PA06

Grace, Jayson
jaysong@unm.edu

Salas, Dominic
dsalas9@unm.edu

Sunday 8th November, 2015

Jayson Grace, Dominic Salas

Problem 1a

To identify a file by its magic number, we created this code:

```
**
 * @file    filetype.c
 * @author Jayson Grace (jaysong@unm.edu)
 * @author Dominic Salas (dominic.salas@gmail.com)
 * @date    11/05/2015
 * @brief  filetype program to identify files without
    dependence on its having an extension
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef int bool;
enum { false, true };

bool DEBUG = false;

/**
 * @brief Get the magic number from the input file
 * @param file The file to get the magic number from
 * @return Magic number
 */
unsigned char* getMagicNumber(FILE *file)
{
  long fileSize;
  unsigned char *fileInfo;
  fseek(file, 0, SEEK_END);
  fileSize = ftell(file);
  rewind(file);
  fileInfo = (char *) malloc(fileSize + 1);


  fread(fileInfo, sizeof(char), 4, file);
```

```
  if (DEBUG)
  {
    printf("%02hhx%02hhx%02hhx%02hhx\n", fileInfo[0],
       fileInfo[1], fileInfo[2], fileInfo[3]);
    if (fileInfo[0] == 0x1f)
    {
      printf("%x\n", fileInfo[0]);
    }
  }
  return fileInfo;
}

/**
 * @brief Get the file type based on its magic number
 * @param fileInfo The information about the file
 */
void getFileType(char* fileInfo)
{
  int i;
  char pdf[4] = {0x25,0x50,0x44,0x00};
  char jpg[4] = {0xff,0xd8,0xff,0x00};
  char elf[4] = {0x7f,0x45,0x4c,0x00};
  char tar[4] = {0x1f,0x8b,0x08,0x00};
  char sh[4] = {0x23,0x21,0x2f,0x00};
  char result[4];
  for (i = 0; i < 3; i++)
  {
    result[i] = fileInfo[i];
    if (DEBUG)
      printf("%c\n", fileInfo[i]);
  }
  result[3] = 0x00;
  if (strcmp(pdf, result) == 0)
    printf("PDF detected\n");
  else if (strcmp(sh, result) == 0)
    printf("sh detected\n");
  else if (strcmp(elf, result) == 0)
    printf("ELF detected\n");
  else if (strcmp(tar, result) == 0)
    printf("tar.gz detected\n");
  else if (strcmp(jpg, result) == 0)
```

```
    printf("jpg detected\n");
  else
    printf("Invalid file type input!\n");
}

/**
  @brief Entry into program
  @param argc Argument count
  @param argv Argument vector
  @return Success (or lackthereof) of program's
     execution
  */
int main(int argc, char **argv)
{
  char* fileInfo;
  if ( argc != 2 )
  {
    printf( "usage: %s filename", argv[0] );
  }
  else
  {
    FILE *file = fopen( argv[1], "r" );

    if ( file == 0 )
    {
      printf( "Could not open file\n" );
      return EXIT_FAILURE;
    }
    else
    {
      fileInfo = getMagicNumber(file);
      getFileType(fileInfo);
    }
  }
  free(fileInfo);
  return EXIT_SUCCESS;
}
```

To get a consistent set of tests to run, we created this bash script:

```
#!/bin/bash
#
# runTests.sh
#
# Run test for problem 1 of pa06
#
# Usage: bash runTests.sh
#
# Jayson Grace, jayson.e.grace@gmail.com, 11/8/2015
#
# Last update 11/8/2015 by Jayson Grace, jayson.e.
   grace@gmail.com
#

compile()
{
  reset
  gcc -g filetype.c -o filetype
}

pdfTest()
{
  wget http://help.adobe.com/en_US/reader/using/
     reader_X_help.pdf
  ./filetype reader_X_help.pdf
  mv reader_X_help.pdf reader_X_help
  ./filetype reader_X_help
  rm reader_X_help
}

jpgTest()
{
  wget https://upload.wikimedia.org/wikipedia/en/1/12/
     Never-Let-Me-Down.jpg
  ./filetype Never-Let-Me-Down.jpg
  mv Never-Let-Me-Down.jpg Never-Let-Me-Down
  ./filetype Never-Let-Me-Down
  rm Never-Let-Me-Down
```

```
}

tarTest()
{
  wget http://ftp.gnu.org/gnu/tar/tar-1.28.tar.gz
  ./filetype tar-1.28.tar.gz
  mv tar-1.28.tar.gz tar-1.28
  ./filetype tar-1.28
  rm tar-1.28
}

shTest()
{
  ./filetype runTests.sh
  cp runTests.sh runTests
  ./filetype runTests
  rm runTests
}

elfTest()
{
  ./filetype /bin/ls
}

compile
pdfTest
jpgTest
tarTest
shTest
elfTest
```

The output of the bash script is as follows:

```
Resolving help.adobe.com (help.adobe.com)...
    23.3.12.17, 23.3.12.10
Connecting to help.adobe.com (help.adobe.com)
    |23.3.12.17|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1930650 (1.8M) [application/pdf]
Saving to:  reader_X_help.pdf

100%
[=============================================================>]
1,930,650    2.85MB/s    in 0.6s

2015-11-08 19:52:28 (2.85 MB/s) -  reader_X_help.
    pdf  saved [1930650/1930650]

PDF detected
PDF detected
--2015-11-08 19:52:28--  https://upload.wikimedia.org/
    wikipedia/en/1/12/Never-Let-Me-Down.jpg
Resolving upload.wikimedia.org (upload.wikimedia.org)
    ... 208.80.153.240, 2620:0:860:ed1a::2:b
Connecting to upload.wikimedia.org (upload.wikimedia.
    org)|208.80.153.240|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 18347 (18K) [image/jpeg]
Saving to:  Never-Let-Me-Down.jpg

100%
[=============================================================>]
 18,347       68.7KB/s    in 0.3s

2015-11-08 19:52:28 (68.7 KB/s) -  Never-Let-Me-Down.
    jpg  saved [18347/18347]

jpg detected
jpg detected
--2015-11-08 19:52:28--  http://ftp.gnu.org/gnu/tar/tar
    -1.28.tar.gz
```

```
Resolving ftp.gnu.org (ftp.gnu.org)... 208.118.235.20,
    2001:4830:134:3::b
Connecting to ftp.gnu.org (ftp.gnu.org)
    |208.118.235.20|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3877043 (3.7M) [application/x-gzip]
Saving to:   tar-1.28.tar.gz

100%
[=====================================================>]
 3,877,043    2.13MB/s    in 1.7s

2015-11-08 19:52:30 (2.13 MB/s) -   tar-1.28.tar.gz
    saved [3877043/3877043]

tar.gz detected
tar.gz detected
sh detected
sh detected
ELF detected
```

Problem 1b

Magic numbers are used to specify what type of data is in a binary file. Because text files aren't binary files, they do not have a magic number. As a result, a file with a magic number can't be a text file.

Problem 2

The following code makes up sizewatch.c:

```
/**
 * @file    sizewatch.c
 * @author Jayson Grace (jaysong@unm.edu)
 * @author Dominic Salas (dominic.salas@gmail.com)
 * @date    11/08/2015
 * @brief  Daemon to keep track of size changes that
    occur in a file over a 3 minute period.
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <time.h>

void getTime()
{
    time_t currentTime;
    struct tm * timeInfo;
    time(&currentTime);
    timeInfo = localtime(&currentTime);
    printf("[%d-%d-%d %d:%02d:%02d] ", (1 + timeInfo->
       tm_mon), timeInfo->tm_mday,  (1900+timeInfo->
       tm_year), timeInfo->tm_hour, timeInfo->tm_min,
       timeInfo->tm_sec);
}

int main(int argc, char* argv[])
{
    struct stat fileInfo;

    if(stat(argv[1], &fileInfo) == -1)
    {
        printf("Error: Input file not found!\n");
        return -1;
    }
```

```
    getTime();
    printf("Monitoring File: %s\n", argv[1]);

    off_t fileSize;
    fileSize = fileInfo.st_size;
    int i;

    getTime();
    printf("Initial Size: %d\n", (int) fileSize);

    for(i = 0; i < 18; i++)
    {
        if(stat(argv[1], &fileInfo) == -1)
        {
            printf("Error: The file has been moved or
               deleted!\n");
            return -1;
        }
        if((fileSize != fileInfo.st_size))
        {
            getTime();
            printf("Size Changed: Old: %d, New = %d\n",
                (int)fileSize, (int)fileInfo.st_size);
            fileSize = fileInfo.st_size;
        }
        sleep(10);
    }

    getTime();
    printf("Monitoring File terminated, BYE.\n");
}
```

While it is running, we make several modifications to testFile.txt.
The output is as follows:

```
[11-8-2015 23:06:43] Monitoring File: testFile.txt
[11-8-2015 23:06:43] Initial Size: 226
[11-8-2015 23:07:33] Size Changed: Old: 226, New = 492
[11-8-2015 23:08:33] Size Changed: Old: 492, New = 734
[11-8-2015 23:08:53] Size Changed: Old: 734, New = 898
[11-8-2015 23:09:33] Size Changed: Old: 898, New = 717
[11-8-2015 23:09:43] Monitoring File terminated, BYE.
```