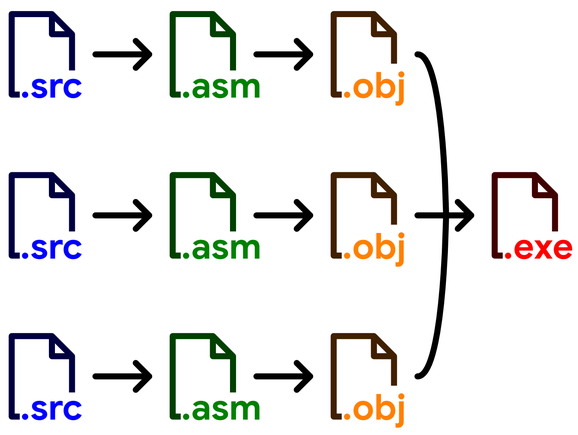
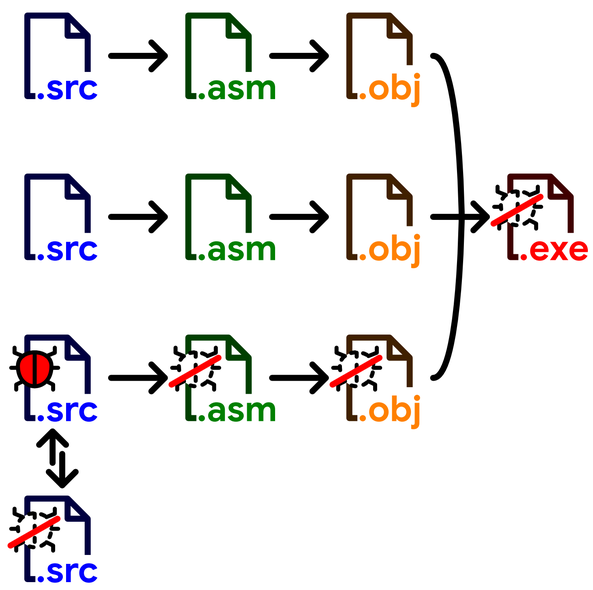
Delinking: stripping programs for parts

What's the problem?

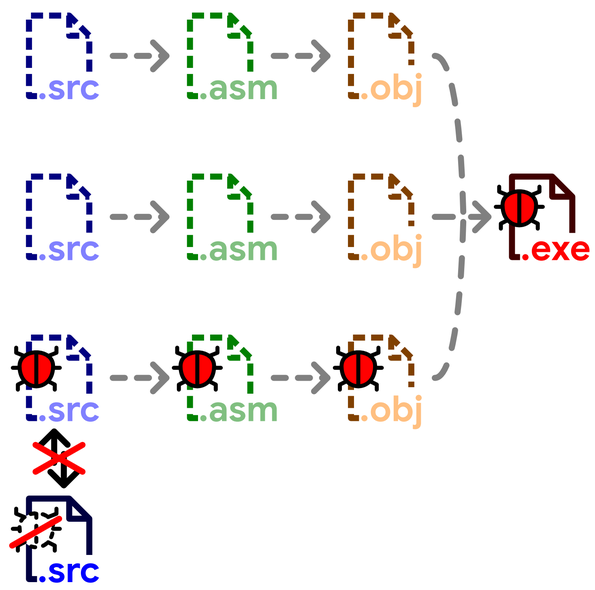


In order to create a new program, the developer writes a set of source code files, then invokes the toolchain:
- Each source code file is compiled into an assembly file.
- Each assembly file is assembled into an object file.
- All object files are linked into an executable file.



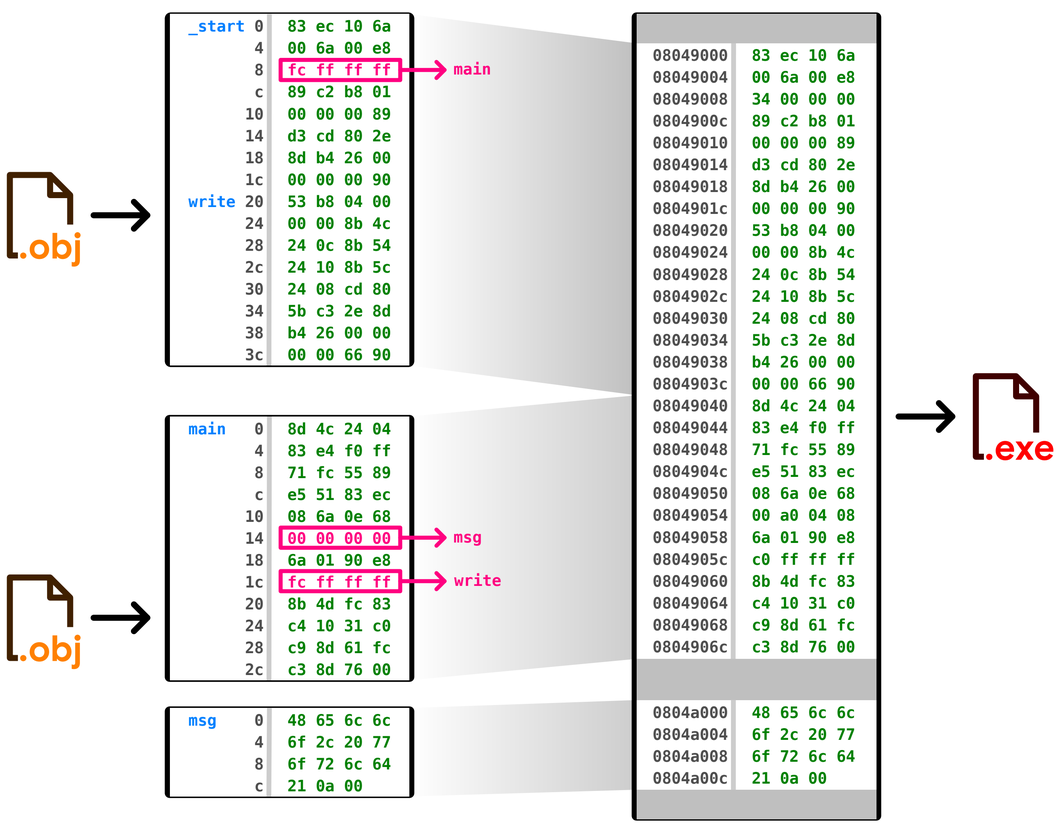
If a program needs a new feature or a bug fix, the developer can create a new executable file by:
- Adding or replacing source code files.
- Invoking the toolchain again.

... as long as they have the sources on hand.



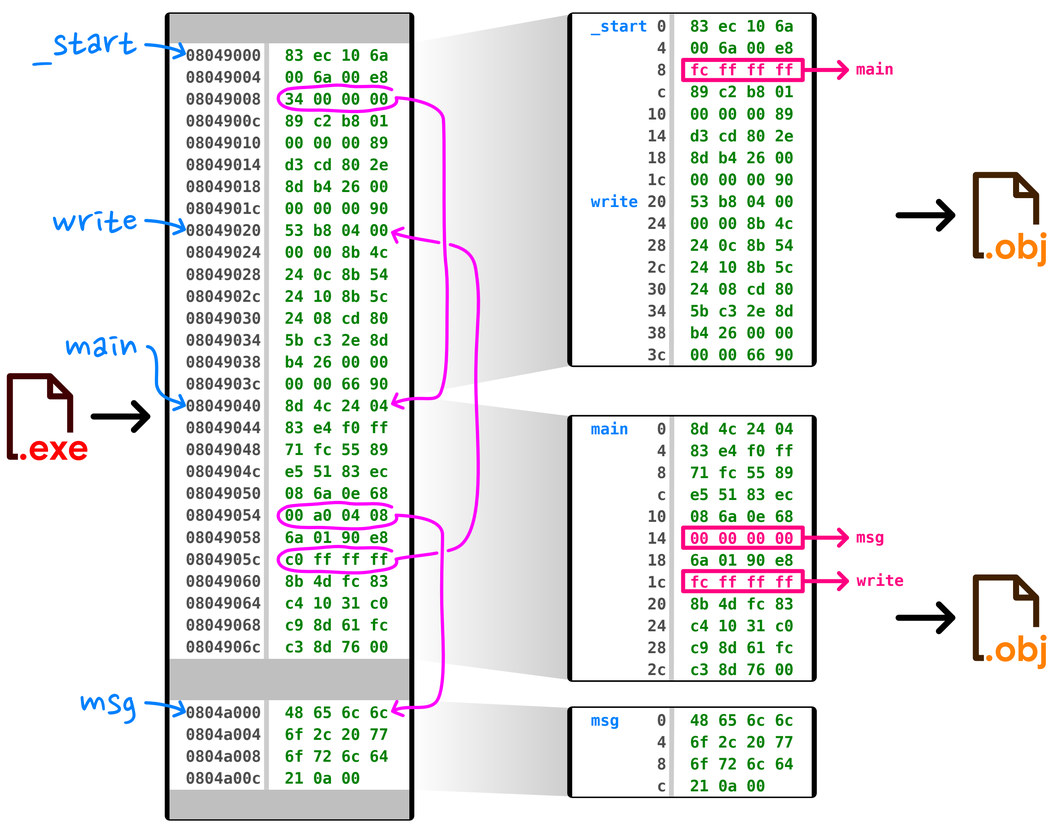
If the source code files are lost, the developer can't use the toolchain to create new executable files anymore, so the program must either:
- Be reimplemented from scratch.
- Be binary patched, hooked...

What's delinking?



Relocatable object files are made up of:
- One or more sections (arrays of bytes).
- A set of symbols.
- A set of relocations.

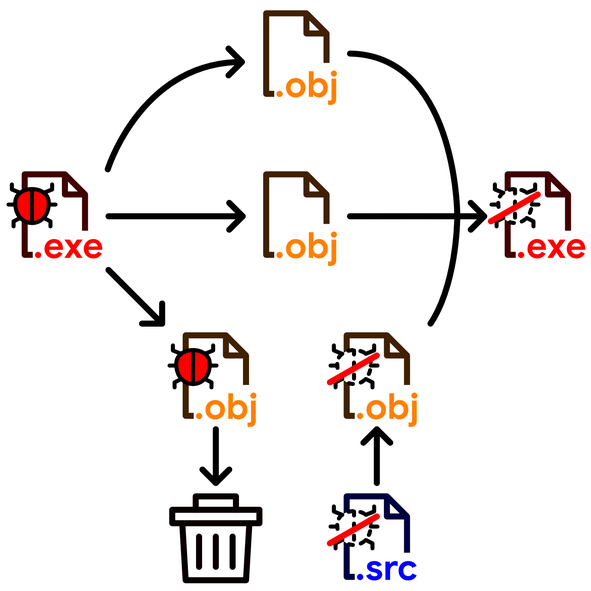
To create an executable file, the linker takes each object file and:
- Lays out their sections in memory.
- Computes the addresses of the symbols.
- Fixes up the relocations.



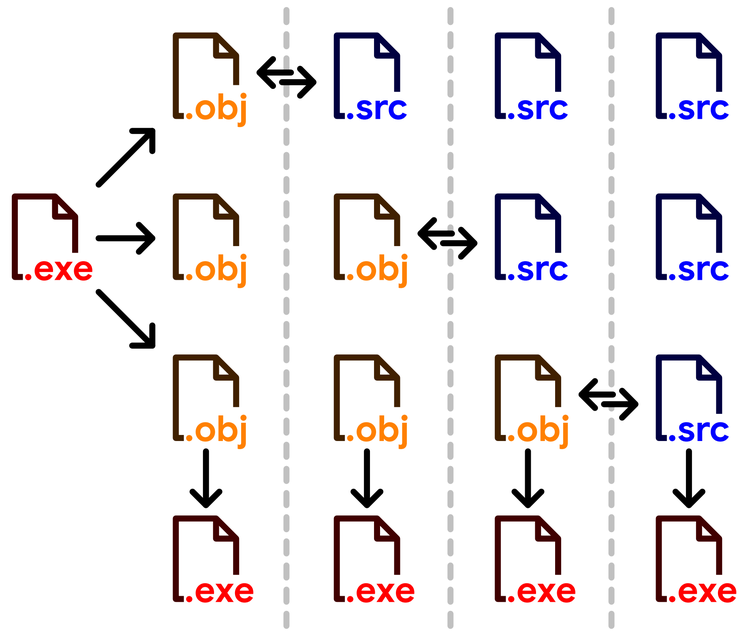
Everyday reverse-engineering work on executable files yields annotations:
- Memory blocks.
- Labels.
- References.

To create object files from these annotations:
- Analyze references to infer relocations.
- Use the labels to create symbols.
- Unapply all relocations to get relocatable sections.

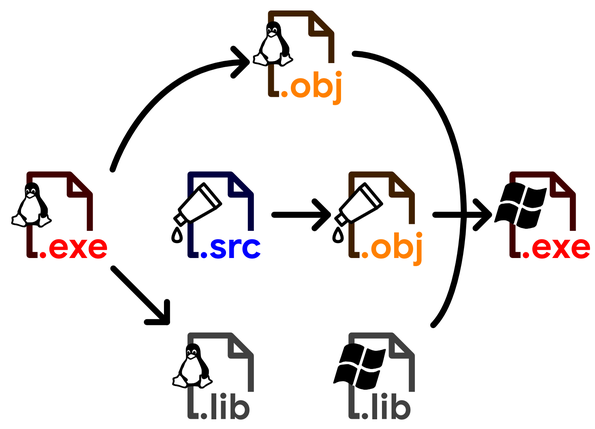
What can I do with this?



Once a program is delinked back into object files, a new executable file can be created by linking both original and new parts together.
Bugfixes and new features can be retrofitted into existing programs, even if their original source code is unavailable.



For a decompilation project, each individual object file can be reimplemented as a source code file, one piece at a time, until the program is fully decompiled.
The linker can produce a working executable file at each step, even if the parts are not matching.



Applications are not limited to just a single platform. Binary code can be delinked across file formats and relinked across ABIs, as long as the pieces can be stitched together...
(may require additional techniques like thunking, glue code and support for custom calling conventions in toolchains)

Wanna know more?

`https://github.com/boricj/ghidra-delinker-extension`

`https://boricj.net`

`jean.baptiste.boric@gmail.com`

`https://boricj.net/assets/acm-ccs-2025-poster-delinking-stripping-programs-for-parts.pdf`

This poster was presented at the 1st edition of the Workshop on Software Understanding and Reverse Engineering (SURE 2025).