Analysis of ptrstew.c

Matt Bishop
For ECS 36A, Fall Quarter 2019
The Program

#include <stdio.h>

char *c[] = {
    "ENTER",
    "NEW",
    "POINT",
    "FIRST"
};
char **cp[] = { c+3, c+2, c+1, c };
char **cpp = cp;
int main(void)
{
    printf("%s", ***++cpp);
    printf("%s ", **++*++cpp+3);
    printf("%s", *cpp[-2]+3);
    printf("%s\n", cpp[-1][-1]+1);
    return(0);
}

This very short, very confusing program is an excellent exercise in using pointers; if you can figure out what this prints, you will be able to understand (almost) any use of C pointers!

This is from Alan Feuer’s marvelous book *The C Puzzle Book*.
char *c[] = {
    "ENTER",
    "NEW",
    "POINT",
    "POINT",
    "FIRST"
};
char **cp[] = { c+3, c+2, c+1, c };
char ***cpp = cp;
**++c: configuration after ++c**
**+++cpp: configuration after *++cpp; red dashed arrow indicates the dereference (what *++cpp points to)**
+++cpp: configuration after +++cpp; red dashed arrow indicates the dereferences, so +++cpp points to a pointer to “POINT”

So printf(“%s”, +++cpp) prints POINT

What is printed (in blue)

POINT
Configuration after previous printf; note cpp is *not* returned to its original value but remains pointing at the second element of cp (that is, cp[1])

What is printed (in blue)

POINT
*--*++cpp+3: parenthesized, (*(--*(++(cpp)))))+3

This shows the order of precedence

What is printed (in blue)

POINT
*--**+cpp+3: parenthesized, (*(-(*(+cpp)))))+3
After ++(cpp)

What is printed (in blue)

POINT
*--*++cpp+3: parenthesized, (*(--(*(*(++(cpp))))))+3
After *(*++(cpp)); red dashed arrow indicates the dereference (what *++cpp points to)

What is printed (in blue)

POINT
*--*++cpp+3: parenthesized, (*(--(*++(cpp))))) + 3

After --(*++(cpp)); red dashed arrow indicates the dereference (what *++cpp points to)

What is printed (in blue)

POINT
*--*++cpp+3: parenthesized, (*(--(*+(cpp))))) + 3
After *(--(*+(cpp)))); red dashed arrow indicates the dereferences

What is printed (in blue)

POINT
*--*++cpp+3: parenthesized, (*-(*)((+(cpp)))))+3

After *-(*)((+(cpp)))))+3; red dashed arrow indicates the dereferences, the purple arrow after the “+3”, so this points to “ER”

So printf(“%s”, *--*++cpp+3) prints ER_
where _ represents a blank

What is printed (in blue)
POINTER_
Configuration after previous printf; note cpp is not returned to its original value but remains pointing at the third element of cp (that is, cp[2])

What is printed (in blue): POINTER_
*cpp[-2]+3: parenthesized, (*cpp[-2])+3
Red dashed arrow shows cpp[-2]  

What is printed (in blue)

POINTER_
*cpp[-2]+3: parenthesized, (*cpp[-2])+3
Red dashed arrow shows *(cpp[-2])

What is printed (in blue)

POINTER_
*cpp[-2]+3: parenthesized, (*cpp[-2])+3
Purple arrow shows (*cpp[-2])+3

So printf(“%s”, *cpp[-2] + 3) prints ST

What is printed (in blue)

POINTER_ST
cpp[-1][-1]+1: red arrow shows deference of cpp[-1]

What is printed (in blue)

POINTER_ST
cpp[-1][-1]+1: red arrow shows deference of cpp[-1][-1]

What is printed (in blue)

POINTER_ST
cpp[-1][-1]+1: this shown the final result, with the purple arrow after the “+1”, so this points to “ER”

So printf("%s ", cpp[-1][-1]+1) prints EW

What is printed (in blue)

POINTER_STEW