1. 응용프로그램 작성

1-1. LCD에 bitmap 그림 파일 표시하기

학습목표	● LCD에 bitmap 그림 파일 표시하는 응용프로그램을 작성한다.

수행 내용 / LCD에 bitmap 그림 표시하는 응용프그램 작성

기기

- 호스트 PC, 타겟 장비, USB 메모리, USB to Serial Cable, USB 메모리 스틱

안전/유의사항

- 장비와 호스트 PC가 네트워크 망에 연결되어 있으면 네트워크를 통해 파일을 타겟 장비에 전송할 수 없으므로 USB 메모리나 minicom의 zmodem을 이용하여 전송한다.

수행/순서

(1) 개요

이장에서 프레임버퍼 디바이스 드라이버를 사용하여 LCD에 사진을 로딩하는 프로그램을 작성해 본다.

프레임 버퍼(Frame Buffer)은 커널에서 한 화면을 구성하기 위해서 할당 해 놓은 메모리라 할수 있다. 그래픽 하드웨어를 사용자 레벨의 응용 프로그램이 제어할 수 있도록 만들어진 디바이스 드라이버를 프레임 버퍼 드라이버라고 한다.

응용 프로그램을 통해 각 픽셀의 색상을 프레임 버퍼 드라이버에 전달하면 프레임 버퍼를 통해 LCD 화면에 표현하게 된다.

(2) 프로그램 작성

bmp 파일을 읽어서 화면에 표시하는 응용프로그램을 작성해 보겠다. 읽을 bmp 파일은 1024*800을 넘지 않고 24bit 픽셀 폼만 지원하는 것으로 작성해 보겠다. 기 작성된 소스가 첨부 파일에 있다.

호스트 PC에서 bitmap.c 와 bitmap.h 파일을 만들고 다음과 같이 작성한다..

① 이미 작성된 소스 파일은 첨부 파일을 참조한다. 호스트 PC에서 coloarbar.c 파일을 만들고 다음과 같이 작성한다.

bitmap.h

```
// BMP File Structure (windows version 3)
// BMP file의 헤더의 구조를 구조체로 만들어 놓았음.
// File Header
typedef struct {
//      unsigned char     bfType;             // 2 byte
      unsigned int      bfSize;             // 4 byte
      unsigned short    bfReserved1;        // 2 byte
      unsigned short    bfReserved2;        // 2 byte
      unsigned int      bfOffBits;          // 4 byte
} BITMAPFILEHEADER;

// Image Header
typedef struct {
      unsigned int      biSize;             // 4 byte
      unsigned int      biWidth;            // 4 byte
      unsigned int      biHeight;           // 4 byte
      unsigned short    biPlanes;           // 2 byte
      unsigned short    biBitCount;         // 2 byte
      unsigned int      biCompression;      // 4 byte
      unsigned int      biSizeImage;        // 4 byte
      unsigned int      biXPelsPerMeter;    // 4 byte
      unsigned int      biYPelsPerMeter;    // 4 byte
      unsigned int      biClrUsed;          // 4 byte
```

```
    unsigned int      biClrImportant;      // 4 byte
} BITMAPINFOHEADER;

// Color Table
typedef struct {
// windows version 3
    unsigned char     rgbBlue;           // 1 byte
    unsigned char     rgbGreen;          // 1 byte
    unsigned char     rgbRed;            // 1 byte
    unsigned char     rgbReserved;       // 1 byte
} RGBQUAD;

// Pixel Data
typedef struct {
    BITMAPINFOHEADER      bmiHeader;
    RGBQUAD               bmiColors[1];
} BITMAPINFO;
```

bitmap.c

```
#include <stdio.h>
#include <stdlib.h>       // for exit
#include <unistd.h>       // for open/close
#include <fcntl.h>        // for O_RDWR
#include <sys/ioctl.h>    // for ioctl
#include <sys/mman.h>
#include <linux/fb.h>     // for fb_var_screeninfo, FBIOGET_VSCREENINFO
#include "bitmap.h"

#define FBDEV_FILE   "/dev/fb0"
#define BIT_VALUE_24BIT     24

void usage(void)
{

printf("=====================================================
```

```
₩n");
   printf("₩nUsage: ./bitmap [FILE.bmp]₩n");

printf("==================================================
₩n");
}
// filename의 파일을 열어 순수 bitmap data은 *data 주소에 저장하고 , pDib은
// malloc 해제하기 위한 메모리 포이트이다.
void read_bmp(char *filename, char **pDib, char **data, int *cols, int *rows)
{
      BITMAPFILEHEADER      bmpHeader;
      BITMAPINFOHEADER      *bmpInfoHeader;
      unsigned int      size;
      unsigned char     magicNum[2];
      int       nread;
      FILE      *fp;
      fp    =   fopen(filename, "rb");
      if(fp == NULL) {
            printf("ERROR₩n");
            return;
      }
      // identify bmp file
      magicNum[0]     =     fgetc(fp);
      magicNum[1]     =     fgetc(fp);
      printf("magicNum : %c%c₩n", magicNum[0], magicNum[1]);
      if(magicNum[0] != 'B' && magicNum[1] != 'M') {
            printf("It's not a bmp file!₩n");
            fclose(fp);
            return;
      }
      nread     =     fread(&bmpHeader.bfSize, 1, sizeof(BITMAPFILEHEADER), fp);
      size      =     bmpHeader.bfSize - sizeof(BITMAPFILEHEADER);
      *pDib     =     (unsigned char *)malloc(size);          // DIB Header(Image Header)
      fread(*pDib, 1, size, fp);
      bmpInfoHeader     =     (BITMAPINFOHEADER *)*pDib;
      printf("nread : %d₩n", nread);
```

```
    printf("size : %d₩n", size);
    // check 24bit
    if(BIT_VALUE_24BIT != (bmpInfoHeader->biBitCount))        // 24bit pixel form만
지원
    {
        printf("It supports only 24bit bmp!₩n");
        fclose(fp);
        return;
    }
    *cols    =    bmpInfoHeader->biWidth;
    *rows    =    bmpInfoHeader->biHeight;
    *data    =    (char *)(*pDib + bmpHeader.bfOffBits - sizeof(bmpHeader) - 2);
    fclose(fp); // 순수  bitmap 데어터
}

void close_bmp(char **pDib)
{
    free(*pDib); // 메모리 해제
}

int main (int argc, char **argv)
{
    int i, j, k, t;
    int fbfd;
    int screen_width;
    int screen_height;
    int bits_per_pixel;
    int line_length;
    int coor_x, coor_y;
    int cols = 0, rows = 0;
    int mem_size;

    char     *pData, *data;
    char     r, g, b;
    unsigned long    bmpdata[1280*800];
    unsigned long    pixel;
```

```
unsigned char    *pfbmap;
unsigned long    *ptr;
struct   fb_var_screeninfo fbvar;
struct   fb_fix_screeninfo fbfix;
printf("================================₩n");
printf("Frame buffer Application - Bitmap₩n");
printf("================================₩n₩n");

if(argc != 2) {
     usage();
     return   0;
}
read_bmp(argv[1], &pData, &data, &cols, &rows);
printf("Bitmap : cols = %d, rows = %d₩n", cols, rows);
for(j = 0; j < rows; j++)
{
     k    =    j * cols * 3;
     t    =    (rows - 1 - j) * cols; // 가로 size가 작을 수도 있다.

     for(i = 0; i < cols; i++)
     {
          b    =    *(data + (k + i * 3));
          g    =    *(data + (k + i * 3 + 1));
          r    =    *(data + (k + i * 3 + 2));

          pixel = ((r<<16) | (g<<8) | b);
          bmpdata[t+i]     =    pixel;          // save bitmap data bottom up
     }
}
close_bmp(&pData); // 메모리 해제
if( (fbfd = open(FBDEV_FILE, O_RDWR)) < 0) //
{
     printf("%s: open error₩n", FBDEV_FILE);
     exit(1);
}
if( ioctl(fbfd, FBIOGET_VSCREENINFO, &fbvar) )
```

```
{
    printf("%s: ioctl error - FBIOGET_VSCREENINFO ₩n", FBDEV_FILE);
    exit(1);
}
if( ioctl(fbfd, FBIOGET_FSCREENINFO, &fbfix) ) // screen info를 얻어옴
{
    printf("%s: ioctl error - FBIOGET_FSCREENINFO ₩n", FBDEV_FILE);
    exit(1);
}
if (fbvar.bits_per_pixel != 32)
{
    fprintf(stderr, "bpp is not 32₩n");
    exit(1);
}
screen_width    =   fbvar.xres;
screen_height   =   fbvar.yres;
bits_per_pixel  =   fbvar.bits_per_pixel;
line_length     =   fbfix.line_length;
mem_size    =   line_length * screen_height;

printf("screen_width : %d₩n", screen_width);
printf("screen_height : %d₩n", screen_height);
printf("bits_per_pixel : %d₩n", bits_per_pixel);
printf("line_length : %d₩n", line_length);
pfbmap  =   (unsigned char *)
    mmap(0, mem_size, PROT_READ|PROT_WRITE, MAP_SHARED, fbfd, 0);

if ((unsigned)pfbmap == (unsigned)-1)
{
    perror("fbdev mmap₩n");
    exit(1);
}
// fb 검은색으로 지움.
for(coor_y = 0; coor_y < screen_height; coor_y++) {
    ptr =   (unsigned long *)pfbmap + (screen_width * coor_y);
    for(coor_x = 0; coor_x < screen_width; coor_x++)
```

```
        {
            *ptr++   =    0x000000;
        }
    }

// direction for image generating : (0,0)-> (1,0)-> (2,0)-> ...-> (x , y)
    for(coor_y = 0; coor_y < rows; coor_y++) {
        ptr =    (unsigned long*)pfbmap + (screen_width * coor_y);
        for (coor_x = 0; coor_x < cols; coor_x++) {
            *ptr++   =    bmpdata[coor_x + coor_y*cols];
       }
    }
    munmap( pfbmap, mem_size);
    close( fbfd);
    return 0;
}
```

② 컴파일을 수행 후 실행 파일을 타겟 장비에 전송한다.

```
cndi@cndi-virtual:~$ cd bitmap/
cndi@cndi-virtual:~/bitmap$ ls
bitmap.c   bitmap.h   flower.bmp
cndi@cndi-virtual:~/bitmap$ arm-linux-gnueabihf-gcc bitmap.c -o bitmap
cndi@cndi-virtual:~/bitmap$ scp bitmap ecube@192.168.10.199:/home/ecube/
ecube@192.168.10.199's password:
bitmap              100%    13KB   12.9KB/s    00:00
cndi@cndi-virtual:~/bitmap$ scp flower.bmp   ecube@192.168.10.199:/home/ecube/
ecube@192.168.10.199's password:
flower.bmp                  100% 2304KB    2.3MB/s    00:00
cndi@cndi-virtual:~/bitmap$
```

③ 타겟 장비에서 실행 bitmap 실행

```
ecube@udoo:~$ ./bitmap flower.bmp
```

```
==================================
Frame buffer Application - Bitmap
==================================

magicNum : BM
nread : 12
size : 2359338
Bitmap : cols = 1024, rows = 768
screen_width : 1024
screen_height : 600
bits_per_pixel : 32
line_length : 4096
Segmentation fault
ecube@udoo:~$
```

그림 1-1 bitmp 실행 LCD 화면