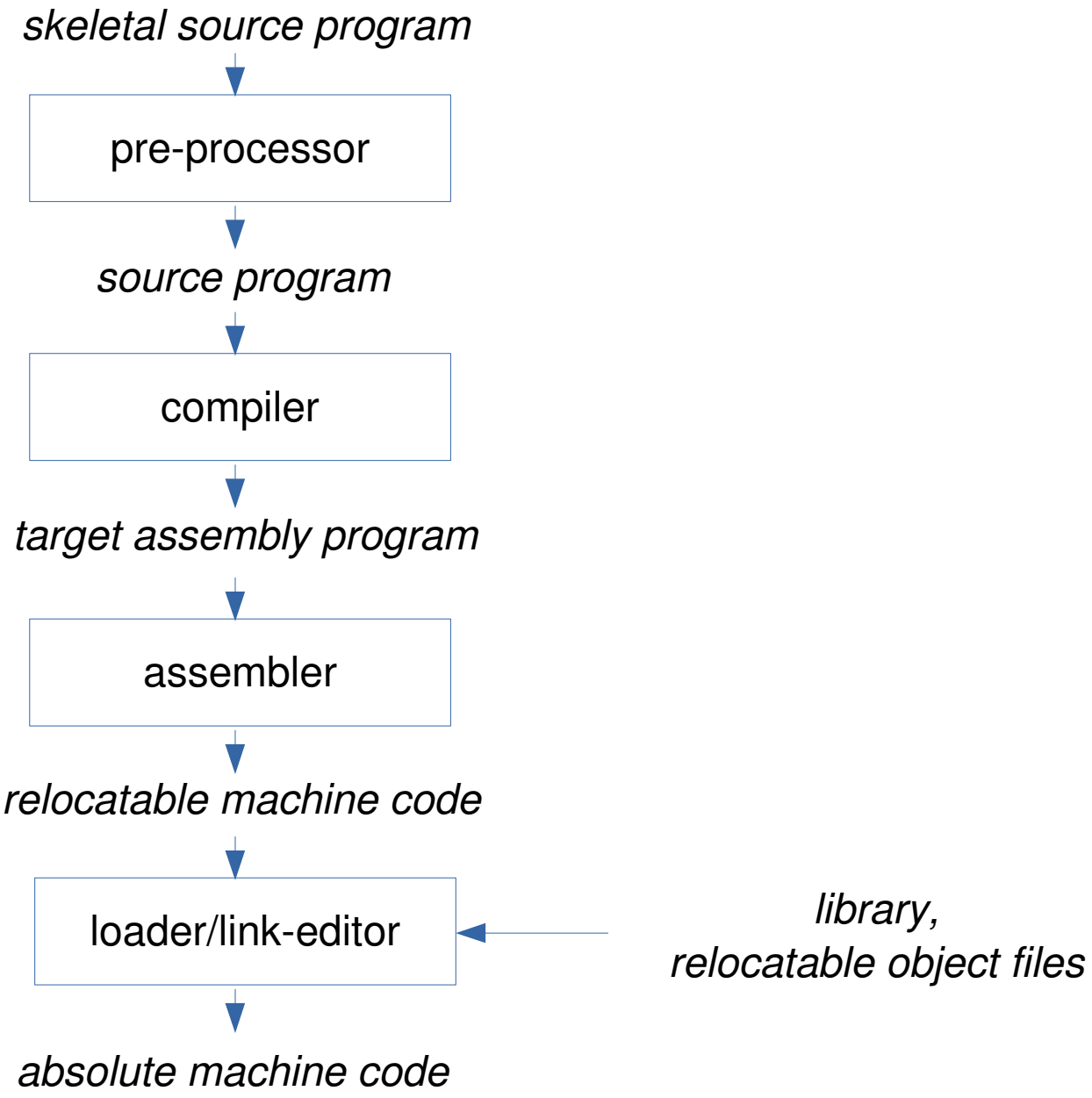
EECS 665 Compiler Construction

Lab 02
Process Execution (Unix/Linux)

Outline

- Software processing system
- Compile-time errors
- Link-time errors
- Run-time errors
- Useful system tools

Software Processing System



Errors During Compilation

- C pre-processor
  - handles directives that start with a '#' sign
  - eg., copies '#include' files
  - eg., expands '#define' definitions
  - eg., resolves conditional compilation with '#if – #endif', or '#ifdef – #endif', etc.
- Compare
  - `gcc pre-proc1.c`
  - `gcc -DCORRECT pre-proc1.c`
- Question-1: How and why does '-DCORRECT' remove errors and warnings?

Errors During Compilation – 2

- Linker
  - resolves external references
  - “pow” and “printf” are external references
- Compare
  - `gcc linker.c`
  - `gcc linker.c -lm`
- Question-2: What does '-lm' do to resolve the error?

GCC Paths

- To find GCC's default “include” and “library” search path
  - `gcc -v linker.c`
  - `gcc -print-search-dirs linker.c`
- `Extend GCC's “include” path`
  - `gcc -I./include linker2.c`
- `Extend GCC's “library search” path`
  - `gcc -I./include -L./lib linker2.c -lmul`
- `First, lets create libmul.so`

Building Shared Object Files

- Shared between multiple running processes
  - only one library instance in memory at a time
- Compile into *position independent code*
  - gcc -c -fpic <file.c>
- Create a shared library
  - gcc -shared -o <libfile.so> <file.o>
- Task
  - Create libmul.so
  - Compile and execute linker2.c

More Linking

- See linker3.c
  - functions 'foo', 'my_mul', 'printf are external references
- Compare 'objdumps' of .o and .exe files
  - `gcc -c linker3.c`
  - `gcc -L./lib linker3.c foo.c -lmul`
  - `readelf --relocs ./linker3.o`
  - `objdump -D linker3.o > temp1`
  - `objdump -D a.out > temp`
- See the entries for 'foo', 'my_mul', and 'printf' in 'temp' and 'temp1'
- Question-3: What has the linker done in 'a.out' compared to 'linker3.o'?

Execution Time Errors

- Run the program
  - `./a.out`
  - error while loading shared libraries: libmul.so:
- LD_LIBRARY_PATH
  - `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./lib`
- See objdump
  - function <my_mul@plt>

Useful system tools

- nm
  - list symbols from object file
- objdump
  - display information from object files
- readelf
  - display information about ELF files
- See 'man' pages

How Program Executes – Briefly

- GCC links program with '*_start'* routine
  - start address of executable is set to address of _start
- The shell calls *execve* with *argc* and *argv*
- *execve* system call handler loads executable, initializes process memory
- Control passed to *_start*
- _start calls *__libc_start_main*
- *__libc_start_main* calls our program's *main*
-
- References
  - http://eli.thegreenplace.net/2012/08/13/how-statically-linked-programs-run-on-linux
  - http://www.tldp.org/LDP/LGNET/issue84/hawk.html