UE18CS351 – Compiler Design

Class 1

12 Jan, 2021

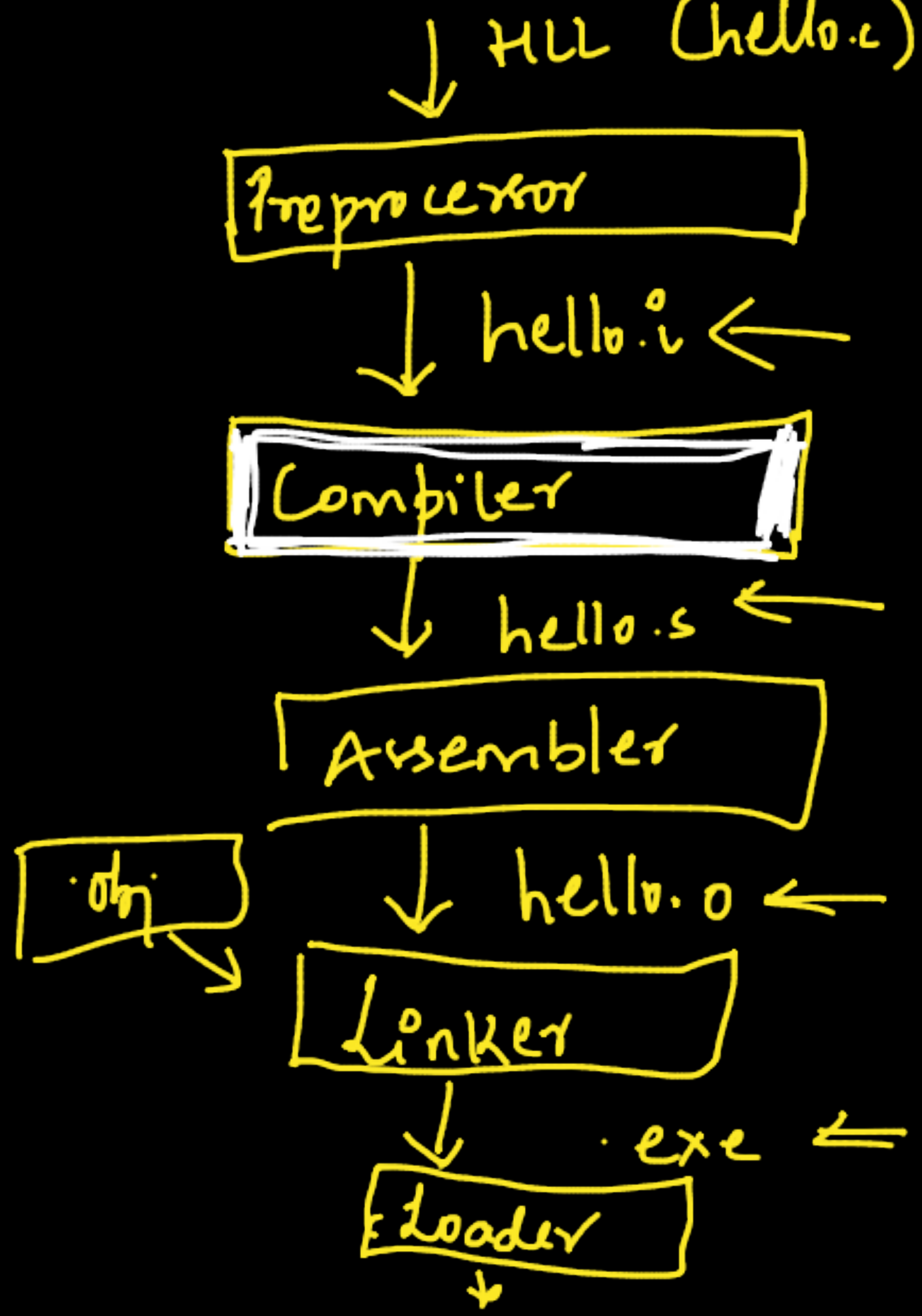
#include <stdio.h>

Lang processing s/m :-

```
1) gcc -E hello.c > hello.i
2) gcc -S hello.i            o/p : hello.s
3) as hello.s -o hello.o
4) ld -
   gcc hello.o
```

gcc

a.out



$$\begin{tikzcd}
\begin{array}{c}\text{HLL} \\ \text{(C lang} \\ \text{as reference)}\end{array} \arrow[r] & \boxed{\text{Compiler}} \arrow[r] & \boxed{\text{LLL}} \arrow[dl] \arrow[dr] & \\
& \boxed{\text{Assembly code}} & & \text{m/c level code}
\end{tikzcd}$$

Donor have to deal with H/w peculiarities

C compiler → Self-compiling / Self-hosting compilers

$\Downarrow$

Compile its own source code

Compiler

$\boxed{\text{Assemblers}}$

m/c code

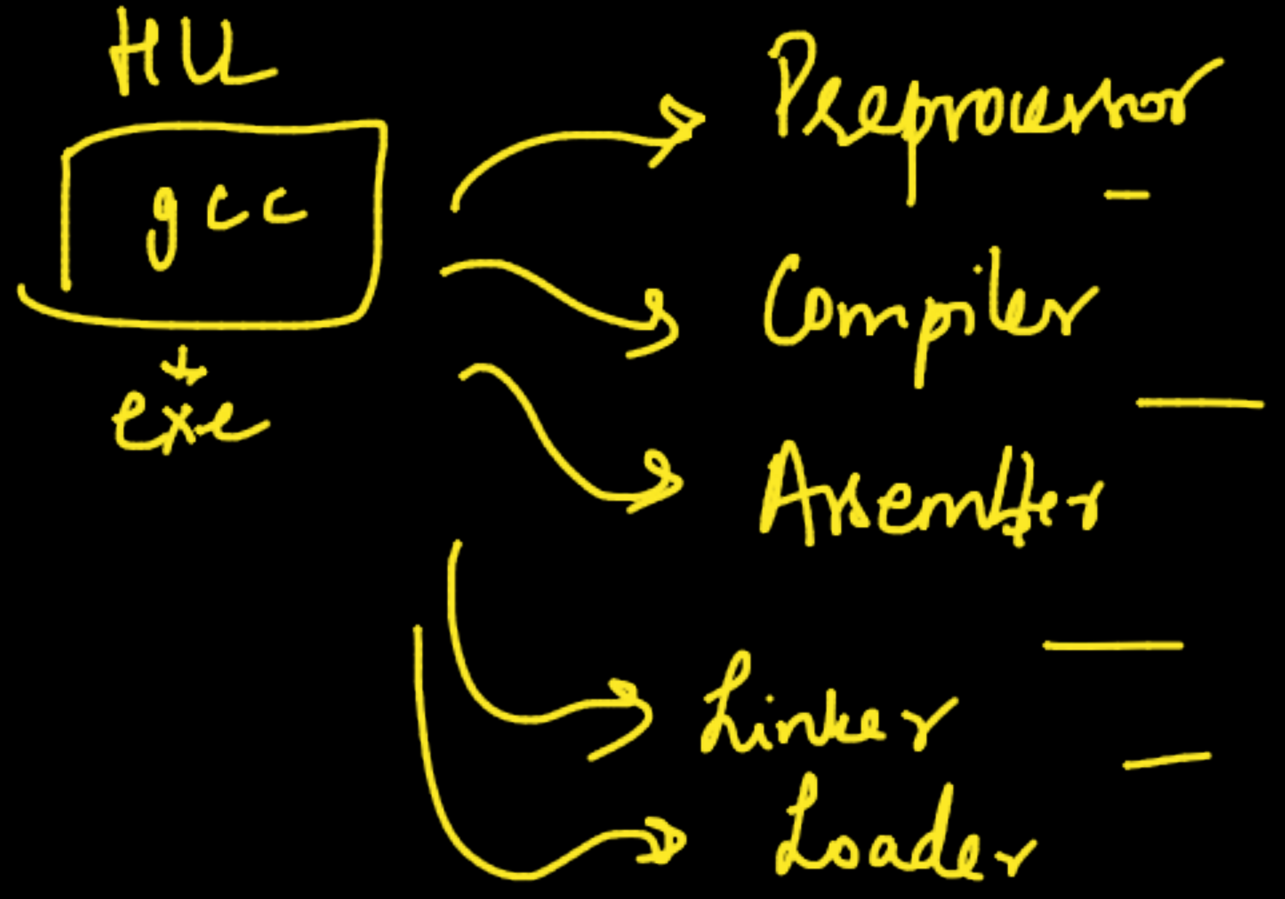
Bootstrapping

What has been discussed till now?

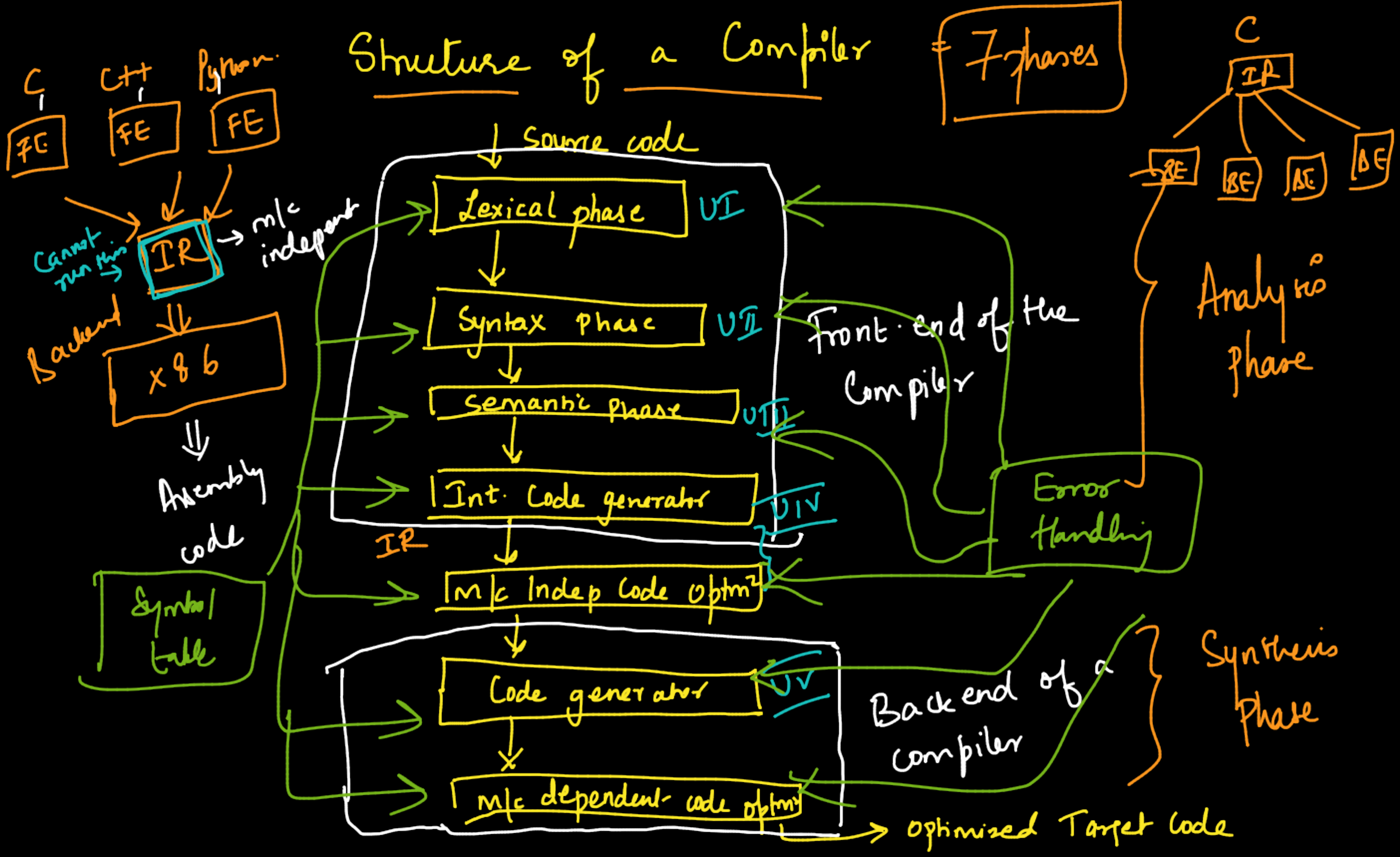
→ Compiler [O/p :- Assembly code]

→ Diff types/ terms
- Transpiler
- Cross-compiler
- Self-hosting compiler (Bootstrapping)
- Decompiler (LLL → HLL)

→ Lang processing system



Structure of a Compiler



$\boxed{\text{Program}}$

$\Uparrow$

String

Source code (hello.c)

$\downarrow$

$\boxed{\text{Lexical Analyzer}}$ or Scanner

format

look like - can be admitted?

alphabet

Is it allowed?

$\boxed{\text{Lang}}$

$\boxed{\text{Metro}}$

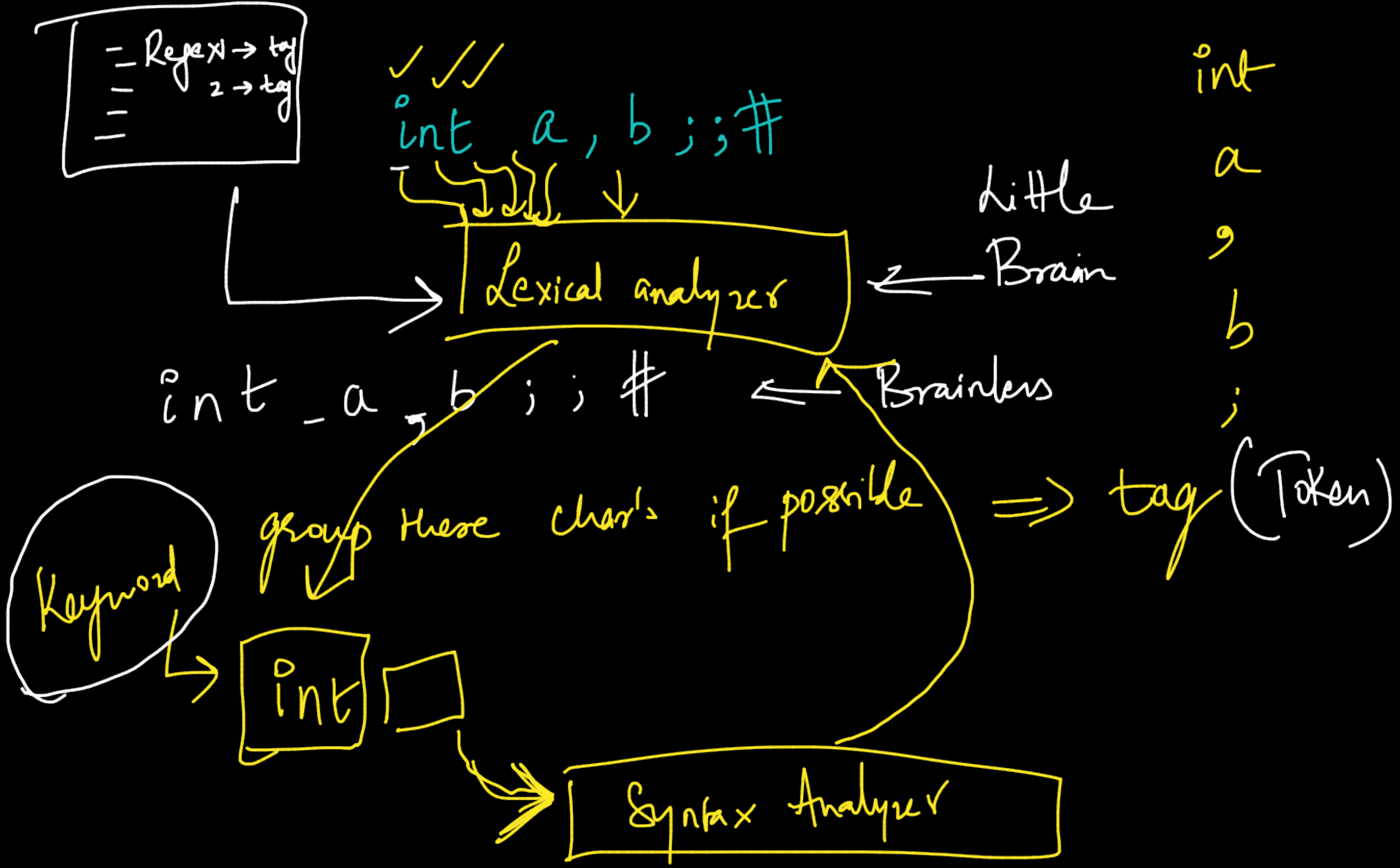
dog

$\Downarrow$

ASCII

Char set

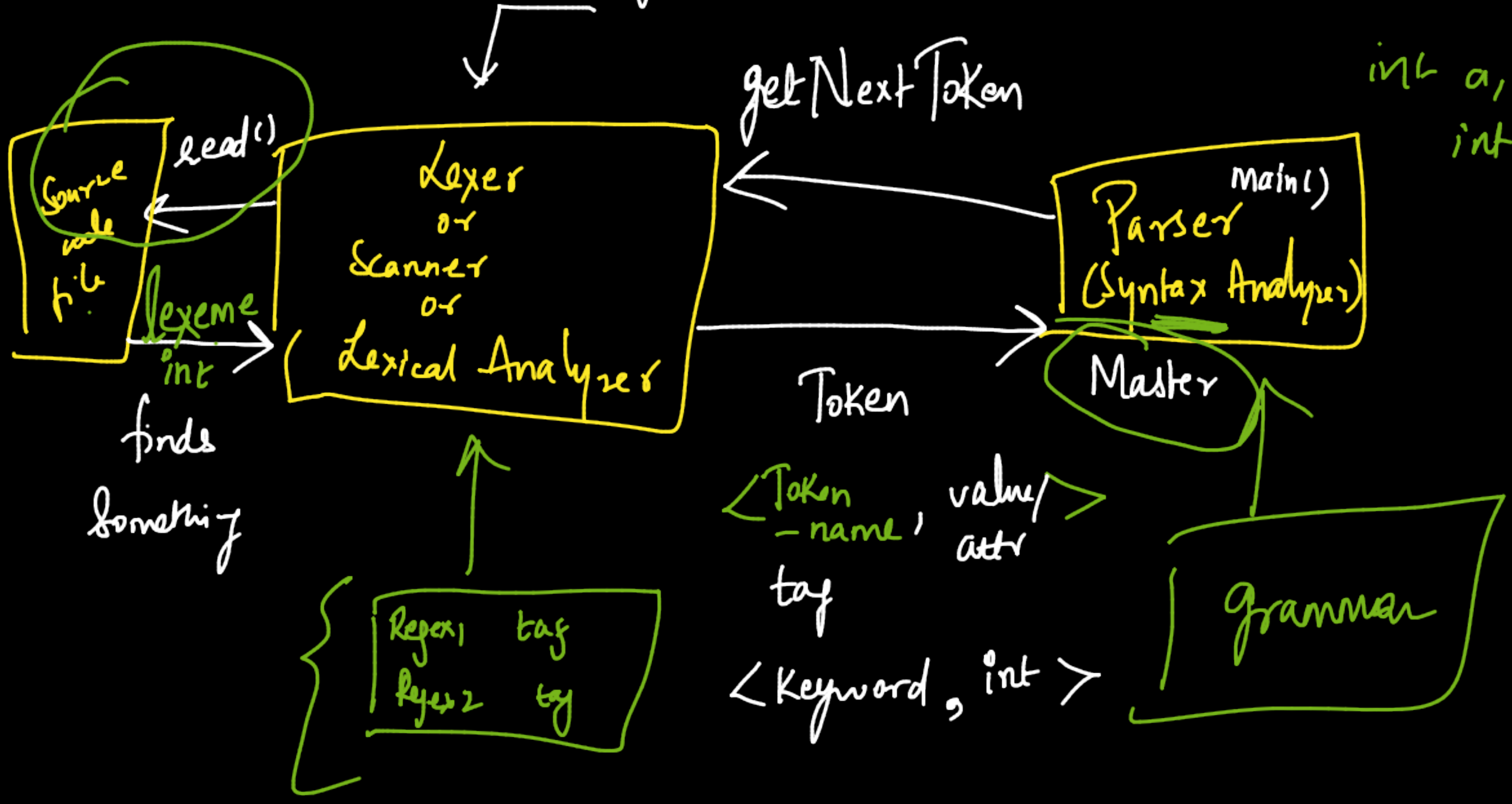
= { *, #, a-z, 0-9, A-Z, ?, $, ... }


Regex → tag
2 → tag
✓ ✓ ✓ ✓
int a , b ; ; #
Lexical analyzer
Little Brain
int _ a _ b ; ; #
Brainless
group these char's if possible ⟹ tag (Token)
int
a
,
b
;
Keyword
int
Syntax Analyzer

int $\boxed{\ }$ a, b ;; #

Lang dependent

Perl ⇒ Scannerless parser



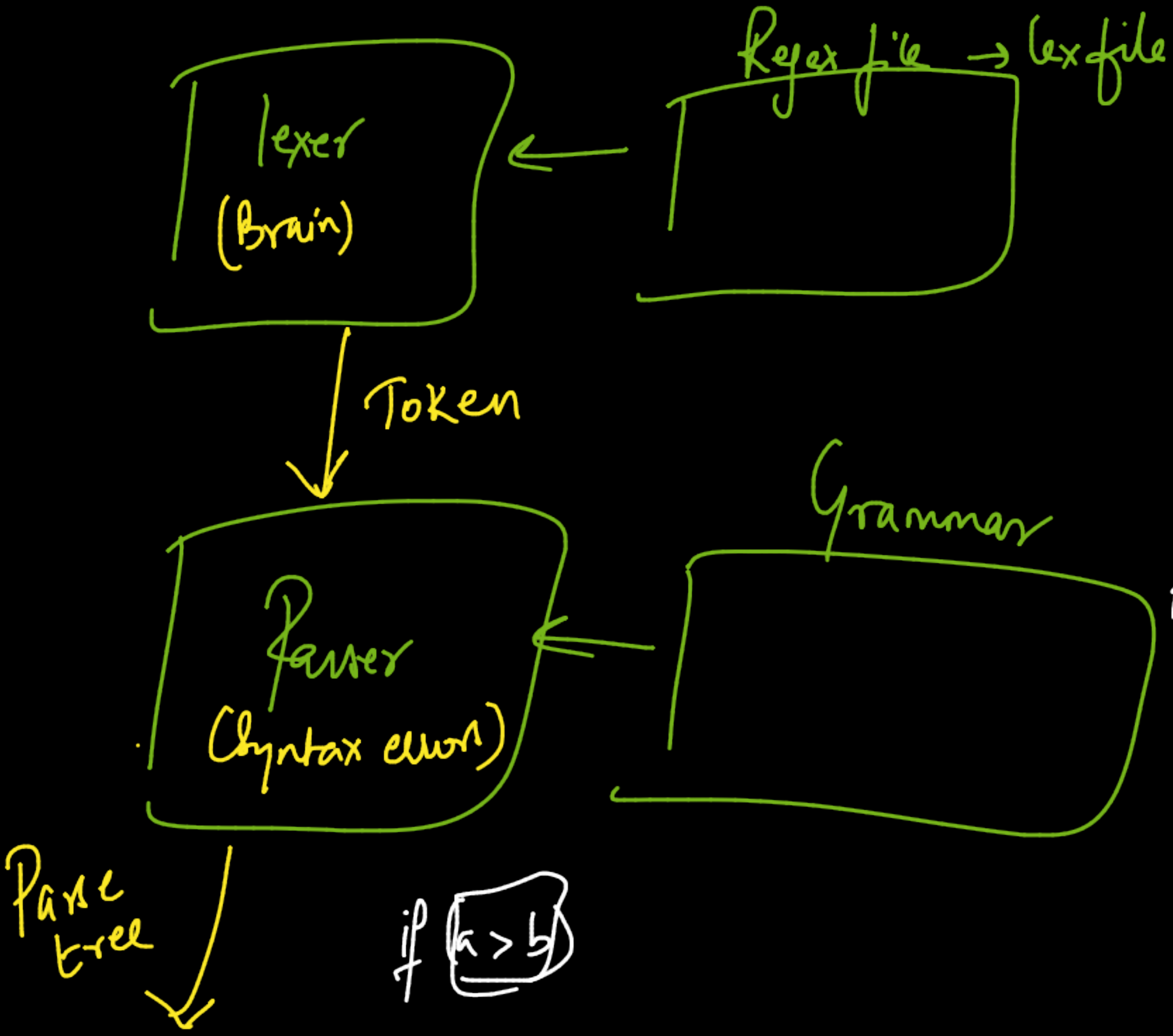
int a, b;

int a, b, c, d;

float a;

CFG for

Var declr



Write a grammar for an entire program
production rules in C lang.

→ Decl of a var
→ if, while, for,
→ Expressions
→ Cond

id =

Lex tool → command in Linux

flex

⇓ never stops ← terminati

[int | float | char] general Key

[a-z A-Z] id

[a-zA-Z0-9]*

{ Regex / Lex }

Decl of a var name in C lang

Type

int a, b;

int a;

float a, b, c;

D → Type List_Var ;

List_Var → id | id , List_var

Type → int | float | Char