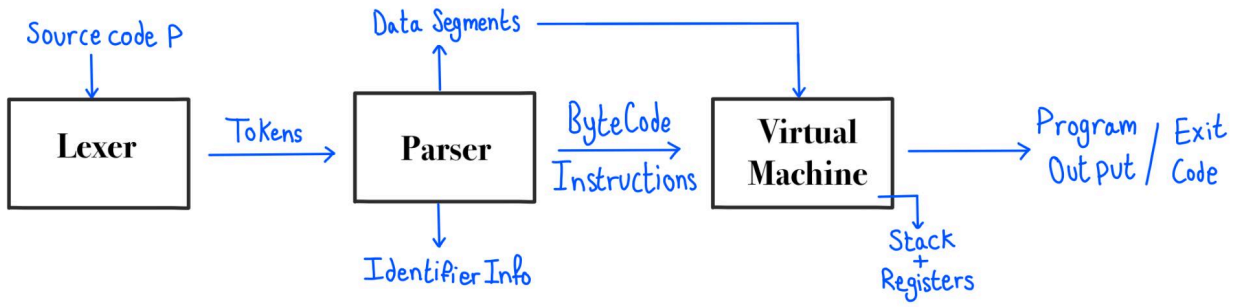
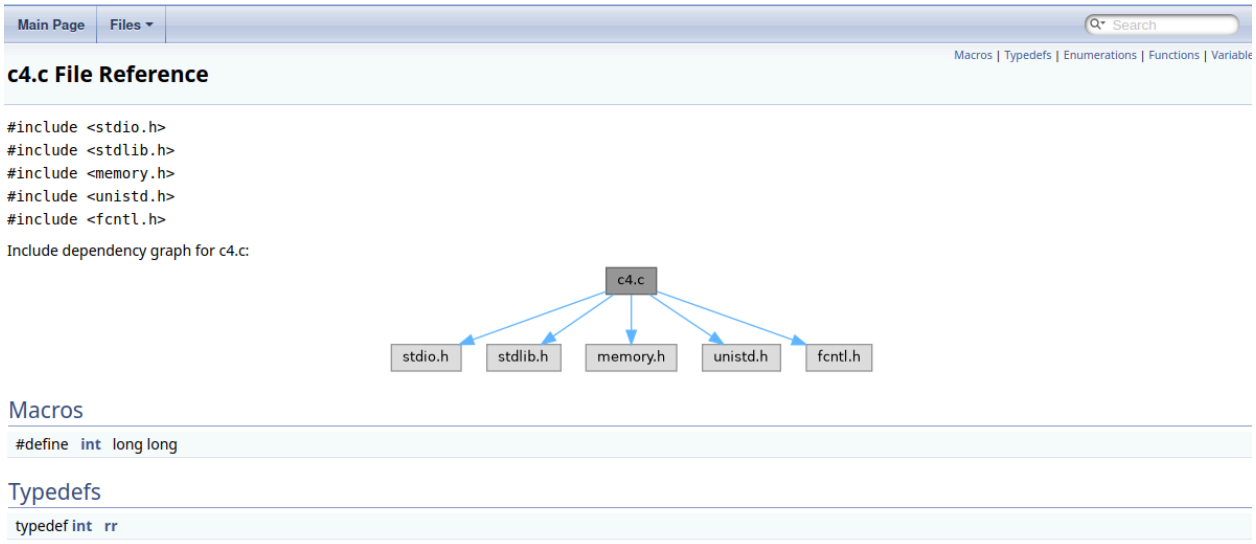


Note: This diagram illustrates the overall flow in the c4 compiler. the lexer converts the source code into tokens, which the parser uses to produce bytecode instructions and update data segments. Finally, the virtual machine executes the bytecode using its stack and registers to generate the program's output.

My C4 Compiler



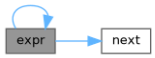
Note: This diagram shows which system headers (stdio.h, stdlib.h, memory.h, unistd.h, fcntl.h) are included by c4.c. The arrows indicate that c4.c depends on these headers for standard I/O, memory operations, and file descriptors.

Function Documentation

◆ expr()

void expr (int lev)

Here is the call graph for this function:

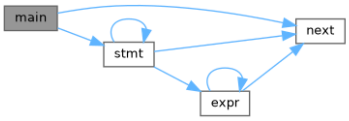


Here is the caller graph for this function:

◆ main()

```
int main ( int      argc,
           char **  argv
         )
```

Here is the call graph for this function:



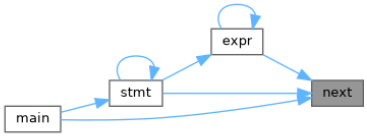
Note: The call graph visualizes how expr() calls next(), and the caller graph visualizes how main() calls stmt(), expr(). The upper box is the function itself; the lower box is a function it calls.

file:///home/joud/projects/c4/docs/html/c4_8c.html

◆ next()

void next ( )

Here is the caller graph for this function:



◆ stmt()

void stmt ( )

Here is the call graph for this function:

Here is the caller graph for this function:

Note: the top diagram shows that next() is called by both expr() and stmt(), and transitively by main() because main() calls stmt(), which calls expr(), and expr() in turn calls next(). next() is the lexer routine that reads characters from the source text and determines the next token (tk, ival, etc.). Whenever the parser needs a fresh token, it calls next(). The bottom graph indicates that stmt() is called directly by main() (to parse statements) and also calls expr() and next() internally. stmt() parses statements like if(...), while(...), or expression statements; it repeatedly calls expr()

to handle expressions and next() to move through tokens. There are arrows showing that once stmt() is invoked by main(), it may call expr() to parse expressions, and expr() in turn calls next() to fetch tokens.

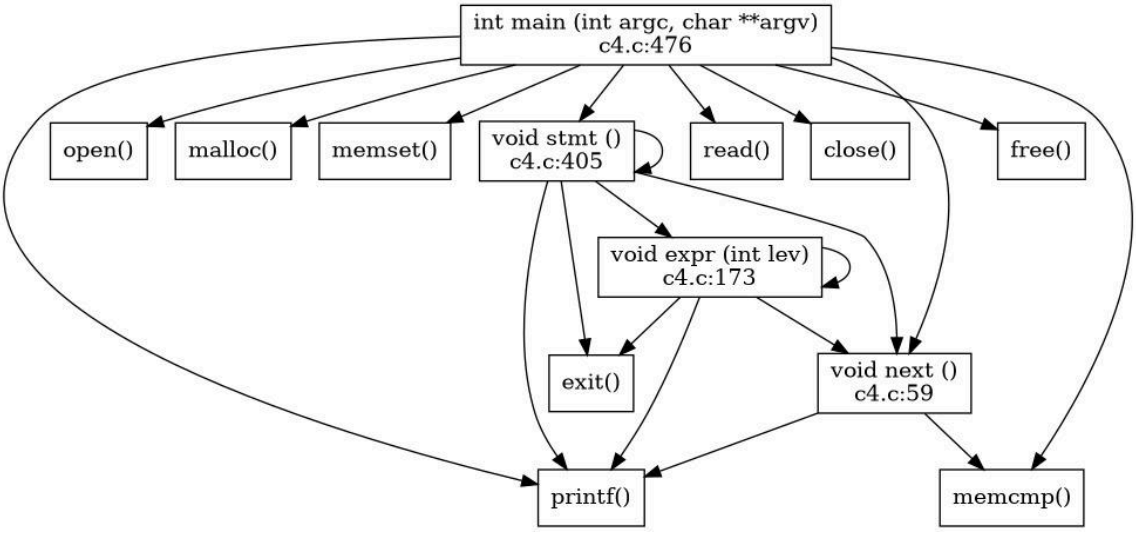
Functions

void	next ()
void	expr (int lev)
void	stmt ()
int	main (int argc, char **argv)

Variables

char *	p
char *	lp
char *	data
int *	e
int *	le
int *	id
int *	sym
int	tk
int	ival
int	ty
int	loc
int	line
int	src
int	debug

Note: This is an index of functions (next(), expr(), stmt(), main()) and global variables (p, lp, tk). It's useful for quickly seeing everything that's declared in c4.c.



Note: This big call graph lumps together all the direct calls from main() and other functions in c4.c. For instance, main() calls stmt(), which calls expr(), and so on. System calls like open(), read(), and printf() are also included since the code references them. Overall, this call graph is helpful to see which functions are "entry points" and which are invoked more deeply (for example, stmt() calls expr(), which calls next()). It's not an architecture or data flow diagram per se, but it does reveal the control flow among c4's major routines and the library calls it uses.