ECE 252

LAB 1 – SYSTEMS PROGRAMMING

Lab 1 Overview

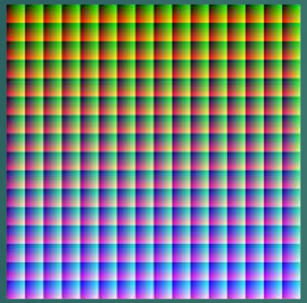
- Basic Linux commands
- Standard C programming tools (gcc, make, gdb/ddd)
- Linux system calls and library functions
- Reading and writing binary files

PNG Files

- **P**ortable **N**etwork **G**raphics
- One of the two most popular graphics formats (other is JPEG)
- Lossless compression
- Allows transparency (using an alpha channel)
- Relatively simple format

Pixels

WIDTH



HEIGHT

R G B A 8 bits x 4 = 32 bits

The PNG “magic number”

First 8 bytes of file are a unique sequence that identifies a PNG file

89 50 4E 47 0D 0A 1A 0A

- 89 is ascii TAB with top bit set
- 50 4E 47 is P N G
- 0D 0A is ascii carriage return and newline
- 1A is ascii End Of File
- 0A is an ascii newline

Chunk format

- After 8-byte magic number, PNG file has series of *chunks*
- Each chunk has:
  - Four-byte length (length of data field)
  - Four-byte type code (ascii characters)
  - (arbitrary data)
  - CRC (computed based on type code and data, not length field

Chunk types

- Common chunk types are:
  - IHDR – information about the file (width, height, depth, etc)
  - IDAT – the actual compressed image data
  - IEND – last chunk in the file
  - *(other chunk types, which we'll ignore)*

IHDR chunk format

- IHDR data field is always 13 bytes:
  - 4 byte width of image in pixels
  - 4 byte height of image in pixels
  - 1 byte bit depth per channel
  - 1 byte colour type
  - 1 byte compression method (always zero)
  - 1 byte filter method (always zero)
  - 1 byte interlace method (0 for no interlace)
- Our files will always have 8 bits per channel (32 total for RGBA)
- Our files will always have colour type 6 (RGBA), and no interlace

IDAT chunk format

- Compressed image data using zlib 1.0
- Filtering is used to reduce the amount of data
- First byte of each row of pixels is filter data
- Raw data size is HEIGHT * (WIDTH * 4 + 1) bytes
- There can be multiple IDAT chunks, but not for this lab

IEND chunk format

- Very simple – length field is zero, no data
- Marks end of image

Byte Ordering

- “little-endian” vs “big-endian”
- Most computers are little-endian
- All network traffic is big-endian
- 0x12AB gets stored as AB 12 but sent over the network as 12 AB
- See Linux man page for “byteorder” for a list of functions

Your Assignment

- You need to write three programs:
  - pnginfo – describes a PNG file
  - findpng – finds valid PNG files in a directory tree
  - catpng – concatenates PNG images into a single image

pnginfo

- Takes single command-line argument giving path to a file
- If file is a valid PNG, prints out the following;
  - filename: width x height
    - e.g. WEEF_1.png: 450 x 229
- If file is not a valid PNG, prints out the following:
  - Filename: Not a PNG file
- You must match this output format!

findpng

- Like Linux `find` command, but for PNGs
- Takes a single command-line argument giving path to a directory
- Walks down that entire directory tree, finding valid PNG files
- Prints out a list of the paths from that directory tree
- e.g.

```
Lab1/yourfile.png
Lab1/whatever/somewhere/anotherfile.png
```

- If no valid PNG files are found, prints:

```
findpng: no PNG file found
```

- Again – you *must* match the output format exactly!

catpng

- Most complex of the three programs
- Like Linux `cat` command, but for PNGs
- Takes any number of command-line arguments
- Each argument is a path to a PNG file
- Combines all the images, stacking them vertically
- Writes combined image to a file called `all.png`
- You can assume all the images are the same width

Doing Research

- You should do some research (using the “man” command and google searches) to find out how to do things like traversing directory trees and reading binary files
- Here are some C functions you might need, found in section 3 of the manual:
  - `fopen, fclose, fread, fwrite, fseek, ftell`
  - `opendir, closedir, readdir`
- Here are some system calls you might need, found in section 2 of the manual:
  - `stat, fstat`
- Read the sample code!

Deliverables

- Source code for all three programs (`pnginfo, findpng, catpng`)
- A `Makefile` that builds all of them, with a "`clean`" target as well
- Must be in a directory called "`Lab1`"
- Zip the directory into "`lab1.zip`"
- Running "`unzip lab1.zip`" must create a directory in the current directory containing the Makefile and your source code
- You ***must*** get this all correct, since marking is automated