CMPE-013/L

Introduction to “C” Programming

Maxwell James Dunne
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

Maxwell James Dunne
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>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

fwrite()

Syntax

```c
size_t fwrite(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**

`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

```
0-255
int -1
```
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
```
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;
    }
    else
    // Otherwise output an error
    puts("Error occurred before reading end of file.");
    fclose(pFile);
    return EXIT_FAILURE;
}
```
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

```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</th>
<th>time_usec</th>
<th>fix_type</th>
<th>lat</th>
<th>lon</th>
<th>alt</th>
<th>eph</th>
<th>epv</th>
<th>vel</th>
<th>cog</th>
</tr>
</thead>
<tbody>
<tr>
<td>57.30</td>
<td>57300</td>
<td>450</td>
<td>-59</td>
<td>-15857</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>57460000</td>
<td>3</td>
</tr>
<tr>
<td>57.55</td>
<td>57550</td>
<td>457</td>
<td>-51</td>
<td>-15855</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>57760000</td>
<td>3</td>
</tr>
<tr>
<td>57.80</td>
<td>57800</td>
<td>469</td>
<td>-42</td>
<td>-15854</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>57960000</td>
<td>3</td>
</tr>
<tr>
<td>58.05</td>
<td>58050</td>
<td>474</td>
<td>-32</td>
<td>-15850</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>58260000</td>
<td>3</td>
</tr>
<tr>
<td>58.30</td>
<td>58300</td>
<td>477</td>
<td>-17</td>
<td>-15847</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>58460000</td>
<td>3</td>
</tr>
<tr>
<td>58.55</td>
<td>58550</td>
<td>474</td>
<td>-9</td>
<td>-15846</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>58760000</td>
<td>3</td>
</tr>
<tr>
<td>58.80</td>
<td>58800</td>
<td>469</td>
<td>-12</td>
<td>-15843</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>58960000</td>
<td>3</td>
</tr>
<tr>
<td>59.05</td>
<td>59050</td>
<td>468</td>
<td>-18</td>
<td>-15839</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>59260000</td>
<td>3</td>
</tr>
<tr>
<td>59.30</td>
<td>59300</td>
<td>471</td>
<td>-14</td>
<td>-15841</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>59460000</td>
<td>3</td>
</tr>
<tr>
<td>59.55</td>
<td>59550</td>
<td>485</td>
<td>-4</td>
<td>-15836</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>59760000</td>
<td>3</td>
</tr>
<tr>
<td>59.80</td>
<td>59800</td>
<td>501</td>
<td>0</td>
<td>-15833</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>59960000</td>
<td>3</td>
</tr>
<tr>
<td>60.05</td>
<td>60050</td>
<td>502</td>
<td>-18</td>
<td>-15839</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>60260000</td>
<td>3</td>
</tr>
<tr>
<td>60.30</td>
<td>60300</td>
<td>507</td>
<td>-28</td>
<td>-15839</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>60460000</td>
<td>3</td>
</tr>
<tr>
<td>60.55</td>
<td>60550</td>
<td>504</td>
<td>-25</td>
<td>-15824</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>60760000</td>
<td>3</td>
</tr>
<tr>
<td>60.80</td>
<td>60800</td>
<td>515</td>
<td>-20</td>
<td>-15824</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>60960000</td>
<td>3</td>
</tr>
<tr>
<td>61.05</td>
<td>61050</td>
<td>524</td>
<td>-14</td>
<td>-15832</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>61260000</td>
<td>3</td>
</tr>
<tr>
<td>61.30</td>
<td>61300</td>
<td>518</td>
<td>-7</td>
<td>-15844</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>61460000</td>
<td>3</td>
</tr>
<tr>
<td>61.55</td>
<td>61550</td>
<td>512</td>
<td>0</td>
<td>-15825</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>61760000</td>
<td>3</td>
</tr>
<tr>
<td>61.80</td>
<td>61800</td>
<td>494</td>
<td>-15825</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>61960000</td>
<td>3</td>
<td>369640833</td>
</tr>
<tr>
<td>62.05</td>
<td>62050</td>
<td>485</td>
<td>-1</td>
<td>-15824</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>62260000</td>
<td>3</td>
</tr>
</tbody>
</table>
File formats

Binary: ZIP

<meso six>

Relative offset 1
Relative offset 2
Relative offset 3
Relative offset n

FILE ENTRY 1
<data>
Local header 1

FILE ENTRY 2
<data>
Local header 2

FILE ENTRY 3
<data>
Local header 3

FILE ENTRY 4
<data>

CENTRAL DIRECTORY

File entry 1
File entry 2
File entry 3

Local header n
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>

Maxwell James Dunne
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>
<tr>
<td>(repeated)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Title
- Item requirements
- Description
- Items contained
- Exits
# File formats

**Binary: RPG**

<table>
<thead>
<tr>
<th>size</th>
<th>ASCII data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</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 |

CMPE-013/L: “C” Programming
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>North</th>
<th>East</th>
<th>South</th>
<th>West</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></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>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</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></td>
<td></td>
<td>Version 1: Requires key, no items</td>
<td></td>
<td></td>
<td>Version 2: 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></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
### File formats

**Binary: RPG**

<table>
<thead>
<tr>
<th>Title</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>1</td>
<td>Item requirements</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>
<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: 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></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
File formats

Binary: RPG

N E S W

0 0 30 0

<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>Version 1</td>
<td>Requires key, no items</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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></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
### 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
gcc (GNU C and C++ compiler)

• First compiler for GNU and adapted by many operating systems
• MPLABX is calling a customized variant of GCC to generate the hex file
• Supports both C and C++
gcc
Basic Usage

Syntax

gcc  -o  outfile  source  files

•  -o  Sets  the  executable  output  name
  –  Without  argument  defaults  to  a  (a.exe  on  cygwin)
•  Source  files  the  set  of  source  files  to compile.
  –  Example
    •  gcc  SimpleMain.c
    •  gcc  -o  mml  mml_tester.c  MatrixMath.c
Object files allow individual compilation of source files (skips the link step)
  – Generally not runnable but have machine code for the source file
• In large projects this is essential as full compiles can take hours to complete
• .o files can then be compiled together without the flag as normal.

15 minute
HTTP://XKCD.COM/303/

THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."

HEY! GET BACK TO WORK!

COMPILING!

OH. CARRY ON.
GCC
Object File Creation

**Syntax**

```sh
gcc -c source files
```

- `-c` Tells the compiler to skip linking
- **Source files** the set of source files to compile into an object
  - Each file will generate a different `.o` file
  - Example
    - `gcc -c mml_tester.c`
make
make

- Command line tool designed to make the process of compiling code easier.
- Parses a makefile to determine which actions to take.
- MPLABX generates a makefile with the project and that is called when the hammer button is clicked.
- Incredibly powerful tool as complex as C itself, will only cover the very basic commands.
Invoking make

• Simply type “make” on the command line
• Make will attempt use a file called makefile and process targets from it
• If called without arguments it attempts to run the target all
• With arguments it attempts to create that specific target
• Smart enough to only execute targets that need updating
make
makefile contents

target: dependencies
actions to make target

newtarget: depedency1 dependency2
more actions
Selective Compilation with make
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 Exp (Light Off): 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

- **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

Maxwell James Dunne – Spring 2016
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.
Messy State Machine

**Reset brush state**
- count = 0
- stop hiccup timer
- stop main timer
- set hiccup timer 30s
- set main timer 2min

**Electric Toothbrush FSM**

- **ES_TIMEOUT(main)**
  - reset brush state

- **ES_TIMEOUT(hiccup)/(count > 3)**
  - reset brush state

- **ES_TIMEOUT(main)**
  - reset brush state

**Brushing**
- **/entry turn motor on**
- **start main timer**
- **start hiccup timer**
- **/exit turn motor off**

**Hiccuping**
- **/entry counter++**
- **start hiccup timer**
- **/exit set hiccup timer 30sec**

**Charging**
- **/entry blink LED**
- **start timer**
- **charger on**
- **/exit LED off**
- **charger off**

**On Charger (Vbat >)**

**Out of Charger**
- **reset brush state**

**On Charger (Vbat <)**

**ES_TIMEOUT(Vbat <)**
- **reset brush state**

**ES_TIMEOUT(Vbat >)**
- **reset brush state**

**Button Press**
- **STOP RESUME TIMER**
- **start resume timer**
- **halt hiccup timer**
- **halt main timer**

**Charged**
- **/entry LED on**
- **start timer**
- **/exit LED off**
- **charger off**

**Charging**
- **/entry blink LED**
- **start timer**
- **charger on**
- **/exit LED off**
- **charger off**

**Off**
- **/entry turn motor off**
- **/exit halt main timer**
In Charging State Machine

- Charging
  - Not charged
  - Fully charged
- Charged
In Brushing State Machine
Late dates
1) Node ats for Lab 10
2) \( \max(p_1, p_2) \)

Battle Boats
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'$'</td>
<td>The start-of-message identifier, always a dollar-sign</td>
</tr>
<tr>
<td>MESSAGE_ID</td>
<td>A 3-character string identifying the type of message.</td>
</tr>
<tr>
<td>','</td>
<td>A comma separates the MESSAGE_ID from the subsequent data</td>
</tr>
<tr>
<td>DATA1,DATA2,DATA3,...</td>
<td>A comma-separated list of data, all encoded as ASCII characters</td>
</tr>
<tr>
<td>'*XX'</td>
<td>A message ends with an asterisk and then a checksum byte encoded as two separate ASCII hexadecimal characters (like '0A'). This checksum is calculated from ALL bytes between the '$' and the '*'</td>
</tr>
<tr>
<td>'\n'</td>
<td>A newline character actually ends the string</td>
</tr>
</tbody>
</table>
• 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.
• 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.
char in

switch...

WAITING
  Input != 'S'
  Return WAITING
  Index = 0
  Return GOOD

RECORDING
  Input == 'S'
  Sentence[Index] = Input
  Index += 1
  Return GOOD

FIRST_CHECKSUM_HALF
  Invalid hex character received
  Return FAILURE
  Valid hex character received
  Save hex value as top 4-bits of stateData->checksum
  Return GOOD

SECOND_CHECKSUM_HALF
  Input != '\n'
  MSGID invalid
  Return FAILURE
  Input is valid hex char && checksum matching succeeded
  Sentence[Index] = '\0'
  Return GOOD

NEWLINE

Input is '\n' && MSGID is one of ('DET', 'CHA', 'COO', 'HIT')
Return one of PARSED*
<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*40</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>
message hashing

\$p + \text{load} \rightleftharpoons \text{EA} \uparrow

\text{Sprintf}(\text{p}, \ldots, \ldots);

11 \times 0, 4, 5 \rightarrow 10

(((<^0)^0)^0)^4)
Generate
key = rand()
guess = rand()

eg = key ^ guess

hash
[guess | key]

↑ ↑ ↑ ↑ ↑

eg → hash

eg → key
guess
\[ eg \land \text{key} \Rightarrow \text{guess} \]

\[ \text{guess} \land \text{key} \]

\[ \text{new hash} \Rightarrow \text{hash} \]
`(rand() % num)`

`0 ↔ 1 - num`

`$c = 000$`
\[ \text{row} -- \quad \text{col} \rightarrow \quad \text{col}++ \]

\[ \text{row}++ \]