Holistic Revision Tree: A Better Version Control System for C Programs

Joseph Tessitore

1. Abstract

According to JetBrain's 2023 developer ecosystem survey, 87% of developers use Git to track revisions in their program [1]. While Git's model is powerful, it has a steep learning curve for new users, owing to its complexity. In this paper, we demonstrate the *Holistic Revision Tree* (HRT), an alternative version control system which takes advantage of the C preprocessor to encode multiple versions inline in a single file. We will show how HRT makes you a better programmer by simplifying version control and sharing code with others.

2. Background

While Git's distributed workflow is appealing to many developers [2], it must be acknowledged that a significant factor in Git's ongoing dominance is due to vendor-lock in. Popular software forges and integrated development environments offer built-in tooling for Git, and not for other version control systems. In addition, many developers are already familiar with Git, and are wary of learning a new, complex system. Because of this, many alternative version control systems are *forced* to offer Git compatibility. For any new version control system to challenge Git's tyrannical rule, it must take advantage of universal software features, and have a very shallow learning curve.

2.1. The C Preprocessor

The C preprocessor is a filter applied to C source code files applied before they are compiled [3]. This is most commonly used to include *header files* in a source file with #include "filename.h". More importantly, it can be used to conditionally compile code. The following example will print "TRUE" when compiled and executed, and the else branch will be excluded entirely from the compiled program:

Listing 1: Conditionally compiling code with math

This can be taken further by defining variables when compiling. For example, we can define the variable VERSION to be equal to 0 or 1 when compiling by using gcc -DVERSION=0 or gcc -DVERSION=1 respectively. This definition will cause the respective branch to be compiled, while the other is filtered out. By using this technology, we can store multiple revisions of a program in the same file.

Listing 2: Conditionally compiling code with a variable

To allow users to compile without manually specifying a version, this block can be added to the top of the file:

```
#ifndef VERSION
#define VERSION 1
#endif
```

Listing 3: A version header to specify the latest version

These lines will define VERSION to 1 (the latest version) if it is not already defined. By updating

the middle line with each update, it can be insured that users will get the newest version.

By combining these factors, the entire revision history of a C file can be stored with in a single file. While this is powerful, as will be elaborated later, we must accept that editing a program like this is very cumbersome. Thus, we must sacrifice a small amount of our method's ideological purity and actually write code instead of just thinking about it.

3. Introducing HRT

The *Holistic Revision Tree* methodology relies on two core files:

- 1. *The Tree* A file containing every revision of the program merged into one file.
- 2. The Work File A file containing a single revision of the program extracted from the tree, without any conditional compilation.

The program consists of two functionalities:

- 1. checkout Extract a specific revision from the tree file into a work file.
- 2. commit Combine a modified revision in a work file back into the tree as a new version.

These functions are named similarly to how they are in Git, to aid in understanding.

4. Methodology

The program uses an *LL parser* [4] to parse some of the relevant preprocessor directives in the C programming language. Thanks to some public domain¹ code from nullprogram.com, the string-handling code necessary was somewhat bearable² [5].

4.1. Checkout

When checking out a revision, the program evaluates any #if statement involving the VERSION

variable³ to decide whether or not to include a block of code⁴.

4.2. Commit

The commit process first creates a work file from the tree of the latest version, which is determined by the version header (Listing 3). While doing this, the program creates a map⁵ from line numbers in the work file to line numbers in the tree file.

It then calls diff to compare the provided and generated work-file, producing a list of changes written by the developer. The output of diff is parsed [6], and the changes made to the work file are spliced into the tree by mapping their indexes back.

diff can output three different types of changes: Additions, deletions, and changes. The following code is generated for each of the three:

```
#if VERSION >= 5
/* Comment added in version 5 */
#endif
```

Listing 4: Code generated for a line addition in version 5

```
#if VERSION < 5
/* Comment removed in version 5 */
#endif</pre>
```

Listing 5: Code generated for a line deletion in version 5

```
#if VERSION < 5
/* Comment changed in version 5 */
#else
/* New comment :3 */
#endif</pre>
```

Listing 6: Code generated for a line change in version 5

It is important to understand that while two lines of code might be next to each-other in a work file, they might be in different conditional-compilation blocks in the tree. While this doesn't need to be handled for addition like in Listing 4, for removal,

¹A fact that we *definitely* checked before we based our program off of it

²also aided in memory allocation

³Assuming VERSION is on the left-hand side

⁴The current version is decided either through a command-line flag or through the version header (Listing 3)

⁵array

it can be handled through recursively parsing conditional-compilation blocks with matching line ranges.

Extra care was required for changes to make sure the new code would not be included multiple times, or so that the wrong likes wouldn't be deleted. The explanation for this process is omitted because I don't fully understand how I fixed it for brevity.

Finally, the program checks the version header shown in Listing 3, and increments it to show the latest version.

5. Usage

Upon obtaining HRT (see Section 7), it can be compiled as follows: gcc -o hrt

hrt.c⁶. To check out the latest version of a program stored in tree.c and save the work file in work.c, you can use hrt checkout < tree.c > work.c.⁷ A specific version, such as version 3, can be checked out by running hrt checkout -v 3 < tree.c > work.c.

After editing your file in your favorite text editor, you can commit your changes by running hrt commit tree.c < work.c > new_tree.c, with new_tree.c intuitively containing the new version of the tree.⁸.

However, hrt isn't needed to simply compile any version of a program. For example, to compile version 3, one can run gcc -DVERSION=3 tree.c. This provides the portability and simplicity that we all so desperately seek.

6. Results

One clear benefit of taking HRT is that it eschews the need for traditional Git forges such as GitHub, since our source tree can be stored wherever a file can be. This could be provided as a link on a website, stored in a Google Drive, or even provided in its entirety at the end of a paper. The latter is useful because it grants readers a deeper insight into the progression of a paper's core, rather than just its final state. It allows them to see how a program changed over time, including deleted functions, confused comments, and code that was never relevant at any point whatsoever. Changes are also simple to share by either sending a work file, or storing the tree on a shared drive that can be changed by all contributers.

6.1. File Size

HRT also has significant size benefits. Table 1 shows a comparison between how much size the source repository takes as a git bundle [7], a tar archive containing each file, and an HRT tree. It also includes the work files of the final revision in bold.

Method	Bytes
tar archive of revisions	450506
HRT tree	44389
Git bundle	25703
Final work file	24256
tar.gz archive of revisions	17517
HRT tree (gzipped)	8645
Final work file (gzipped)	5791

Table 1: Comparison of file sizes of hrt source code

As shown, the HRT tree is the smallest way that every version can be stored with compression. It's compression ratio can be calculated through size of all revisions, yielding 10.15x for an uncompressed tree, and 52.11x for a compressed tree. Other compression systems like Facebook's ZSTD,

⁶Or whatever C compiler you prefer

⁷Standard input and output are used throughout this program because it makes it feel more UNIX-y

^{*}Because of how shells work, you can't do hrt commit tree.c < work.c > tree.c. You can do hrt commit tree.c < work.c > new_tree.c && mv new_tree.c tree.c to accomplish the same thing

⁹hrt commit is non-atomic. Careful!

meanwhile, can only offer 5.5x on source-code [8]. Further research is needed to investigate HRT's clear potential as a general purpose compression algorithm.

7. Obtaining HRT

HRT can be obtained either by downloading the source tree from https://jonot.me/hrt.c, or by copying the source tree provided at the end of the paper.¹⁰

¹⁰Programmers in the UK must go through a waiting list to make sure that they *really* don't want to use Git instead, and that they're not just pretending.

```
8. Source Listing
                                                                                                                                                                                                                                                                                               Line parse_line(Str line) {
  Line ret = {0};
  ret.span = line;
  if ((line.len || line.data[0] != '#') {
    return ret;
}
 #ifndef VERSION
#define VERSION 23
                                                                                                                                                 typedef struct {
                                                                                                                                                 Str head;
Str tail;
_Bool ok;
} Cut;
  #endif
 #endif
#include <assert.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdib.h>
                                                                                                                                                                                                                                                                                                #if VERSION < 1
                                                                                                                                                 Cut cut(Str s, char c) {
                                                                                                                                                                                                                                                                                                    Cut c = cut(line, '');
                                                                                                                                                     Cut r = {0};
if (!s.len) {
  #include <string.h>
                                                                                                                                                                                                                                                                                                    Cut c = cut(trimleft(line), ' ');
  #if VERSION >= 4
 #II VERSION >= 4
#include <unistd.h>
#include <sys/wait.h>
#if VERSION < 9
#include <organia</pre>
                                                                                                                                                                                                                                                                                                    endif
const Str IFDEF = S("#ifdef");
if (equals(c.head, IFDEF)) {
    ret.operator = OP_DEF;
    ret.variable = c.tail;
    ret.line_type = LINE_TYPE_IF;
    return ret;
                                                                                                                                                     }
char *begin = s.data;
char *end = s.data + s.len;
char *cut = begin;
for (; cut < end && *cut != c; cut++) {</pre>
 #include <errno.h>
#endif
 #endif
                                                                                                                                                     }
r.ok = cut < end;
r.head = span(begin, cut);
r.tail = span(cut + r.ok, end);
return r;</pre>
                                                                                                                                                                                                                                                                                                    }
const Str IFNDEF = S("#ifndef");
if (equals(c.head, IFNDEF)) {
    ret.operator = OP_NDEF;
    ret.variable = c.tail;
    ret.line_type = LINE_TYPE_IF;
}
 #if VERSION >= 11
 #if VERSION >= 11

#if VERSION < 13

static int version = 1;

#else

static int version = 2;
                                                                                                                                                 #if VERSION >= 1
 #define max(x, y) (((x) > (y)) ? (x) : (y)) #define min(x, y) (((x) > (y)) ? (y) : (x)) #endif
                                                                                                                                                                                                                                                                                                          return ret;
                                                                                                                                                  #if VERSION < 6
                                                                                                                                                                                                                                                                                                    }
const Str IF = S("#if");
if (equals(c.head, IF)) {
   ret.line_type = LINE_TYPE_IF;
   parse_expression(c.tail, &ret);
   return ret;
                                                                                                                                                  void printstr(Str s) {
  for (int i = 0; i < s.len; i++) {
    putchar(s.data[i]);</pre>
 #endif
/* Some code from nullprogram.com */
#define new(a, n, t) (t*)alloc(a, n, sizeof(t), _Alignof(t))
                                                                                                                                                  #else
                                                                                                                                                 void fprintstr(FILE *f, Str s) {
  fwrite(s.data, sizeof(char), s.len, f);
                                                                                                                                                                                                                                                                                                      const Str ELIF = S("#elif");
 typedef struct {
  char *begin;
  char *end;
} Arena;
                                                                                                                                                                                                                                                                                                    if (equals(c.head, ELIF)) {
  ret.line_type = LINE_TYPE_ELIF;
  parse_expression(c.tail, &ret);
  return ret;
                                                                                                                                                  #endif
                                                                                                                                                 #endif
Str read_file(Arena *a, FILE *f) {
    Str ret = {.data = a->begin, .len = a->end - a->begin};
    ret.len = fread(ret.data, l, ret.len, f);
    a->begin += ret.len;
    return ret;
 #if VERSION >= 4
#if VERSION < 5</pre>
                                                                                                                                                                                                                                                                                                     const Str ELSE = S("#else");
                                                                                                                                                                                                                                                                                                    if (equals(c.head, ELSE)) {
  ret.line_type = LINE_TYPE_ELSE;
  return ret;
 """
"" Returns 3. Behavior if x == 4 is undefined */
int foo(int x) {
    if (x == 4) {
        return 5;
    }
}
                                                                                                                                                                                                                                                                                                    const Str ENDIF = S("#endif");
if (equals(c.head, ENDIF)) {
  ret.line_type = LINE_TYPE_ENDIF;
                                                                                                                                                 typedef enum {
    OP_LT = 0,
    OP_GT = 1,
    OP_LE = 2,
    OP_EG = 3,
    OP_EQ = 4,
    OP_NDEF = 6,
    OP_NDEF = 7,
    OP_CTATOR:
     }
return 3;
 }
#endif
                                                                                                                                                                                                                                                                                                         return ret:
                                                                                                                                                                                                                                                                                                      return ret:
 #if VERSION < 5
#endif
void *alloc(Arena *a, ptrdiff_t count, ptrdiff_t size,
ptrdiff_t align) {
    ptrdiff_t pad = -(uintptr_t)a->begin & (align - 1);
    assert(count < (a->end - a->begin - pad) / size);
    void *r = a->begin + pad;
    a->begin += pad + count * size;
    return memset(r, 0, count * size);
}
                                                                                                                                                                                                                                                                                                #if VERSION >= 1
typedef struct IfBlockParse IfBlockParse;
                                                                                                                                                  } Operator;
                                                                                                                                                                                                                                                                                                 #if VERSION < 4
                                                                                                                                                 typedef enum {
  LINE_TYPE_PLAIN = 0,
  LINE_TYPE_IF = 1,
  LINE_TYPE_ELIF = 2,
  LINE_TYPE_ELSE = 3,
  LINE_TYPE_ENDIF = 4,
} LineType;
                                                                                                                                                                                                                                                                                                void print_line(const Line *line) {
  puthar('"');
  printstr(line->span);
  putchar('"');
 #define S(s)
  (Str) { s, sizeof(s) - 1 }
                                                                                                                                                                                                                                                                                                 typedef struct PlainBlockParse {
                                                                                                                                                                                                                                                                                                Line line;

struct PlainBlockParse *next;

PlainBlockParse;
                                                                                                                                                  typedef struct {
 typedef struct {
  char *data;
  ptrdiff_t len;
} Str;
                                                                                                                                                     Str variable;
Operator operator;
LineType line_type;
                                                                                                                                                                                                                                                                                                 #if VERSION < 4
                                                                                                                                                     int value;
Str span;
                                                                                                                                                                                                                                                                                                void print_plain_block(const PlainBlockParse *b) {
  if (b == NULL) {
 Str span(char *begin, char *end) {
                                                                                                                                                        VERSION >= 5
     str r = {0};
r.data = begin;
r.len = begin ? end - begin : 0;
return r;
                                                                                                                                                 #if VERSION >= 5
#if VERSION < 9
int lineno;
#else
int no;
#endif</pre>
                                                                                                                                                                                                                                                                                                    if (b == NULL) {
  printf("in");
} else {
  putchar('(');
  print_line(&b->line);
  print_line(&b->line);
  print_f(" . ");
  print_plain_block(b->next);
  putchar(')');
 _Bool equals(Str a, Str b) {
  if (a.len != b.len) {
                                                                                                                                                  #endif
                                                                                                                                                 } Line;
          return 0;
                                                                                                                                                 /* Parse a base 10 integer. Very basic routine, does not
handle things
    like negatives */
int parse_integer(Str s) {
    int ret = 0;
    for (int i = 0; i < s.len; i++) {
        ret *= 10;
    }
}</pre>
    }
if (a.len == 0) {
    return 1;
                                                                                                                                                                                                                                                                                                 #endif
typedef struct {
    PlainBlockParse *plain_block;
      return !memcmp(a.data, b.data, a.len);
                                                                                                                                                                                                                                                                                                 IfBlockParse* if_block;
} StatementBlockParse:
 Str trimleft(Str s) {
    for (; s.len 66 *s.data <= ' '; s.data++, s.len--) {
    }
}</pre>
                                                                                                                                                          ret += s.data[i] - '0';
                                                                                                                                                     }
return ret;
                                                                                                                                                                                                                                                                                                #if VERSION < 4
void print_if_block(const IfBlockParse *);</pre>
      return s;
                                                                                                                                                                                                                                                                                                 void print_statement_block(const StatementBlockParse *b) {
                                                                                                                                                                                                                                                                                                   roid print_statement_block(const Statem
if (b = NULL) {
    printf("nil");
} else {
    putchar('(');
    print_plain_block(b->plain_block);
    print_f(".");
    print_if_block(b->if_block);
    putchar(')');
                                                                                                                                                 Operator parse operator(Str op) {
                                                                                                                                                     const Str strings[] = {
    S("<"), S(">="), S("=="), S("!="), S("!="),
 Str trimright(Str s) {
       for (; s.len && s.data[s.len - 1] <= ' '; s.len--) {
                                                                                                                                                         ,
or (size_t i = 0; i < sizeof(strings) / sizeof(Str); i+
      return s;
                                                                                                                                                 +) {
    if (equals(op, strings[i])) {
 Str substring(Str s, ptrdiff_t i) {
                                                                                                                                                         }
      if (i) {
   s.data += i;
                                                                                                                                                     }
/* DEADLINE I DONT HAVE TIME */
                                                                                                                                                                                                                                                                                                }
         s.len -= i;
                                                                                                                                                                                                                                                                                                #endif
typedef struct ElifBlockParse {
   Line line;
   StatementBlockParse *block;
   struct ElifBlockParse *next;
} ElifBlockParse;
     return s;
                                                                                                                                                  void parse_expression(Str expr, Line *out) {
                                                                                                                                                     Old parse_expression(Sti
Cut c = cut(expr, ' ');
Str variable = c.head;
c = cut(c.tail, ' ');
Str operator = c.head;
Str value = c.tail;
  _Bool starts_with(Str s, Str prefix) {
  if (!prefix.len) {
                                                                                                                                                                                                                                                                                                 #if VERSION < 4
                                                                                                                                                                                                                                                                                                #IT VERSION < 4
void print_elif_block(const ElIfBlockParse *b) {
   if (b == NULL) {
      printf("nil");
}</pre>
       if (s.len < prefix.len) {
                                                                                                                                                     out->variable = variable;
out->operator = parse_operator(operator);
out->value = parse_integer(value);
                                                                                                                                                                                                                                                                                                    print( "....,
} else {
  printf("(elif ");
  print_line(&b->line);
       s.len = prefix.len;
      return equals(s, prefix);
```

}

```
putchar(' ');
print_statement_block(b->block);
putchar(' ');
print_elif_block(b->next);
                                                                                                                                                                                                                        case OP_EQ:
    return version == line->value;
case OP_NEQ:
                                                                                                              ret->if_block = parse_if_block(a, p);
                                                                                                                                                                                                                        return version != line->value;
default:

/* These shouldn't be handled here at all */

return 0;
                                                                                                           ElIfBlockParse *parse_elif_block(Arena *, ParserState *);
ElseBlockParse *parse_else_block(Arena *, ParserState *);
                                                                                                            IfBlockParse *parse_if_block(Arena *a, ParserState *p) {
#endif
 typedef struct ElseBlockParse {
                                                                                                              Line next:
                                                                                                              if (peek(p, &next) && next.line_type == LINE_TYPE_IF) {
   IfBlockParse *ret = new(a, 1, IfBlockParse);
 #if VFRSTON >= 9
                                                                                                                                                                                                                      #if VFRSTON < 3
                                                                                                                                                                                                                      void dump_statement_block(StatementBlockParse *b);
   Line line;
                                                                                                                 vold dump_statement_block(StatementBlockParse *b);
#else
#if VERSION < 6
void dump_statement_block(StatementBlockParse *, _Bool);</pre>
    StatementBlockParse *block;
#if VERSION < 4
                                                                                                                                                                                                                      #if VERSION < 9
                                                                                                                                                                                                                       void dump_statement_block(FILE *, StatementBlockParse *,
Bool);
void print_else_block(const ElseBlockParse *b) {
  if (b == NULL) {
                                                                                                          i (peek(p, &next) && ni
feed(p);
#if VERSION >= 2
    ret->end_line = next;
#endif
                                                                                                                                                                                                                      void
        printf("nil");
   printf("nil");
} else {
printf("(else ");
print_statement_block(b->block);
putchar(')');
                                                                                                                                                                                                                     #if VERSION < 15
                                                                                                                                                                                                                      **I thouses are line numbers in the reduced tree file. Indexes are values in the full tree */
                                                                                                                    ret->next = parse_statement_block(a, p);
                                                                                                                 return ret;
} else {
  /* This is a failed parse actually. TODO maybe do an
                                                                                                                                                                                                                     #else
/* Indexes are line numbers in the reduced tree file. Values are line numbers in the full tree */
                                                                                                           error here? */
                                                                                                                                                                                                                      static int *line_map;
static int current_line;
                                                                                                                     return NULL;
#enolT
typedef struct IfBlockParse {
   Line line;
#if VERSION >= 2
                                                                                                                                                                                                                      #if VERSION >= 15
                                                                                                                 return NULL;
                                                                                                                                                                                                                     #if VERSION >= 15
int line_map_reverse(int full_line_number) {
   int "ptr = line_map + 1;
   while (1) {
      if "ptr == full_line_number) {
      return ptr - line_map;
   }
}
                                                                                                              }
   Line end line;
#endif
StatementBlockParse *statement_block;
ElIfBlockParse *elif_block;
ElseBlockParse *else_block;
StatementBlockParse *next;
} IfBlockParse;
                                                                                                           ElifBlockParse *parse_elif_block(Arena *a, ParserState *p) {
   Line next;
                                                                                                                 ine next;
f (peek(p, &next) && next.line_type == LINE_TYPE_ELIF) {
   ElifBlockParse *ret = new (a, 1, ElifBlockParse);
                                                                                                                                                                                                                           ptr++;
                                                                                                                  feed(p);
ret->line = next;
#if VERSION < 4
#IT VERSION < 4
void print_if_block(const IfBlockParse *b) {
   if (b == NULL) {
      printf(*nit*);
   } else {
      printf(*(if *);
   }
}</pre>
                                                                                                              ret->lane = next;
ret->lock(a, p);
ret->next = parse_elif_block(a, p);
return ret;
} else {
    return NULL;
                                                                                                                                                                                                                      #if VERSION < 10
                                                                                                                                                                                                                     printf("(if ");
print_line(&b->line);
putchar(" ');
print_statement_block(b->statement_block);
putchar(" ');
print_elif_block(b->elif_block);
putchar(" ');
print_else_block(b->else_block);
putchar(" ');
print_statement_block(b->else_block);
putchar(" ');
print_statement_block(b->ensext).
                                                                                                              }
                                                                                                                                                                                                                      #else
                                                                                                                                                                                                                      static _Bool should_trim = 1;
                                                                                                                                                                                                                      #endif
                                                                                                           ElseBlockParse *parse_else_block(Arena *a, ParserState *p) {
                                                                                                            Line next;

if (peek(p, &next) && next.line_type == LINE_TYPE_ELSE) {
    ElseBlockParse *ret = new (a, 1, ElseBlockParse);

#if VERSION >= 9
                                                                                                                                                                                                                     #if VERSION < 3
       print_statement_block(b->next);
                                                                                                                                                                                                                     void dump_plain_block(PlainBlockParse *b) {
                                                                                                                 ret->line = next;
                                                                                                           #endif
       putchar(')'):
                                                                                                                                                                                                                      #else
                                                                                                                  feed(n):
                                                                                                                                                                                                                      #if VFRSTON < 6
                                                                                                                  ret->block = parse_statement_block(a, p);
                                                                                                                                                                                                                      void dump_plain_block(PlainBlockParse *b, _Bool final) {
                                                                                                              return ret;
} else {
                                                                                                                                                                                                                     #else
#if VERSION < 10
#endif
typedef struct {
                                                                                                                                                                                                                     void dump_plain_block(FILE *f, PlainBlockParse *b, _Bool final) {
                                                                                                                  return NULL;
                                                                                                              }
    Str file;
#if VERSION >= 5
                                                                                                                                                                                                                     #else
#if VERSION < 11</pre>
 int lineno;
#endif
                                                                                                                                                                                                                                                ment_block(FILE *, StatementBlockParse *,
                                                                                                           #if VERSION < 2
/* Returns the
} ParserState;
                                                                                                           #11 VERSION < 2
/* Returns the "turning number" for the line. If opening something, 1. If closing, -1 */
int parse_turning(Str line) {
 _Bool peek(ParserState *p, Line *out) {
                                                                                                                                                                                                                      void dump_statement_block(FILE *, StatementBlockParse *);
    if (!p->file.len) {
                                                                                                                                                                                                                      #endit
                                                                                                             nt parse_turning(Str Line) {
Cut c = cut(Line, '');
const Str IF = S("#if");
const Str IFDEF = S("#ifdef");
const Str IFNDEF = S("#ifdef");
const Str ENDIF = S("#ifdef");
if (equals(c.head, IF)) {
   return 1;
}
       return Θ;
                                                                                                                                                                                                                      void dump_plain_block(FILE *f, PlainBlockParse *b) {
}
Cut c = cut(p->file, '\n');
*out = parse_line(c.head);
#if VERSION >= 5
#if VERSION < 9
out->lineno = p->lineno + 1;
                                                                                                                                                                                                                     #endif
#endif
#endif
if (b == NULL) {
                                                                                                                                                                                                                            return;
                                                                                                                                                                                                                      }
#if VERSION >= 5
                                                                                                              if (equals(c.head, IFDEF)) {
    out->no = p->lineno + 1;
                                                                                                                                                                                                                      #if VERSION < 6
                                                                                                           #else
/* Returns true if the condition deals with the VERSION
variable */
Bool condition_version(const Line *line) {
    const Str version_str = S("VERSION");
                                                                                                                                                                                                                     printf("%d: ", b->line.lineno);
#else
    fprintstr(f, b->line.span);
#endif
#endif
 /* To be called after peek to advance the parser state */
                                                                                                                                                                                                                      #endit
void feed(ParserState *p) {
  Cut c = cut(p->file, '\n');
#if VERSION >= 5
                                                                                                              if (!equals(line->variable, version_str)) {
                                                                                                                                                                                                                      #if VERSION < 6
                                                                                                                  return Θ;
                                                                                                                                                                                                                         printstr(b->line.span);
                                                                                                              }
/* Support that fallback in the file */
if (line->operator == OP_DEF || line->operator== OP_NDEF) {
                                                                                                                                                                                                                      #if VERSION < 3
p->file = c.tail;
}
                                                                                                                                                                                                                        putchar('\n');
dump_plain_block(b->next);
                                                                                                               return 1:
                                                                                                                                                                                                                      #if VERSION < 5
PlainBlockParse *parse plain block(Arena *a, ParserState *p)
                                                                                                                                                                                                                      if (!final || b->next != NULL) {
#else
tline next;
if (peek(p, &next) && next.line_type == LINE_TYPE_PLAIN) {
#if VERSION < 5</pre>
                                                                                                                                                                                                                     #etse
#if VERSION < 9
if (!final || b->next == NULL) {
#else
                                                                                                           #if VERSION < 9
_Bool evaluate_condition(const Line *line) {</pre>
       PlainBlockParse *ret = new(a, 1, PlainBlockParse);
                                                                                                                                                                                                                     fputc('\n', f);
#if VERSION < 10</pre>
                                                                                                             Bool evaluate condition(const Line *line, int version) {
       PlainBlockParse *ret = new (a. l. PlainBlockParse);
                                                                                                                                                                                                                        line_map[current_line++] = b->line.no;
                                                                                                            _
#else
PlainBlockParse *ret = new (a, 1, Pla

#endif

feed(p);

ret >line = next;

ret >next = parse_plain_block(a, p);

return ret;

} else {

return NULL;
                                                                                                             Bool evaluate_condition(const Line *line) {
                                                                                                                                                                                                                     #etse
if (should_trim) line_map[current_line++] = b->line.no;
dump_plain_block(f, b->next);
#endif
#endif
                                                                                                            #endif
                                                                                                           #endif
#endif
/* Assumes that the variable is VERSION */
#if VERSION < 9
const int version = 1;
#endif</pre>
                                                                                                                                                                                                                      #if VERSION < 6
                                                                                                               switch (line->operator) {
                                                                                                                                                                                                                            putchar('\n');
                                                                                                             switch (line->operator) {
case OP_LT:
    return version < line->value;
case OP_CT:
    return version > line->value;
case OP_LE:
                                                                                                                                                                                                                     ....iUN < 9
fputc('\n', f);
#endif
#er'''</pre>
IfBlockParse *parse_if_block(Arena *, ParserState *);
StatementBlockParse *parse_statement_block(Arena *a,
                                                                                                                                                                                                                      #endif
ParserState *p) {
StatementBlockParse *ret = new(a, 1, StatementBlockParse);
ret->plain_block = parse_plain_block(a, p);
                                                                                                                 return version <= line->value;
                                                                                                                                                                                                                      #if VERSION < 9
                                                                                                              case OP GE:
                                                                                                                  return version >= line->value;
                                                                                                                                                                                                                      #endif
```

```
#if VERSION < 6
                                                                                     void dump_else_block(FILE *f, ElseBlockParse *b) {
                                                                                                                                                                         #if VERSION < 11
                                                                                                                                                                              dump_statement_block(f, b->statement_block, version);
dump_elif_block(f, b->elif_block, version);
   dump_plain_block(b->next, final);
                                                                                     #endif
#else
                                                                                     #endif
#if VERSION < 10
                                                                                     #endif
                                                                                                                                                                              dump_else_block(f, b->else_block, version);
dum_plain_block(f, b->next, final);
#endif
#endif
                                                                                      if (b == NULL) {
  return;
                                                                                                                                                                              dump_statement_block(f, b->statement_block);
dump_elif_block(f, b->elif_block);
dump_else_block(f, b->else_block);
                                                                                       dump_statement_block(b->block);
                                                                                                                                                                         #endif
                                                                                                                                                                         #endif
                                                                                     #if VFRSTON < 6
#if VFRSTON < 6
                                                                                                                                                                         #endif
                                                                                                                                                                         fprintstr(f, b->end_line.span);
  fputc('\n', f);
## VERSION >= 9
  line_map[current_line++] = b->end_line.no;
 Bool dump_elif_block(ElIfBlockParse *b) {
                                                                                                     ment block(b->block, 0):
                                                                                    #etse
#if VERSION < 9
dump_statement_block(f, b->block, 0);
#else
#if VERSION < 9
_Bool dump_elif_block(FILE *f, ElIfBlockParse *b) {</pre>
                                                                                                                                                                         #endif
#endif
#if VERSION < 11
_Bool dump_elif_block(FILE *f, ElIfBlockParse *b, int version) {
                                                                                    #if VERSION < 10
                                                                                       dump_statement_block(f, b->block, 0, version);
                                                                                                                                                                         #if VFRSTON < 6
version) {
#else
_Bool dump_elif_block(FILE *f, ElifBlockParse *b) {
#endif
#endif
                                                                                      if (!should_trim) {
  fprintstr(f, b->line.span);
  fputc('\n', f);
                                                                                                                                                                              printstr(b->line.span);
putchar('\n');
                                                                                    }
#if VERSION < 11
                                                                                                                                                                              dump_statement_block(b->statement_block);
 if (b == NULL) {
                                                                                       dump_statement_block(f, b->block, version);
return \theta; #endif
                                                                                     #else
                                                                                                                                                                         #if VERSION < 6
                                                                                                                                                                         dump_statement_block(b->statement_block, 0);
#endif
                                                                                      dump_statement_block(f, b->block);
                                                                                    #endif
#endif
#endif
}
#if VERSION < 2
                                                                                                                                                                         #if VERSION >= 5
#if VERSION < 6
  if (equals(c.head, IFNDEF)) {
  return 1;
                                                                                                                                                                             printf("%d: ", b->end_line.lineno);
                                                                                     #endif
                                                                                     #endif
#if VERSION < 10
                                                                                                                                                                         #endif
   if (condition_version(&b->line)) {
                                                                                    #if VERSION < 6
void dump_if_block(IfBlockParse *b) {</pre>
                                                                                                                                                                         ### VERSION < 6
    printstr(b->end_line.span);
    putchar('\n');
   if (should_trim && condition_version(&b->line)) {
#if VERSION < 9
                                                                                     void dump_if_block(FILE *f, IfBlockParse *b) {
     if (evaluate_condition(&b->line)) {
                                                                                                                                                                         }
#if VERSION < 3
#if VERSION < 11
                                                                                     #if VERSION < 11
                                                                                                                                                                                          ent_block(b->next);
      if (evaluate_condition(&b->line, version)) {
                                                                                      void dump_if_block(FILE *f, IfBlockParse *b, int version) {
                                                                                    void dump_if_block(FILE *f, IfBlockParse *b) {
#endif
                                                                                                                                                                         #etse
#if VERSION < 6
dump_statement_block(b->next, 1);
#else
#if VERSION < 9</pre>
#etse
   if (evaluate_condition(&b->line)) {
      dump_statement_block(f, b->block);
#endif
                                                                                     #endif
#if VERSION < 10
                                                                                                                                                                           dump_statement_block(f, b->next, 1);
                                                                                     #endif
dump_statement_block(f, b->block, 0, version);
#else
                                                                                      if (b == NULL) {
                                                                                    return;
#endif
                                                                                                                                                                         #if VERSION < 10
                                                                                                                                                                         dump_statement_block(f, b->next, 1, version);
#else
dump_statement_block(f, b->block, version);
#endif
                                                                                    }
#if VERSION < 2
                                                                                      if (equals(c.head, ENDIF)) {
  return -1;
                                                                                                                                                                            dump_statement_block(f, b->next, version);
 #endif
                                                                                                                                                                         #else
                                                                                                                                                                           dump statement block(f. b->next):
#if VERSION < 3
dump_statement_block(b->block);
#else
                                                                                     #if VFRSTON < 10
                                                                                                                                                                         #endit
                                                                                       if (condition_version(&b->line)) {
                                                                                                                                                                         #endif
                                                                                    #else
if (should_trim && condition_version(&b->line)) {
#if VERSION < 6
dump_statement_block(b->block, θ);
#else
                                                                                                                                                                         #endif
                                                                                    #if VERSION < 9
#if VERSION < 9
       dump_statement_block(f, b->block, 0);
                                                                                          if (evaluate condition(&b->line)) {
#endif
                                                                                                                                                                         #if VFRSTON < 3
                                                                                    ##fif VERSION < 11
if (evaluate_condition(&b->line, version)) {
##else
if (evaluate_condition(&b->line)) {
                                                                                                                                                                                             ment_block(StatementBlockParse *b) {
                                                                                                                                                                        void dump_statement_block(StatementBlockParse *b, _Bool final) {
                                                                                         #if VERSION < 6
       return dump_elif_block(b->next);
                                                                                                                                                                         #else
#if VERSION < 9</pre>
                                                                                                                                                                         #11 ..
void dump_stace
_Bool final) {
#if VFRSTON < 9
                                                                                                                                                                                      statement_block(FILE *f, StatementBlockParse *b,
       return dump_elif_block(f, b->next);
                                                                                     #if VERSION < 10
#else
#if VERSION < 11
                                                                                            dump_statement_block(f, b->statement_block, 0, version);
                                                                                     #if VERSION < 11
                                                                                                                                                                         void dump_statement_block(FILE *f, StatementBlockParse *b,
_Bool final, int version) {
       return dump_elif_block(f, b->next, version);
                                                                                    dump_statement_block(f, b->statement_block, version);
#endif
#else
       return dump elif block(f, b->next);
#endif
                                                                                                                                                                         #if VERSION < 11
                                                                                    } else if (!dump_elif_block(f, b->elif_block, version)) {
dump_else_block(f, b->else_block, version);
#endif
#endif
#endif
                                                                                                                                                                                             ment_block(FILE *f, StatementBlockParse *b,
                                                                                                                                                                         void dump_state
int version) {
#else
void dump_statement_block(FILE *f, StatementBlockParse *b) {
                                                                                                                                                                         #endit
                                                                                     #if VERSION < 3
                                                                                                                                                                         #endif
                                                                                            dump_statement_block(b->statement_block);
                                                                                                                                                                         #endif
#if VFRSTON >= 9
/* This should dump the elif pragma, but I'm not using that in my program so it's okay that it doesn't work at all */
#if VERSION >= 10
                                                                                     #if VERSION < 6
                                                                                            dump_statement_block(b->statement_block, 0);
                                                                                                                                                                         return;
#endif
                                                                                     #if VERSION < 9
fprintstr(f, b->line.span);
fputc('\n', f);
#if VERSION < 11</pre>
                                                                                         dump_statement_block(f, b->statement_block, 0);
} else if (!dump_elif_block(f, b->elif_block)) {
   dump_else_block(f, b->else_block);
                                                                                                                                                                         }
#if VERSION < 2
     VERSION < 11
dump_statement_block(f, b->block, version);
dump_elif_block(f, b->next, version);
                                                                                     #endif
                                                                                                                                                                         #if VERSION < 3
  dump_plain_block(b->plain_block);
     dump_statement_block(f, b->block);
dump_elif_block(f, b->next);
                                                                                    ##if VERSION < 6
} else if (!dump_elif_block(b->elif_block)) {
dump_else_block(b->else_block);
                                                                                                                                                                         #if VERSION < 6
                                                                                                                                                                         dump_plain_block(b->plain_block, final && b->if_block ==
NULL);
#endif
#endif
                                                                                     #endif
#endif
                                                                                                                                                                         #else
#if VERSION < 10
dump_plain_block(f, b->plain_block, final && b->if_block
== NULL);
     return 0;
                                                                                    } else {
#if VERSION >= 5
#if VERSION < 6
   printf("%d: ", b->line.lineno);
                                                                                                                                                                         #else
  dump_plain_block(f, b->plain_block);
#if VERSION < 6
                                                                                     #else
 void dump_else_block(ElseBlockParse *b) {
                                                                                          fprintstr(f, b->line.span):
                                                                                    ipintstr(i, b->Line.span);
fputc('n', f);
#if VERSION < 9
dump_statement_block(f, b->statement_block, 0);
                                                                                                                                                                         #if VERSION < 9
  dump_if_block(f, b->if_block);
#if VERSION < 9
 void dump_else_block(FILE *f, ElseBlockParse *b) {
                                                                                          line_map[current_line++] = b->line.no;
                                                                                                                                                                           dump_if_block(f, b->if_block, version);
void dump_else_block(FILE *f, ElseBlockParse *b, int version)
                                                                                    #if VERSION < 10
                                                                                                                                                                         #else
                                                                                          dump_statement_block(f, b->statement_block, 0, version);
                                                                                                                                                                           dump_if_block(f, b->if_block);
#else
```

```
if (line.mode == 'a') {
  lines_to_skip = line.tree.end - line.tree.begin + 1;
                                                                                                                                                                                                                                                                                                               void diff(Arena *a, StatementBlockParse *tree) {
  #endif
                                                                                                                                                                                                                                                                                                                   const int version = 1;
                                                                                                                                                        } else if (line.mode == 'c') {
    lines_to_skip = line.tree.end - line.tree.begin +
    line.work.end - line.work.begin + 2;
  #endif
                                                                                                                                                                                                                                                                                                               #else
                                                                                                                                                                                                                                                                                                                              splice(Arena *a. StatementBlockParse *tree.
 #if VERSION < 6
dump_if_block(b->if_block);
#endif
#endif
                                                                                                                                                                                                                                                                                                               PlainBlockParse *dna,
DiffLine range) {
   if (tree->if_block && range.tree.begin >= tree->if_block-
                                                                                                                                                             } else {
  lines_to_skip = line.tree.end - line.tree.begin + 3;
                                                                                                                                                                                                                                                                                                               >line.no) {
#if VERSION < 21
                                                                                                                                                            Cut c = {0};
  #if VERSION >= 8
                                                                                                                                                                                                                                                                                                                        /* TODO: Parse into the tree */
#if VERSION >= 8
Str read_from_popen(Arena *a, FILE *f) {
    Str ret = {0};
    ret.data = a->begin;
    while (!feof(f)) {
    #if VERSION < 9
        ret.len += fread(ret.data + ret.len, sizeof(char), a->end - (ret.data + ret.len), f);
}
                                                                                                                                                            for (int i = 0; i < lines_to_skip; i++) {
  if (!c.tail.len) {</pre>
                                                                                                                                                                                                                                                                                                               if (tree->if_block->next &&

#if VERSION >= 21

(tree->if_block->next->if_block || tree->if_block->next->plain_block) &&
                                                                                                                                                        break;
}
#else
Str skip_lines(Str str) {
                                                                                                                                                         Cut c = cut(str, '\n');
while (c.tail.data[0] == '<' || c.tail.data[0] == '>' ||
c.tail.data[0] == '-') {
#endif
                                                                                                                                                                                                                                                                                                                                         range.tree.begin >= first line statement(tree-
                                                                                                                                                                                                                                                                                                               >if block->next)) {
                                                                                                                                                                                                                                                                                                                            cuck->next)) {
splice(a, tree->if_block->next, dna, range);
return;
         #endif
                                                                                                                                                                c = cut(c.tail, '\n');
                                                                                                                                                            }
return c.tail;
     }
a->begin += ret.len;
.
     return ret;
}
                                                                                                                                                         StatementBlockParse *parse_from_filename(Arena *a, const
                                                                                                                                                            FILE *f = fopen(filename, "r");
assert(f != NULL);
ParserState p = {
 #endif
                                                                                                                                                                                                                                                                                                                             return:
 #if VERSION >= 4
#if VERSION < 6
void diff() {
                                                                                                                                                                                                                                                                                                               } // TODO: Elif is completely not handled here. Shouldn't matter, but IDK */ splice(a, tree->if_block->statement_block, dna, range);
      /* We aren't ready yet */
                                                                                                                                                                 .file = read_file(a, f),
      return;
                                                                                                                                                                                                                                                                                                                      return;
      /* Child read, parent write */
                                                                                                                                                             return parse_statement_block(a, &p);
     /* Child redu, ,...
int pl[2];
/* Parent read, child write */
                                                                                                                                                                                                                                                                                                                   PlainBlockParse *prefix = new (a, 1, PlainBlockParse);
PlainBlockParse *suffix = new (a, 1, PlainBlockParse);
                                                                                                                                                        #if VERSION >= 10
#if VERSION < 11
                                                                                                                                                                                                                                                                                                                   char *prefix_line_buf = new (a, 32, char);
sprintf(prefix_line_buf, "#if VERSION >= %d", version + 1);
                                                                                                                                                         int first_line_if(const IfBlockParse *);
      pipe(p2);
      pid_t pid = fork();
pid t pid = fork();
if (pid = 0) {
    /* Child */
    close(p1[1]);
    close(p2[0]);
    dup2(p2[0]);
    dup2(p2[1], 1);
    char *args[] = ("/usr/bin/env", "diff", "./test.c", "./
main.c", NULL);
    if (vscw/carg[0], args)) //
    if (vscw/carg[0], args) //

                                                                                                                                                                                                                                                                                                                   prefix->line.span.data = prefix_line_buf;
prefix->line.span.len = strlen(prefix->line.span.data);
prefix->next = dna;
suffix->line.span = S("#endif");
                                                                                                                                                         /* Returns the first line number of a statement block */
                                                                                                                                                         int first_line_statement(const StatementBlockParse *b) {
                                                                                                                                                            if (b->plain_block) {
  return b->plain_block->line.no;
                                                                                                                                                             return b->if_block->line.no;
                                                                                                                                                                                                                                                                                                               #if VERSION >= 11
         if (execv(args[0], args)) {
    printf("Error #%d: %s\n", errno, strerror(errno));
                                                                                                                                                                                                                                                                                                                   PlainBlockParse *ptr = tree->plain block;
                                                                                                                                                                                                                                                                                                                   while (ptr) {
  if (ptr->line.no == range.tree.begin) {
    break;
                                                                                                                                                        #if VERSION >= 12
                                                                                                                                                        int last_line_statement(const
inf (b->if_block) {
#if VERSION < 13
   if (b->if_block->next) {
                                                                                                                                                                                             atement(const StatementBlockParse *b) {
          /* Parent */
close(p1[0]);
          close(p2[1]);
                                                                                                                                                                if (b->if_block->next && (b->if_block->next->if_block ||
                                                                                                                                                                                                                                                                                                                   PlainBlockParse *dna ptr = dna:
             * Wait for child to exit */
                                                                                                                                                                                                                                                                                                                   while (dna_ptr->next) {
  dna_ptr = dna_ptr->next;
          int status:
                                                                                                                                                         b->if_block->next->plain_block)) {
          while (1) {
  wait(&status);
  if (WIFEXITED(status)) {
                                                                                                                                                                     return last_line_statement(b->if_block->next);
                                                                                                                                                                                                                                                                                                                    suffix->next = ptr->next;
                                                                                                                                                                  return b->if_block->end_line.no;
      break;
                                                                                                                                                                                                                                                                                                                   ptr->next = prefix;
dna_ptr->next = suffix;
             }
                                                                                                                                                            PlainBlockParse *ptr = b->plain_block;
                                                                                                                                                             while (ptr->next) {
  ptr = ptr->next;
  #if VFRSTON < 7
                                                                                                                                                                                                                                                                                                              void diff(StatementBlockParse *b) {
  #else
#if VERSION < 9
void diff(Arena *a, StatementBlockParse *b) {
                                                                                                                                                             return ptr->line.no;
                                                                                                                                                                                                                                                                                                                   assert(range.begin >= first_line_statement(tree) &&
  #else
typedef struct {
                                                                                                                                                                                                                                                                                                                        range.end <= last_line_statement(tree) + 1);
if (tree->if_block && range.begin > tree->if_block-
      int begin;
                                                                                                                                                         PlainBlockParse
                                                                                                                                                                                                      *get_line_range_descend(Arena
                                                                                                                                                                                                                                                                                                               >line.no) {
    if (tree->i_block->next &&
        range.begin > first_line_statement(tree->if_block->next)) {
 int end;
} DiffLineRange;
                                                                                                                                                         DiffLineRange range,
                                                                                                                                                                                                                                              const PlainBlockParse
                                                                                                                                                        *b) {
    if (b == NULL || range.end - range.begin == 0) {
DiffLineRange parse_diff_range(Str range) {
   DiffLineRange ret = {0};
   Cut c = cut(range, ',');
   if (c.ok) {
                                                                                                                                                                                                                                                                                                                             return splice_delete(a, tree->if_block->next, range);
                                                                                                                                                       }
/* Start cutting */
if (b->line.no == range.begin) {
  PlainBlockParse *copy = new (a, 1, PlainBlockParse);
  copy->line = b->line;
#if VERSION >= 13
  copy->line.no = -1;
#endif
                                                                                                                                                                                                                                                                                                                         if (tree->if block->else block &&
         f (c.ok) {
ret.begin = parse_integer(c.head);
ret.end = parse_integer(c.tail) + 1; /* This might be
                                                                                                                                                                                                                                                                                                               range.begin > tree->if_block->else_block->line.no) {
    return splice_delete(a, tree->if_block->else_block-
>block, range);
    } else {
  ret.begin = parse_integer(range);
  ret.end = ret.begin + 1;
                                                                                                                                                                                                                                                                                                               #else
if (tree == NULL) {
                                                                                                                                                                DiffLineRange new_range = range;
                                                                                                                                                                                                                                                                                                                 new_range_begin++;
new_range_begin++;
copy->next = get_line_range_descend(a, new_range, b->next);
return copy;
      return ret:
 typedef struct {
  DiffLineRange tree;
  DiffLineRange work;
                                                                                                                                                             return get_line_range_descend(a, range, b->next);
                                                                                                                                                                                                                                                                                                                         ;
int plain_begin = tree->plain_block->line.no;
       char mode:
                                                                                                                                                                                                                                                                                                                        int plain_end = ptr->line.no + 1;
 3 Diffline:
                                                                                                                                                        /* Copies the range of lines from the workfile referred to
DiffLine parse_diff_line(Str line) {
  char mode = 'a';
  Cut c = cut(line, 'a');
  if (!c.ok) {
    mode = 'c';
    c = cut(line, 'c');
  if (!c.ok) {
    c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {
        c.ok) {

                                                                                                                                                        /* copies the range of lines from the workfile referred to
by the
range. Assumes that it doesn't contain any pragmas, which
I don't
                                                                                                                                                                                                                                                                                                                      if ((range.begin >= plain_begin && range.begin < plain_end)
                                                                                                                                                                                                                                                                                                               (range.end > plain_begin \&\& range.end <= plain_end)) { #if VERSION < 14 printf("(\&d, \&d)[\&d, \&d]\n", range.begin, range.end,
                                                                                                                                                        need for the paper anyways. */
PlainBlockParse *get_line_range(Arena *a, DiffLineRange
                                                                                                                                                                                                                                                                                                              princ.;
plain_begin,
    plain_end);
                                                                                                                                                        range, const StatementBlockParse *b) {

/* First check if we need to go deeper into the if block */
          if (!c.ok) {
                                                                                                                                                         mode = 'd';
c = cut(line, 'd');
                                                                                                                                                                                                                                                                                                                   ptr = tree->plain_block;
    PlainBlockParse *parent = NULL;
    while (ptr->line.no == -1 || ptr->line.no < range.begin) {
    parent = ptr;</pre>
                                                                                                                                                            return get_line_range(a, range, b->if_block->next);
}
                                                                                                                                                                                                                                                                                                                   ptr = ptr->next;
      DiffLine ret = {0};
      ret.mode = mode;
ret.tree = parse_diff_range(c.head);
ret.work = parse_diff_range(c.tail);
return ret;
                                                                                                                                                               } /* TODO: This path isn't needed for the paper but wow */ \,
                                                                                                                                                                                                                                                                                                                            PlainBlockParse *prefix = new (a, 1, PlainBlockParse);
char *prefix_line_buf = new (a, 32, char);
sprintf(prefix_line_buf, "#if VERSION < %d", version
                                                                                                                                                            return get_line_range_descend(a, range, b->plain_block);
 #if VERSION < 18
Str skip_lines(Str str, DiffLine line) {</pre>
                                                                                                                                                                                                                                                                                                                            prefix->line.span.data = prefix_line_buf;
prefix->line.span.len = strlen(prefix->line.span.data);
      int lines_to_skip;
                                                                                                                                                         #if VERSION < 11
```

```
prefix->next = ptr;
                                                                                                                   if (parent == NULL) {
tree->plain_block = prefix;
                                                                                                                                                                                                                                 #endif
    if (parent == NULL) {
tree->plain_block = prefix;
                                                                                                                                                                                                                                 #if VERSION < 23
                                                                                                                          } else {
         } else {
                                                                                                                              parent->next = prefix:
                                                                                                                                                                                                                                    if (tree->if_block->else_block) {
             parent->next = prefix;
                                                                                                                           | PlainBlockParse *back_ptr = ptr;
| while (back_ptr &&
| (back_ptr->line.no == -1 || back_ptr->line.no <
                                                                                                                                                                                                                                splice_change(a, tree->if_block->else_block->block, work, dma, range);
          }
PlainBlockParse *back_ptr = ptr;
while (back_ptr &&
    (back_ptr->line.no == -1 || back_ptr->line.no <
                                                                                                                 range.tree.end)) {
 range.end)) {
                                                                                                                                                                                                                                 #if VERSION < 23
                                                                                                                    parent = back ptr:
                                                                                                                                                                                                                                 splice_change(a, tree->if_block->else_block->block, work, range, delete); #endif #endif
    parent = back_ptr;
back_ptr = back_ptr->next;
                                                                                                                    back_ptr = back_ptr->next;
                                                                                                                          }
PlainBlockParse *interlude = new (a, 1, PlainBlockParse);
interlude->line.span = S(*#else*);
parent->next = interlude;
PlainBlockParse *suffix = new (a, 1, PlainBlockParse);
suffix->line.span = S(*#endif*);
suffix->next = back_ptr;
cerranu = 72
cerranu = 72
cerranu = 72
          }
PlainBlockParse *suffix = new (a, 1, PlainBlockParse);
suffix->line.span = S("#endif");
suffix->next = back_ptr;
parent->next = suffix;
                                                                                                                                                                                                                                 }
#if VERSION < 22
                                                                                                                                                                                                                                    splice_change(a, tree->if_block->next, work, dna, range);
#endi1
                                                                                                                                                                                                                                 #if VFRSTON < 23
                                                                                                                 #if VERSION >= 22
#if VERSION < 23
#if VERSION < 13
                                                                                                                                                                                                                                          splice_change(a, tree->if_block->next, work, range,
        /* TODO: Again, this doesn't handle elif */
return splice_delete(a, tree->if_block->statement_block,
                                                                                                                   if (delete) {
  parent->next = suffix;
                                                                                                                                                                                                                                 #else
    splice_change(a, tree->if_block->next, work, range);
                                                                                                                 #else
                                                                                                                                                                                                                                 #endif
#if VERSION < 13
   if VERSION < 13
PlainBlockParse "ptr = tree->plain_block;
if (ptr->line.no == range.begin) {
    PlainBlockParse "back_ptr = ptr;
    while (back_ptr->line.no != range.end - 1) {
        back_ptr = back_ptr->next;
}
                                                                                                                 #endif
                                                                                                                #if VERSION >= 17
                                                                                                                                                                                                                                 void update_version_tree(StatementBlockParse *);
                                                                                                                                                                                                                                 #endif
       PlainBlockParse *prefix = new (a, 1, PlainBlockParse);
                                                                                                                                                                                                                                 #endif
       realmouteraise prila - new (a, 32, fearmouteraise), char "prefix_line_buf = new (a, 32, char); sprint(prefix_line_buf, "#if VERSION < %d", version + 1); prefix_>line_span_data = prefix_line_buf; prefix>-line.span.len = strlen(prefix->line.span.data);
                                                                                                                                                                                                                                 void diff(Arena *a, StatementBlockParse *tree) {
#endif
#endif
#endif
                                                                                                                                 + range.work.begin,
                                                                                                                                                                                                                                    char tree_file_name[] = "/tmp/tree.cXXXXXXX";
       prefix->next = ptr;
tree->plain_block = prefix;
                                                                                                                 #if VERSION < 23
                                                                                                                                                                                                                                int fileno = mkstemp(tree_file_name);
FILE *tree_file = fdopen(fileno, "w");
#if VERSION < 9
dump_statement_block(tree_file, b, 0);
#else
      PlainBlockParse 'suffix = new (a, 1, PlainBlockParse);
suffix->line.span = 5("#endif");
suffix->next = back_ptr->next;
back_ptr->next = suffix;
                                                                                                                    range_to_get.end = range_to_get.begin + (range.work.end
range.work.begin);
                                                                                                                           if (range_to_get.end - range_to_get.begin >
    range.work.end - range.work.begin) {
#else
  if (!tree->if_block) {
                                                                                                                                                                                                                                 #if VERSION < 10
line_map = new (a, 1024 * 1024, int);</pre>
#endif
       return:
                                                                                                                          range_to_get.end =
                                                                                                                                                                                                                                    current line = 1:
                                                                                                                                                range_to_get.begin + (range.work.end
#if VERSION < 13
                                                                                                                 range.work.begin);
                                                                                                                                                                                                                                 #if VERSION < 10
dump_statement_block(tree_file, tree, 0, version);
#else</pre>
                                                                                                                range.work.begin);
if (range_to_get.begin > splice_change_delete &&
range_to_get.end <= range.work.end) {
    splice_change_delete = range_to_get.end;
    PlainBlockParse *fresh_dna = get_line_range(a,
    range_to_get, work);
    if (fresh_dna) {
        interlude>-next = fresh_dna;
        while (fresh_dna) -anext) {
        fresh_dna = fresh_dna>-next;
    }
}
   stse
splice_delete(a, tree->if_block->statement_block, range);
if (tree->if_block->else_block) {
    splice_delete(a, tree->if_block->else_block->block,
                                                                                                                                                                                                                                     dump_statement_block(tree_file, tree, version);
                                                                                                                                                                                                                                 #else
 range):
                                                                                                                                                                                                                                    dump statement block(tree file, tree):
                                                                                                                                                                                                                                 #endit
     ,
splice_delete(a, tree->if_block->next, range);
                                                                                                                                                                                                                                 #endif
                                                                                                                                                                                                                                 #endif
fclose(tree_file);
                                                                                                                        fresh_dna->next = suffix;
                                                                                                                                                                                                                                    char work_file_name[] = "/tmp/work.cXXXXXXX";
fileno = mkstemp(work_file_name);
FILE *work_file = fdopen(fileno, "w");
/* Assuming C file under lmb for the paper */
#if VERSION >= 15
#if VERSION < 22</pre>
                                                                                                                       } else {
parent->next = suffix;
 void splice_change(Arena *a, StatementBlockParse *tree, const
StatementBlockParse *work, PlainBlockParse *dna, DiffLine
                                                                                                                                                                                                                                **Assuming C file under lmb for the pap

#if VERSION < 7

can *buf = malloc(1024 * 1024);

#else

char *buf = new (a, 1024 * 1024, char);
 range) {
                                                                                                                 ##If VERSION < 23
PlainBlockParse *fresh_dna = get_line_range(a,
 #else
#if VERSION < 23</pre>
                                                                                                                 range_to_get, work);
                    _change(Arena *a, StatementBlockParse *tree, const
                                                                                                                 #if VERSION < 22
 StatementBlockParse *work, DiffLine range, _Bool delete) {
                                                                                                                                                                                                                                 #endif
                                                                                                                                                                                                                                     unsigned long fssize = fread(buf, sizeof(char), 1024 *
                                                                                                                           interlude->next = fresh dna:
                                                                                                                                                                                                                                unsigned long fssize = fread(buf, sizeof(ch
1024, stdin);
fwrite(buf, sizeof(char), fssize, work_file);
#if VERSION < 8
free(buf);
#endif</pre>
#etse
static int splice_change_delete;
void splice_change(Arena *a, StatementBlockParse *tree, const
StatementBlockParse *work, DiffLine range) {
                                                                                                                    while (fresh_dna->next)
fresh_dna = fresh_dna->next;
                                                                                                                 #else
#if VERSION < 23
#endif
#endif
                                                                                                                   if (fresh_dna) {
  if (!delete &&range_to_get.end >= range.work.end) {
                                                                                                                                                                                                                                 fclose(work_file);
#if VERSION >= 9
   if (tree == NULL) {
       return;
                                                                                                                       delete = 1;
    f if (tree->plain_block) {
  PlainBlockParse *ptr = tree->plain_block;
  while (ptr->next) {
    ptr = ptr->next;
}
                                                                                                                 interlude->next = fresh_dna;
while (fresh_dna->next) {
   fresh_dna = fresh_dna->next;
#endif
                                                                                                                                                                                                                                         Give the buffer some extra space */
                                                                                                                                                                                                                                 /* Give the burrer some extra space */
char cmdbwf[256];
    sprintf(cmdbuf, "diff %s %s", tree_file_name,
work_file_name);
      }
int plain_begin = tree->plain_block->line.no;
int plain_end = ptr->line.no + 1;
                                                                                                                 #if VERSION < 23
                                                                                                                                                                                                                                 #if VERSION < 8
                                                                                                                    fresh_dna->next = suffix;
                                                                                                                                                                                                                                    FILE *diff_output = popen(cmdbuf, "r");
                                                                                                                 face valid output = popen(chaudur, fr),
for (;) {
  char outbuf (4096);
  unsigned long r = fread(outbuf, sizeof(char), 4096,
  diff_output);
        /* Check for overlap */
#if VERSION < 22
if ((range.tree.begin >= plain_begin && range.tree.begin
                                                                                                                 #endif
#endif
                                                                                                                                                                                                                                       if (r == 0) {
    break;
 < plain end) ||
               (range.tree.end > plain_begin && range.tree.end <=
                                                                                                                                                                                                                               #else
FILE "diff_out = popen(cmdbuf, "r");
Str diff_output = read_from_popen(a, diff_out);
#if VERSION < 9
fprintstr(stdout, diff_output);
#else</pre>
plain end)) {
                                                                                                                 #if VFRSTON < 22
 ##else
if (((range.tree.begin >= plain_begin && range.tree.begin
                                                                                                                           fresh_dna->next = suffix;
                                                                                                                 #endif
if (((range.tree.begin ~ pican_regan & to.gctree.end ~ plain_end) || (range.tree.end > plain_begin & range.tree.end <= plain_end)) || (plain_begin >= range.tree.begin & plain_end <= range.tree.end)) {
                                                                                                                   }
if (!tree->if_block) {
                                                                                                                                                                                                                                         StatementBlockParse *work = parse from filename(a.
 ptr = tree->plain_block;
PlainBlockParse *parent = NULL;
while (ptr->line.no == -1 || ptr->line.no <
range.tree.begin) {</pre>
                                                                                                                 splice_change(a, tree->if_block->statement_block, work,
dna, range);
                                                                                                                                                                                                                                 work_file_name);
                                                                                                                                                                                                                                 while (diff_output.len) {
  Cut c = cut(diff_output, '\n');
  DiffLine d = parse_diff_line(c.head);
#if VERSION < 10</pre>
                                                                                                                 #else
#if VERSION < 23</pre>
   parent = ptr;
ptr = ptr->next;
}
                                                                                                                 splice_change(a, tree->if_block->statement_block, work,
range, delete);
                                                                                                                                                                                                                                #11 VERSIUW < 10
    printf("%c %d %d - %d %d\n", d.mode, d.tree.begin,
d.tree.end, d.work.begin,
d.work.end);</pre>
          PlainBlockParse "prefix = new (a, 1, PlainBlockParse);
char "prefix_line_buf = new (a, 32, char);
sprintf(prefix_line_buf, "#if VERSION < %d", version
                                                                                                                   if (evaluate_condition(&tree->if_block->line)) {
    splice_change(a, tree->if_block->statement_block, work,
                                                                                                                 splice_cnange(a, tree->iT_block->
range);
} else {
   if (tree->if_block->else_block) {
                                                                                                                                                                                                                                 prefix->line.span.data = prefix_line_buf;
prefix->line.span.len = strlen(prefix->line.span.data);
prefix->next = ptr;
                                                                                                                                                                                                                                                  d.work.begin, d.work.end);
                                                                                                                              splice_change(a, tree->if_block->else_block->block,
                                                                                                                 work, range);
                                                                                                                                                                                                                                        PlainBlockParse *dr = get_line_range(a, d.work, work);
```

```
should_trim = 0;
                                                                                                                                                                                                        \label{eq:statement_block} StatementBlockParse \ \ ^*s = parse\_statement\_block(\&a, \&p); \\ \ \#if \ VERSION < 6
dump_plain_block(stderr, dr);
#endif
                                                                                                     #endif
                                                                                                                                                                                                              dump_statement_block(s, 0);
#endif
#if VERSION < 12
                                                                                                                                                                                                        #if VERSION < 9
                                                                                                                                                                                                               dump_statement_block(stdout, s, 0);
      if (d.mode != 'a') {
    fprintf(stderr, "Diff mode currently not supported");
                                                                                                     #if VERSION < 5
                                                                                                     int main() {
#else
                                                                                                                                                                                                       /* TODO: Get the version from the program */
#else
         return;
                                                                                                    woid print_usage(const char *name) {
    fprintf(stderr, "Usage: %s checkout\n%s commit <tree-
file>\n", name, name);
#enoil
#if VERSION >= 11
    d.tree.begin = line_map[d.tree.begin];
#if VERSION < 12
    d.tree.end = line_map[d.tree.end];</pre>
                                                                                                                                                                                                        #if VERSION < 20
                                                                                                                                                                                                              version = get_version_tree(s);
                                                                                                                                                                                                       version = get_version_args(argc, argv);
if (version == -1) {
    version = get_version_tree(s);
}
                                                                                                    #if VERSION >= 16
/* Returns the version number, or -1 if none is given */
int get_version_args(int argc, char *argv[]) {
    for (int i = 1; i < argc - 1; i++) {
        if (!strcmp(argv[i], "-v")) {
            return atoi(argv[i + 1]);
        }
    }
}</pre>
a.tree.end = line_map[d.tree.end];
#else
#if VERSION < 22
d.tree.end = line_map[d.tree.end - 1] + 1;</pre>
#else
                                                                                                                                                                                                        #endif
#else
    d.tree.end = line_map[d.tree.end - 1] + 1;
    should_trim = 0;
#endif
#if VERSION < 15</pre>
                                                                                                                                                                                                       #endif
#endif
#if VERSION < 11
const int version = 1;
#else
    dump_statement_block(stdout, s);
#endif</pre>
      fprintf(stderr, "[%c] TREE %d %d - WORK %d %d\n", d.mode,
/* Returns the version number from the block, or -1 if none
                                                                                                                                                                                                        #if VERSION < 10
                                                                                                                                                                                                              dump_statement_block(stdout, s, 0, version);
                                                                                                     int get_version_tree(StatementBlockParse *b) {
                                                                                                                                                                                                        #etse
#if VERSION < 11
    dump_statement_block(stdout, s, version);
#endif</pre>
                                                                                                       if (!b) {
    return -1;
#if VERSION < 12
      PlainBlockParse *dna = get_line_range(a, d.work, work);
                                                                                                       }
if (b->plain_block) {
      should_trim = 0;
                                                                                                                                                                                                        #endif
                                                                                                                                                                                                        #endif
      splice(a, tree, dna, d);
dump_statement_block(stdout, tree);
                                                                                                       if (!b->if_block) {
  return -1;
                                                                                                                                                                                                          } else if (!strcmp("commit", argv[1])) {
   if (argc < 3) {
      print_usage(argv[0]);
      goto end;
}</pre>
#etse
if (d.mode == 'a') {
    PlainBlockParse *dna = get_line_range(a, d.work, work);
#if VERSION < 17
    should_trim = 0;</pre>
                                                                                                       }
if (!equals(b->if_block->line.variable, S("VERSION"))) {
                                                                                                                                                                                                              }
FILE *tree_file = fopen(argv[2], "r");
                                                                                                       }
if (b->if_block->line.operator != OP_NDEF) {
   return -1;
                                                                                                                                                                                                        Str file_contents = read_file(&a, tree_file);
#if VERSION >= 6
splice(a, tree, dna, d);
#if VERSION < 17
dump_statement_block(stdout, tree);
#endif</pre>
                                                                                                       }
Str line = b->if_block->statement_block->plain_block-
                                                                                                                                                                                                             ParserState p = {
    .file = file_contents,
                                                                                                    >line.span;

/* Use offset to get the version number alone */

line.data += 16;

line.len -= 16;
                                                                                                                                                                                                               ;,
StatementBlockParse *s = parse_statement_block(&a, &p);
#if VERSION >= 16
version = get_version_tree(s);
                                                                                                     return parse_integer(line);
#if VERSION >= 17
                                                                                                                                                                                                        #if VERSION < 7
         should_trim = 0;
dump_statement_block(stdout, tree);
                                                                                                                                                                                                        #IN VERSION < /
diff(s);
#else
    diff(&a, s);
#endif</pre>
                                                                                                    static char version_tree_buf[32];
void update_version_tree(StatementBlockParse *b) {
                                                                                                                                                                                                        #endif
         PlainBlockParse *dna = get_line_range(a, d.work, work);
                                                                                                        /* TODO: Add version tree if it doesn't exist */
sprintf(version_tree_buf, "#define VERSION %d", version
                                                                                                                                                                                                              fclose(tree_file);
        splice_change(a, tree, work, dna, d);
                                                                                                                                                                                                          } else {
  print_usage(argv[0]);
                                                                                                       1);
b->if_block->statement_block->plain_block->line.span.data
version_tree_buf;
b->if_block->statement_block->plain_block->line.span.len =
#if VERSION < 23
                                                                                                                                                                                                        end:
#endif
#if VERSION < 2
         splice_change(a, tree, work, d, 0);
        splice_change_delete = -1;
splice_change(a, tree, work, d);
                                                                                                     strlen(version tree buf);
                                                                                                                                                                                                           print_statement_block(s);
                                                                                                     #endif
#endif
#endif
#endif
#endif
#if VER
                                                                                                                                                                                                        #etse
#if VERSION < 3
dump_statement_block(s);
#else
#if VERSION < 5</pre>
                                                                                                    #endif
int main(int argc, char *argv[]) {
  if (argc < 2) {
    print_usage(argv[0]);
    return 1;</pre>
      VERSION < 17
should_trim = 0;
                                                                                                                                                                                                        dump_statement_block(s, 0);
#endif
         dump_statement_block(stdout, tree);
                                                                                                   } else {
   fprintf(stderr, "Diff mode currently not supported\n");
   return;
                                                                                                                                                                                                        #endif
                                                                                                                                                                                                        #if VERSION < 5
                                                                                                                                                                                                       putchar('\n');
#endif
#endif
#if VERSION >= 17
should_trim = 0;
#endif
                                                                                                                                                                                                           free(buf);
#endif
                                                                                                          .begin = buf,
.end = buf + buf_size,
                                                                                                                                                                                                        #if VERSION >= 4
                                                                                                                                                                                                        #if VERSION < 5
                                                                                                                                                                                                        diff();
#else
return 0;
#endif
#if VERSION < 12
                                                                                                        Str file_contents = read_file(&a, stdin);
                                                                                                    #else
#if VERSION < 10</pre>
      diff_output = skip_lines(diff_output, d);
                                                                                                          .begin = buf,
.end = buf + buf_size,
                                                                                                                                                                                                        #endif
                                                                                                    .begin = buf,
.end = buf + buf_size,
#endif
      diff_output = skip_lines(diff_output);
#endif
}
#if VERSION >= 17
   update_version_tree(tree);
dump_statement_block(stdout, tree);
                                                                                                    #endif
#if VERSION < 1</pre>
                                                                                                      ir VeRSION < 1
cut c = {0};
c.tail = file_contents;
while (c.tail.len) {
    c = cut(c.tail, '\n');
    Str line = trimleft(c.head);
    Line l = parse_line(tine);
    printf("%d\n", l.line_type);</pre>
#endif
#endif
 #endif
 #if VERSION < 8
#if VERSION < 5
      char buf[200];
                                                                                                     #if VFRSTON < 5
                                                                                                    ParserState p = {
    .file = file_contents,
#endif
 #if VERSION < 6
______ < 6
char buf[2000];
#else
fwrite(outbuf, sizeof(char), r, stdout); #endif
#if VERSION < 8
                                                                                                      );
if VERSION >= 10
line_map = new (&a, 1024 * 1024, int);
#endif
#endif
#if VERSION < 6
    size_t s = read(p2[0], &buf, 200);
    sleep(1);
    buf[s] = '\0';
    buf[s] = '\0';
    sleep(d: %s\n", s, buf);</pre>
                                                                                                     #if VFRSTON < 5
                                                                                                       StatementBlockParse *s = parse_statement_block(&a, &p);
                                                                                                    ParserState p = {
    .file = file_contents,
};
#if VFRSTON < 8
```

Bibliography

- [1] JetBrains, "Team Tools The State of Developer Ecosystem in 2023 Infographic." [Online]. Available: https://www.jetbrains.com/lp/devecosystem-2023/team-tools/
- [2] Linus Torvalds, "Git." [Online]. Available: https://git-scm.org/
- [3] cppreference.com, "Preprocessor." [Online]. Available: https://en.cppreference.com/w/c/preprocessor
- [4] Wikipedia, "LL parser." [Online]. Available: https://en.wikipedia.org/wiki/LL_parser
- [5] Chris Wellons, "Robust Wavefront OBJ model parsing in C." [Online]. Available: https://nullprogram.com/blog/2025/03/02/
- [6] OpenBSD, [Online]. Available: https://man.openbsd.org/diff
- [7] "Git git-bundle Documentation." [Online]. Available: https://git-scm.com/docs/git-bundle
- [8] Matt Mahoney, "Silesia Open Source Compression Benchmark." [Online]. Available: http://mattmahoney.net/dc/silesia.html