CSc 352
Binary File IO

Benjamin Dicken

File Content

- Recall that files on a UNIX system are iNodes, that have pointers to data blocks, where the actual data of a file is stored
- Those blocks contain a sequence of 1’s and 0’s

- We can choose how to interpret when we read
- We can choose the format when we write

File Content

- Many of the files we have dealt with on UNIX in this course have been “text” files
  - *.c *.py *.txt *.stl makefile
  - This is just because we wrote text to those, and used programs that interpret files as text (vim)
- What have we used that are *NOT* text files?
- A “binary file” is just a file that we treat as information represented in RAW binary, rather than ASCII characters

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

int main() {
  uint32_t number = 10000000;

  FILE* text = fopen("text", "w");
  fprintf(text, "%u", number);
  fclose(text);

  FILE* binary = fopen("binary", "wb");
  fwrite(&number, 1, sizeof(number), binary);
  fclose(binary);

  return 0;
}
```

What is this?

What is this?

Tools for viewing file contents

```
$ hexdump file_name
```

```
$ xxd -b file_name
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

int main() {
  uint64_t number = 20;

  FILE* text = fopen("text", "w");
  fprintf(text, "%lu", number);
  fclose(text);

  FILE* binary = fopen("binary", "wb");
  fwrite(&number, 1, sizeof(number), binary);
  fclose(binary);

  return 0;
}
```

Which file represents the number more efficiently?

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

int main() {
  uint64_t number = 517;

  FILE* text = fopen("text", "w");
  fprintf(text, "%lu", number);
  fclose(text);

  FILE* binary = fopen("binary", "wb");
  fwrite(&number, 1, sizeof(number), binary);
  fclose(binary);

  return 0;
}
```

Which file represents the number more efficiently?

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

int main() {
  uint64_t number = 129481231210;

  FILE* text = fopen("text", "w");
  fprintf(text, "%lu", number);
  fclose(text);

  FILE* binary = fopen("binary", "wb");
  fwrite(&number, 1, sizeof(number), binary);
  fclose(binary);

  return 0;
}
```

Which file represents the number more efficiently?

Data Representation

Each row represents:
studentID, exam 1, exam 2, final exam

How many bytes would it take to represent this with a CSV ASCII file?

How many bytes would it take to represent this in binary? How compact could we get it?

grade_info.csv

```
19311233,80,90,100
91246834,75,85,82
21245122,43,76,87
18673124,90,75,90
```

Implement Conversion

Write the code to:

- Open this text file
- Re-write the same data to **`binary_grade_info.bin`**
- Close the file

grade_info.csv

```
19311233,80,90,100
91246834,7,85,82
21245122,43,100,87
18673124,90,75,90
```

Sum the numbers

Write a program that:

1. Asks the user for a file name
2. Sums the numbers
3. Prints the result

Assume that the file is formatted in binary and has alternating 8-byte integers (uint64_t) and 4-byte floats (float)

Use **fread**

Use:
`/tmp/352numbers`
and
`/tmp/more352numbers`
to test

```c
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>

int main(int argc, char* argv[]) {
  FILE* f = fopen(argv[1], "rb");
  int i = 0;
  int r = 1;
  double sum = 0;
  while(r) {
    if (i%2 == 0) {
      uint64_t temp = 0;
      r = fread(&temp, sizeof(uint64_t), 1, f);
      sum += temp;
    } else {
      float temp = 0.0;
      r = fread(&temp, sizeof(float), 1, f);
      sum += temp;
    }
    i++;
  }
  fclose(f);
  printf("SUM: %lf\n", sum);
  return 0;
}
```