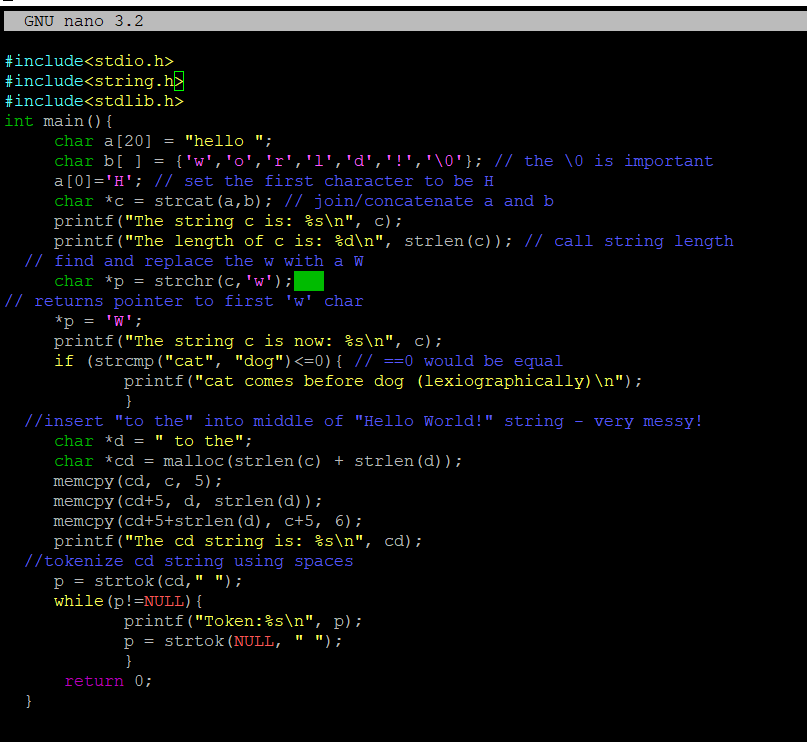
return 0; //จบการทำงาน

}

การจำลองการทำงาน



**Exercise 7 : cstrings.c**

****

Description

#include<stdio.h> //เรียกใช้งานฟังก์ชัน “stdio.h”

#include<string.h> //เรียกใช้งานฟังก์ชัน “string.h”

#include<stdlib.h> //เรียกใช้งานฟังก์ชัน “stdlib.h”

int main(){ // เปิดใช้งาน main function

char a[20] = "hello "; // กำหนด Array a รับค่า “hello”แบบ string

char b[ ] = {'w','o','r','l','d','!','\0'}; // the \0 is important //กำหนดตัวแปร array b รับค่า w , o , r , l และ \0 โดยที่ \0 หมายถึง null เป็นการระบุว่าจบ Array

a[0]='H'; // set the first character to be H //กำหนดให้ Array a ตำแหน่งที่ 0 รับค่า H

char \*c = strcat(a,b); // join/concatenate a and b // กำหนดตัวแปร c เป็นตัวแปร Pointer มาชี้ ณ ตำแหน่งแรกที่เกิดจากการนำ Array b[] มาต่อท้าย Array a[]

printf("The string c is: %s\n", c); // แสดงข้อความ (%s) ที่ตัวแปร Pointer c ชี้อยู่ทั้งหมดจนกว่าจะเจอ \0 แล้วทำการขึ้นบรรทัดใหม่

printf("The length of c is: %d\n", strlen(c)); // call string length // แสดงความยาวของ c โดยใช้ฟังก์ชัน strlen ()

// find and replace the w with a W

char \*p = strchr(c,'w'); // ให้ Pointer p กลับไปชี้ยังตำแหน่ง ‘w’ ตัวแรกที่โปรแกรมพบ ที่เกิดจากการนำ Array b มาต่อท้าย array a โดยใช้ ฟังก์ชัน strchr ()

// returns pointer to first 'w' char

\*p = 'W'; // ให้ Pointer p ชี้ไปยังตำแหน่งของ ‘W’

printf("The string c is now: %s\n", c); แสดงข้อความ (%s) ที่ตัวแปร Pointer c ชี้อยู่ทั้งหมดจนกว่าจะเจอ \0 แล้วทำการขึ้นบรรทัดใหม่

if (strcmp("cat", "dog")<=0){ // ==0 would be equal // ใช้คำสั่งเปรียบเทียบ (strcmp()) ถ้าตัวอักษรที่เรานำไปเปรียบเทียบมาก่อนตัวเทียบจะให้ผลลัพธ์เป็น 1 แต่ถ้ามาทีหลังจะให้ผลลัพธ์เป็น -1 และถ้าเท่ากันหรือเหมือนกัน จะเป็น 0 ซึ่งในที่นี่ c มาก่อน d หากผ่านเงื่อนไข if ให้ไปบรรทัดต่อไป

printf("cat comes before dog (lexicographically)\n"); // แสดงข้อความ “cat comes before dog (lexicographically)” แล้วขึ้นบรรทัดใหม่

}

//insert "to the" into middle of "Hello World!" string - very messy!

char \*d = " to the"; // กำหนดให้ Pointer d เก็บข้อความ “to the” ซึ่งเป็นข้อความชนิด String

char \*cd = malloc(strlen(c) + strlen(d)); //ทำการจองหน่วยความจำโดยใช้ฟังก์ชัน malloc()โดยมีขนาดเท่ากับความยาว string ของ c และ d โดยหาความยาวได้จากฟังก์ชัน strlen() แล้วให้ Pointer cd ชี้ไปยังตำแหน่งดังกล่าว

memcpy(cd, c, 5); // ทำการคัดลอกข้อความจากตำแหน่งที่ c ชี้ มาใส่ยังตำแหน่งที่ cd ชี้ จำนวน 5 ตัว (เนื่องจากตัวอักษร 1 ตัว กินพื้นที่ 5 bytes)

memcpy(cd+5, d, strlen(d)); // ทำการคัดลอกข้อความจากตำแหน่งที่ d ชี้ มาใส่ยังตำแหน่งที่ cd โดยทำการเลื่อนตัวชี้ไป +5 (cd+5) จำนวนเท่ากับความของข้อความที่ Pointer d ชี้อยู่

memcpy(cd+5+strlen(d), c+5, 6); // ทำการคัดลอกข้อความจากตำแหน่งที่ c ชี้ มาใส่ยังตำแหน่งที่ cd ชี้อยู่โดยเลื่อนตำแหน่งไป 5 + ความยาวของข้อความที่ Pointer d ชี้อยู่ จำนวน 6 ตัว

printf("The cd string is: %s\n", cd); // แสดงข้อความทั้งหมดที่ cd ชี้อยู่ จนกว่าจะเจอ \0 แล้วทำการขึ้นบรรทัดใหม่

//tokenize cd string using spaces

p = strtok(cd," "); //ทำการแบ่งข้อความโดยใช้คำสั่ง “strtok()”

while(p!=NULL){ // ทำการวนลูป เมื่อ p ยังไม่เจอ \0 หรือ วนลูปจนกว่าจะเจอ \0 ใน Array

printf("Token:%s\n", p); // แสดงข้อความ (%s) ที่ถูกแบ่ง แล้วขึ้นบรรทัดใหม่

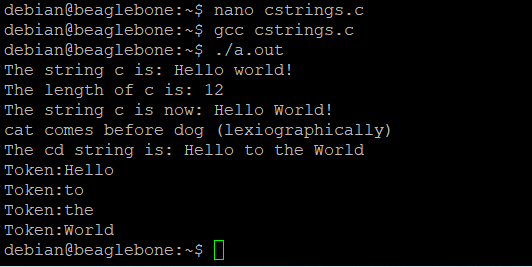
p = strtok(NULL, " "); //ทำการแบ่งข้อความต่อจากครั้งก่อน โดยกำหนดให้ str เป็น NULL ไม่เช่นนั้นโปรแกรมจะทำการแบ่งข้อความเดิมใหม่ ไม่ต่อจากที่ทำครั้งก่อนหน้า

}

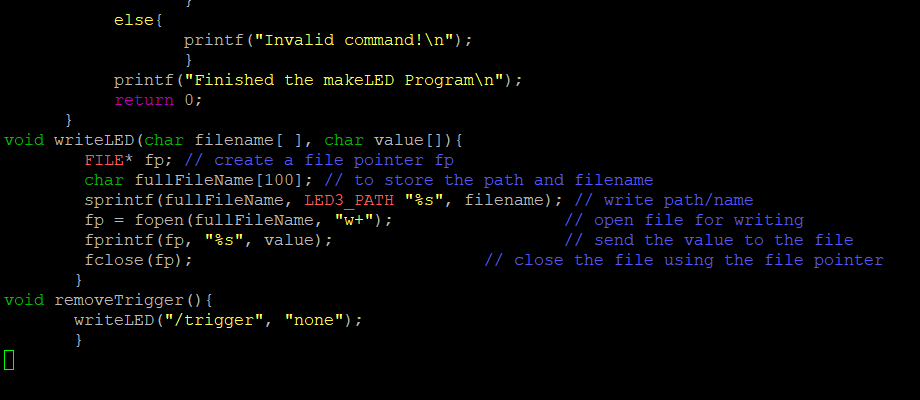
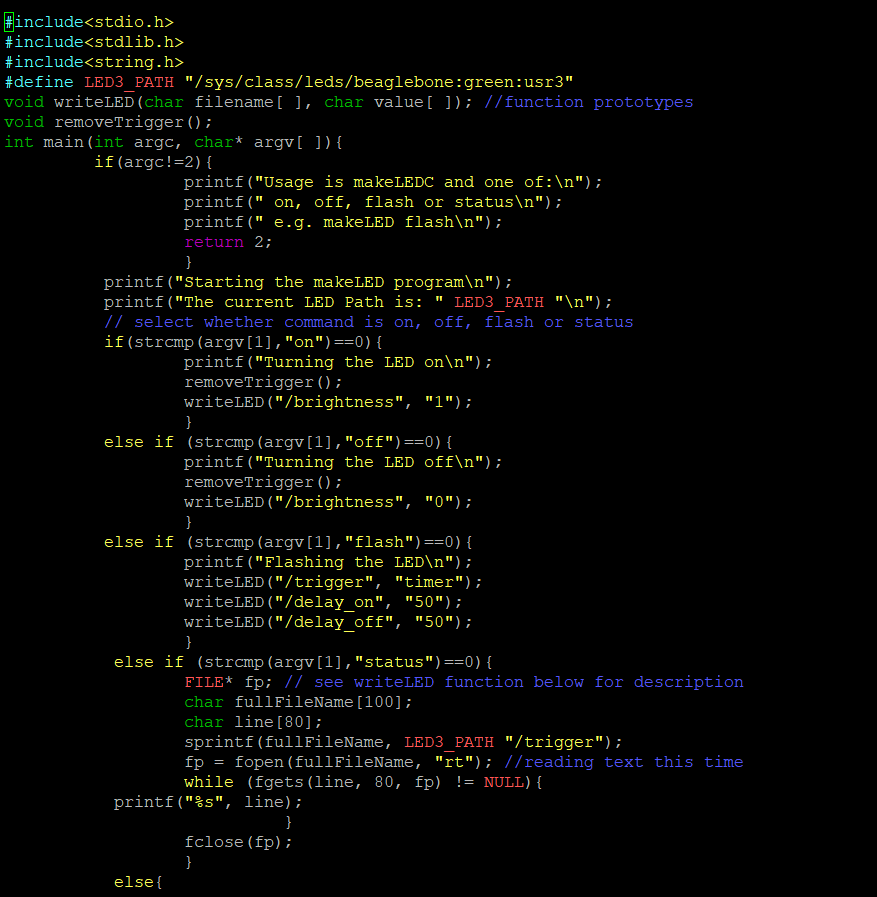
return 0; // จบการทำงาน

}

การจำลองการทำงาน



**Exercise 8 : makeLED.c**



Description

#include<stdio.h> // เรียกใช้งาน Library “stdio.h”

#include<stdlib.h> // เรียกใช้งาน Library “stdlib.h”

#include<string.h> // เรียกใช้งาน Library “string.h”

#define LED3\_PATH "/sys/class/leds/beaglebone:green:usr3" //กำหนดเส้นทางของ LED3\_path

void writeLED(char filename[ ], char value[ ]); //function prototypes // สร้างฟังก์ชันที่ชื่อว่า writeLED โดยภายในมี Parameters 2 ตัวที่เป็น Array คือ filename และ value

void removeTrigger(); // สร้างฟังก์ชันที่ชื่อว่า removeTrigger

int main(int argc, char\* argv[ ]){ // เปิดใช้งานฟังก์ชันหลัก โดยที่กำหนดให้มี 2 argument โดยมี Argument Cout คือ argc ทำหนีที่นับจำนวน Argument ที่รับเข้ามา และ Vector Argument หรือ argv ที่จะทำหน้าที่เป็น Pointer

if(argc!=2){ // ถ้า argument ที่เข้ามานับได้ไม่ถึง 2 ให้ไปบรรทัดต่อไป

printf("Usage is makeLEDC and one of:\n"); // แสดงข้อความ “Usage is makeLEDC and one of ” แล้วขึ้นบรรทัดใหม่

printf(" on, off, flash or status\n"); // แสดงข้อความ “on, off, flash or status” แล้วขึ้นบรรทัดใหม่

printf(" e.g. makeLED flash\n"); //แสดงข้อความ “e.g. makeLED flash” แล้วขึ้นบรรทัดใหม่

return 2; //Return ค่า 2 ออกไปแล้วจบการทำงาน

}

printf("Starting the makeLED program\n"); // แสดงข้อความ “Starting the makeLED program” แล้วขึ้นบรรทัดใหม่

printf("The current LED Path is: " LED3\_PATH "\n"); //แสดงจำนวนของเส้นทาง LED ในบรรจุบัน ที่ดึงมาจาก LED3\_PATH

// select whether command is on, off, flash or status

if(strcmp(argv[1],"on")==0){ // ถ้าทำการเปรียบเทียบระหว่างค่าใน argv[1] และ string “on” เหมือนกัน ให้ไปบรรทัดต่อไป

printf("Turning the LED on\n"); //แสดงข้อความ “Turning the LED on” แล้วขึ้นบรรทัดใหม่

removeTrigger(); //เรียกใช้ฟังก์ชัน removeTringger()

writeLED("/brightness", "1"); //เรียกใช้ฟังก์ชัน writeLED โดยกำหนดค่าที่จะส่งเข้าไปคือ “/brightness” และ “1”

}

else if (strcmp(argv[1],"off")==0){ //ถ้าทำการเปรียบเทียบระหว่างค่าใน argv[1] และ String “Off” เหมือนกัน ให้ไปบรรทัดต่อไป

printf("Turning the LED off\n"); //แสดงข้อความ “Turning the LED off” แล้วขึ้นบรรทัดใหม่

removeTrigger(); //เรียกใช้ฟังก์ชัน removeTrigger()

writeLED("/brightness", "0"); //เรียกใช้ฟังก์ชัน writeLED โดยกำหนดค่าที่จะส่งเข้าไปคือ “/brightness” และ “0”

}

else if (strcmp(argv[1],"flash")==0){ //ถ้าทำการเปรียบเทียบระหว่างค่าใน argv[1] และ String “flash” เหมือนกัน ให้ไปบรรทัดต่อไป

printf("Flashing the LED\n"); //แสดงข้อความ “Flashing the LED” แล้วขึ้นบรรทัดใหม่

writeLED("/trigger", "timer"); //เรียกใช้ฟังก์ชัน writeLED โดยกำหนดค่าที่จะส่งเข้าไปคือ “/trigger” และ “timer”

writeLED("/delay\_on", "50"); //เรียกใช้ฟังก์ชัน writeLED โดยกำหนดค่าที่จะส่งเข้าไปคือ “/delay\_on” และ “50”

writeLED("/delay\_off", "50"); //เรียกใช้ฟังก์ชัน writeLED โดยกำหนดค่าที่จะส่งเข้าไปคือ “/delay\_off” และ “50”

}

else if (strcmp(argv[1],"status")==0){ //ถ้าทำการเปรียบเทียบระหว่างค่าใน argv[1] และ String “status” เหมือนกัน ให้ไปบรรทัดต่อไป

FILE\* fp; // see writeLED function below for description สร้างตัวแปรไฟล์ Pointer ชื่อ fp

char fullFileName[100]; //ประกาศตัวแปรชนิด char ชื่อ fullFileName ขนาด Array 100

char line[80]; //ประกาศตัวแปรชนิด char ชื่อ line ขนาด Array 80

sprintf(fullFileName, LED3\_PATH "/trigger"); //นำเอาข้อความ “trigger” ไปใส่ต่อท้าย LED3\_PATH ที่เรา define ไว้ แล้วนำไปเก็บไว้ที่ fullFileName

fp = fopen(fullFileName, "rt"); //reading text this time // เปิดไฟล์ fullFileName เพื่ออ่านเพียงอย่างเดียว

while (fgets(line, 80, fp) != NULL){ //อ่านไฟล์จาก fp 80 ตัว มาเก็บที่ line โดยที่ถ้าไม่เท่ากับ NULL ให้วนลูปต่อไป

printf("%s", line); //แสดงข้อความจาก line

}

fclose(fp); //ปิดไฟล์ที่ชี้อยู่

}

else{ //อื่นๆ

printf("Invalid command!\n"); // แสดงข้อความ “Invalid command!” แล้วขึ้นบรรทัดใหม่

}

printf("Finished the makeLED Program\n"); // แสดงข้อความ “Finished the makeLED Program”

return 0; //จบการทำงาน

}

void writeLED(char filename[ ], char value[]){ //ฟังก์ชัน writeLED

FILE\* fp; // create a file pointer fp // สร้างตัวแปรไฟล์ Pointer ชื่อ fp

char fullFileName[100]; // to store the path and filename //สร้าง Array fullFileName ชนิด char

sprintf(fullFileName, LED3\_PATH "%s", filename); // write path/name //นำเอาข้อความ “trigger” ไปใส่ต่อท้าย LED3\_PATH ที่เรา define ไว้ แล้วนำไปเก็บไว้ที่ fullFileName

fp = fopen(fullFileName, "w+"); // open file for writing// เปิดไฟล์ fullFileName เพื่ออ่านเพียงอย่างเดียว

fprintf(fp, "%s", value); // send the value to the file//นำค่าจาก value เขียนลงไปใน fp

fclose(fp); // close the file using the file pointer // ปิดไฟล์ที่ fp ชี้อยู่

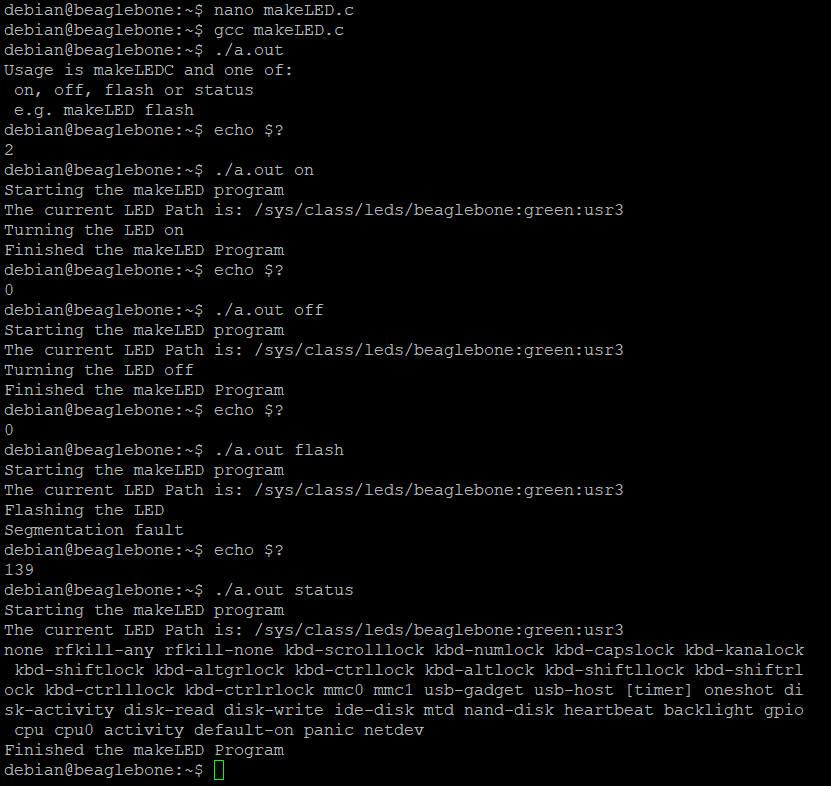
}

void removeTrigger(){ //ฟังก์ชัน removeTrigger

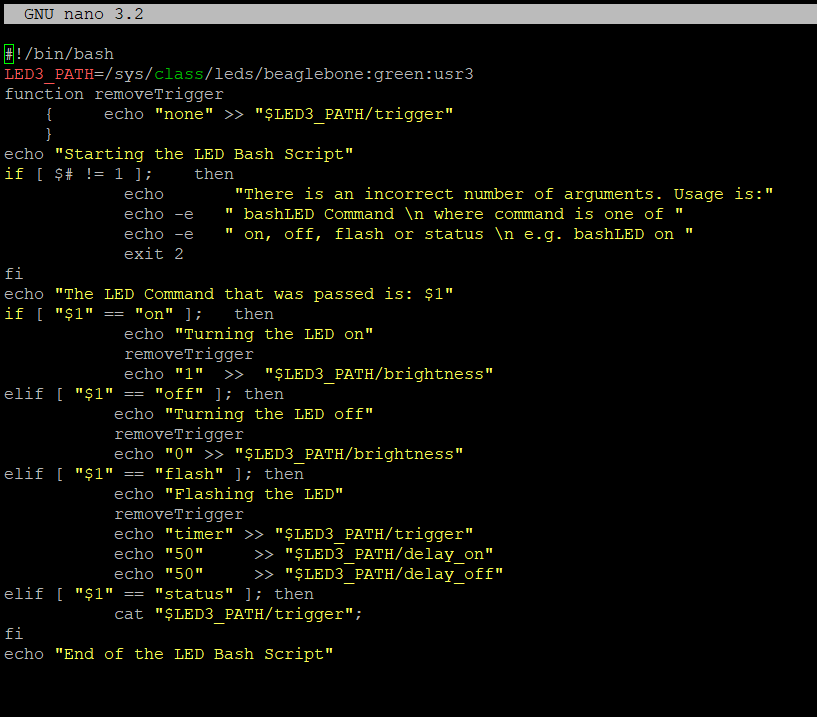
writeLED("/trigger", "none"); //เรียกใช้ฟังก์ชัน writeLED แล้วส่งค่า “/trigger” และ “none” ไป

}

การจำลองการทำงาน



**Exercise 9 : bashLED**



Description

#!/bin/bash

LED3\_PATH=/sys/class/leds/beaglebone:green:usr3 //กำหนดเส้นทางของ LED\_PATH

function removeTrigger //ฟังก์ชัน removeTrigger

{ echo "none" >> "$LED3\_PATH/trigger"

//ใส่คำว่า “none” ลงไปที่ส่วนท้ายของ "$LED3\_PATH/trigger

}

echo "Starting the LED Bash Script" //แสดงข้อความ “Starting the LED Bash Script”

if [ $# != 1 ]; then //ถ้าจำนวน Argument ที่เพิ่มเข้ามาในสคริปต์นี้เกิน 1 ให้กระทำบรรทัดต่อไป

echo "There is an incorrect number of arguments. Usage is:"

//แสดงข้อความ “There is an incorrect number of arguments. Usage is:”

echo -e " bashLED Command \n where command is one of "

//แสดงข้อความ “bashLED Command \n where command is one of”

echo -e " on, off, flash or status \n e.g. bashLED on "

//แสดงข้อความ “on, off, flash or status \n e.g. bashLED on”

exit 2 // return ค่า 2 แล้วจบการทำงาน

fi //จบเงื่อนไข if

echo "The LED Command that was passed is: $1" // แสดงข้อความและ Argument ที่เข้ามา

if [ "$1" == "on" ]; then // ถ้าตรงตามเงื่อนไขให้ไปบรรทัดต่อไป

echo "Turning the LED on" //แสดงข้อความ “Trning the LED on”

removeTrigger // เรียกใช้งานฟังก์ชัน removeTrigger

echo "1" >> "$LED3\_PATH/brightness" //ใส่เลข 1 ลงไปต่อท้าย LED3\_PATH/brightness

elif [ "$1" == "off" ]; then // ถ้าตรงตามเงื่อนไขให้ไปบรรทัดต่อไป

echo "Turning the LED off" //แสดงข้อความ “Turning the LED off”

removeTrigger // เรียกใช้ฟังก์ชัน removeTrigger

echo "0" >> "$LED3\_PATH/brightness" //ใส่เลข 0 ลงไปต่อท้าย LED3\_PATH/brightness

elif [ "$1" == "flash" ]; then // ถ้าตรงตามเงื่อนไขให้ไปบรรทัดต่อไป

echo "Flashing the LED" //แสดงข้อความ “Flashing the LED”

removeTrigger // เรียกใช้ฟังก์ชัน removeTrigger

echo "timer" >> "$LED3\_PATH/trigger" //ใส่ timer ลงไปต่อท้าย LED3\_PATH/trigger

echo "50" >> "$LED3\_PATH/delay\_on" //ใส่ 50 ลงไปต่อท้าย LED3\_PATH/delay\_on

echo "50" >> "$LED3\_PATH/delay\_off" //ใส่ 50 ลงไปต่อท้าย LED3\_PATH/delay\_off

elif [ "$1" == "status" ]; then // ถ้าตรงตามเงื่อนไขให้ไปบรรทัดต่อไป

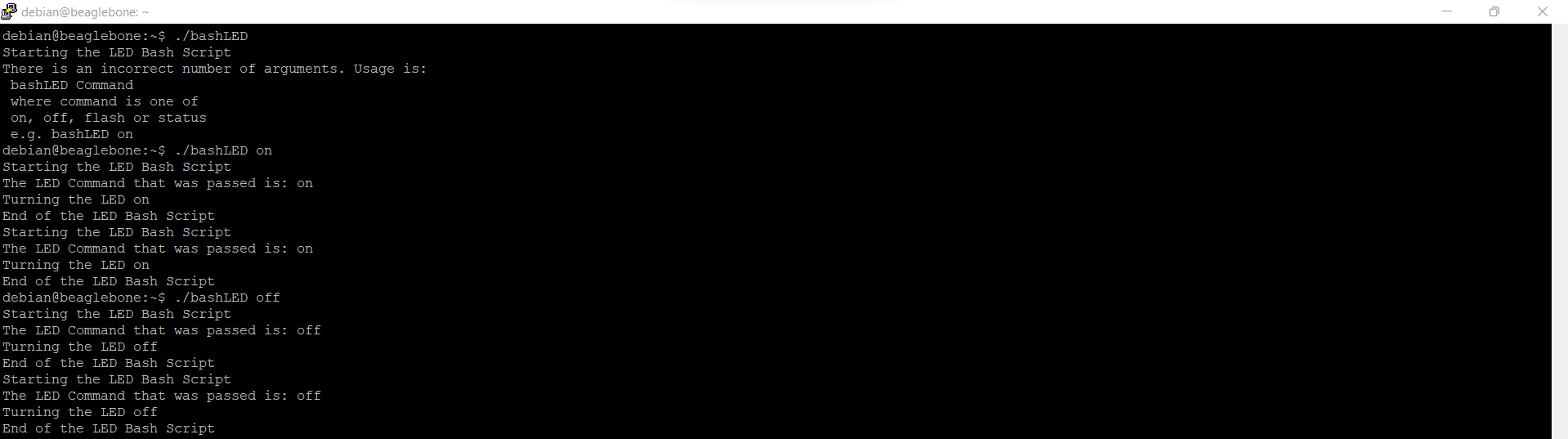
cat "$LED3\_PATH/trigger"; //อ่านไฟล์จาก LED3\_PATH/trigger

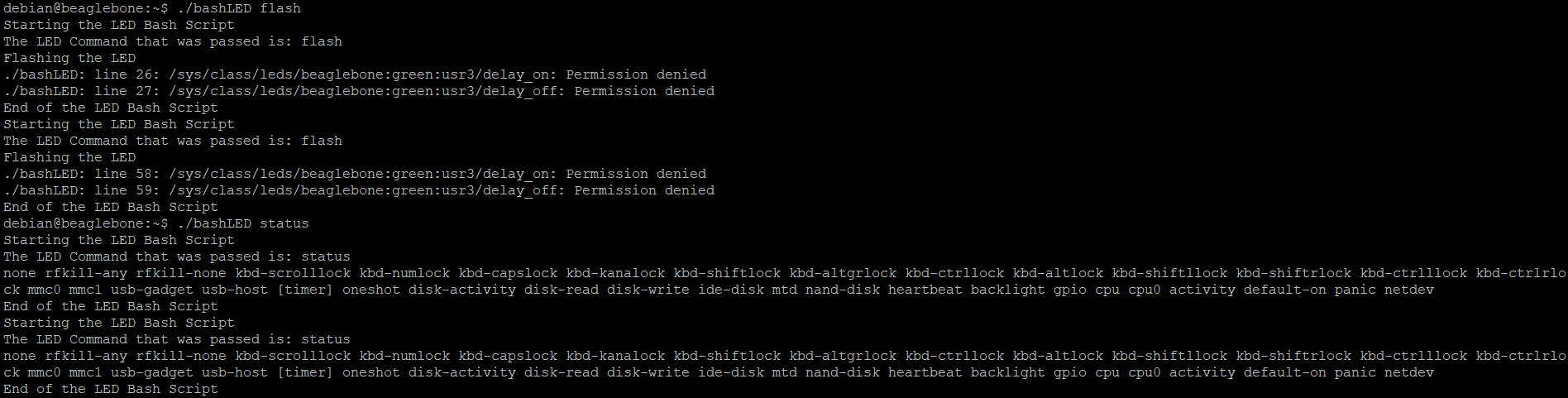
fi //จบเงื่อนไข if

echo "End of the LED Bash Script" //แสดงข้อความ “End of the LED Bash Script”

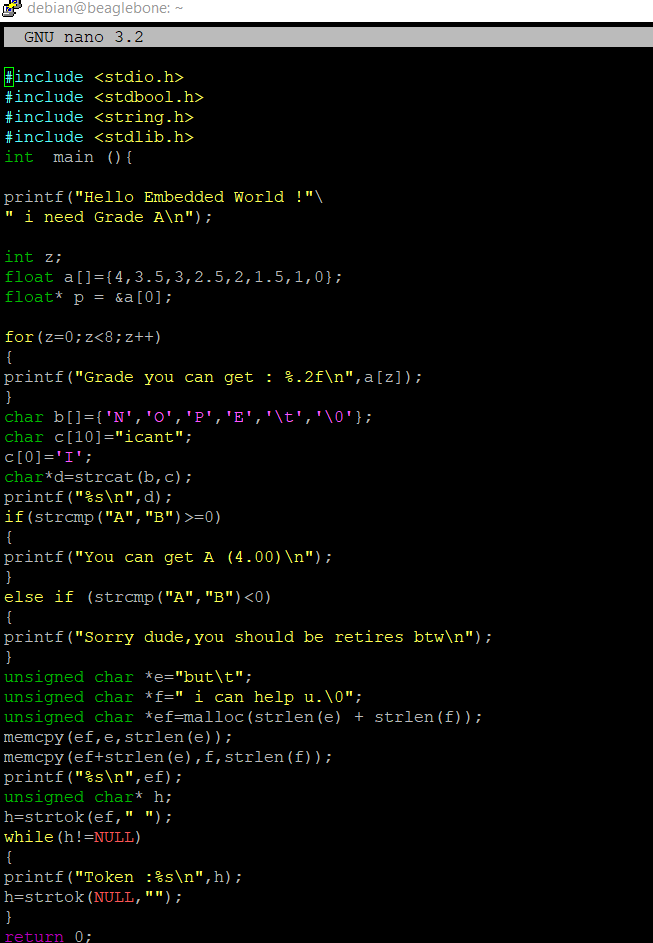
การจำลองการทำงาน

เนื่องจากยังไม่สามารถจำลองผลได้ ต้องเพิ่มคำสั่ง chmod u+x \_\_\_\_ ตามด้วยชื่อไฟล์ที่ต้องการ เพื่อให้สามารถประมวลผลไฟล์ได้ นั่นเอง และใช้ command “./bashLED \_\_\_” เพื่อจำลองการทำงาน





การดัดแปลงคำสั่งเพื่อนำไปใช้ประโยชน์



ผมได้ทำการนำคำสั่งใน exercise 1-7 มาปรับใช้ โดยเป็นคำสั่งที่เกี่ยวกับการเกรด เสมือนการตอบโต้ของระบบ เพราะคำสั่งได้ถูก set คำตอบไปแล้ว แต่ก็ได้นำมาประยุกต์ใช้ หากทำการศึกษาต่อยอด ก็อาจจะมีการพิมพ์ตอบโต้กับระบบ เช่น ฉันอยากได้เกรด A แล้วระบบถามว่า อ่านหนังสือหรือยัง ถ้าเราตอบ yes ระบบจะตอบกลับมาว่า คุณมีโอกาสได้ A แต่ถ้าตอบ No ระบบก็จะบอกคุณว่าเสี่ยง F แต่ถ้าตอบว่า idk ระบบจะบอกว่า why you don’t know . ประมาณนั้นครับ

การจำลองการทำงาน

