ECS 36A, April 19, 2023

How to Copy to/from the CSIF

- Download file *f* to CSIF:
  1. Activate the VPN that connects you to the CSIF (like Pulse)
  2. Give the following command on *your* computer:

     ```
     scp f pc12.cs.ucdavis.edu:
     ```

  The file *f* is now in your CSIF home directory
- Upload file *f* from CSIF:
  1. Activate the VPN that connects you to the CSIF (like Pulse)
  2. Give the following command on *your* computer:

     ```
     scp pc12.cs.ucdavis.edu:f .
     ```

  The file *f* is now in the current working directory/folder on your system

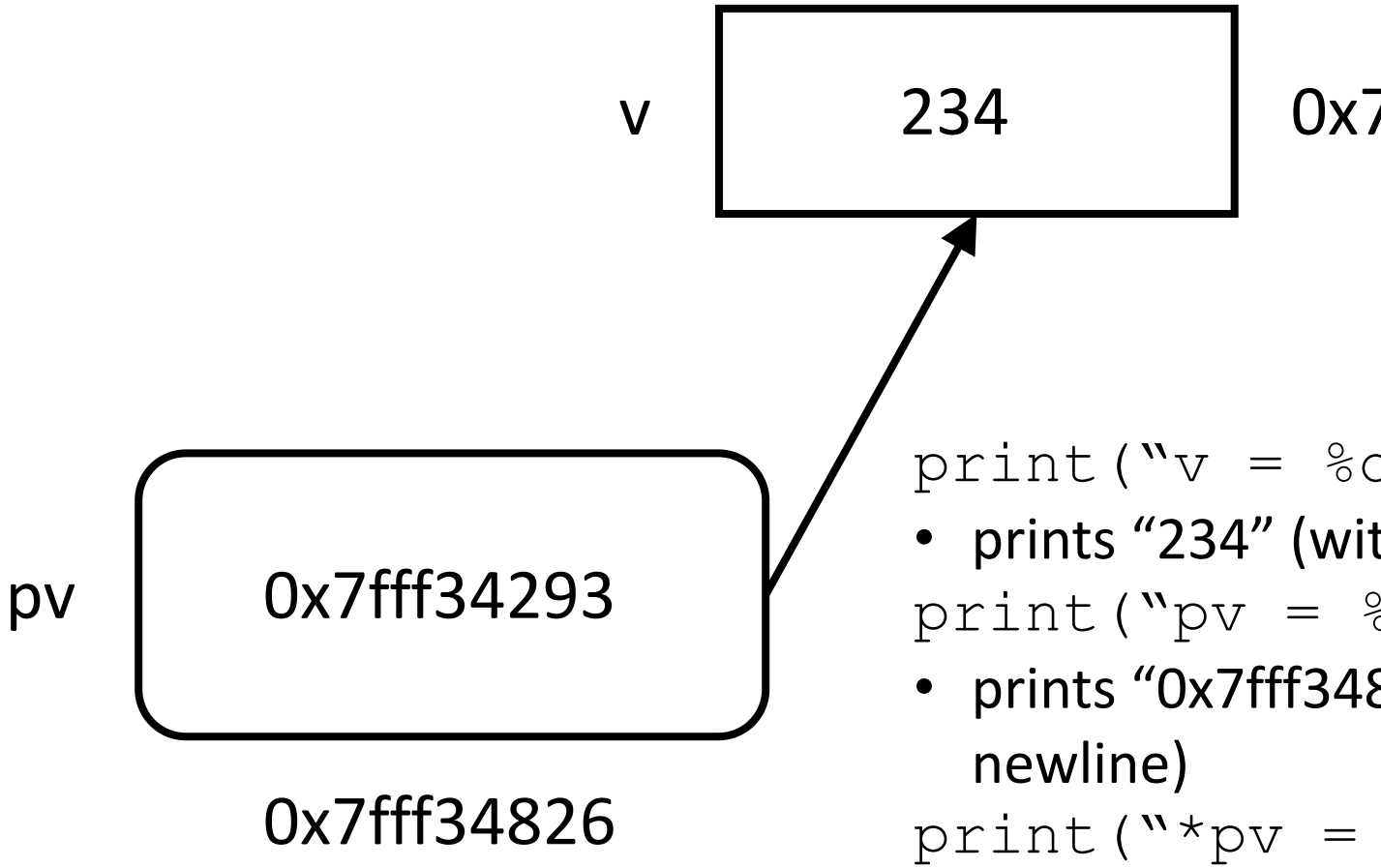
Pointers

- A variable containing the address of another variable
- Example:

```
int x = 0;
int *px;
px = &x;
printf("x = %d, px = %p, *px = %d\n", x, (void *)px, *px);
```

- Operators:
  - &*variable*: address of *variable*
  - **variable*: what is in the memory location with the address stored in *variable*

In Pictures




```
print("v = %d\n", v);
```

- prints "234" (without the "s, ending in newline)

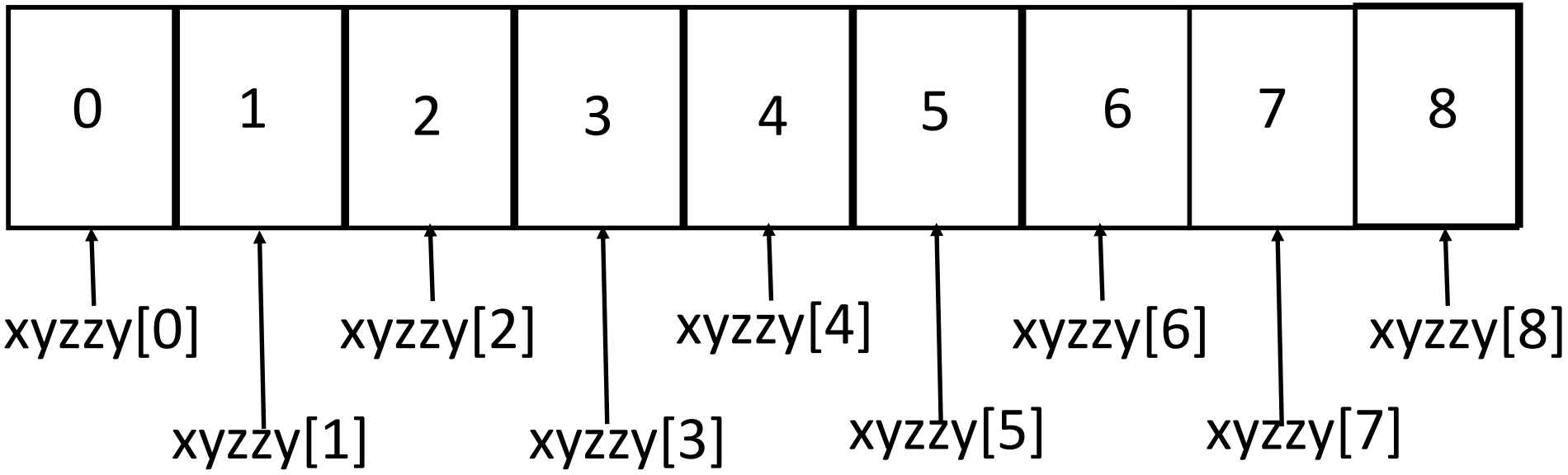
```
print("pv = %p\n", (void *)pv);
```

- prints "0x7fff34826" (without the "s, ending in newline)

```
print("*pv = %d\n", *py);
```

- prints "234" (without the "s, ending in newline)

C Arrays


0
1
2
3
4
5
6
7
8
xyzzy[0]
xyzzy[1]
xyzzy[2]
xyzzy[3]
xyzzy[4]
xyzzy[5]
xyzzy[6]
xyzzy[7]
xyzzy[8]

Arrays as Pointers and *Vice Versa*

- Arrays are simply another way to express pointers
  - So xarray[0] and *xarray refer to the same memory location
  - And xarray[12] and *(xarray+12) refer to the same memory location

Initializations

- Initializing an array

```
int iarr[5] = { 1, 2, 3, 4, 5 };
```

  or

```
int iarr[] = { 1, 2, 3, 4, 5};
```

- Initializing a pointer

```
int ivar;
int *iptr = &ivar;
```

Strings

- An array of characters terminated with a 0 byte
  - 0 byte is a byte with all bits set to 0; also called a NUL byte
  - You can use either an array or a pointer
- Examples:

```
char carr[6] = { 'h', 'e', 'l', 'l', 'o', '\0' };
char carr[6] = { 'h', 'e', 'l', 'l', 'o', '\0' };
char *cstr = "hello";
```

  - For the last, when a string (in "…") ends, the compiler adds a NUL byte

A Warning

- You want to make a copy of a string

```
char *cstr = "hello";
```

- Do *not* do this:

```
cdupstr = cstr;
```

- If `cdupstr` is a pointer, this simply copies the *pointer*, so `cdupstr` and `cstr` point to the same string; if it's an array, you get an error

Doing It Right

- You want to make a copy of a string

```
char *cstr = “hello”;
char cdupstr[100];
```

  - Be sure `cdupstr` is an array with enough room to hold “hello” *plus the trailing NUL byte!*
- This works:

```
(void) strcpy(cdupstr, cstr);
```

- But this is better!

```
(void) strncpy(cdupstr, cstr, 99);
cdupstr[99] = ‘\0’;
```

Reading a Line of Input

- Use fgets(buf, n, stdin)
  - On success, returns address of buf
  - On failure or EOF, if nothing has been read, returns a NULL pointer; otherwise, it returns all the characters read up to that error or the end of file
- Example use:

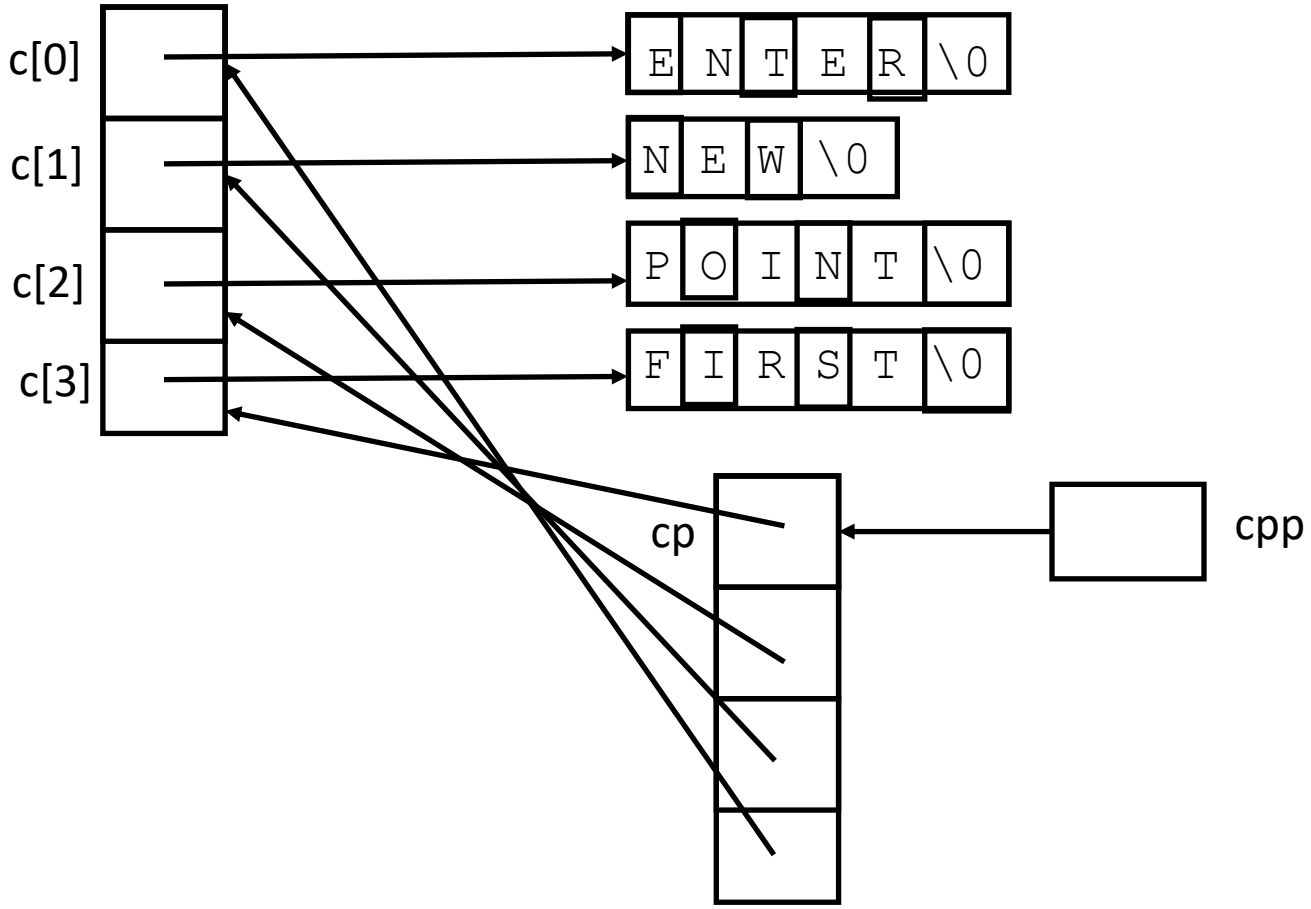
```
if (fgets(buf, 100, stdin) == NULL){
   fprintf(stderr, "Bad input\n"); . . .
```

  - If there is a new line, it reads up to that and *then*appends the '\0' byte
- Another way (but do *not* do this!)

```
if (gets(buf) == NULL){ fprintf(stder, "Bad input\n"); . . . }
```

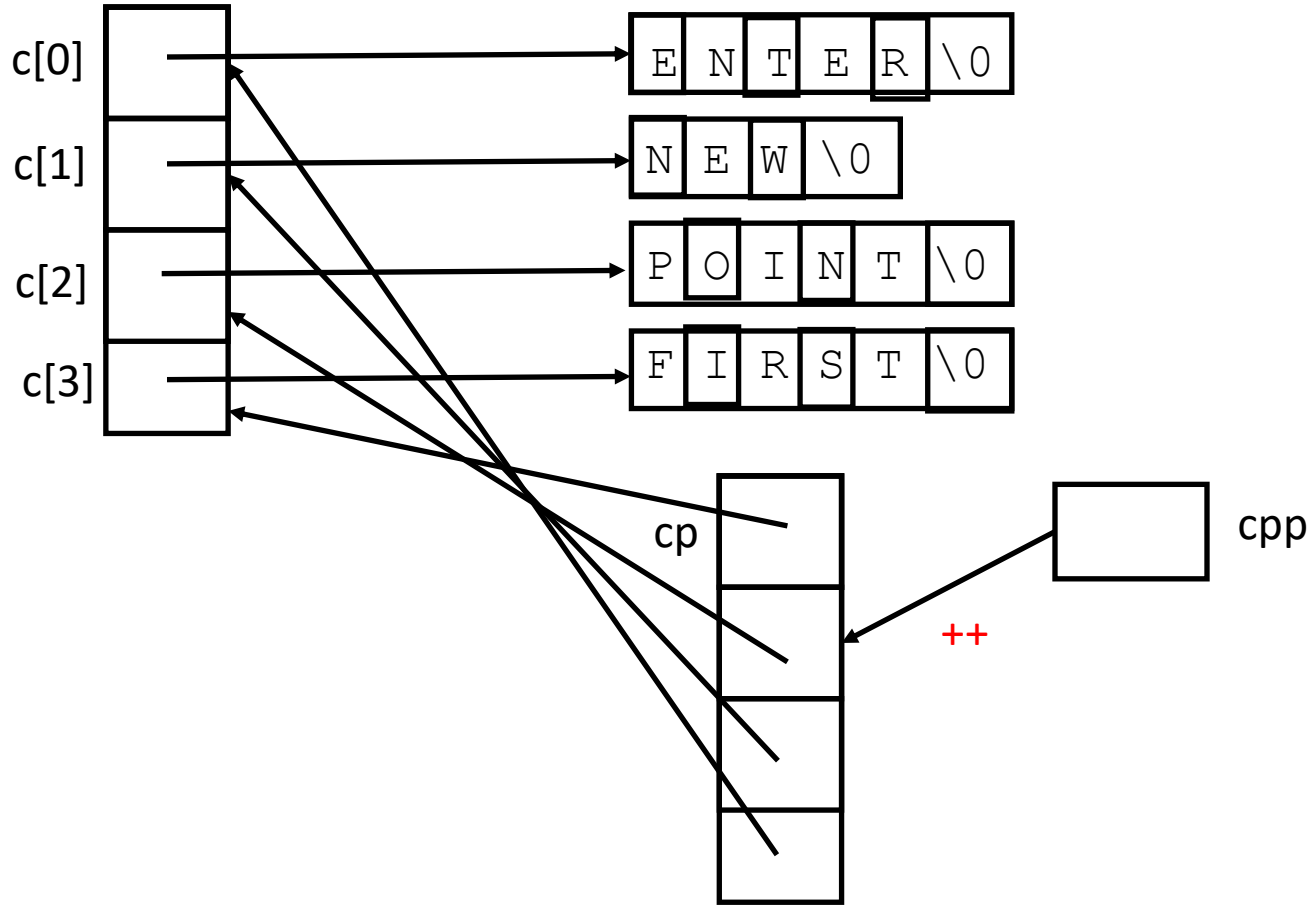
Pointer Stew (Initially)

```
#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);
}
```



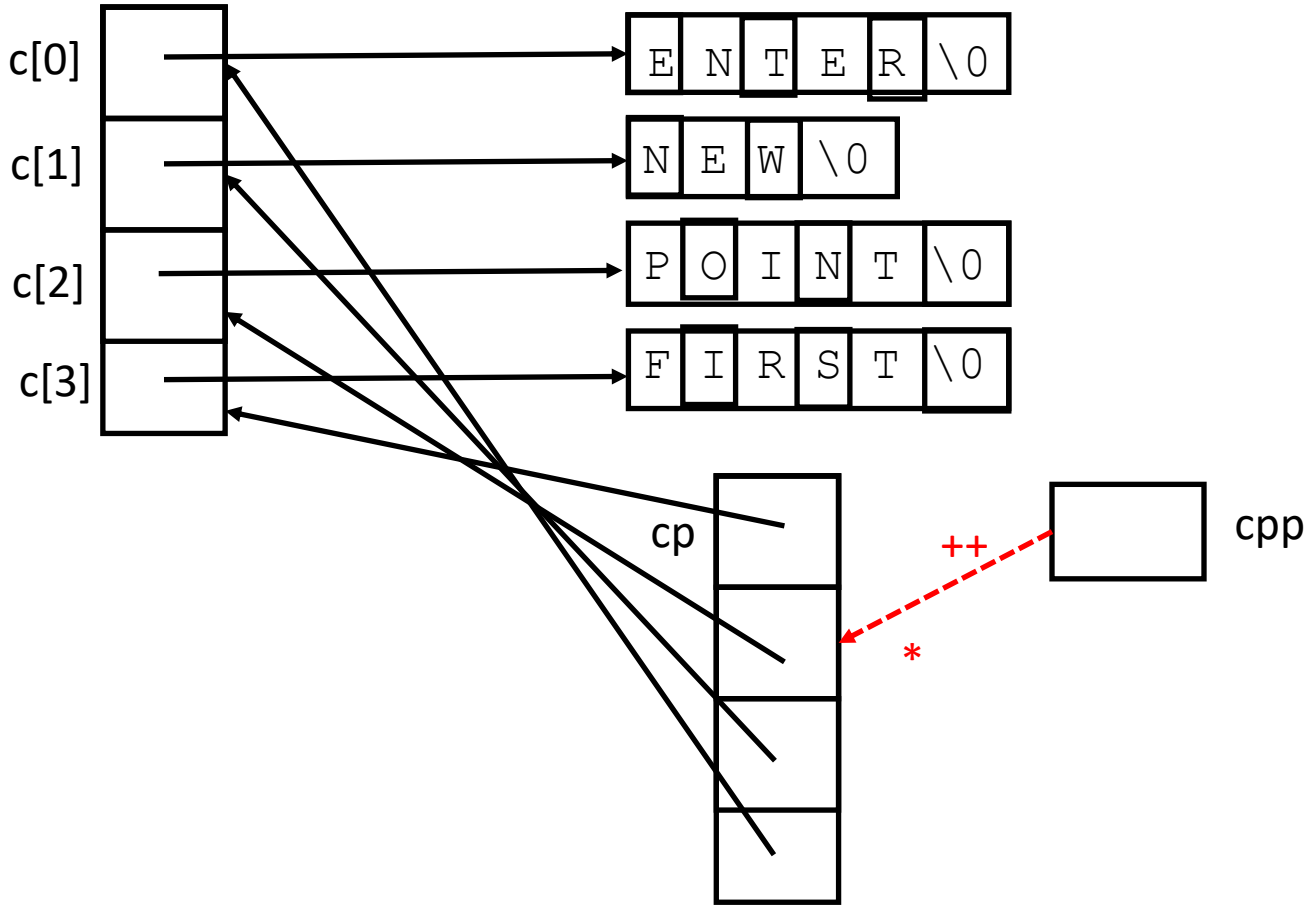
Pointer Stew (++cpp)

```
#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);
}
```



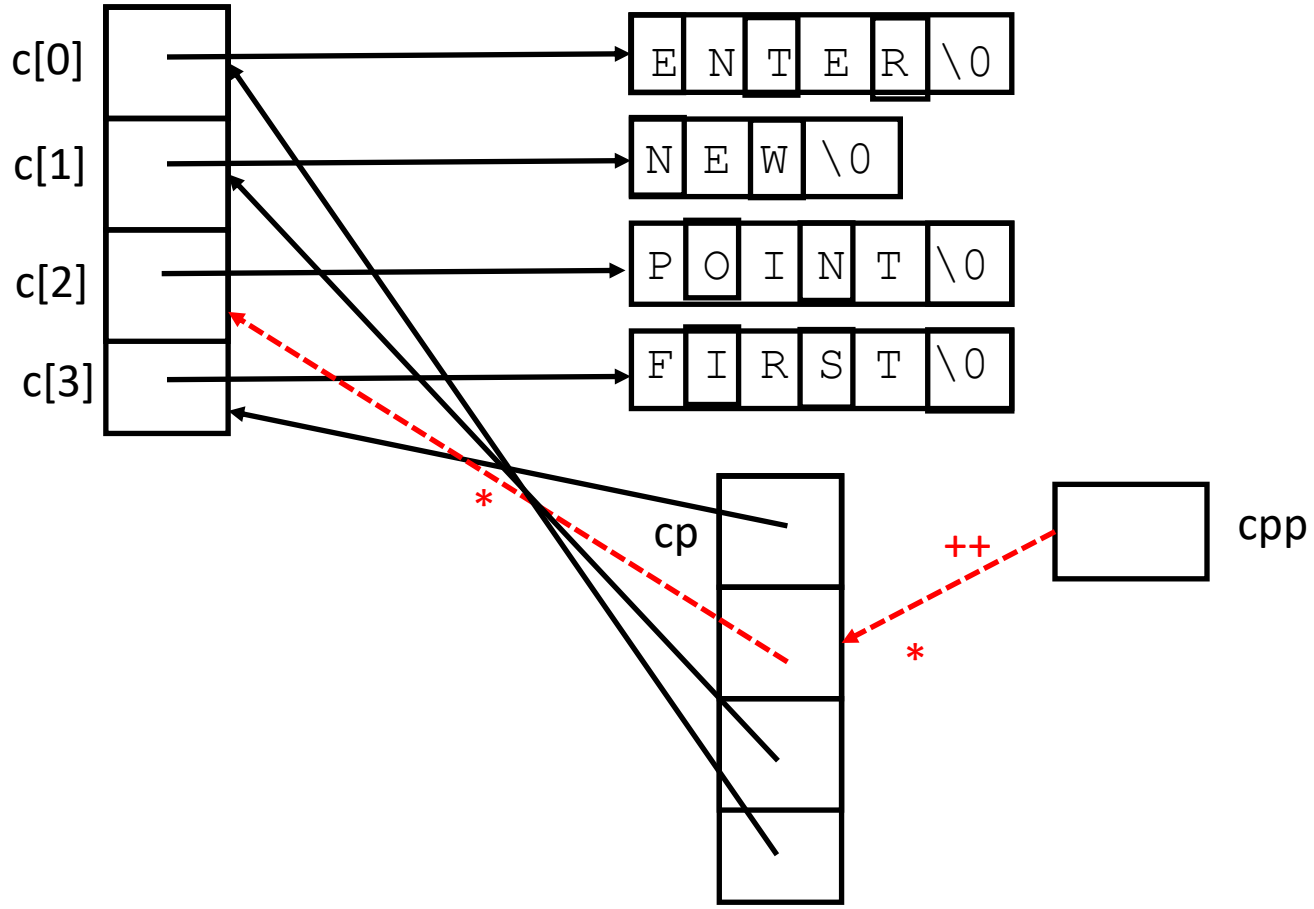
Pointer Stew (*++ccp)

```
#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);
}
```



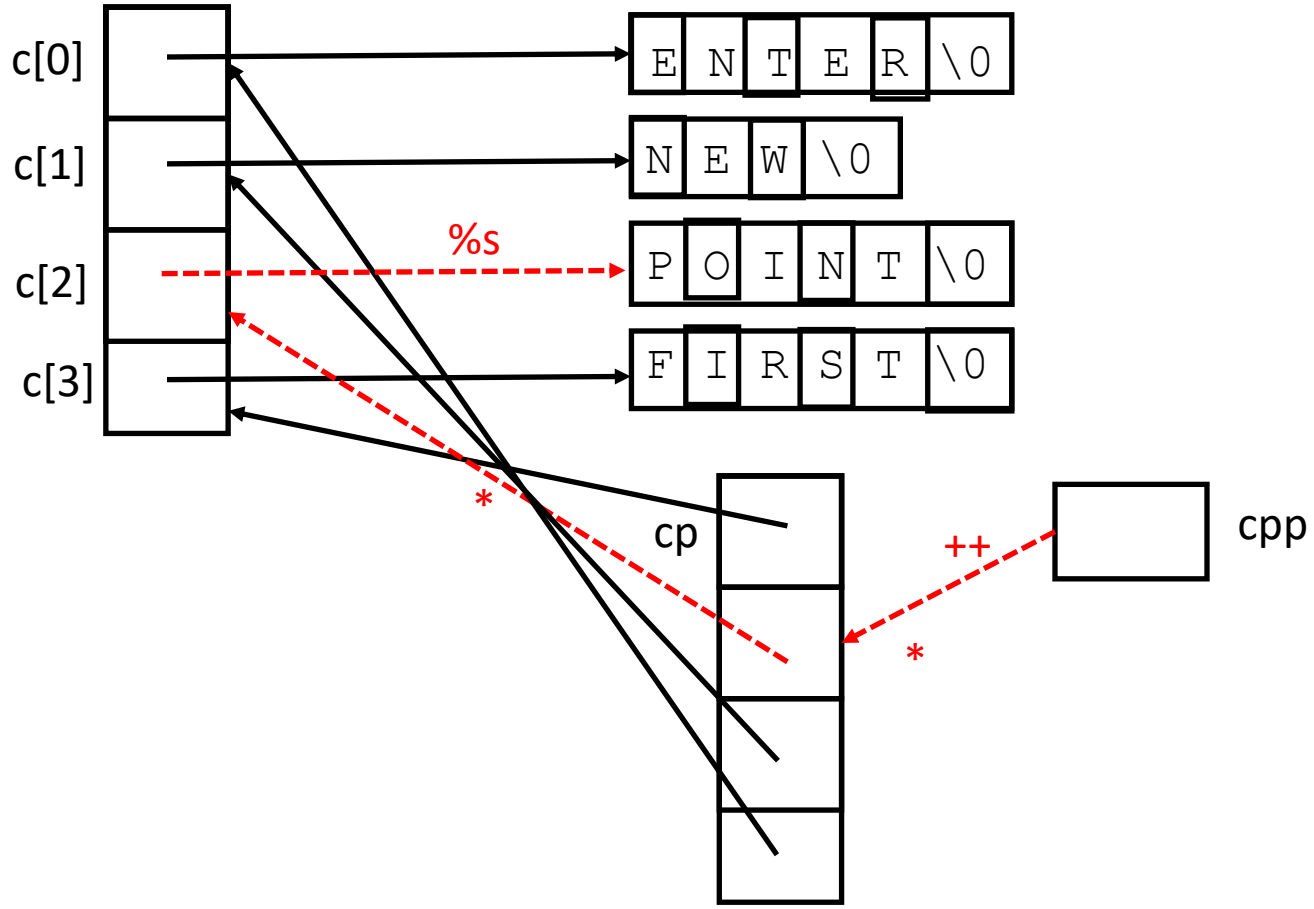
Pointer Stew (**++cpp)

```
#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);
}
```



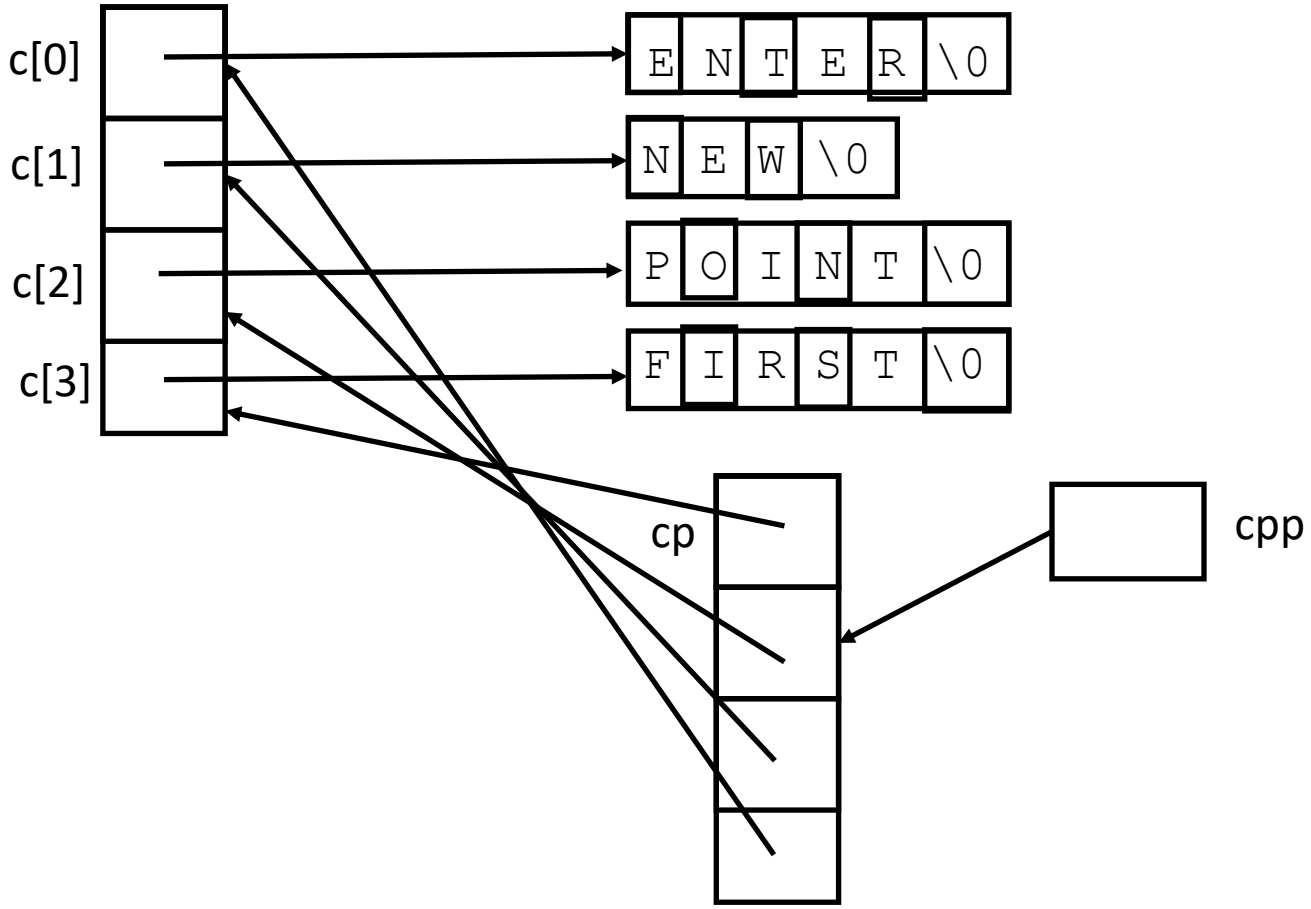
Pointer Stew (End of First *printf)*

```
#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);
}
```



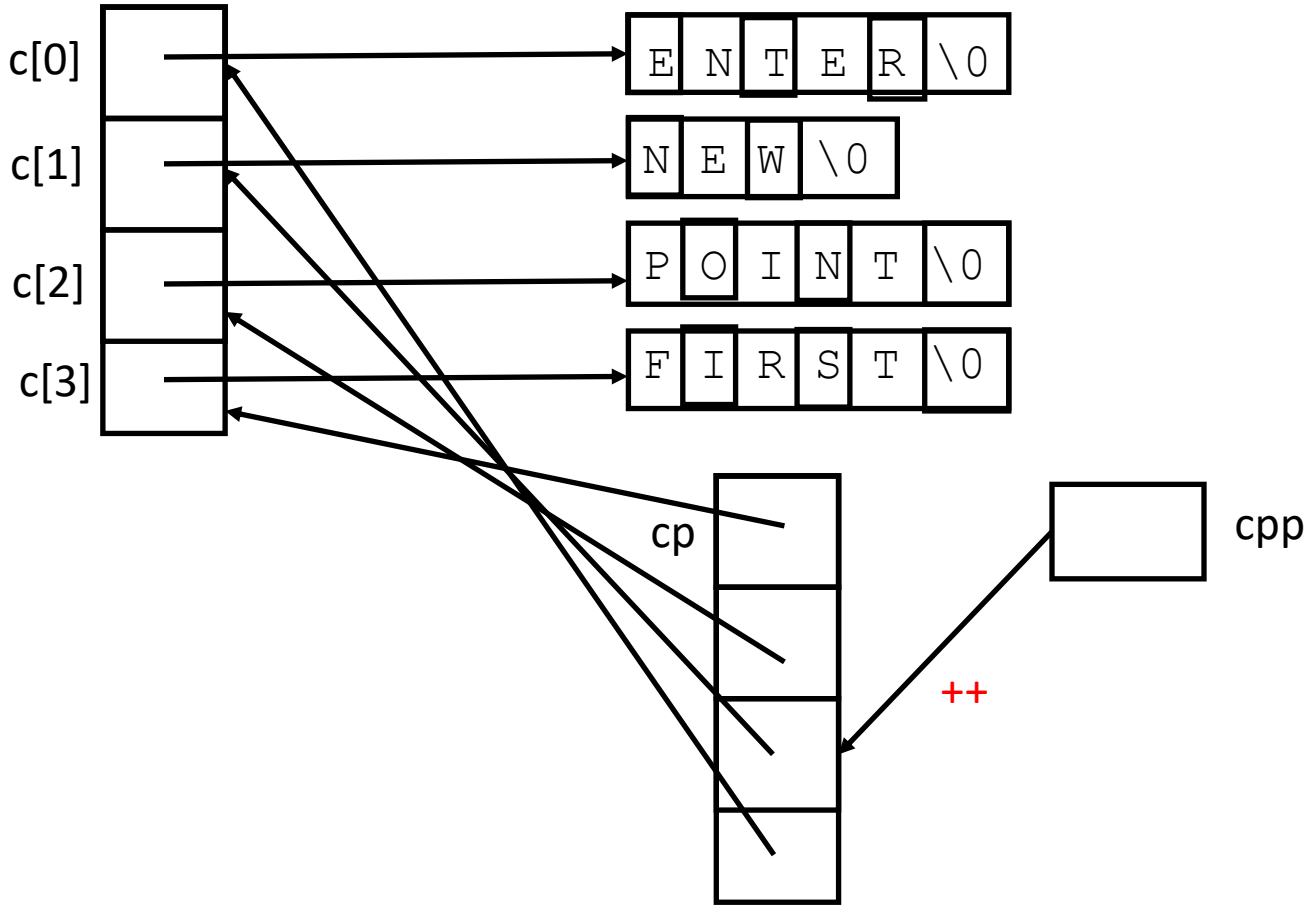
Pointer Stew (Current State of Variables)

```
#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);
}
```



Pointer Stew (++cpp)

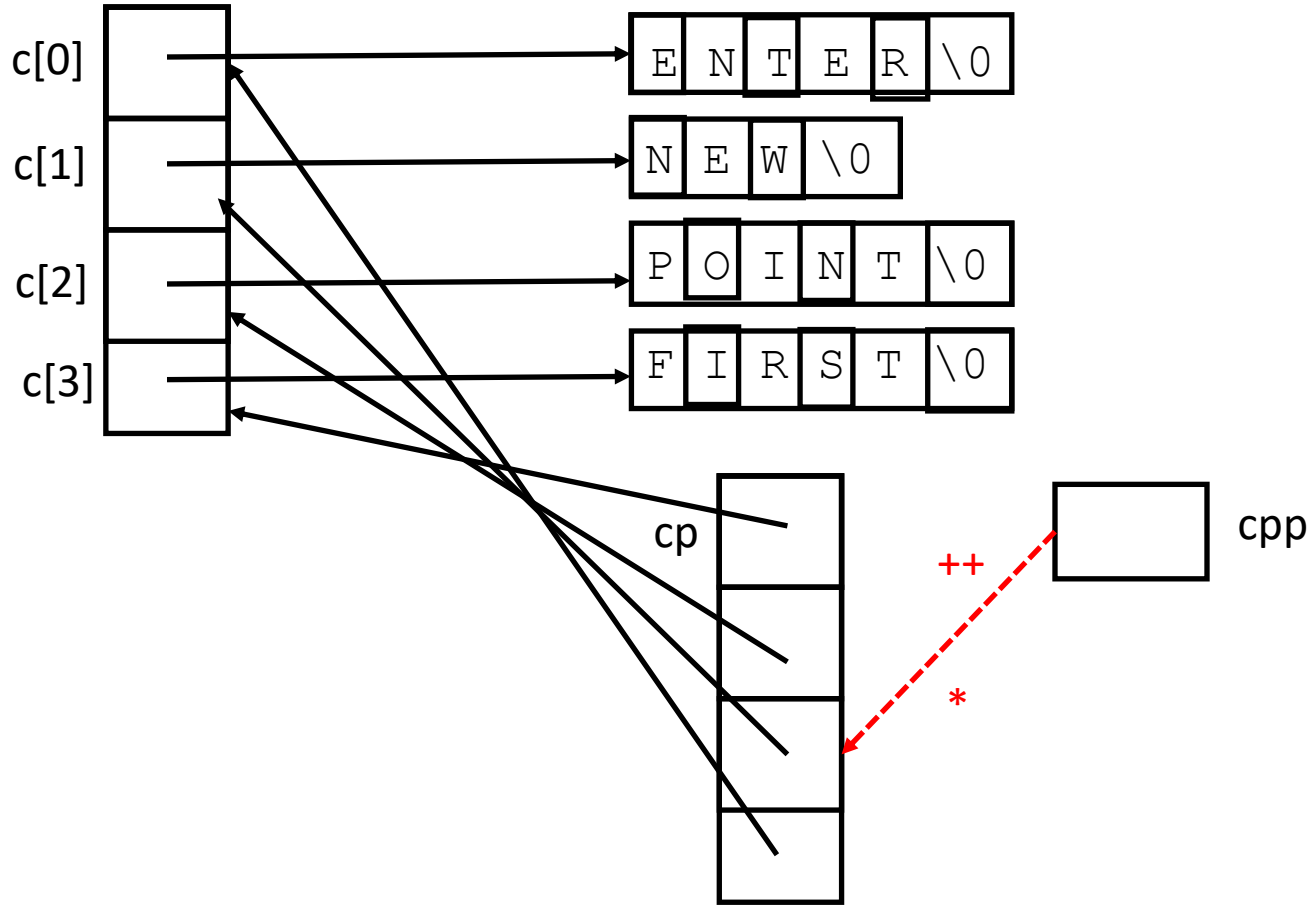
```
#include <stdio.h>
char *c[] = {
    "ENTER",
    "NEW",
    "POINT",
    "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
    printf("%s", **++cpp );
    printf("%s ", *--*++cpp+3 );
    printf("%s", *cpp[-2]+3 );
    printf("%s\n", cpp[-1][-1]+1 );
    return(0);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



POINT

Pointer Stew (*++cpp)

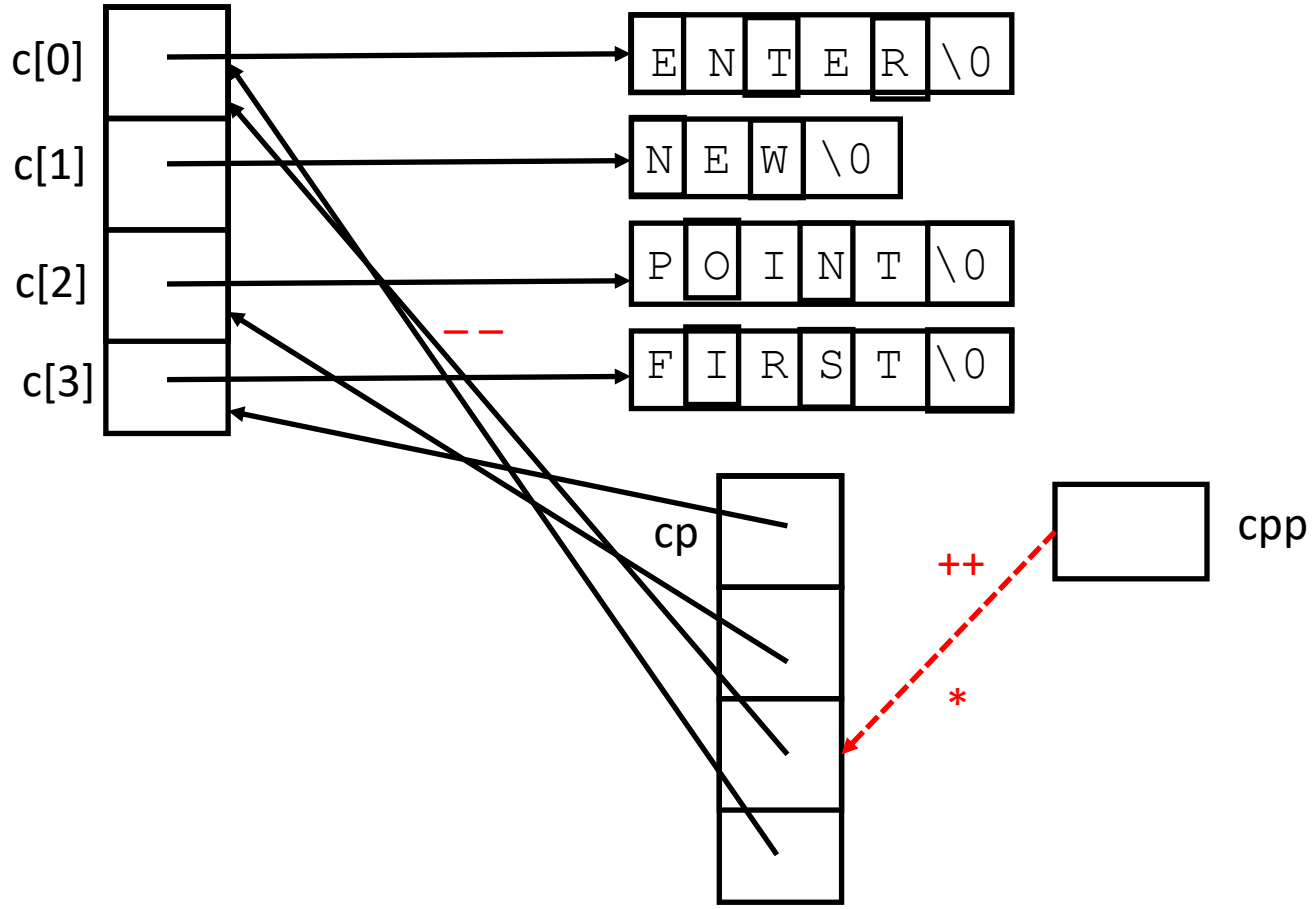
```
#include <stdio.h>
char *c[] = {
    "ENTER",
    "NEW",
    "POINT",
    "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
    printf("%s", **++cpp );
    printf("%s ", *--*++cpp+3 );
    printf("%s", *cpp[-2]+3 );
    printf("%s\n", cpp[-1][-1]+1 );
    return(0);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



POINT

Pointer Stew (– –*++cpp)

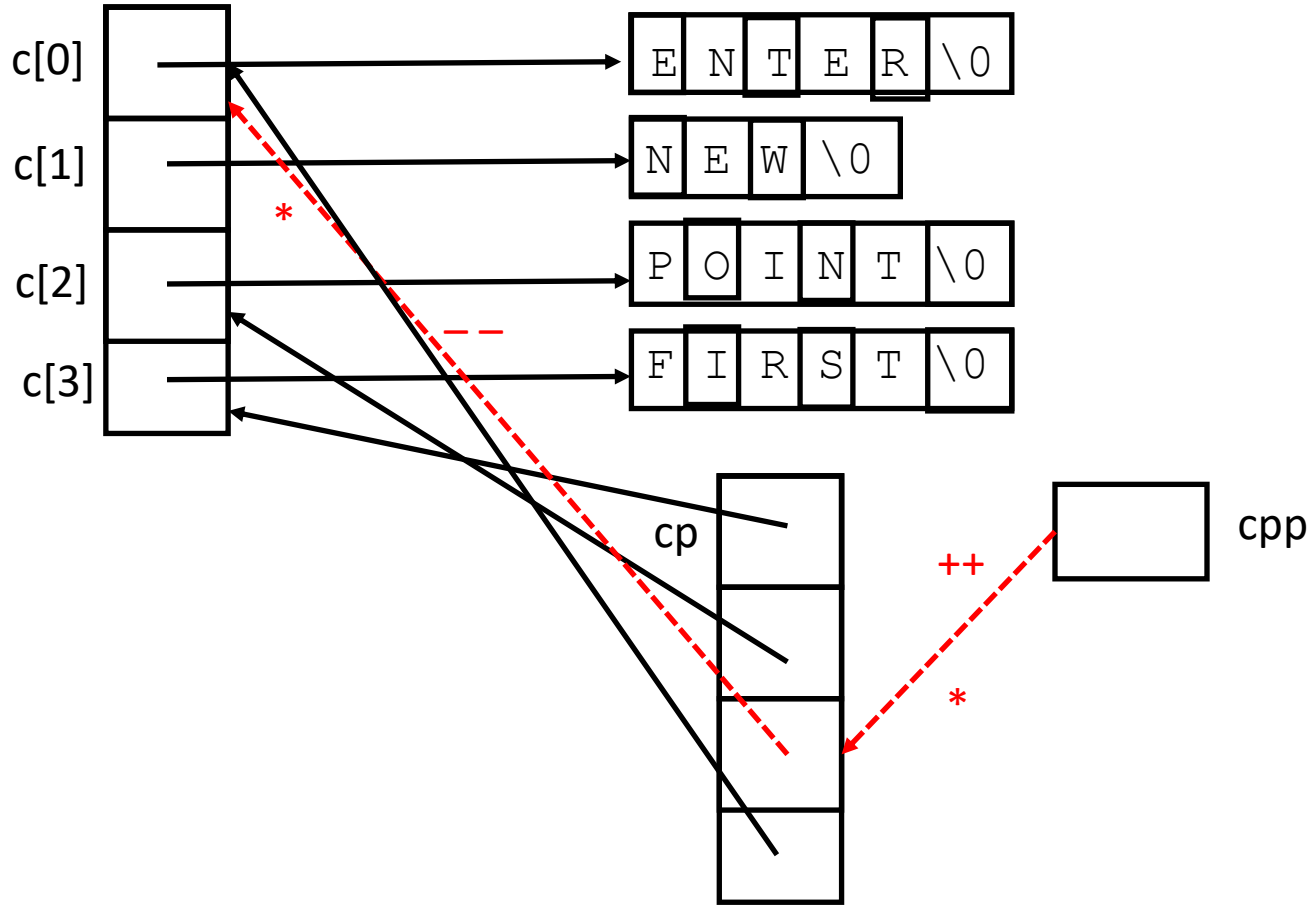
```
#include <stdio.h>
char *c[] = {
    "ENTER",
    "NEW",
    "POINT",
    "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
    printf("%s", **++cpp );
    printf("%s ", *--*++cpp+3 );
    printf("%s", *cpp[-2]+3 );
    printf("%s\n", cpp[-1][-1]+1 );
    return(0);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



POINT

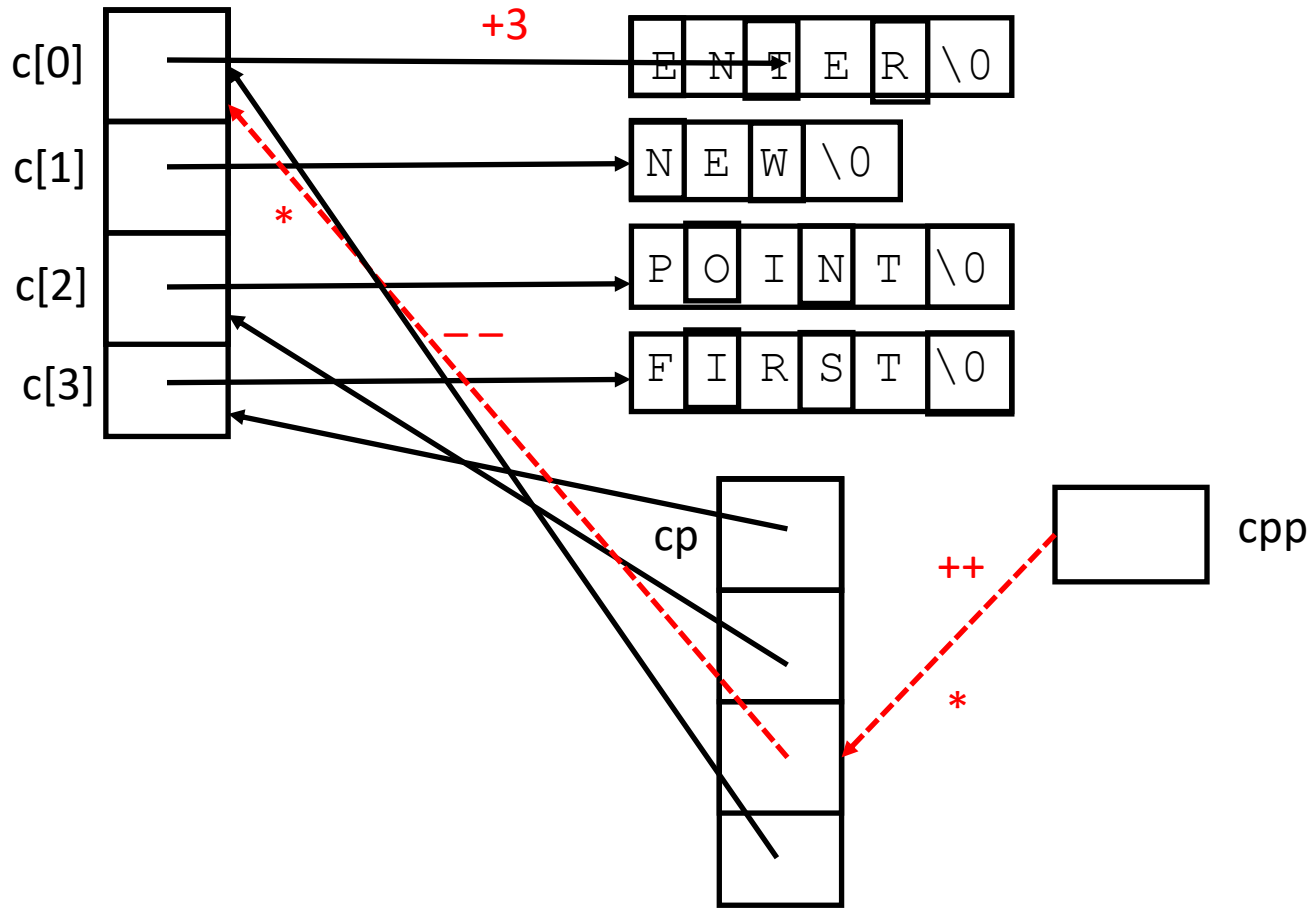
Pointer Stew (*– –*++cpp)

```
#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);
}
```



Pointer Stew (*– –*++cpp+3)

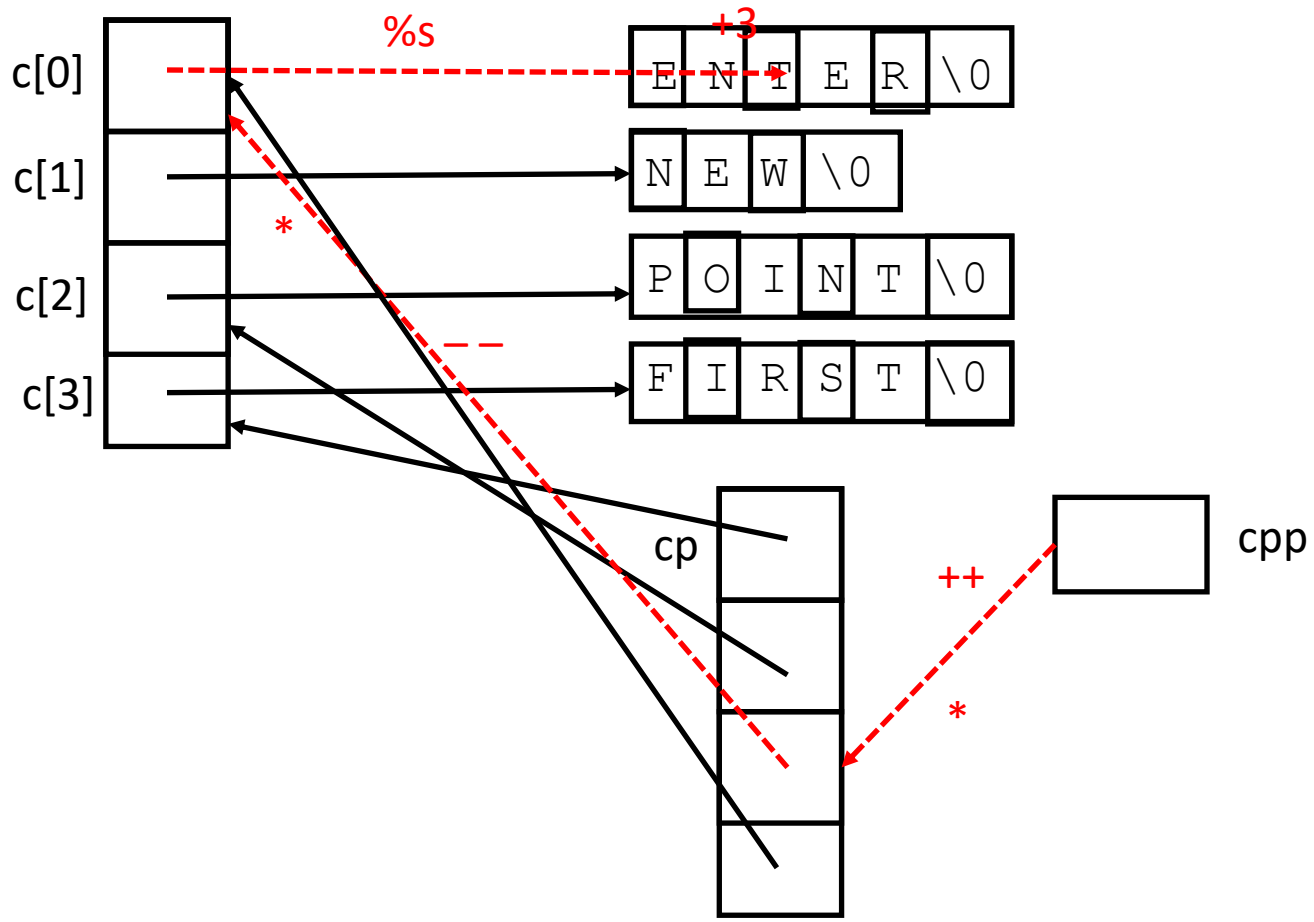
```
#include <stdio.h>
char *c[] = {
     "ENTER",
     "NEW",
     "POINT",
     "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
     printf("%s", **++cpp );
     printf("%s ", *--*++cpp+3 );
     printf("%s", *cpp[-2]+3 );
     printf("%s\n", cpp[-1][-1]+1 );
   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



POINTER_

Pointer Stew (End of Second *printf*)

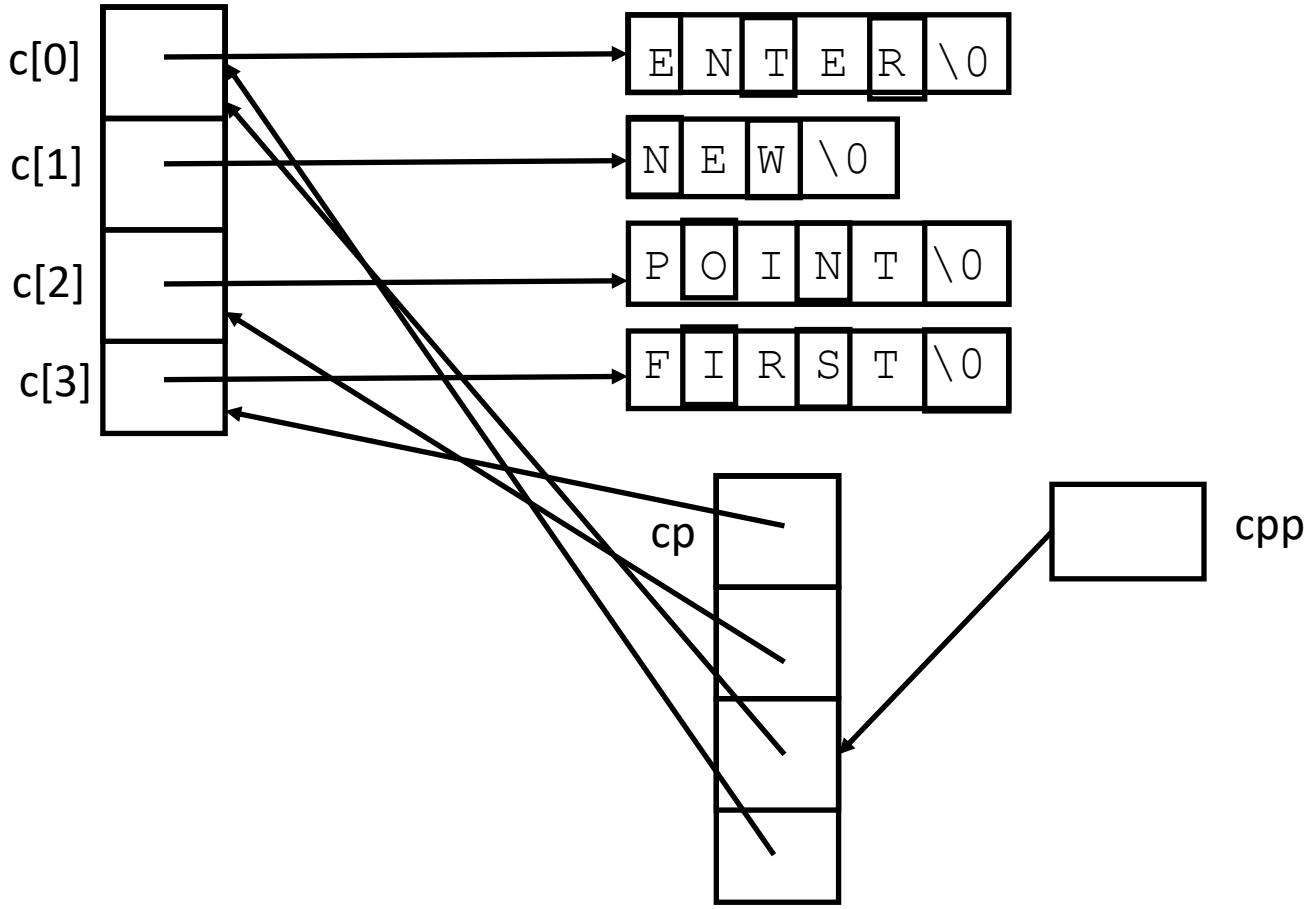
```
#include <stdio.h>
char *c[] = {
     "ENTER",
     "NEW",
     "POINT",
     "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
     printf("%s", **++cpp );
     printf("%s ", *--*++cpp+3 );
     printf("%s", *cpp[-2]+3 );
     printf("%s\n", cpp[-1][-1]+1 );
   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



POINTER_

Pointer Stew (Current State of Variables)

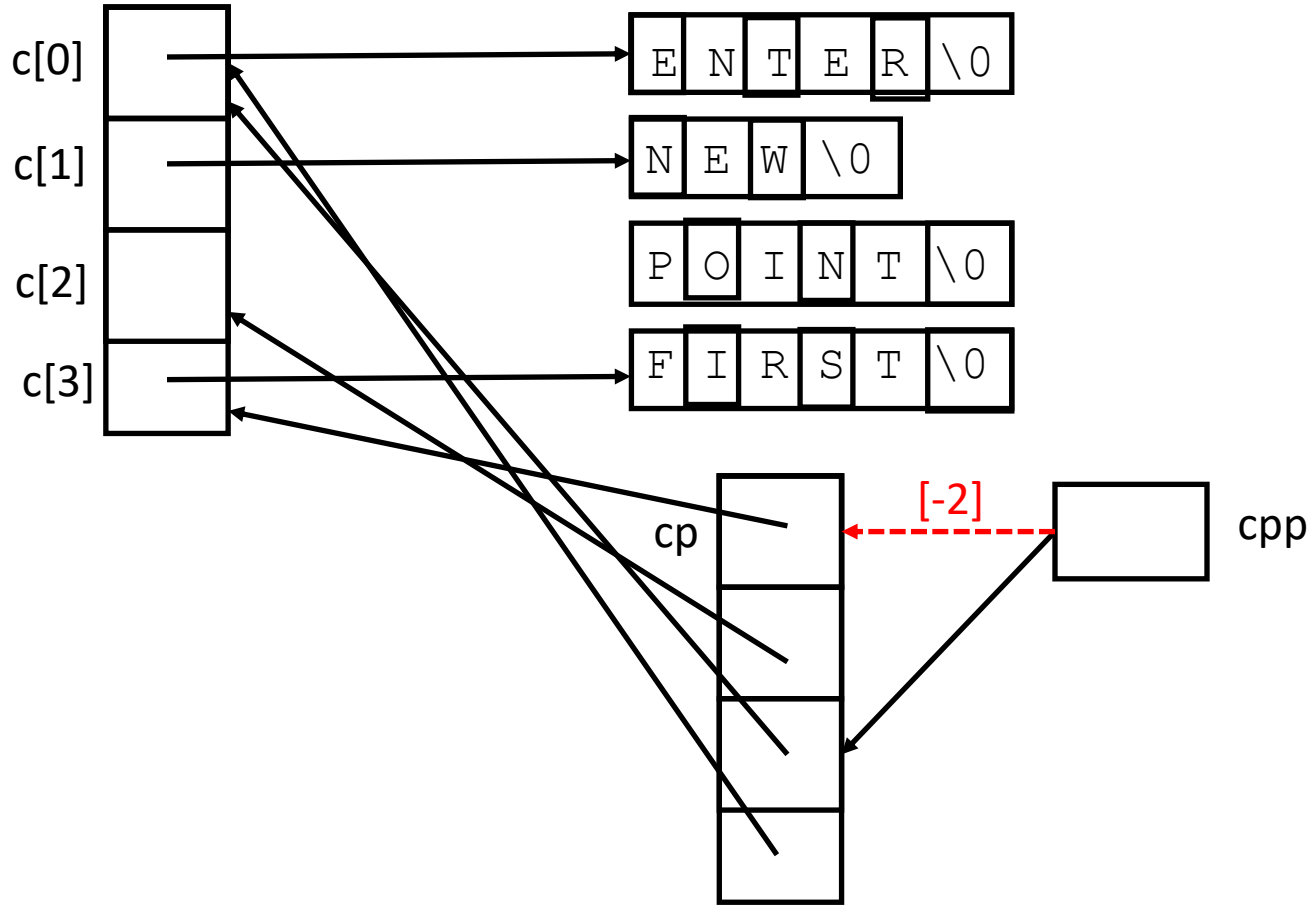
```
#include <stdio.h>
char *c[] = {
    "ENTER",
    "NEW",
    "POINT",
    "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
    printf("%s", **++cpp );
    printf("%s ", *--*++cpp+3 );
    printf("%s", *cpp[-2]+3 );
    printf("%s\n", cpp[-1][-1]+1 );
    return(0);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



POINTER_

Pointer Stew (cpp[–2])

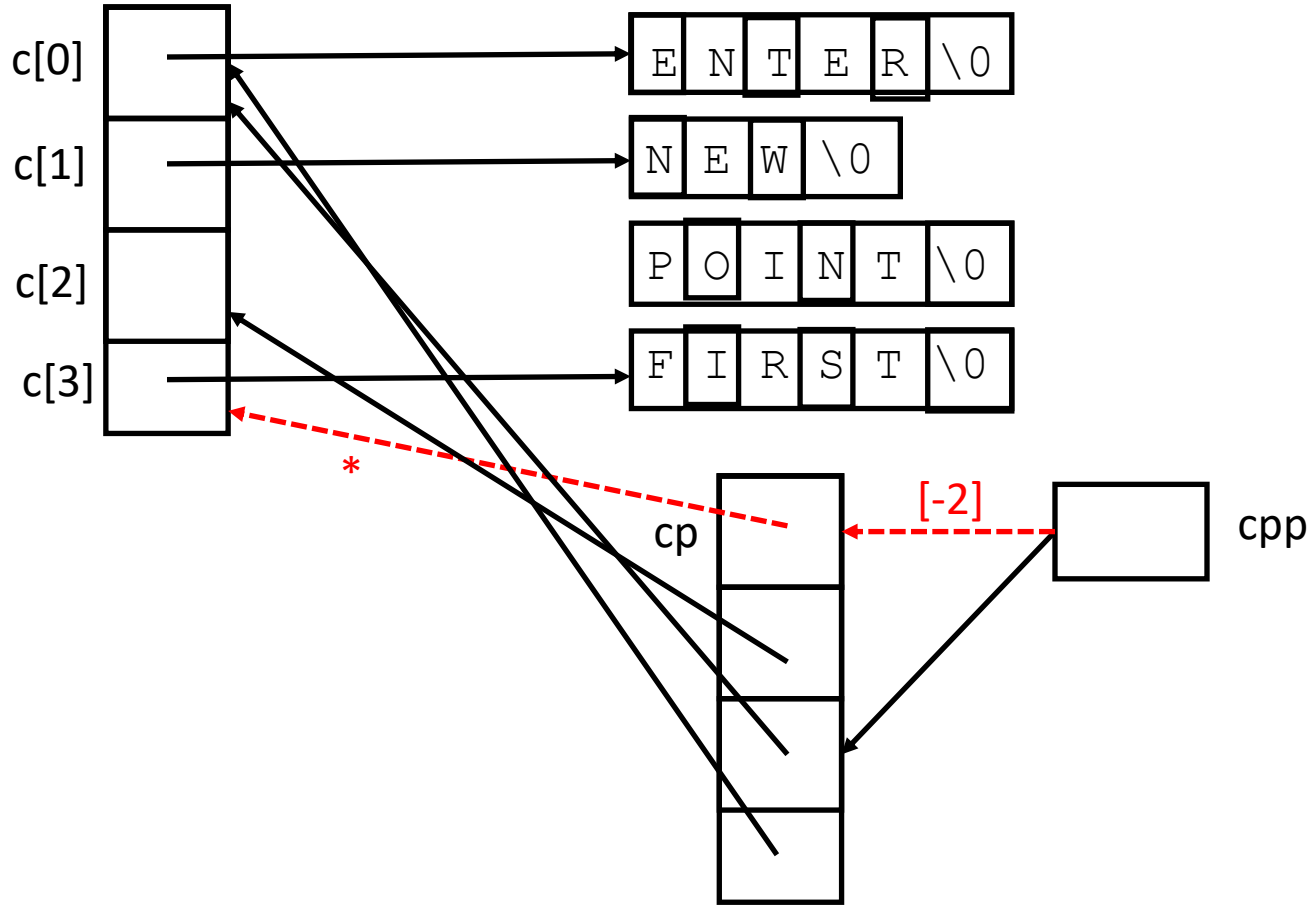
```
#include <stdio.h>
char *c[] = {
    "ENTER",
    "NEW",
    "POINT",
    "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
    printf("%s", **++cpp );
    printf("%s ", *--*++cpp+3 );
    printf("%s", *cpp[-2]+3 );
    printf("%s\n", cpp[-1][-1]+1 );
    return(0);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



POINTER_

Pointer Stew (*cpp[-2])

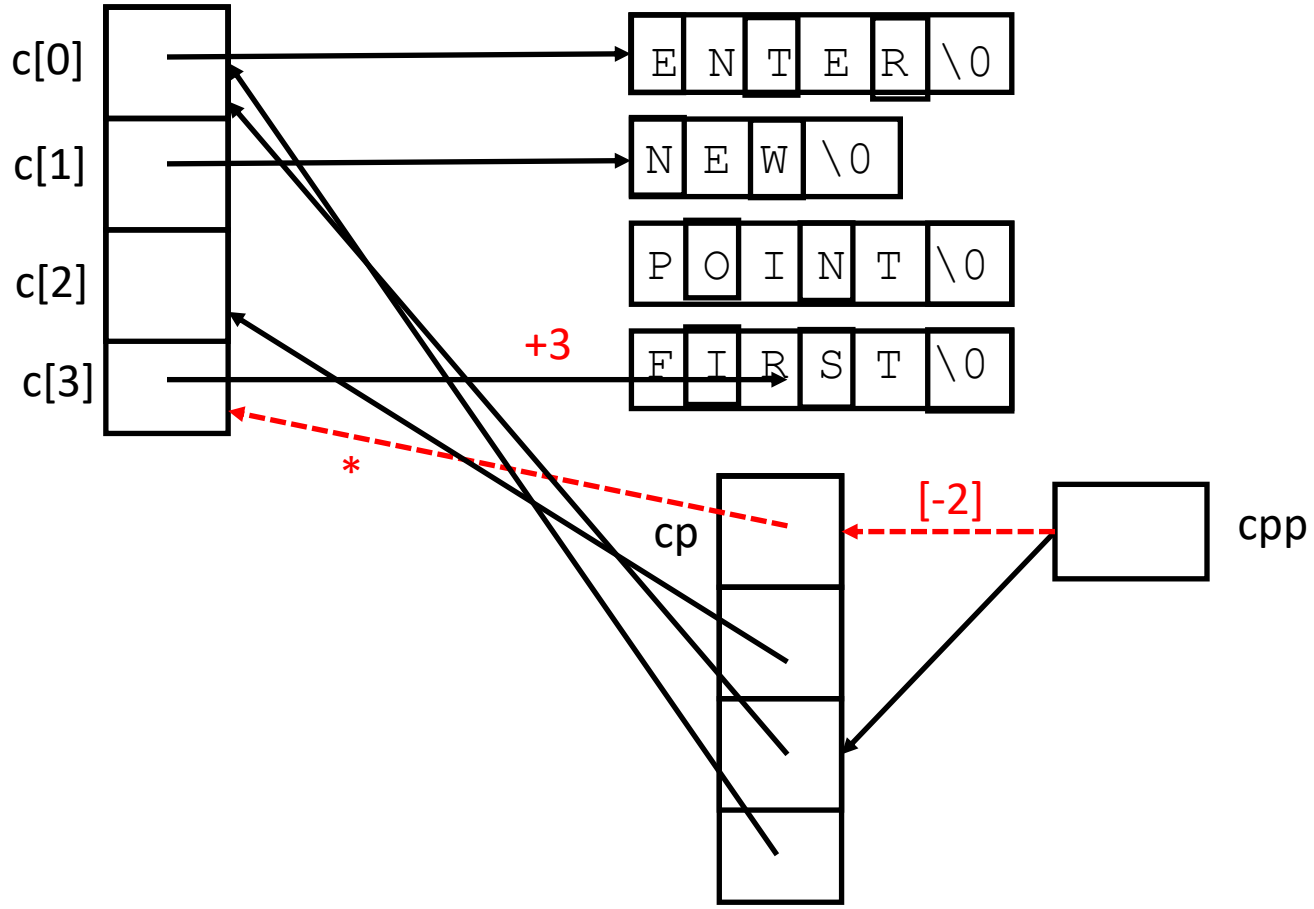
```
#include <stdio.h>
char *c[] = {
    "ENTER",
    "NEW",
    "POINT",
    "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
    printf("%s", **++cpp );
    printf("%s ", *--*++cpp+3 );
    printf("%s", *cpp[-2]+3 );
    printf("%s\n", cpp[-1][-1]+1 );
    return(0);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



POINTER_

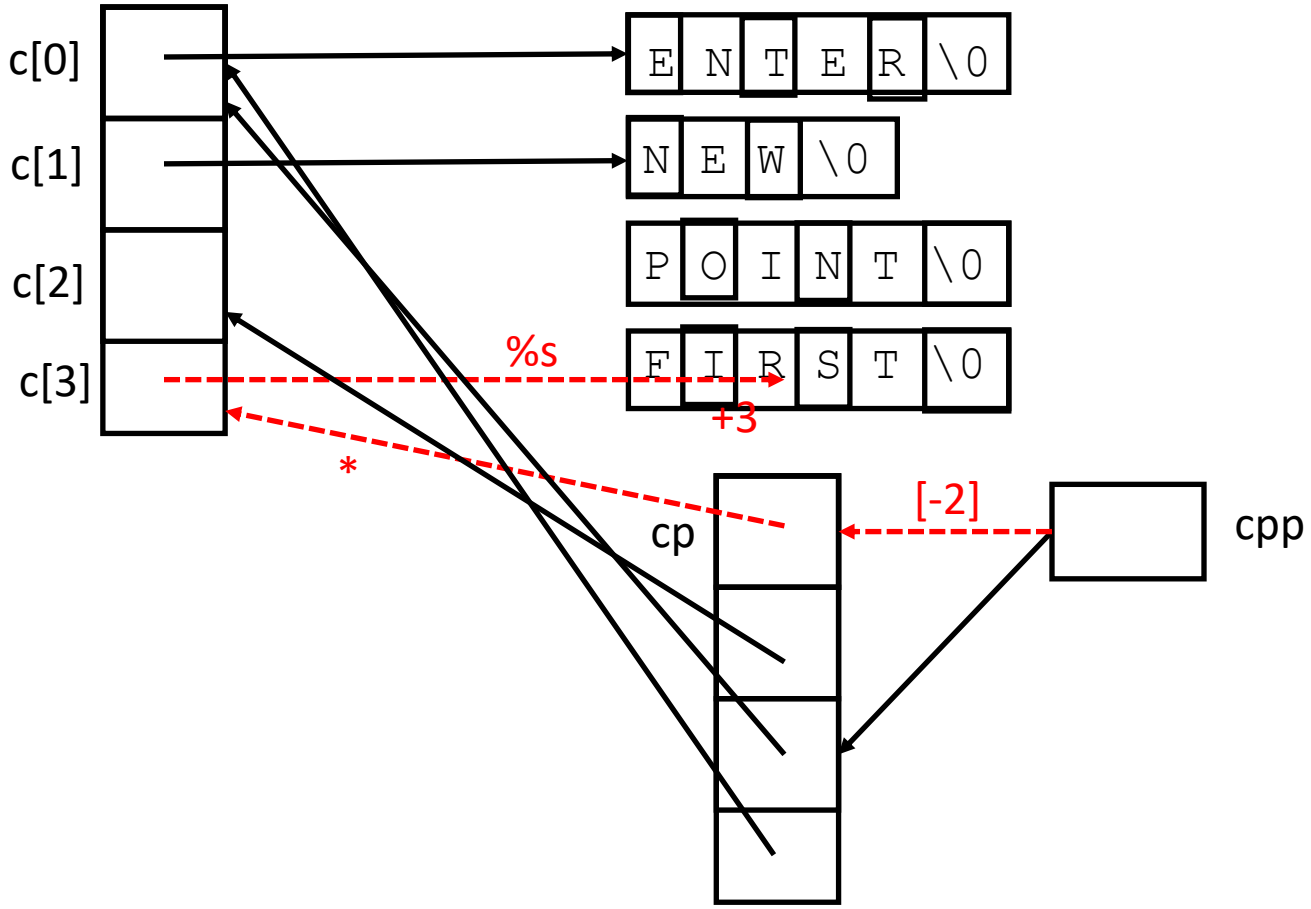
Pointer Stew (*cpp[–2] + 3)

```
#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);
}
```



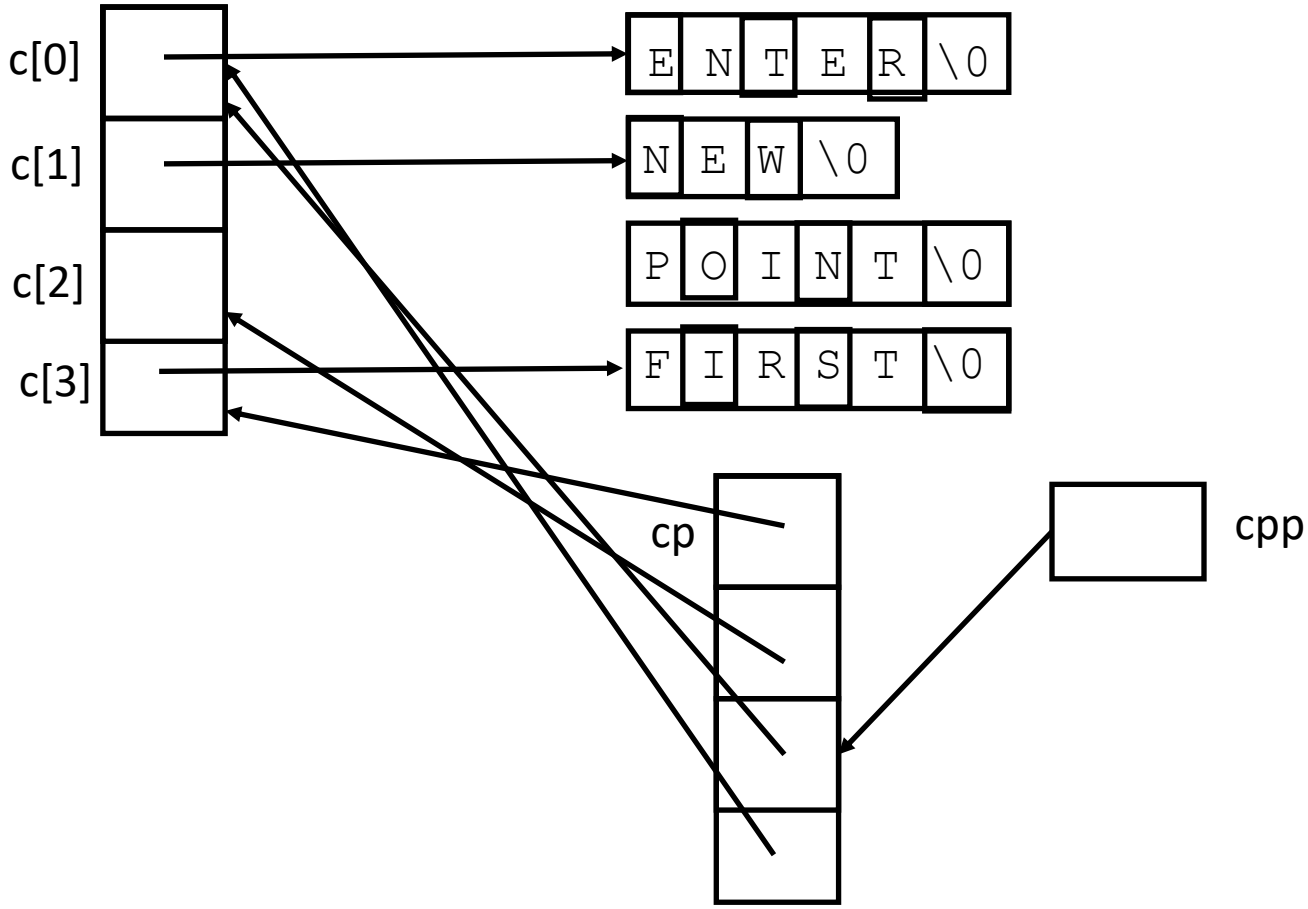
Pointer Stew (End of Third *printf*)

```
#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);
}
```



Pointer Stew (Current State of Variables)

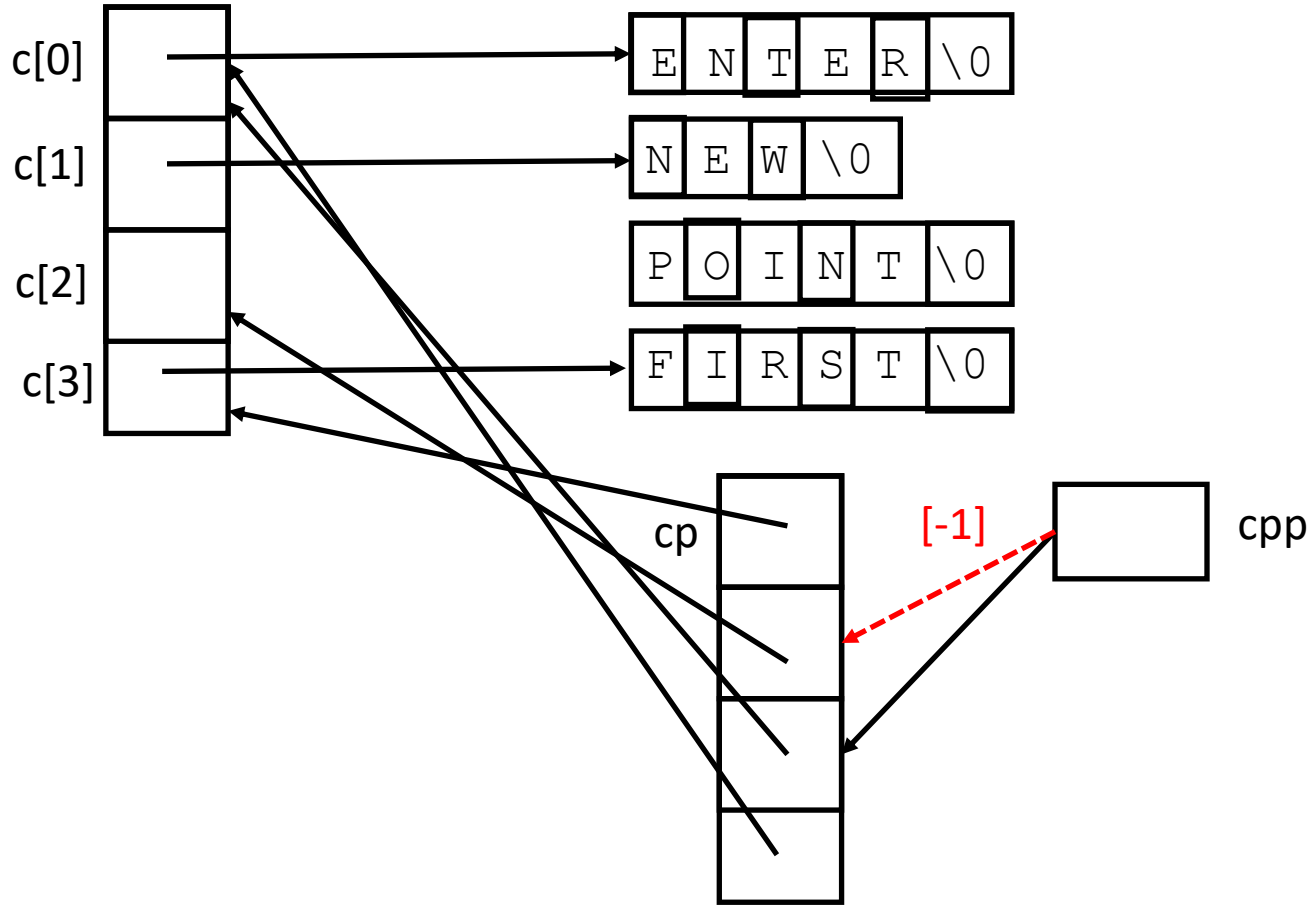
```
#include <stdio.h>
char *c[] = {
    "ENTER",
    "NEW",
    "POINT",
    "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
    printf("%s", **++cpp );
    printf("%s ", *--*++cpp+3 );
    printf("%s", *cpp[-2]+3 );
    printf("%s\n", cpp[-1][-1]+1 );
    return(0);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



POINTER_ST

Pointer Stew (cpp[–1])

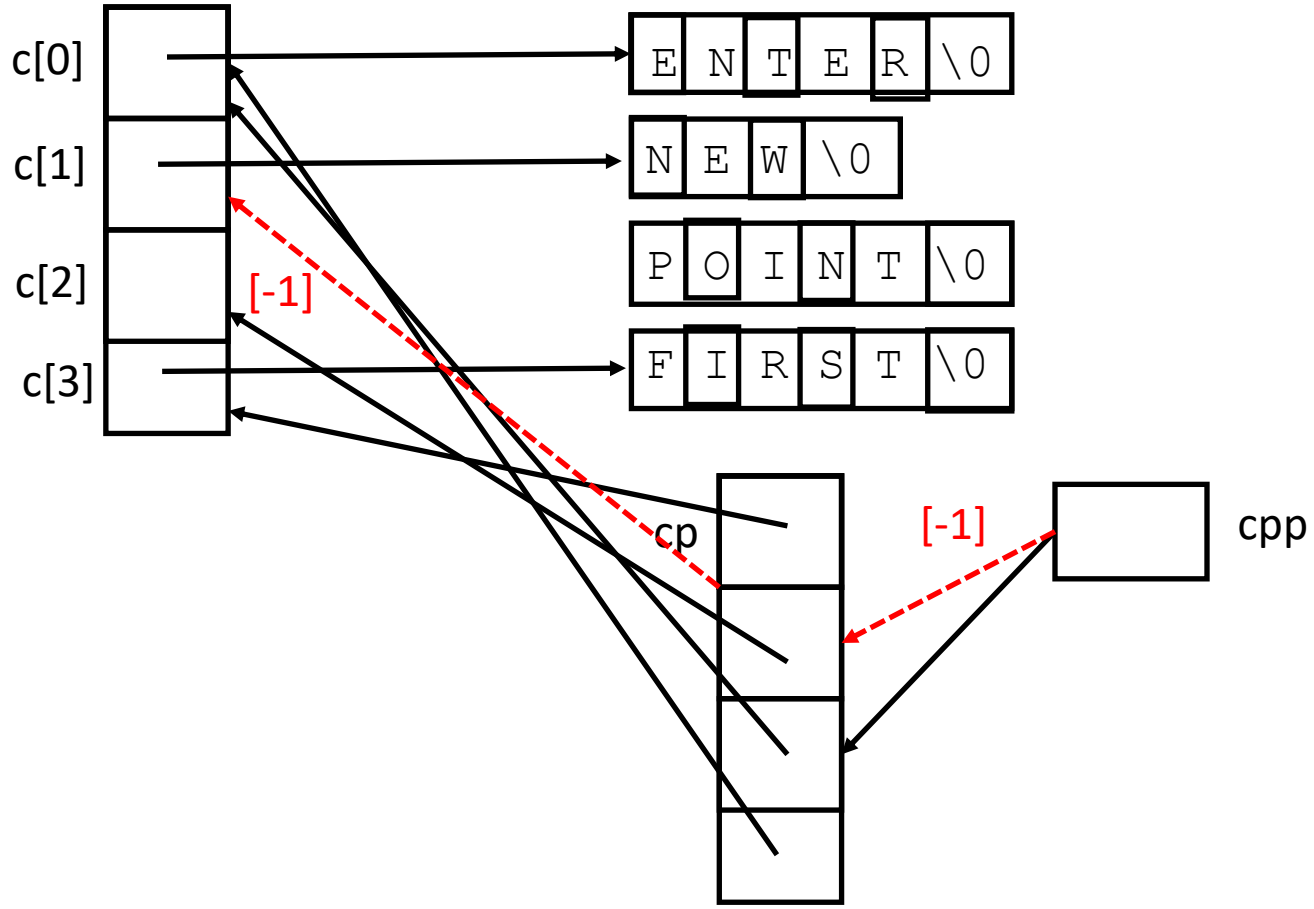
```
#include <stdio.h>
char *c[] = {
    "ENTER",
    "NEW",
    "POINT",
    "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
    printf("%s", **++cpp );
    printf("%s ", *--*++cpp+3 );
    printf("%s", *cpp[-2]+3 );
    printf("%s\n", cpp[-1][-1]+1 );
    return(0);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



POINTER_ST

Pointer Stew (cpp[–1][–1])

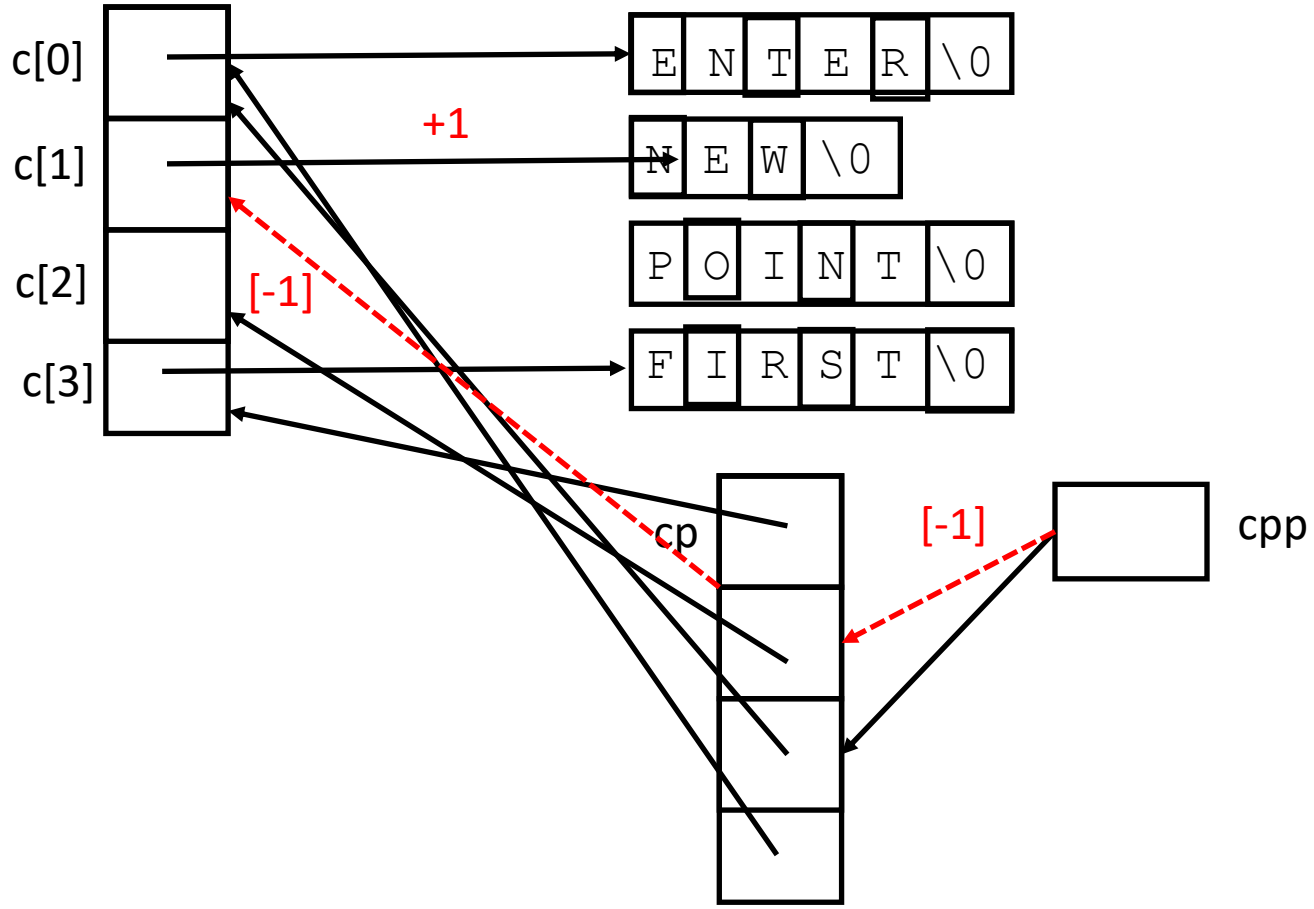
```
#include <stdio.h>
char *c[] = {
     "ENTER",
     "NEW",
     "POINT",
     "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
     printf("%s", **++cpp );
     printf("%s ", *--*++cpp+3 );
     printf("%s", *cpp[-2]+3 );
     printf("%s\n", cpp[-1][-1]+1 );
   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



POINTER_STEW↲

Pointer Stew (cpp[–1][–1] + 1)

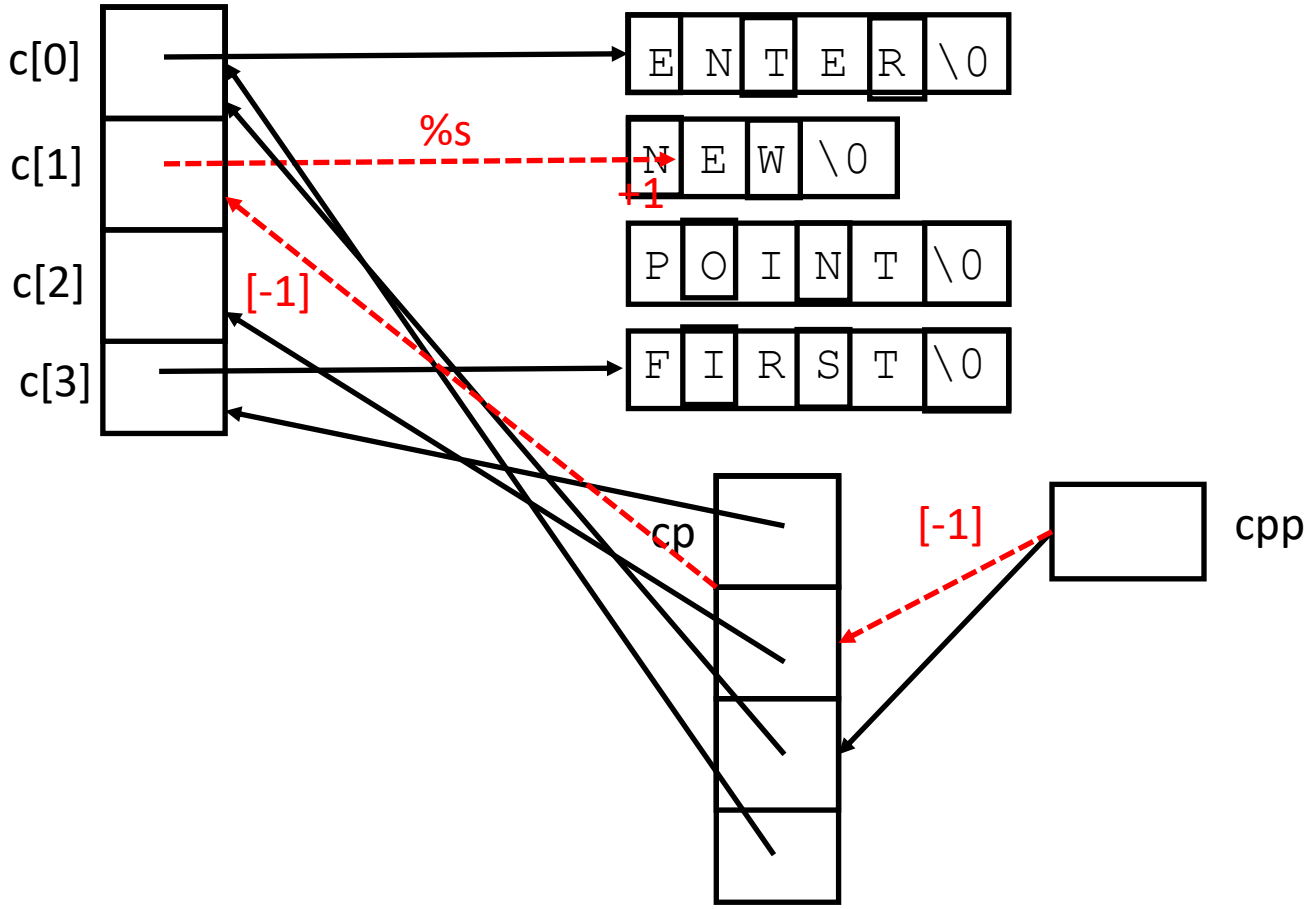
```
#include <stdio.h>
char *c[] = {
     "ENTER",
     "NEW",
     "POINT",
     "FIRST"
};
char **cp[] = {  c+3, c+2, c+1, c
};
char ***cpp = cp;
int main(void)
{
     printf("%s", **++cpp );
     printf("%s ", *--*++cpp+3 );
     printf("%s", *cpp[-2]+3 );
     printf("%s\n", cpp[-1][-1]+1 );
   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



POINTER_STEW↲

Pointer Stew (End of Last *printf*)

```
#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);
}
```



POINTER_STEW↲