

ECS 36A, May 1, 2023

Announcements

- Tutoring is available from the CS Tutoring Club
- All homework 2 and extra credit 2 problems have Gradescope running
- Executables of the answers to problems in homework 2 are on the CSIF in `~bishop/ecs36a/hw2`

Reversing a String

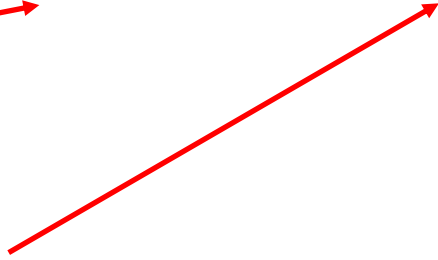
- Approach:
 - If 0 or 1 characters in string, it's reversed
 - Otherwise swap the first and last chars, and reverse the string between them
- Suggested interface:

```
void reverse(char *str, int b, int e)
```


The string to be reversed



Index of the first char in
the string to be reversed



Index of the last char in
the string to be reversed

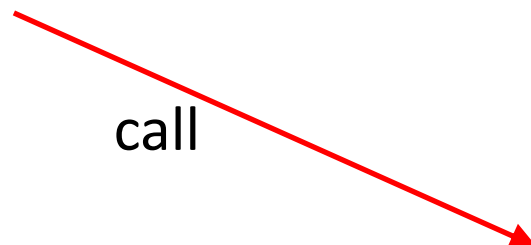


Here's What It Looks Like

`reverse("abcde", 0, 4)`

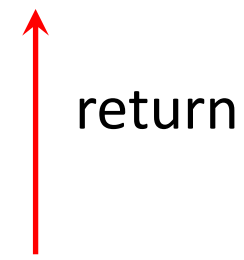


`reverse("ebcda", 1, 3)`

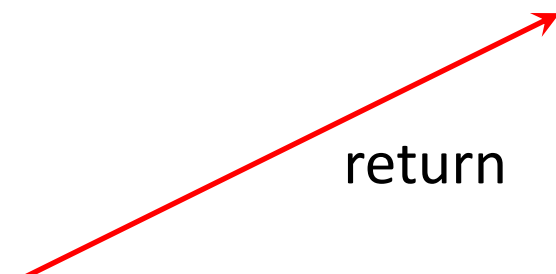


`reverse("edcba", 2, 2)`

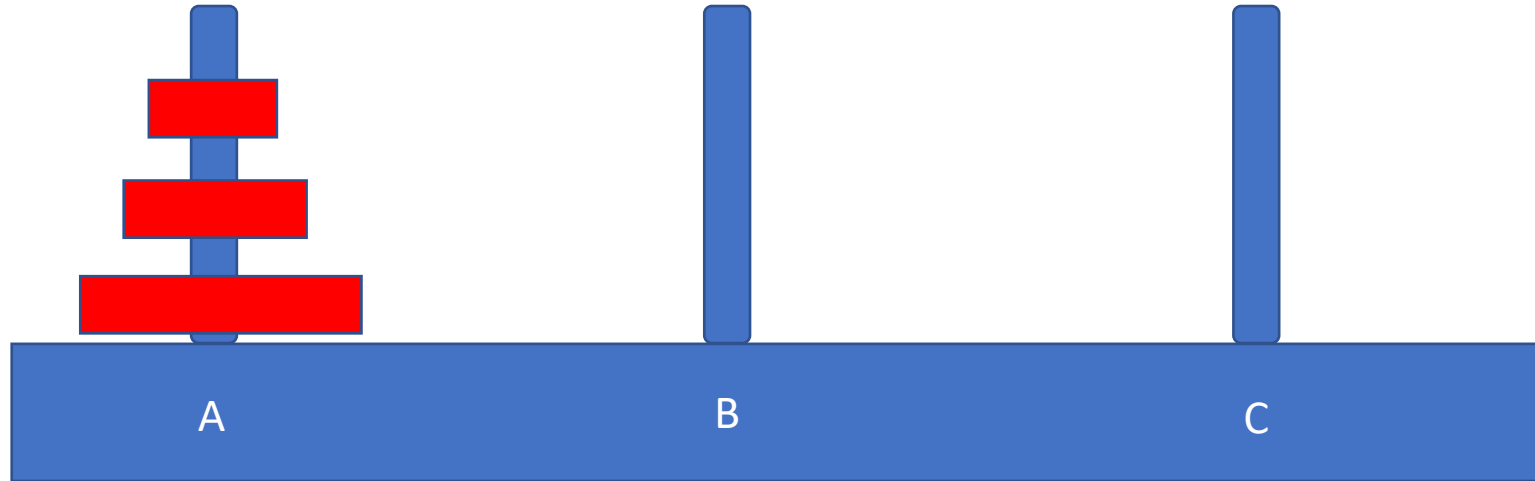
`reverse("edcba", 0, 4)`



`reverse("edcba", 1, 3)`

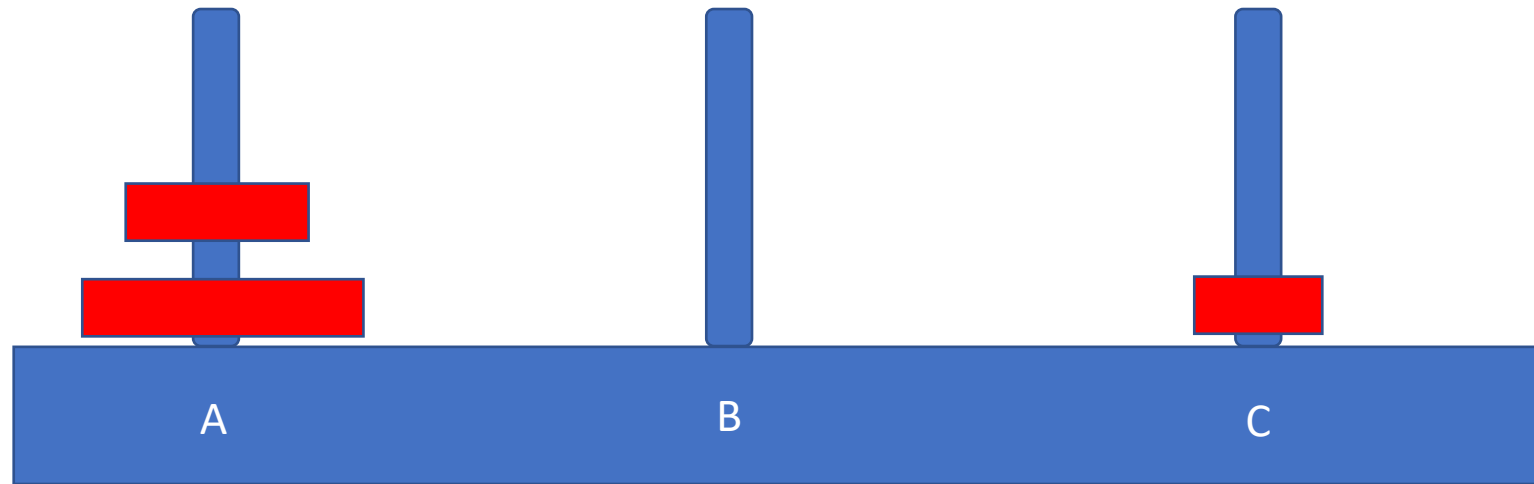


Tower of Hanoi



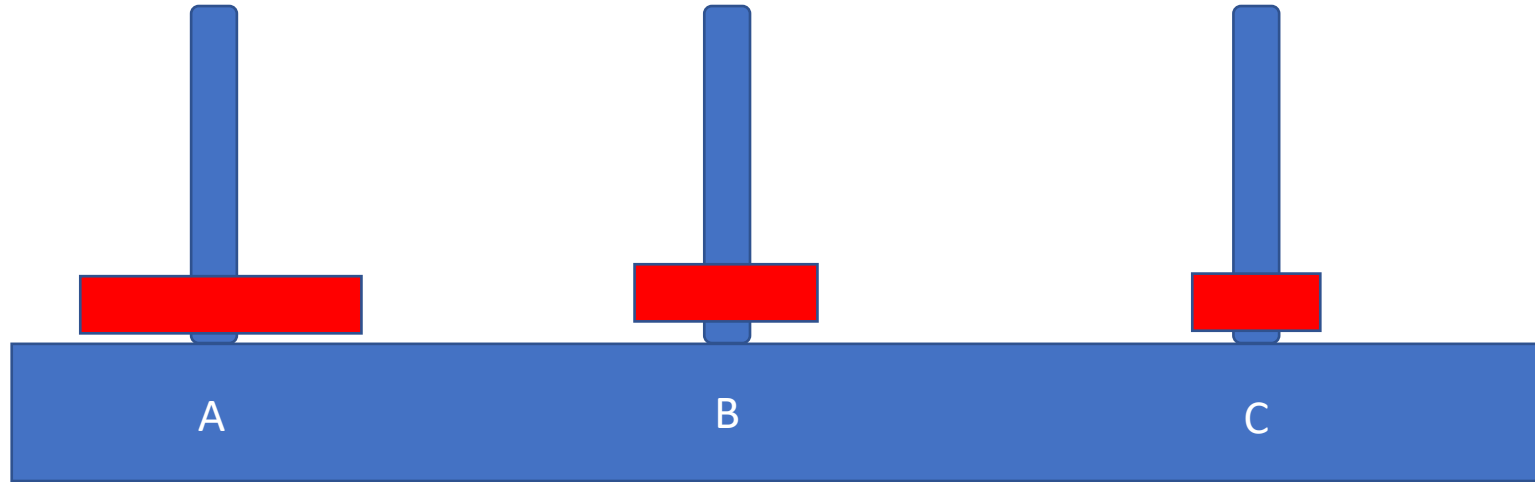
- Problem: move all 3 disks from peg A to peg C
- *Restriction*: can never put a larger disk on a smaller one!
- Approach: move all but bottom to peg B from peg A, move bottom one from A to C, then move stack from B to C

Tower of Hanoi



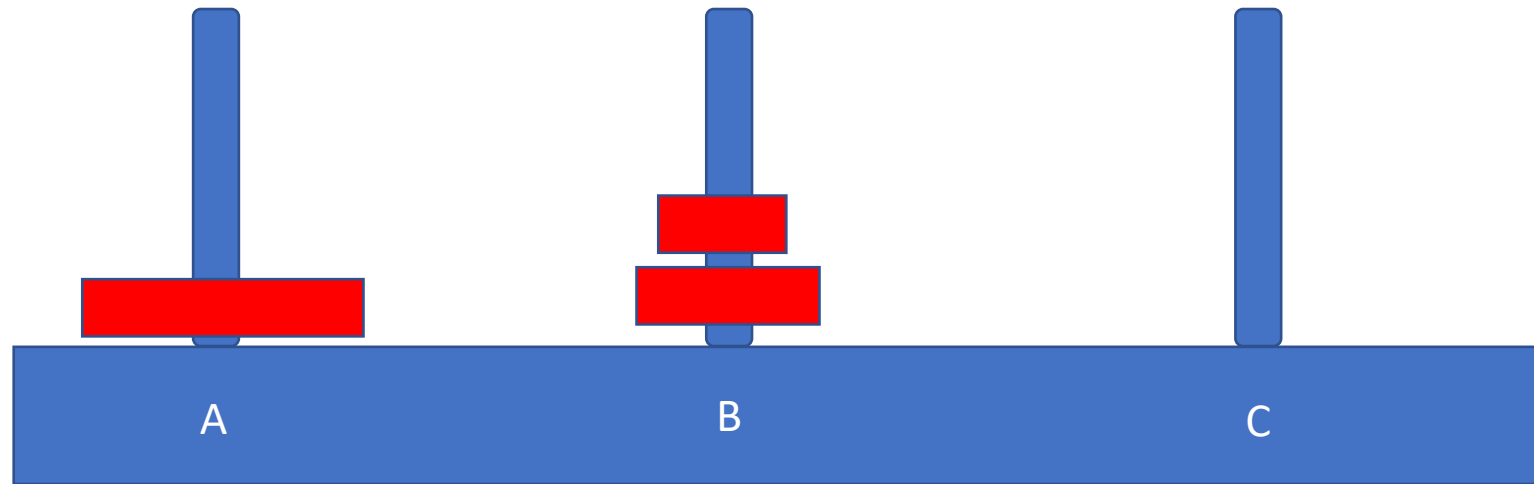
- Move top disk from A to C

Tower of Hanoi



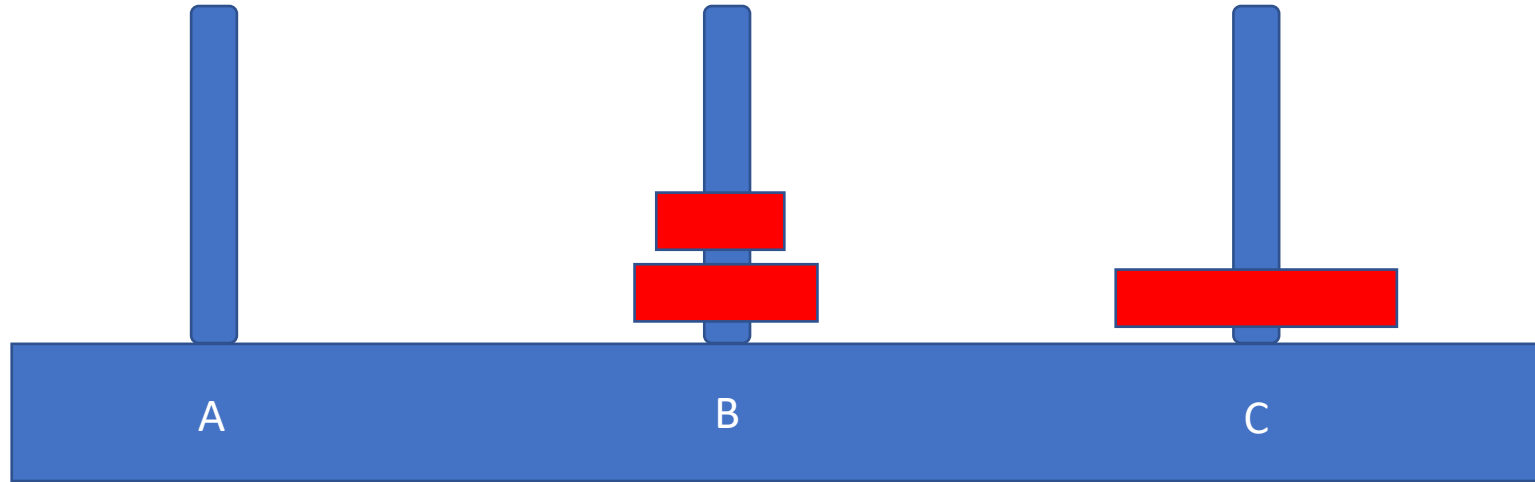
- Move top disk from A to B
- Now we can put C onto B and we have transferred the stack

Tower of Hanoi



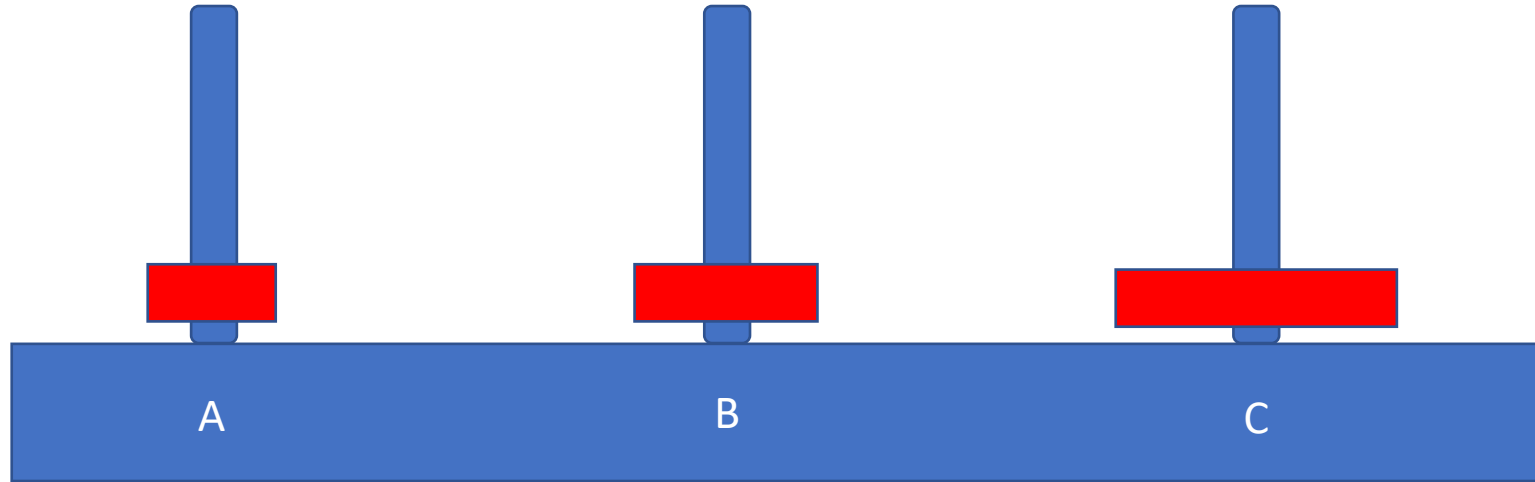
- Move top disk from C to B
- Stack is moved

Tower of Hanoi



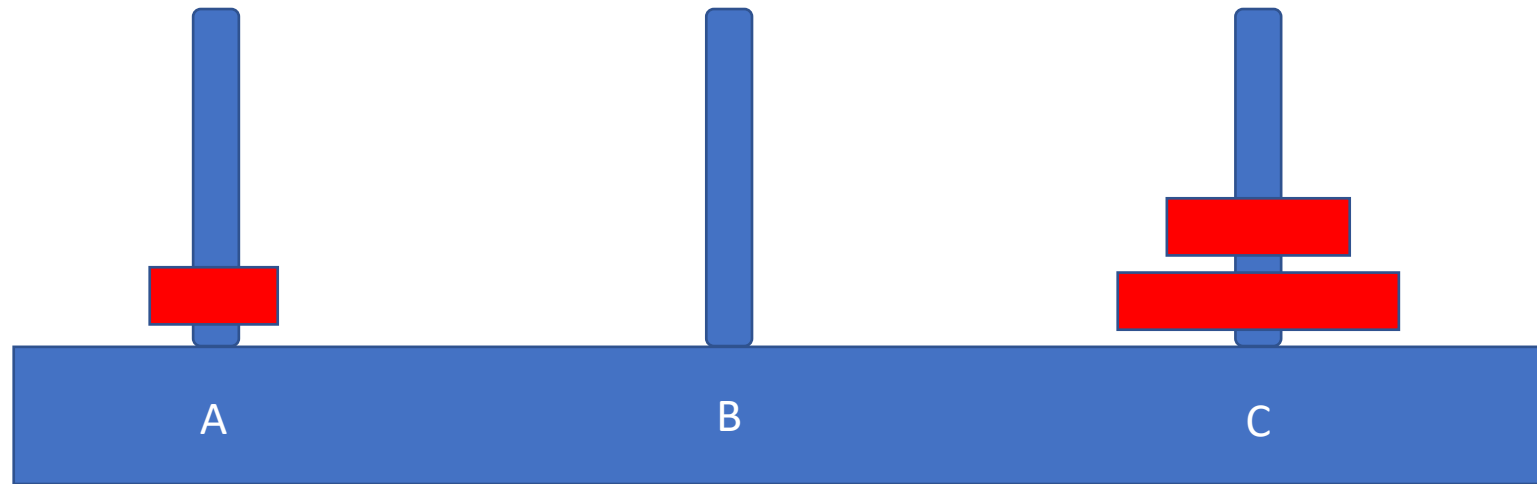
- Move top disk from A to B
- Now move disks from peg B to C
- Cannot do it directly; if we put the top one on peg C, we must move last peg from B to A. So put the top one on A

Tower of Hanoi



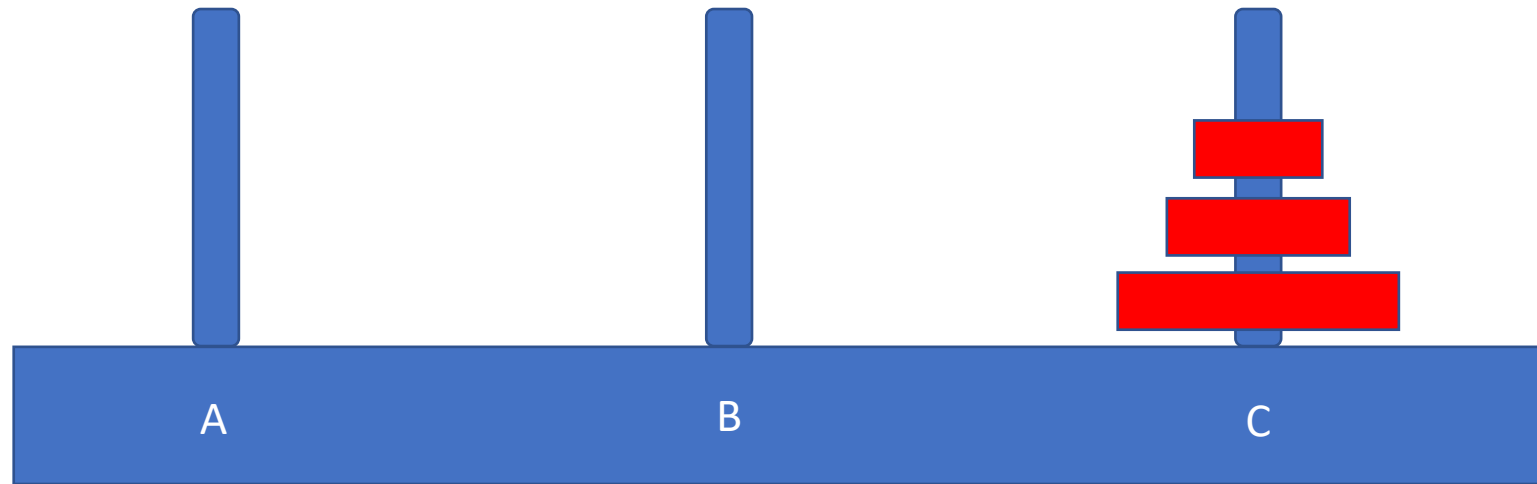
- Move top disk from A to B
- Now move disks from peg B to C
- Cannot do it directly; if we put the top one on peg C, we must move last peg from B to A. S put the top one on A

Tower of Hanoi



- Move top disk from B to C
- Almost done!

Tower of Hanoi



- Move top disk from A to C
- And done!

Sequence of Moves

- Move top disk from tower A to tower C
- Move top disk from tower A to tower B
- Move top disk from tower C to tower B
- Move top disk from tower A to tower C
- Move top disk from tower B to tower A
- Move top disk from tower B to tower C
- Move top disk from tower A to tower C

Sequence of Moves

- Move top disk from tower A to tower C
- Move top disk from tower A to tower B
- Move top disk from tower C to tower B
- Move top disk from tower A to tower C
- Move top disk from tower B to tower A
- Move top disk from tower B to tower C
- Move top disk from tower A to tower C

Move top 2 disks from A to B

Move top 2 disks from B to C

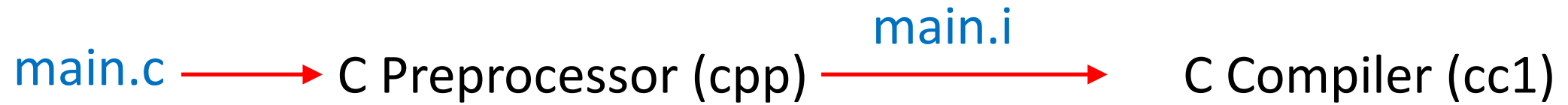
In Program, Reading Integer

```
/* read in a line, including the newline */
while (fgets(buf, MAXINPUT, stdin) != NULL) {
    /* convert this to an integer; report an error if needed */
    if (sscanf(buf, "%d", &n) != 1 || n <= 0) {
        fprintf(stderr, "Enter a positive integer\n");
        continue;
    }
    . . . . .
}
/* got an EOF */
```

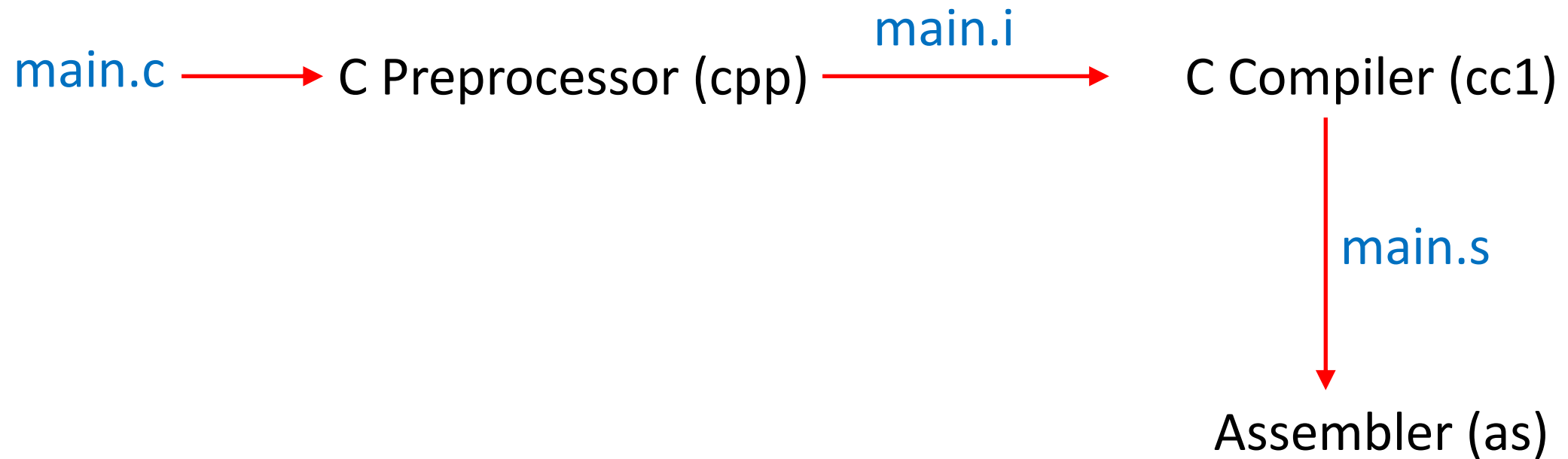
C Compiler Structure

`main.c`  C Preprocessor (cpp)

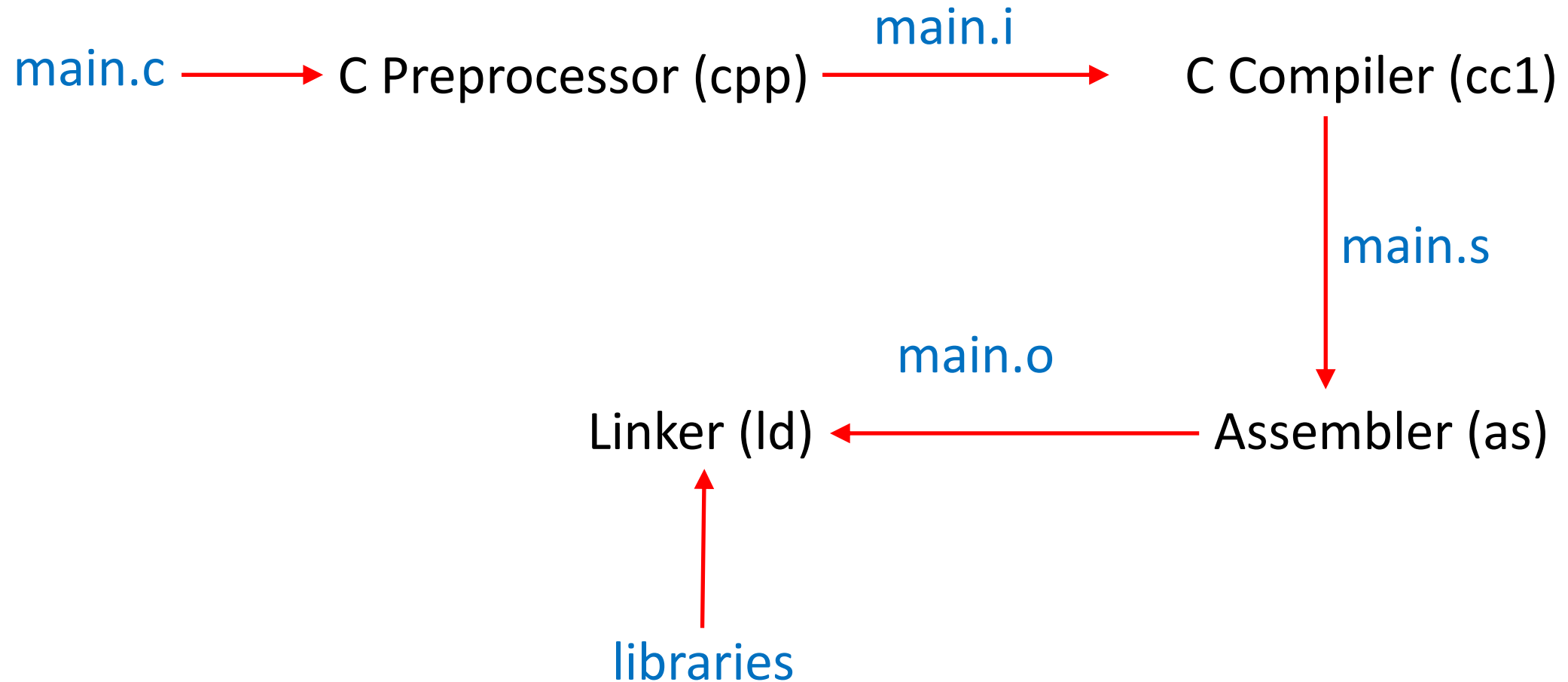
C Compiler Structure



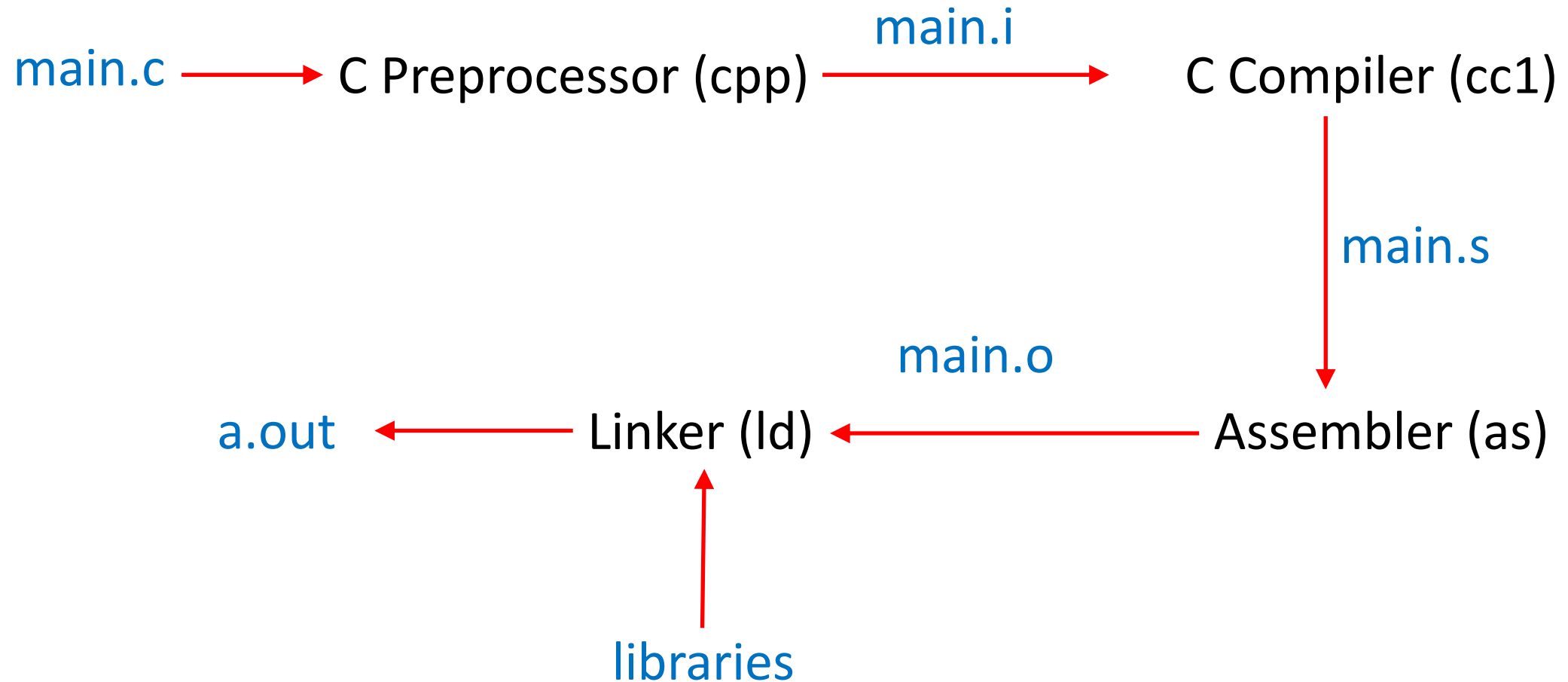
C Compiler Structure



C Compiler Structure



C Compiler Structure



C Preprocessor

- All lines begin with #
- `#define`
- `#undef`
- `#include`

#define

```
#define BOARD 8*8
```

- Replace every occurrence of the word “BOARD” with “8*8”
- Usually used to parameterize something; examples from stdio.h:
 - NULL is a macro (0)
 - EOF is a macro (-1)
- *Warning: this is textual substitution, so do not treat them as variables!*

Watch Out For This

- Goal: create a chessboard, each side being 8 squares, and 2 extra squares for computation, for a total of 100 squares

```
#define SIDE 8+2
```

- Now every occurrence of SIDE is replaced by 8+2

```
char chess[SIDE*SIDE];
```

becomes

```
char chess[8+2*8+2];
```

So the board has 26 squares

Do This

- Goal: create a chessboard, each side being 8 squares, and 2 extra squares for computation, for a total of 100 squares

```
#define SIDE (8+2)
```

- Now every occurrence of SIDE is replaced by (8+2)

```
char chess[SIDE*SIDE];
```

becomes

```
char chess[(8+2)*(8+2)];
```

So the board has 100 squares

Parameterized Macro

```
#define isbetween0and9 (x)    ( (0<= (x) ) && ( (x) <=9) )
```

- `isbetween0and9(4)` returns 1 and `isbetween0and9(-100)` returns 0
- Beware — whatever is put for `x` is evaluated every time `x` occurs in the macro definition

```
x = 9; . . . isbetween0and9 (x++)
```

becomes

```
x = 9; . . . ( (0<= (x++) ) && ( (x++) <=9) )
```

or

```
x = 9; . . . ( (0<= (9) ) && ( (10) <=9) )
```

which returns false (as `10 > 9`)

#undef

- Delete a macro definition

```
#define XYZZY    "dizzy"
```

```
. . .
```

```
#undef XYZZY
```

```
int XYZZY = -20;
```

- Without the `#undef`, the declaration becomes:

```
int "dizzy" = -20;
```

which gives an error

#include

- Interpolate file into current source code
- When it does this, it preserves the line numbers of the original files by using these:

```
# 9 "macros.c"
```

Next line is treated as line 9 by the compiler and debuggers

C and Files

- Files represented by a *file pointer*
 - Note: the actual representation in Linux is a *file descriptor*, which is a non-negative integer, but that is non-portable; the file pointer is
 - 3 predefined file pointers: stdin, stdout, stderr
- File pointer contains information:
 - Which file is being referenced (ie, the file descriptor)
 - Whether opened for reading, writing, or appending
 - Where in the file the next access is to occur
 - And lots of other information not relevant here