CMPE-013/L

Introduction to “C” Programming

Gabriel Hugh Elkaim
Winter 2015

File I/O
File I/O

• Most data on computers are stored in files.
• So accessing data reads and writes to these files.
• And in a Unix environment, everything is a file:
  – Serial ports
  – Network connections
  – Hard drives
  – Displays

So everything can be controlled via file access.

File I/O

Standard files

• Three special files that are automatically opened and closed:
  – stdin: standard input (keyboard/serial port)
  – stdout: standard output (screen)
  – stderr: standard error (screen)

```
a.out > myfile.txt
```
File I/O

The standard library

• `<stdio.h>` contains functions for working with files

• Its concept of a file includes:
  – Filename
  – File access mode
  – File size
  – Current position

File I/O

Using files

• Files are opened with `fopen()`

• Files are read and written to:
  – `fprintf()`, `fscanf()` — Formatting strings
  – `fputc()`, `fgetc()` — Characters
  – `fputs()`, `fgets()` — Lines
  – `fread()`, `fwrite()` — Blocks

• Files are closed with `fclose()`
File I/O
Using files

- Only a limited number of files can be opened at a time
  - Per process
  - Also per OS
- Very large on modern Oses
  - \( \geq 2048 \) usually
- For the XC32: 8

File I/O
FILE

- The standard library uses a single struct to store the metadata of the file

```c
typedef struct _iobuf {
    char * _ptr;
    int _cnt;
    char * _base;
    unsigned short _flag;
    short _file;
    size_t _size;
} FILE;
```
**File I/O**

**fopen()**

**Syntax**

```
FILE *fopen(const char *name, const char *mode);
```

- **name** is a C string with the filename
- **mode** is the mode to open the file in
  - "r" opens for **reading**
  - "w" opens for **writing**
  - "a" opens for appending
  - "b" specifies binary

- Returns the file pointer

---

**File I/O**

**File modes**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>r</code></td>
<td>Open a text file for <strong>reading</strong>.</td>
</tr>
<tr>
<td><code>w</code></td>
<td>Truncate to zero length or create a text file for <strong>writing</strong>.</td>
</tr>
<tr>
<td><code>a</code></td>
<td>Append; open or create a text file for <strong>writing at the end-of-file</strong>.</td>
</tr>
<tr>
<td><code>rb</code></td>
<td>Open a binary file for <strong>reading</strong>.</td>
</tr>
<tr>
<td><code>wb</code></td>
<td>Truncate to zero length or create a binary file for <strong>writing</strong>.</td>
</tr>
<tr>
<td><code>ab</code></td>
<td>Append; open or create a binary file for <strong>writing at the end-of-file</strong>.</td>
</tr>
<tr>
<td><code>r+</code></td>
<td>Open a text file for <strong>read/write</strong>.</td>
</tr>
<tr>
<td><code>w+</code></td>
<td>Truncate to zero length or create a binary file for <strong>read/write</strong>.</td>
</tr>
<tr>
<td><code>wb+</code></td>
<td>Append; open or create a binary file for <strong>read/write</strong>. You can read data anywhere in the file, but you can write data only at the end-of-file.</td>
</tr>
<tr>
<td><code>r+b</code> or <code>rb+</code></td>
<td>Open a binary file for <strong>read/write</strong>.</td>
</tr>
<tr>
<td><code>w+b</code> or <code>wb+</code></td>
<td>Truncate to zero length or create a binary file for <strong>read/write</strong>.</td>
</tr>
<tr>
<td><code>a+b</code> or <code>ab+</code></td>
<td>Append; open or create a binary file for <strong>read/write</strong>. You can read data anywhere in the file, but you can write data only at the end-of-file.</td>
</tr>
</tbody>
</table>
File I/O
fread()

Syntax

```c
size_t fread(void *ptr, size_t size,
             size_t count, FILE *stream);
```

- `ptr` – The buffer to write into
- `size` – The size of each element to read
- `count` – The number of elements to read
- `stream` – The pointer to the file
- Returns the number of elements read
  - Less than count indicates error or EOF
File I/O

feof()

Syntax

```c
int feof(FILE *stream);
```

- `stream` — The pointer to the file
- Returns a non-zero value if the stream is at the end of the file, 0 otherwise

File I/O

fseek()

Syntax

```c
int fseek(FILE *stream, long offset, int origin);
```

- `stream` — The pointer to the file
- `offset` — The bytes to move from the current location
- `origin` — The reference location: either SEEK_SET, SEEK_CUR, or SEEK_END
- Returns 0 if successful, otherwise returns a non-zero value
**File I/O**

**fclose()**

Syntax

```c
int fclose(FILE *stream);
```

- **stream** — The pointer to the file
- Returns 0 if successful, otherwise returns EOF
  - EOF is a macro, generally -1

**Example**

```c
int main(void)
{
    // Open the file, terminating if there was an error
    FILE *pFile = fopen("/room1.txt", "rb");
    if (pFile == NULL) {
        puts("Error opening file.");
        return EXIT_FAILURE;
    }

    // Count the characters in the file.
    int n = 0;
    while (fgetc(pFile) != EOF) {
        ++n;
    }

    // Output the results, if we succeeded
    if (feof(pFile)) {
        printf("Total bytes read: %d\n", n);
        fclose(pFile);
        return EXIT_SUCCESS;
    }

    // Otherwise output an error
    puts("Error occurred before reading end of file.");
    fclose(pFile);
    return EXIT_FAILURE;
}
```
File formats

Types

- Two groups:
  - Text
  - Binary

- Text are easier to process, but larger
- Binary are harder to process, but smaller
- Many formats are now zipped text files so the data is easy to parse, but the size is small
  - .docx/.xlsx for example
File formats

**Text: XML**

```
<messageInfo name = "System Time" pgn = "126992" size = "8">
  <field
    name = "Days since epoch"
    type = "int"
    offset = "16"
    length = "16"
    signed = "no"
    units = "days"
    endian = "little"

  <messageInfo name = "Rudder" pgn = "127245" size = "6">
  <field
    name = "Position"
    type = "int"
    offset = "32"
    length = "16"
    signed = "yes"
    units = "rad"
    scaling = "0.0001"
    endian = "little"
```

**Text: CSV**

```
timestamp, time_usec, fix_type, lat, lon, alt, mph, spw, vel, cog
57.300000000000004, 87300, 450, -59, -15887, 0.0, 0.0, 0.0, 0.0, 87460000, 3, 369640780, -1220013611, 0, 150, 159, 1, 13186
57.550000000000004, 57550, 457, -51, -15851, 0.0, 0.0, 0.0, 0.0, 57760000, 3, 369640785, -1220013613, 0, 149, 159, 1, 13141
57.800000000000004, 57800, 469, -42, -15554, 0.0, 0.0, 0.0, 0.0, 57960000, 3, 369640786, -1220013615, 0, 149, 159, 1, 13450
58.050000000000004, 58050, 474, -32, -15550, 0.0, 0.0, 0.0, 0.0, 58260000, 3, 369640785, -1220013615, 0, 149, 159, 2, 13620
58.300000000000004, 58300, 477, -17, -15846, 0.0, 0.0, 0.0, 0.0, 58460000, 3, 369640785, -1220013615, 0, 149, 159, 2, 13620
58.550000000000004, 58550, 474, -9, -15846, 0.0, 0.0, 0.0, 0.0, 58760000, 3, 369640793, -1220013616, 0, 150, 159, 1, 13607
58.800000000000004, 58800, 469, -12, -15843, 0.0, 0.0, 0.0, 0.0, 58960000, 3, 369640796, -1220013616, 0, 149, 159, 2, 13616
59.050000000000004, 59050, 468, -18, -15839, 0.0, 0.0, 0.0, 0.0, 59260000, 3, 369640798, -1220013618, 0, 150, 159, 2, 13486
59.300000000000004, 59300, 471, -14, -15841, 0.0, 0.0, 0.0, 0.0, 59460000, 3, 369640798, -1220013618, 0, 150, 159, 2, 13486
59.550000000000004, 59550, 485, -4, -15836, 0.0, 0.0, 0.0, 0.0, 59760000, 3, 369640803, -1220013618, 0, 149, 159, 1, 13441
59.800000000000004, 59800, 502, 0, -15833, 0.0, 0.0, 0.0, 0.0, 59960000, 3, 369640804, -1220013618, 0, 150, 159, 2, 13313
60.050000000000004, 60050, 502, -18, -15833, 0.0, 0.0, 0.0, 0.0, 60260000, 3, 369640808, -1220013618, 0, 150, 159, 2, 13030
60.300000000000004, 60300, 507, -28, -15839, 0.0, 0.0, 0.0, 0.0, 60460000, 3, 369640808, -1220013618, 0, 150, 159, 2, 13030
60.550000000000004, 60550, 504, -25, -15828, 0.0, 0.0, 0.0, 0.0, 60760000, 3, 369640815, -1220013620, 0, 149, 159, 1, 12704
60.800000000000004, 60800, 505, 20, -15824, 0.0, 0.0, 0.0, 0.0, 60960000, 3, 369640818, -1220013620, 0, 150, 159, 2, 12492
61.050000000000004, 61050, 524, -14, -15822, 0.0, 0.0, 0.0, 0.0, 61260000, 3, 369640823, -1220013621, 0, 149, 159, 1, 12492
61.300000000000004, 61300, 518, -7, -15844, 0.0, 0.0, 0.0, 0.0, 61460000, 3, 369640823, -1220013621, 0, 149, 159, 1, 12492
61.550000000000004, 61550, 512, 0, -15825, 0.0, 0.0, 0.0, 0.0, 61760000, 3, 369640830, -1220013623, 0, 150, 159, 5, 11498
61.800000000000004, 61800, 499, 0, -15825, 0.0, 0.0, 0.0, 0.0, 61960000, 3, 369640833, -1220013623, 0, 150, 159, 2, 11094
62.050000000000004, 62050, 485, -1, -15824, 0.0, 0.0, 0.0, 0.0, 62260000, 3, 369640836, -1220013623, 0, 149, 159, 1, 11094
```
File formats

Binary: ZIP

File formats

Binary: RPG

- Needed a format to store each room in a dungeon

- Requirements
  - Title
  - Description
  - Items in the room
  - Exits:
    - Which room
    - What direction
File formats
Binary: RPG

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
</table>

- But it would be cool if the rooms could change depending on items the player has encountered
  - Like keys
- So we want different versions of the room for:
  - Description
  - Items
  - Exits
## File formats

**Binary: RPG**

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(repeated)

---

## File formats

**Binary: RPG**

- **size**
- ASCII data

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
File formats
Binary: RPG

size binary data

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
</table>

File formats
Binary: RPG

size ASCII data

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
</table>
File formats
Binary: RPG

size | binary data

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
</table>

File formats
Binary: RPG

1D:

<table>
<thead>
<tr>
<th>North</th>
<th>East</th>
<th>South</th>
<th>West</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
</table>
### File formats

**Binary: RPG**

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Version 1: Requires key, no items*

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Version 2: No requirements, contains key*

/Room32.txt
A large metal throne forged of swords of previous kings sits prominently here. Your dad is rarely in it, however, instead ruling the kingdom from his council’s chambers.
## File formats

### Binary: RPG

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1</td>
<td>Requires key, no items</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version 2</td>
<td>No requirements, contains key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/Room32.txt

---

### Binary: RPG

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1</td>
<td>Requires key, no items</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version 2</td>
<td>No requirements, contains key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/Room32.txt
A large metal throne forged of swords of previous kings sits prominently here. Your dad is rarely in it, however, instead ruling the kingdom from his council’s chambers. You feel the weight of the castle key stolen earlier in your pocket.
### File formats

**Binary: RPG**

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Version 1: Requires key, no items</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Version 2: No requirements, contains key</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/Room32.txt