Typical C/C++ Language Processing

1. `file.c` or `file.cpp`
   To do all the steps, type:
   - `gcc file.c` (for C, or)
   - `g++ file.c` (for C++)

2. `cpp` (preprocessor)
   Handles `#include`, `#define`, `#if`, etc.

3. `file.i`

4. `cc1` (C compiler)
   This is the actual compiler

5. `file.s`
   To stop here, type:
   - `g++ -S file.cpp`
C/C++ Language Processing (cont’d)

file.s

libc
user libraries
other modules

as
(assembler)

file.o

ld
(linker)

a.out

Translates from assembly language into machine (binary) – unique to each processor type

Object file – to stop here:
g++ -c file.cpp
It is common to stop here

"Knits" together (links) all of the independent program modules

This is probably called file.exe on Windows systems
ELF - Relocatable Object File

<table>
<thead>
<tr>
<th>Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>.text</td>
</tr>
<tr>
<td>.rodata</td>
</tr>
<tr>
<td>.data</td>
</tr>
<tr>
<td>.bss</td>
</tr>
<tr>
<td>.symtab</td>
</tr>
<tr>
<td>.rel.text</td>
</tr>
<tr>
<td>.rel.data</td>
</tr>
<tr>
<td>.debug</td>
</tr>
<tr>
<td>.line</td>
</tr>
<tr>
<td>.strtab</td>
</tr>
</tbody>
</table>

Describes object file sections.
ELF - Executable File

<table>
<thead>
<tr>
<th>ELF Header</th>
<th>Section Header Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment Header Table</td>
<td></td>
</tr>
<tr>
<td>.init</td>
<td>Read-only memory</td>
</tr>
<tr>
<td>.text</td>
<td>segment (code)</td>
</tr>
<tr>
<td>.rodata</td>
<td>Read-Write memory</td>
</tr>
<tr>
<td>.data</td>
<td>segment (data)</td>
</tr>
<tr>
<td>.bss</td>
<td>Symbol table &amp;</td>
</tr>
<tr>
<td>.symtab</td>
<td>debugging info</td>
</tr>
<tr>
<td>.debug</td>
<td>(Not loaded into</td>
</tr>
<tr>
<td>.line</td>
<td>memory)</td>
</tr>
<tr>
<td>.strtab</td>
<td></td>
</tr>
</tbody>
</table>
LINUX Runtime Memory Image

- Kernel virtual memory
- User stack
- Shared Libraries
- Run-time heap
- Read-Write segment (.data, .bss)
- Read-only segment (.init, .text, .rodata)
- Unused

Inaccessible to user code
%esp
brk
Loaded from executable file

0x40000000
0x00400400
0xc0000000
LINUX Startup Pseudo-Code

/* crt1.o */

_start: /* entry point in .text */
    call __libc_init_first /* startup code in .text */
call _init /* startup code in .init */
call atexit /* startup code in .text */
/* set up argument list for main here */
call main /* application main code */
call _exit /* returns control to shell */