

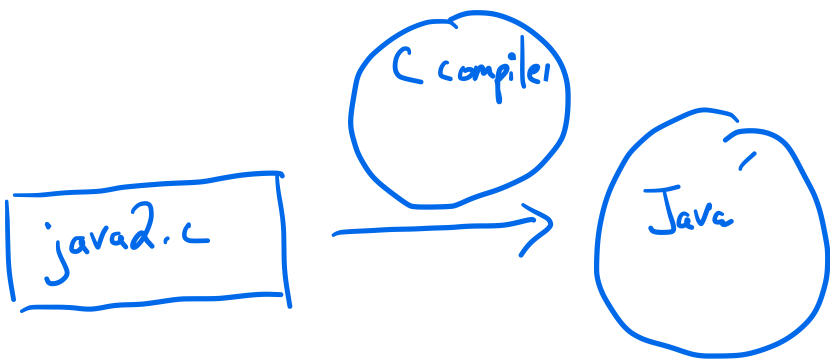
- 1. Last time
- 2. Trusting trust
 - Background
 - Adding a new feature to a language
 - Context
 - Goal: bug login [and keep it bugged]
 - Self-reproducing programs
 - Result
 - Moral, discussion
- 3. Further thoughts on trust

2. Trusting trust

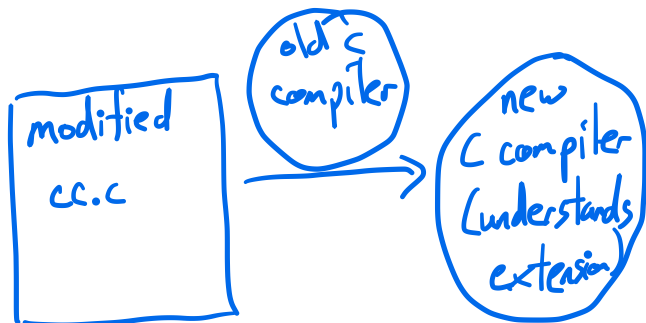
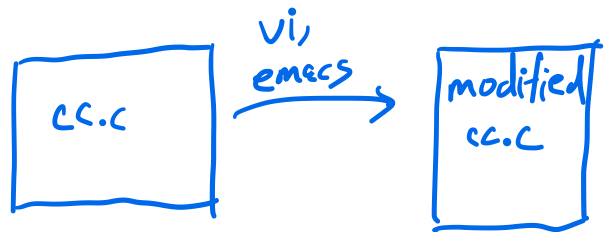
A. Background

B.1 Adding a feature to a language

Say we want the token "cs202" in a .java file to produce `System.print("hi")`. We do this:



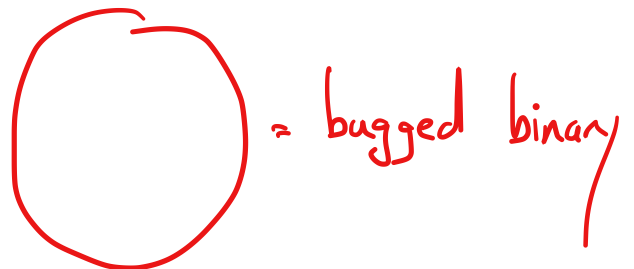
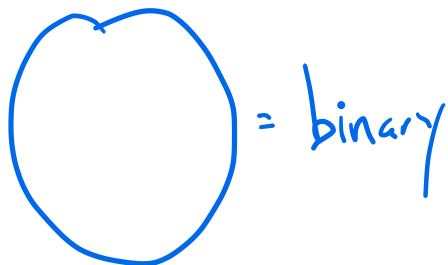
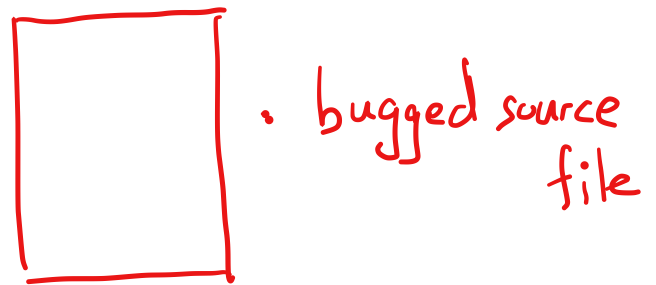
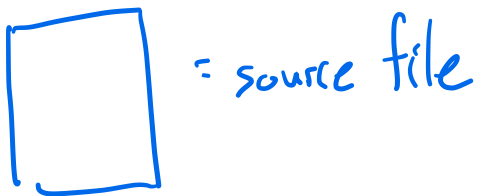
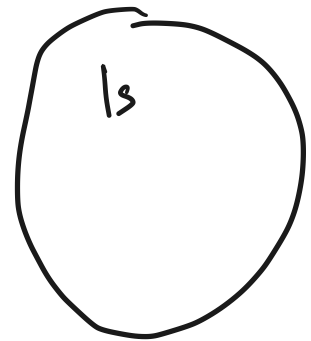
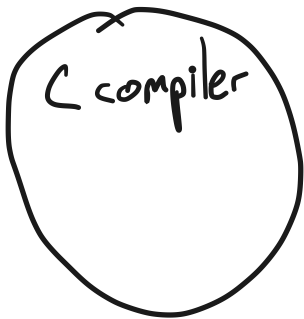
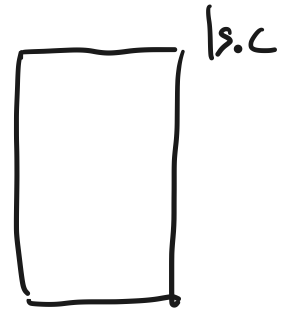
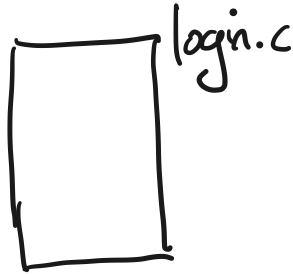
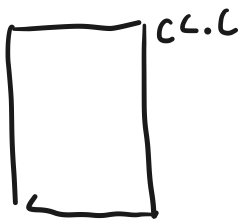
We can do the same thing to extend C:



The compiler for Java now recognizes "cs202" as a token in Java files



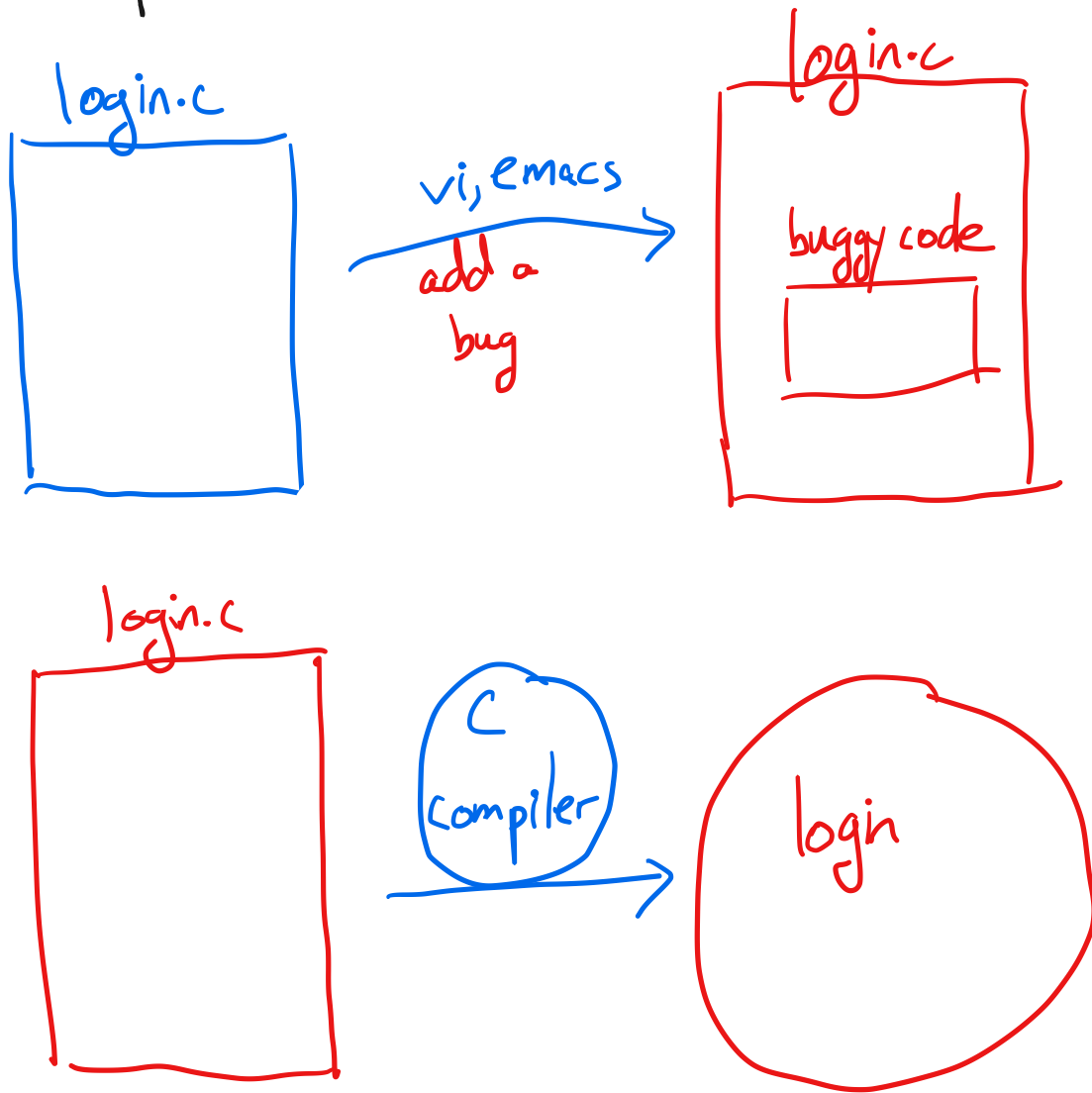
C. Context



... trace in any source files keep

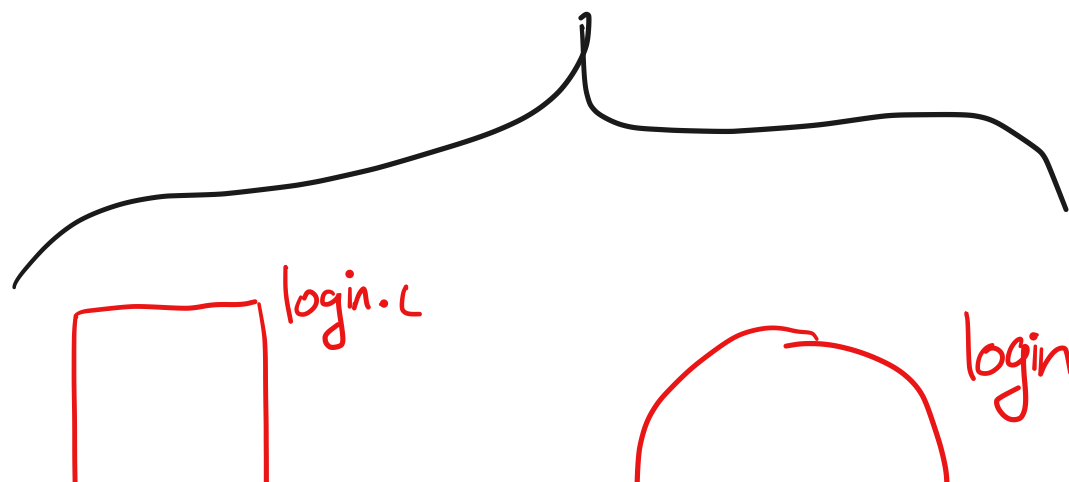
D. Goal: bug login, leave no trace in any source code, so that it bugged across recompilations.

D.1 First attempt:

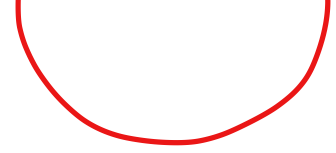
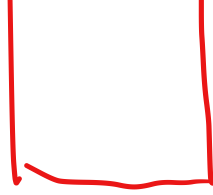


Distribute:

"Problem":
anyone looking



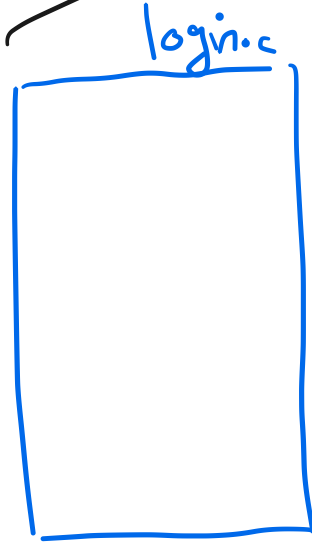
at login.c can see
that it is bugged.



D.2 Second attempt:

Distribute

"Problem": recompile
login.c and login
becomes non-bugged
again

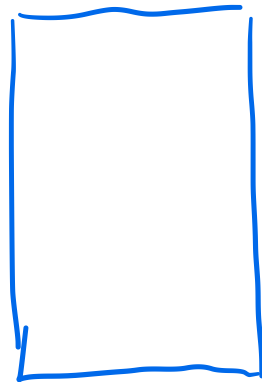


login



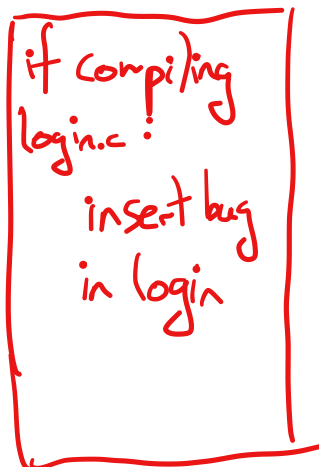
D.3. Third attempt:
bug the C compiler!

cc.c

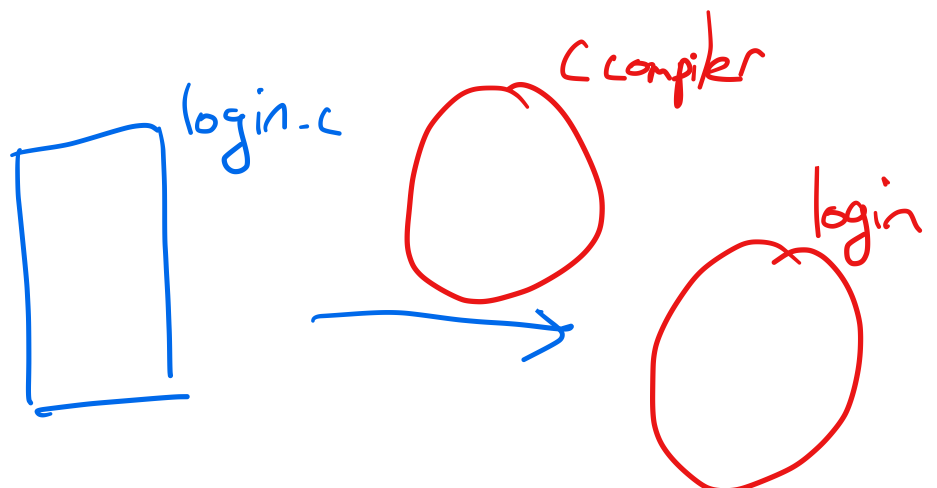
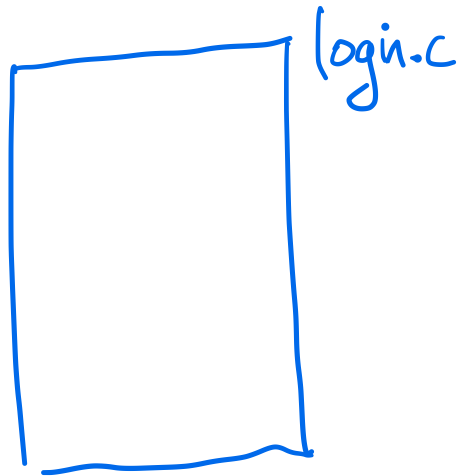
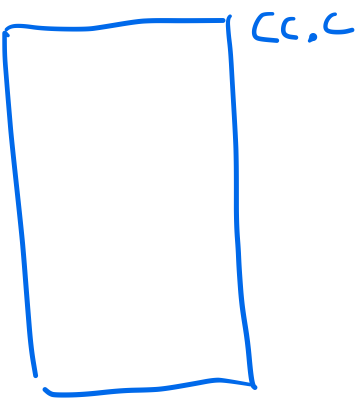
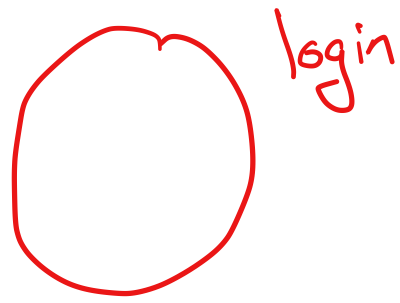
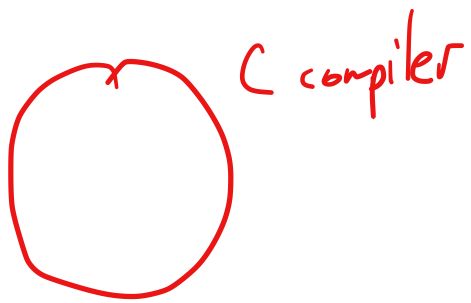
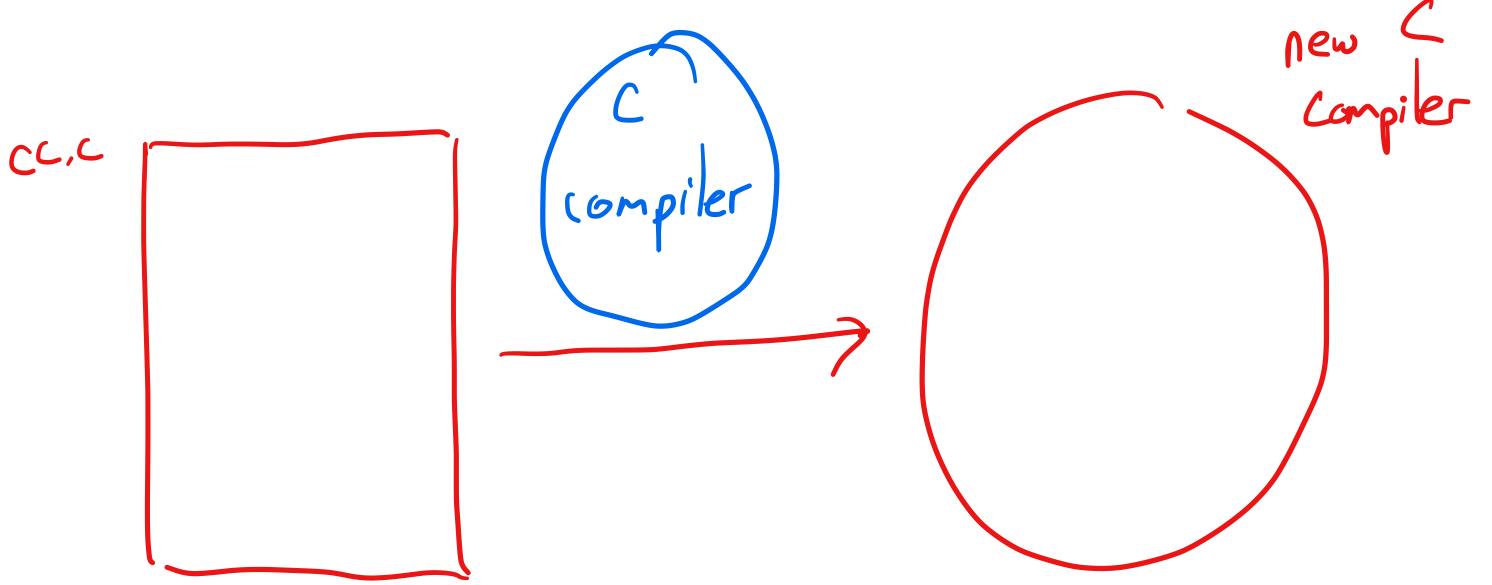


vi, emacs

add a
bug

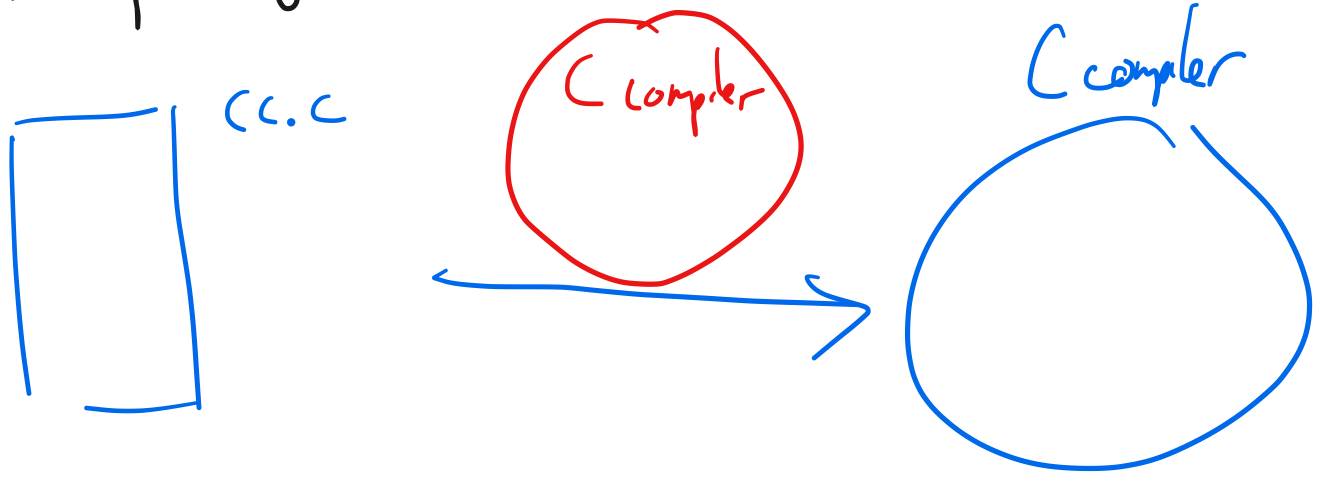


cc.c

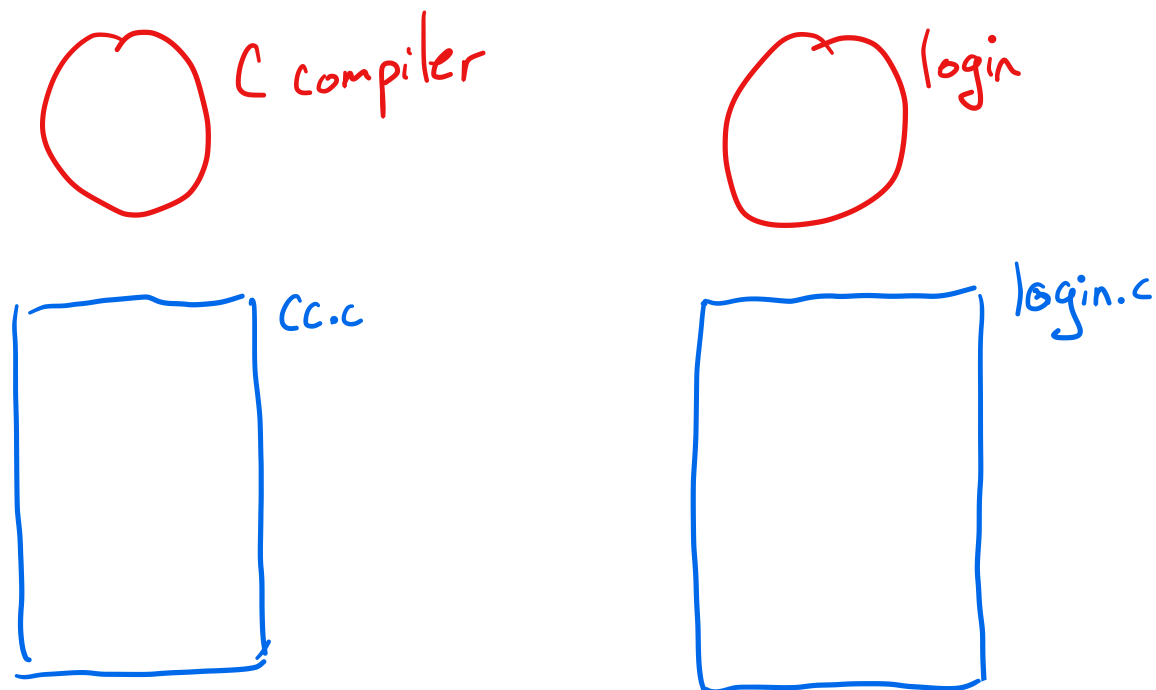


Close!! No trace in source

- Recompile login \Rightarrow bugged login



Goal: make this picture stable:



E. Self-reproducing programs

[exec stuff]

X = "Output 'X'. Output '='. Output quote mark.

Output X. Output quote mark. Output X.

Exec stuff

Output 'X'. Output '=', Output quote mark. Output X.

Output quote mark. Output X.

X = 'Output 'X'. Output '='. Output quote mark.

Output X. Output quote mark. Output X.

Output 'X'. Output '=', Output quote mark.

Output X. Output quote mark. Output X.

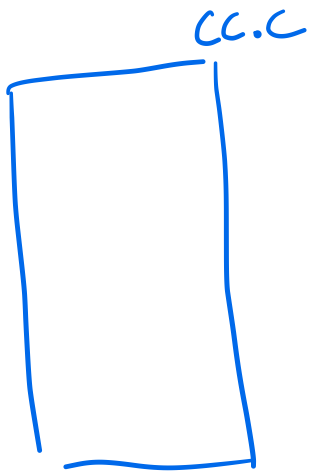
Print this followed by its quotation:

"Print this followed by its quotation."

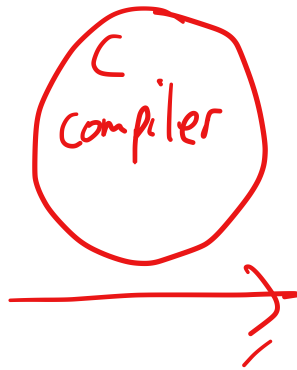
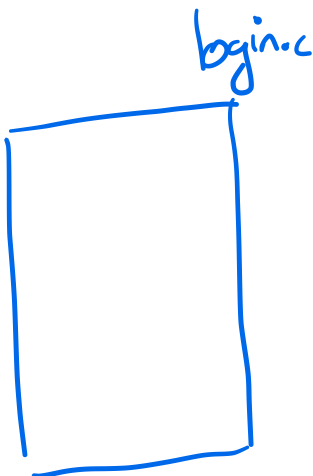
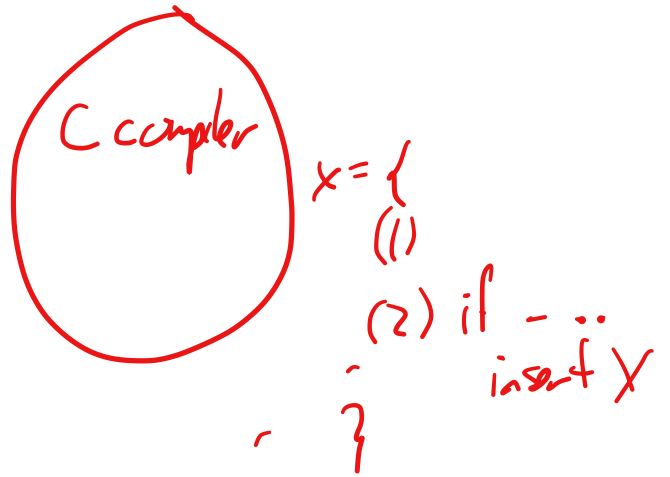
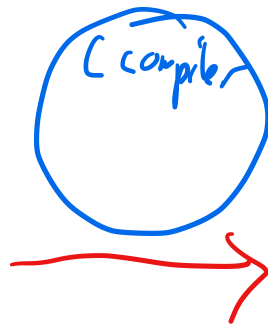
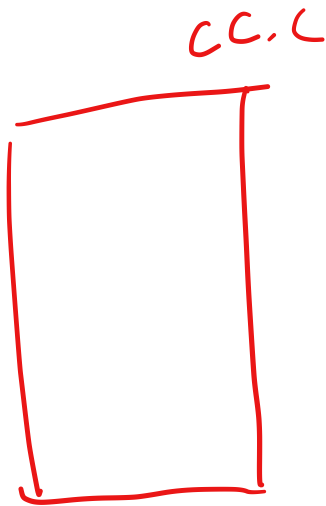
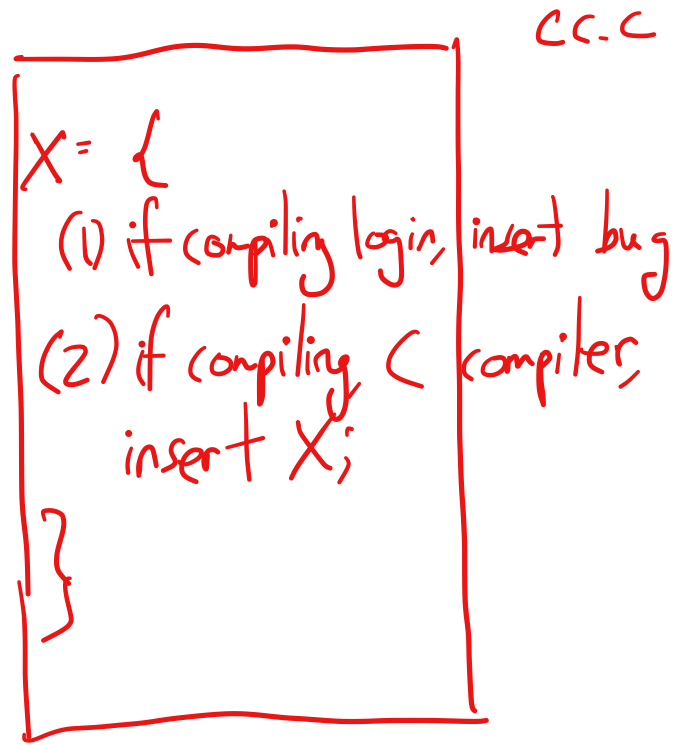
((lambda (x) `(,x 'x)))

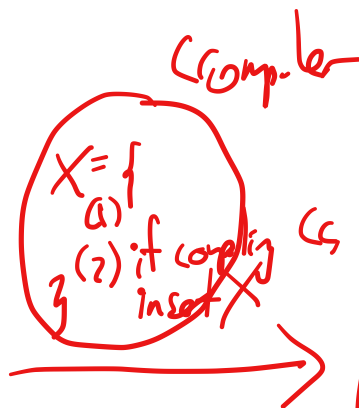
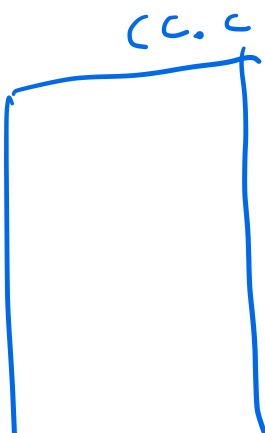
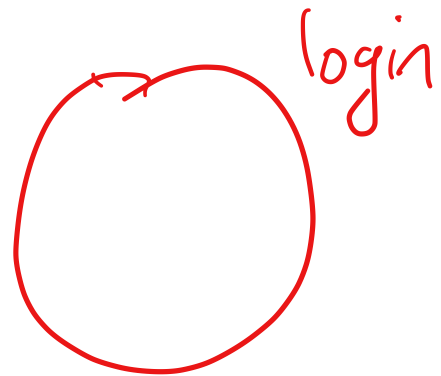
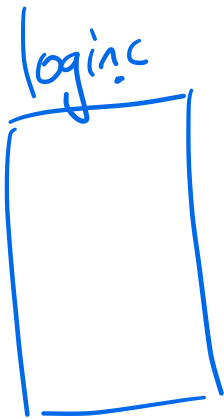
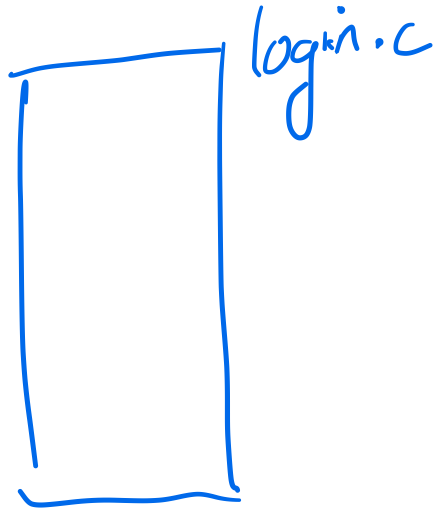
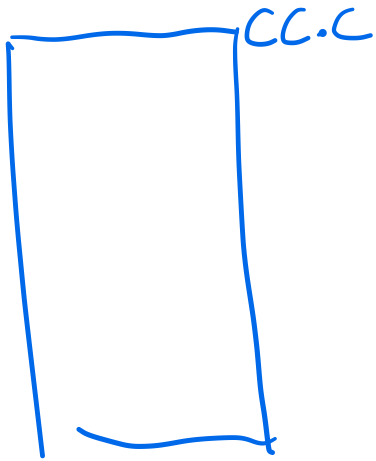
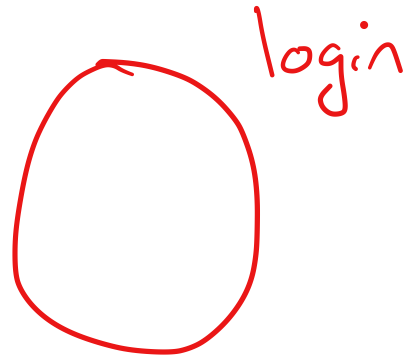
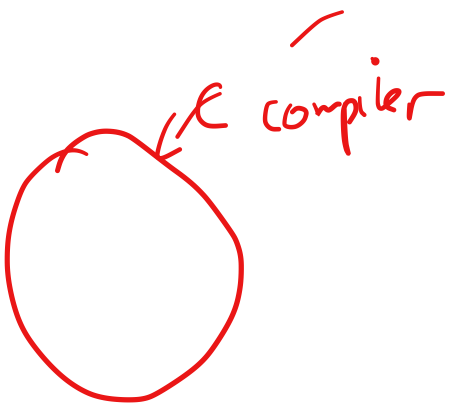
'(lambda (x) `(,x 'x)))

quine

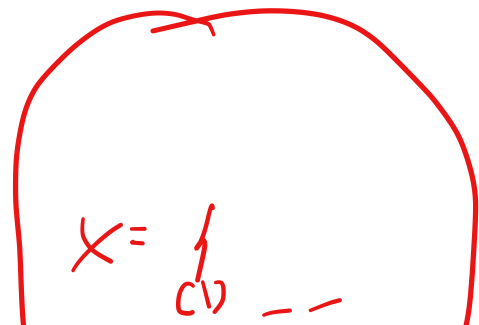


vi, emacs





C compiler



1

(2) ---



cc -S cc.c



cc.S

as cc.S