CMPE-013/L

Introduction to “C” Programming

Maxwell James Dunne
gcc
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
**gcc**

Object Files

- 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.
HTTP://XKCD.COM/303/

CPU sim
~900
6 hours

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

```
 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 to 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: dependency1 dependency2
  more actions
Selective Compilation with make

- Mml.c
  - Mml.o
- Mm.c
  - Mm.o
  - Mml.out
Battle Boats

Late days

max(partner1, partner2)

Lab 10
The NMEA Protocol

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

A

<table>
<thead>
<tr>
<th>guess</th>
<th>xor</th>
<th>key</th>
<th>checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>guess xor</th>
<th>key</th>
<th>checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>guess</th>
<th>xor</th>
<th>key</th>
<th>checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>guess</th>
<th>xor</th>
<th>key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E guess</th>
<th>key</th>
<th>guess</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5 \texttt{scanf} \\
5 \texttt{scanf} \left(\'c 00,4,5\', \%u, \%u\right) \texttt{scanf} \left(\right)
for i in range(1, n):
    print(decoder(i))
    print(encode)
[DET, 4 3 2 1, 1 4 2 1 3]
"<00, 1, 2, 310

calculate_checksum(char* foo, int length)
```c
for(i in String)
    printfDecode(i);
```

<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>
</tbody>
</table>

**HIT messages**

- $HIT, 3, 8, 1*43
- $HIT, 0, 2, 0*4b
- $HIT, 2, 3, 1*49
- $HIT, 5, 6, 4*4e
- $HIT, 0, 3, 0*4a
- $HIT, 1, 7, 1*4e
- $HIT, 4, 8, 0*45
- $HIT, 5, 3, 3*4c
- $HIT, 0, 5, 0*4c
- $HIT, 5, 6, 1*4b
- $HIT, 1, 1, 1*48
- $HIT, 1, 0, 0*48
- $HIT, 5, 2, 5*4b
- $HIT, 2, 8, 0*43
- $HIT, 0, 6, 0*4f
- $HIT, 5, 9, 0*45
- $HIT, 2, 8, 2*41
Partners
someone else

T W uno 32
// delay

~15

Protocol or field

1 test ← AA → 2 test

Underline: ~15
Agent run(1charin)

\[
\text{if } \overrightarrow{\text{in}} = \overrightarrow{0}
\]
message checksum

# * UE
Fert Decodr

generate $AE$

state machine
Protocol

turn 1

printf("%s", message)
Agent run

\(\text{if}(\text{in})\)
\(\text{protodecode}\)

\$\$

State Machine
127 + 0 = 204
both six repos