

Report of C Coursework

Chi Wai Kong

H00278458

8th March, 2018

Introduction

This is the report of a steganography system in C programming language. The aim of this program is to encode and decode message into a PPM image. PPM image is a very simple open source RGB colour bitmap format and the program can hide the text message in the red field.

Program Design

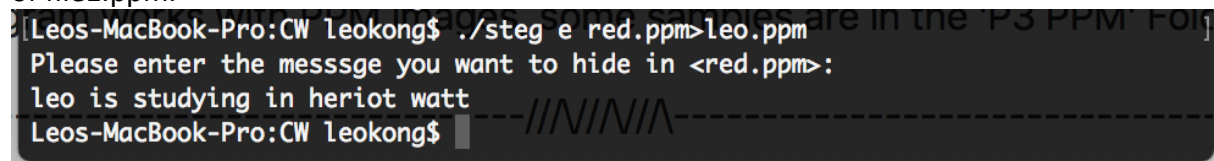
Front-end design

The main function of the program which can take 3 or 4 arguments. Depending on the number and the signal of arguments, the program can provide encoding or decoding function. The second argument has to be an “e” if user wants to encode the PPM image or a “d” if user want to decode the image.

This program is very easy to run. For the call:

```
$ steg e file1.ppm > file2.ppm
```

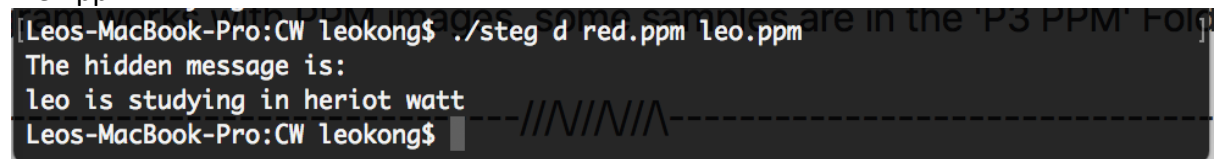
steg prompts a message and ask user to enter what they need to encode in file1.ppm. After entering the message, the new ppm image (file2.ppm) will be generate into the same folder of file1.ppm.



```
Leos-MacBook-Pro: CW leokong$ ./steg e red.ppm>leo.ppm
Please enter the messsge you want to hide in <red.ppm>:
leo is studying in heriot watt
Leos-MacBook-Pro: CW leokong$
```

```
$ steg d file1.ppm file2.ppm
```

steg shows the message hidden in file2.ppm after decoding by comparison with original file1.ppm.



```
Leos-MacBook-Pro: CW leokong$ ./steg d red.ppm leo.ppm
The hidden message is:
leo is studying in heriot watt
Leos-MacBook-Pro: CW leokong$
```

if user enter the wrong arguments, the program will automatically show the error message.

```
[Leos-MacBook-Pro:~ leokong$ ./steg d red.pp leo.ppm]
Cannot open the file<red.pp>
Leos-MacBook-Pro:~ leokong$
```

```
[Leos-MacBook-Pro:~ leokong$ ./steg red.ppm leo.ppm]
PLEASE ENTER
steg e file1.ppm>file2.ppm for encoding
OR
steg d file1.ppm file2.ppm for decoding
Leos-MacBook-Pro:~ leokong$
```

Back-end design

As I mentioned before, the program can take the arguments that entered by user and check which action they want. For the encoding action, the program first scans all the data from the image into the structure named "img". Second, asking user to enter the message that they want to hide. Third, after scanning the message entered by user, the program will create a new structure called "copy_img" that included the hidden message in the red fields. Finally, before freeing the memory, the program needs to output the new PPM image with the hidden message to the same path of the original PPM image.

```
if(argc==3 && strcmp(argv[1],"e")==0){ /*steg e file1.ppm >
file2.ppm*/
    if(NULL==(f1=fopen(argv[2],"rb")))
    {
        fprintf(stderr, "Cannot open the file<%s>\n",
            argv[2]);
        exit(1);
    }
    //printf("Please enter the message that you want to
    hide:\n");
    img=getPPM(f1);
    setbuf(stdin,NULL);
    fprintf(stderr, "Please enter the messsge you want to
    hide in <%s>:\n",argv[2]);
    fgets(a, 128, stdin);
    copy_img=encode(a,img);
    showPPM(copy_img);
    free(img->data);
    free(copy_img->data);
    free(img);
    free(copy_img);
}
```

If user want to decode the image, the data of new image and the original image will be scanned into two data structures. And then the program will start to compare both of the images and do the decoding process. After the decoding process, the hidden message will be shown before freeing the memory.

```
else if(argc==4 && strcmp(argv[1],"d")==0){/*steg d file1.ppm
file2.ppm*/
    if(NULL==(f1=fopen(argv[2],"rb")))
    {
        fprintf(stderr, "Cannot open the file<%s>\n",
            argv[2]);
        exit(1);
    }
    img=getPPM(f1);
    if(NULL==(f2=fopen(argv[3],"rb")))
    {
        fprintf(stderr, "Cannot open the file<%s>\n",
            argv[3]);
        exit(1);
    }
    copy_img=getPPM(f2);
    char*s=decode(img,copy_img);
    fprintf(stderr, "The hidden message is:\n%s\n",s);
    free(img->data);
    free(copy_img->data);
    free(img);
    free(copy_img);
}
```

If user cannot enter the correct arguments, the error message will be shown. From the code, expect the 3 arguments with a “e” in second argument and 4 arguments with a “d” in second argument, all of the other are considered as an error.

```
else {
    fprintf(stderr, "PLEASE ENTER\n");
    fprintf(stderr, "steg e file1.pmm>file2.pmm for
encoding\n");
    fprintf(stderr, "OR\n");
    fprintf(stderr, "steg d file1.pmm file2.pmm for
decoding\n");
    exit(1);
}
```

Data Structure

In the program, I used structure to store data from PPM image. Using structure is one of the easy ways to keep and management the data because in each structure can contain more than one different variables. After we understand what data can be found in PPM image, then we can use structure to store them.

I created a structure called "struct PPM". In this structure, it can handle the content of the PPM image. For example, the flag of the PPM format, the width and height of the PPM image and the maximum colour value of the image (usually it is 255). We can store the value as integer numbers. However, for the pixel value from the image, I used "unsigned char *" to store and handle the data.

```
▼ struct PPM{  
  
    int flag; /*flag of the PPM format*/  
    int width; /*integer number of columns*/  
    int height; /*integer number of rows*/  
    int max; /*Integer maximum colour value - usually 255*/  
    unsigned char *data; /*pixel value*/  
};
```

Next, I chose to use the unsigned char to store the pixel value of PPM image. Signed char will have values ranging from -127 to +127. Whereas unsigned char will have values from 0 to 255.

So, after scanning the pixel values into the integer called red, green and blue, I directly put them into the unsigned char.

```
int size=img->width*img->height;  
int i;  
int red ,green,blue;  
if(img->flag==3){  
    img->data=(unsigned char*)calloc(size,3);  
    for(i=0;i<size;i++){  
        fscanf(fd,"%d%d%d",&red,&green,&blue);  
        /* printf("%d %d %d \n",red,green,blue);*/  
        img->data[i*3]=red;  
        img->data[i*3+1]=green;  
        img->data[i*3+2]=blue;  
    }  
}
```

Algorithms

The algorithms of the program are very simple. From the beginning of the program, there is a set of “random numbers” which are stored in the array of integer and assigned by me randomly. In other words, the “random number” is the key to encode and decode and cannot be changed after the message is encoded. The first letter uses the first number from the random number to encode, then the second letter use the second number to encode and so on.

```
int rand_num[]=
{9,1,2,4,1,4,6,3,5,5,9,6,4,4,1,4,3,5,1,9,2,6,3,5,9,8,5,1,9,8,2,6,8
,1,9,2,2\
,5,4,1,4,5,5,7,9,3,7,0,8,5,8,4,1,6,1,1,5,0,3,2,1,2,3,3,9,4,2\
,3,5,2,5,0,6,0,7,2,8,6,2,5,1,0,5,6,7,0,7,3,7,8,0,0,9,4,7,5,8,\
0,3,5,5,3,2,7,2,8,1,5,6,7,7,5,7,1,9,6,0,0,3,5,4,8,8,8,8,1,4,9};
```

For encoding, the formula of encoding is the sum of the value of the original pixel, the text and the key random number. The method is to put the encoding message value in the random pixel. Of course, the first letter is put into the pixel near front but not guarantee which one. Using “%255” which means to prevent the original pixel value is in red colour valued at 255. If the new value is less than 255, nothing will be change, else the new value will be deducted by 255.

```
for(index=0;index<len;index++){

    pos=positions[index];

    copy_img->data[pos]=((int)copy_img->data[pos]+
(int)text[index]+rand_num[index])%255;

}
```

The message is hidden in the red field of PPM, so comparing the image with hidden message and the original image, the program can decode the message. “*3” is meaning that the program can only compare the red fields because comparing the green and blue fields is meaningless. Thinking of the reverse case of encoding. If the new value is smaller than the original, which means after the encoding calculating the value is greater than 255. Therefore, we need to add 255 to the new value before the decoding calculation. Also, there is not allow more than 128 letters in a hidden message, so, the error message will be shown if the program find more than 128 differences from two images which can mean the pixel values of the image may be changed my user manually.

```
for(i=0;i<size;i++){ /*find the different pixels and get
characters*/

    if(i1->data[i*3]<i2->data[i*3]){

        a[count]=i2->data[i*3]-i1->data[i*3]-
        rand_num[count];
        count++;
    }
    else if(i1->data[i*3]>i2->data[i*3]){

        a[count]=((int)i2->data[i*3]+255)-(int)i1-
        >data[i*3]-rand_num[count];
        count++;
    }
    if(count==128){
        printf("Error.\n");
        exit(1);
    }
}

return a;
```