CMPE-013/L

Introduction to “C” Programming

Maxwell James Dunne
Spring 2016
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)
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

*Close() did nothing*
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
ascii
unicode
\texttt{a}\texttt{b}
16\texttt{b}i+s
# File I/O

## File modes

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

\texttt{fread()}\linebreak
\texttt{size_t fread(void *ptr, size_t size, size_t count, FILE *stream);}\linebreak

- \textit{ptr} – The buffer to write into \texttt{STDSIN}\linebreak
- \textit{size} – The size of each element to read\linebreak
- \textit{count} – The number of elements to read\linebreak
- \textit{stream} – The pointer to the file\linebreak
- Returns the number of elements read\linebreak
  – Less than count indicates error or \texttt{EOF}
File I/O

fred()

Syntax

```
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

```
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
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The grid is labeled with rows and columns, indicating a matrix or table structure. The question mark (?) suggests uncertainty or a missing 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;
}
```
Structs

File formats
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

```
<MetaData>
  <messageInfo name = "System Time" pgn = "126992" size = "8">
    <desc>Represents the current data and time</desc>
    <field
      name = "Days since epoch"
      type = "int"
      offset = "16"
      length = "16"
      signed = "no"
      units = "days"
      endian = "little"
    />
  </messageInfo>
  <messageInfo name = "Rudder" pgn = "127245" size = "6">
    <desc>Represents the current rudder position</desc>
    <field
      name = "Position"
      type = "int"
      offset = "32"
      length = "16"
      signed = "yes"
      units = "rad"
      scaling = "0.0001"
      endian = "little"
    />
  </messageInfo>
</MetaData>
```
### File formats

Text: CSV

<table>
<thead>
<tr>
<th>timestamp, time_usec, fix_type, lat, lon, alt, eph, epv, vel, cog</th>
</tr>
</thead>
<tbody>
<tr>
<td>57.300000000000000004, 57300, 450, -59, -15857, 0.0, 0.0, 0.0, 57460000, 3, 369640780, -1220013611, 0, 150, 159, 1, 13186</td>
</tr>
<tr>
<td>57.550000000000000004, 57550, 457, -51, -15855, 0.0, 0.0, 0.0, 57760000, 3, 369640785, -1220013613, 0, 149, 159, 1, 13411</td>
</tr>
<tr>
<td>57.800000000000000004, 57800, 469, -42, -15854, 0.0, 0.0, 0.0, 57960000, 3, 369640786, -1220013615, 0, 149, 159, 1, 13458</td>
</tr>
<tr>
<td>58.050000000000000004, 58050, 474, -32, -15850, 0.0, 0.0, 0.0, 58260000, 3, 369640788, -1220013615, 0, 149, 159, 2, 13620</td>
</tr>
<tr>
<td>58.300000000000000004, 58300, 477, -17, -15847, 0.0, 0.0, 0.0, 58460000, 3, 369640788, -1220013615, 0, 149, 159, 2, 13620</td>
</tr>
<tr>
<td>58.550000000000000004, 58550, 474, -9, -15846, 0.0, 0.0, 0.0, 58760000, 3, 369640793, -1220013616, 0, 150, 159, 1, 13607</td>
</tr>
<tr>
<td>58.800000000000000004, 58800, 469, -12, -15843, 0.0, 0.0, 0.0, 58960000, 3, 369640796, -1220013616, 0, 149, 159, 2, 13616</td>
</tr>
<tr>
<td>59.050000000000000004, 59050, 468, -18, -15839, 0.0, 0.0, 0.0, 59260000, 3, 369640798, -1220013618, 0, 150, 159, 2, 13486</td>
</tr>
<tr>
<td>59.300000000000000004, 59300, 471, -14, -15841, 0.0, 0.0, 0.0, 59460000, 3, 369640798, -1220013618, 0, 150, 159, 2, 13486</td>
</tr>
<tr>
<td>59.550000000000000004, 59550, 485, -4, -15836, 0.0, 0.0, 0.0, 59760000, 3, 369640803, -1220013618, 0, 149, 159, 1, 13441</td>
</tr>
<tr>
<td>59.800000000000000004, 59800, 501, -15833, 0.0, 0.0, 0.0, 59960000, 3, 369640804, -1220013618, 0, 150, 159, 2, 13313</td>
</tr>
<tr>
<td>60.050000000000000004, 60050, 502, -18, -15839, 0.0, 0.0, 0.0, 60260000, 3, 369640808, -1220013618, 0, 150, 159, 2, 13303</td>
</tr>
<tr>
<td>60.300000000000000004, 60300, 507, -28, -15839, 0.0, 0.0, 0.0, 60460000, 3, 369640808, -1220013618, 0, 150, 159, 2, 13300</td>
</tr>
<tr>
<td>60.550000000000000004, 60550, 504, -25, -15824, 0.0, 0.0, 0.0, 60760000, 3, 369640815, -1220013620, 0, 149, 159, 1, 12704</td>
</tr>
<tr>
<td>60.800000000000000004, 60800, 515, -20, -15824, 0.0, 0.0, 0.0, 60960000, 3, 369640818, -1220013620, 0, 149, 159, 2, 12492</td>
</tr>
<tr>
<td>61.050000000000000004, 61050, 524, -14, -15832, 0.0, 0.0, 0.0, 61260000, 3, 369640823, -1220013621, 0, 149, 159, 1, 12492</td>
</tr>
<tr>
<td>61.300000000000000004, 61300, 518, -7, -15844, 0.0, 0.0, 0.0, 61460000, 3, 369640823, -1220013621, 0, 149, 159, 1, 12492</td>
</tr>
<tr>
<td>61.550000000000000004, 61550, 512, 0, -15825, 0.0, 0.0, 0.0, 61760000, 3, 369640830, -1220013623, 0, 150, 159, 5, 11498</td>
</tr>
<tr>
<td>61.800000000000000004, 61800, 494, -0, -15825, 0.0, 0.0, 0.0, 61960000, 3, 369640833, -1220013623, 0, 150, 159, 2, 11094</td>
</tr>
<tr>
<td>62.050000000000000004, 62050, 485, -1, -15824, 0.0, 0.0, 0.0, 62260000, 3, 369640836, -1220013623, 0, 149, 159, 1, 11094</td>
</tr>
</tbody>
</table>
import zipfile

File formats

Binary: ZIP

Relative offset 1

Relative offset 2

Relative offset 3

Relative offset n

FILE ENTRY 1

<data>

FILE ENTRY 2

<data>

FILE ENTRY 3

<data>

FILE ENTRY 4

<data>

CENTRAL DIRECTORY

File entry 1

File entry 2

File entry 3

Local header 1

Local header 2

Local header 3

Local header n

Maxwell James Dunne – Spring 2016
File formats

Binary: RPG

- Needed a format to store each room in a dungeon
- Requirements
  - Title: the throne room
  - Description:
  - Items in the room:
  - Exits:
    - Which room
    - What direction

Ph (encrypted)
## File formats

**Binary: RPG**

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

Maxwell James Dunne – Spring 2016
File formats

Binary: RPG

- 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)

46
File formats

Binary: RPG

<table>
<thead>
<tr>
<th>size</th>
<th>ASCII data</th>
</tr>
</thead>
</table>

Title

Item requirements

Description

Items contained

Exits
File formats

Binary: RPG

<table>
<thead>
<tr>
<th>size</th>
<th>binary data</th>
</tr>
</thead>
</table>

| Title | Item requirements | Description | Items contained | Exits |
File formats

Binary: RPG

<table>
<thead>
<tr>
<th>size</th>
<th>ASCII data</th>
</tr>
</thead>
</table>

Title | Item requirements | Description | Items contained | Exits |
File formats

Binary: RPG

0-2^55

<table>
<thead>
<tr>
<th>size</th>
<th>binary data</th>
</tr>
</thead>
</table>

Title | Item requirements | Description | Items contained | Exits
File formats

Binary: RPG

<table>
<thead>
<tr>
<th>North</th>
<th>East</th>
<th>South</th>
<th>West</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>15</td>
<td>42</td>
<td>0</td>
</tr>
</tbody>
</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>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version 1:</strong> Requires key, no items</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Version 2:</strong> No requirements, contains key</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/Room32.txt
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>The Throne Room</td>
<td>Version 1: Requires key, no items</td>
<td>Version 2: No requirements, contains key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/Room32.txt
### 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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Version 1:** Requires key, no items
- **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.

<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
## 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
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>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>
</tr>
</tbody>
</table>

Version 2: No requirements, contains key

/Room32.txt
# 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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Version 1:** Requires key, no items

**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. You feel the weight of the castle key stolen earlier in your pocket.

<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: Requires key, no items</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version 2: No requirements, contains key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/Room32.txt
## 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
File formats

Binary: RPG

Version 1: Requires key, no items
Version 2: No requirements, contains key

/Rom32.txt
CMPE-013/L

Introduction to “C” Programming

Maxwell James Dunne

Spring 2016
More FSM
Our Roach State Machine

- **Hiding**
  - Light Goes On
    - Drive Fwd.
  - Light Goes Off
    - Stop

- **Driving_Forward**
  - Hit Obj
    - Left Reverse, Start Timer
  - Timer Expires (Light On)
    - Straight, Drive Fwd.

- **Backing_Up**
  - Hit Rear Obj
    - Forward Right, Start Timer
  - Timer Expired
    - Left Reverse, Start Timer

- **Evade_Forward**
  - Timer Expired
    - Stop
Entry and Exit Events

- We can combine these duplicated actions into entry and exit events
- Instead of having a repeated actions we now have actions associated with states
  - For example: everytime we enter hiding we should stop the roach
Entry Roach State Machine

EVENT

Hiding
entry: STOP

Light Goes On    Light Goes Off

Driving Forward
entry: Drive Forward

Hit Obj    Timer Expires (Light On)

Backing Up
entry: Left Reverse, Start Timer

Hit Rear Obj    Timer Expired

Evade Forward
entry: Forward Left, Start Timer
Coding Entry and Exit Events

- State machines with these events can no longer merely change state. They need to make use of a makeTransition flag.
  - This is needed to ensure the entry and exit events are handled correctly.

  - nextState = BACKING_UP;
  - makeTransition = TRUE;
Making the state transition

- The actual state transition requires a recursive call to ensure that the events are handled correctly.

```c
if (makeTransition == TRUE) {
    RunRoachStateMachine(EXIT_EVENT);
    myState = nextState;
    RunRoachStateMachine(ENTRY_EVENT);
}
```
FSM to HSM

• With the addition of the exit and entry events we can now combine similar actions and remove duplicate behavior.

• This allows for a cleaner design but more complex problems still cause “state explosion”
  – Many of these extra states come from repetition

• HSM’s address this problem by introducing superstates with substates below them
Top Level State Machine

- **Off**
  - ON
  - Button_press
  - DUNE Brushing

- **Brushing**
  - Off
  - DUNE Brushing

- **Powerdock**
  - ON
  - Off
  - LED off

- **LED off**
  - ON

Maxwell James Dunne – Spring 2016
In Charging State Machine

- Charging
- Charged
- Timeout

Diagram showing the state transitions between Charging and Charged states, with Timeout as a possible transition.
In Brushing State Machine
Code Example of the toothBrush
Line follow

Find beacon

Drift to B

30-40 states

Integration testing
Introduction to “C” Programming

Maxwell James Dunne
Spring 2016
Battle Boats

3) Late hours max(p1, p2)
4) RPG no late hours
5) Quizzes 30% Labs: 70%

3) due Thursday
2) partnered
| | The start-of-message identifier, always a dollar-sign |
| | **MESSAGE_ID** | A 3-character string identifying the type of message. |
| | **,'** | A comma separates the MESSAGE_ID from the subsequent data |
| | **DATA1,DATA2,DATA3,...** | A comma-separated list of data, all encoded as ASCII characters |
| | **XX** | A message ends with an asterisk and then a checksum byte encoded as two separate ASCII hexadecimal characters (like '0A'). This checksum is calculated from ALL bytes between the '$' and the '*'. |
| | **\n** | A newline character actually ends the string. |
• Agent A generates a random 16-bit number that is its "guess" along with another 16-bit number that is used as the encryption key.

• Agent A then transmits a checksum of both its guess and key (which is an 8-bit XOR of all of their bytes) along with an encrypted version of its guess (which is a 16-bit XOR of the guess with the encryptionKey).

• During this time Agent B is doing the same thing.
guess 16
Ekey 16

16 16  checksum

Checksum

guess XOR Ekey

unencrypt
• Once Agent A has received Agent B's encrypted guess and checksum, it transmits the unencrypted guess and the encryption key (and Agent B does the same).

• 5. Agent B can now verify Agent A's information by verifying both the checksum and the encryption key (and Agent A does the same).

• 6. Now both can agree on who should go first by having either guessed higher or lower than the other agent depending on if the XOR of the LSB of their guesses is 1 or 0.
$DET,0.1 \times 6E$
<table>
<thead>
<tr>
<th>Negotiation Data Set 1</th>
<th>$CHA, 37348, 117*46</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$DET, 9578, 46222*66</td>
</tr>
<tr>
<td>Negotiation Data Set 2</td>
<td>$CHA, 54104, 139*45</td>
</tr>
<tr>
<td></td>
<td>$DET, 32990, 21382*5e</td>
</tr>
<tr>
<td>Negotiation Data Set 3</td>
<td>$CHA, 62132, 70*79</td>
</tr>
<tr>
<td></td>
<td>$DET, 52343, 16067*50</td>
</tr>
<tr>
<td>Negotiation Data Set 4</td>
<td>$CHA, 36027, 55*7a</td>
</tr>
<tr>
<td></td>
<td>$DET, 7321, 36898*6e</td>
</tr>
<tr>
<td>HIT messages</td>
<td>$HIT, 3, 8, 1*43</td>
</tr>
<tr>
<td></td>
<td>$HIT, 0, 2, 0*4b</td>
</tr>
<tr>
<td></td>
<td>$HIT, 2, 3, 1*49</td>
</tr>
<tr>
<td></td>
<td>$HIT, 5, 6, 4*4e</td>
</tr>
<tr>
<td></td>
<td>$HIT, 0, 3, 0*4a</td>
</tr>
<tr>
<td></td>
<td>$HIT, 1, 7, 1*4e</td>
</tr>
<tr>
<td></td>
<td>$HIT, 4, 8, 0*45</td>
</tr>
<tr>
<td></td>
<td>$HIT, 5, 3, 3*4c</td>
</tr>
<tr>
<td></td>
<td>$HIT, 0, 5, 0*4c</td>
</tr>
<tr>
<td></td>
<td>$HIT, 5, 6, 1*4b</td>
</tr>
<tr>
<td></td>
<td>$HIT, 1, 1, 1*48</td>
</tr>
<tr>
<td></td>
<td>$HIT, 1, 0, 0*48</td>
</tr>
<tr>
<td></td>
<td>$HIT, 5, 2, 5*4b</td>
</tr>
<tr>
<td></td>
<td>$HIT, 2, 8, 0*43</td>
</tr>
<tr>
<td></td>
<td>$HIT, 0, 6, 0*4f</td>
</tr>
<tr>
<td></td>
<td>$HIT, 5, 9, 0*45</td>
</tr>
<tr>
<td></td>
<td>$HIT, 2, 8, 2*41</td>
</tr>
</tbody>
</table>
while(old_addboat((rand, rand)))
\[ 70\% < 62\% \]

\[ |A| |B| |C| < 1 \]
What files needed

protocol

Field →

BB

BB

-1

x

\text{switch}

\text{Random}

\text{Agent} \rightarrow H_{agent} \to
'0'-'9', 'A-F', 'a-f'  
13'-65+10'  
'8'-48'  

switch  
case '0'... '9'  
case 'a'-... 'f'  

grade separately