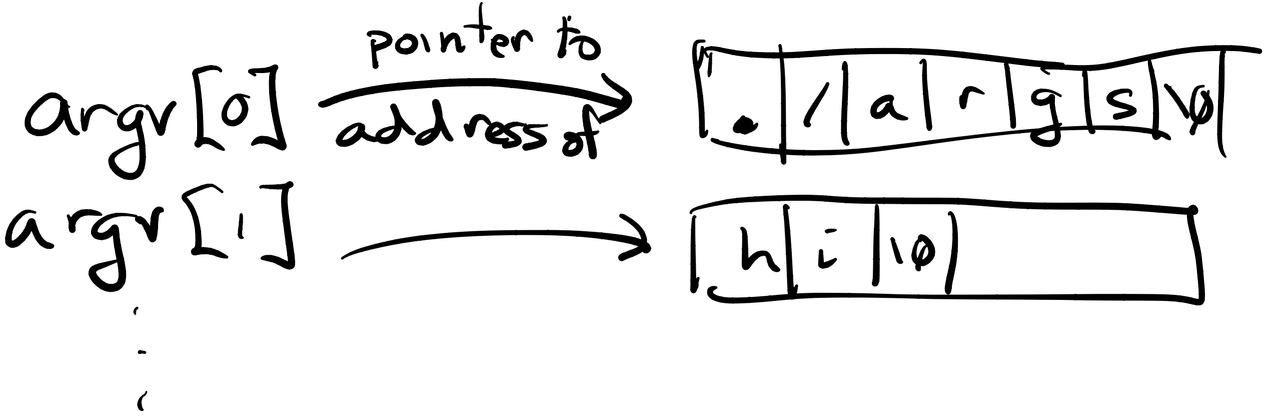
CS 208

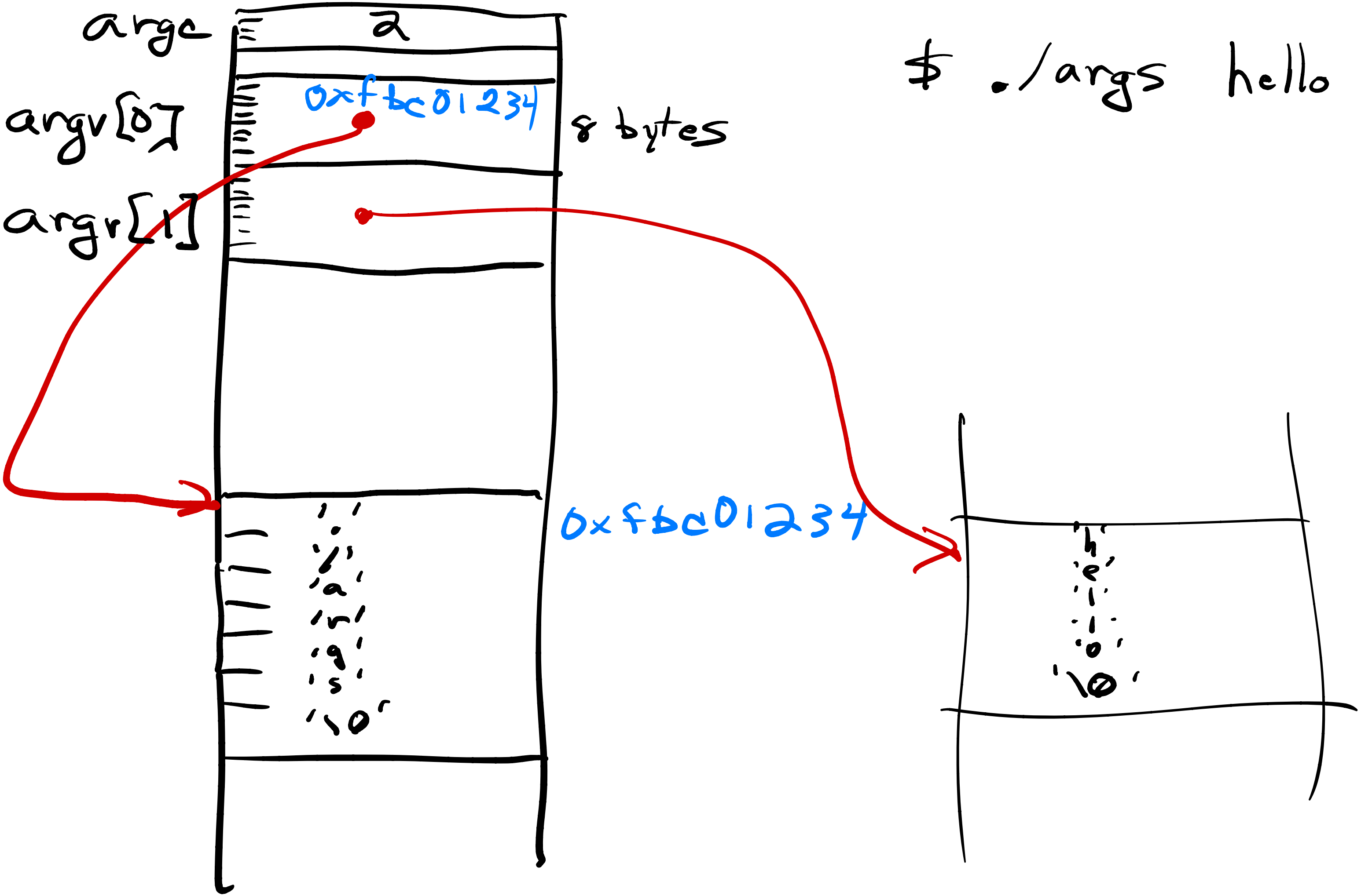
M, 29 Sep 2025

char *argv[]

"array of char*'s"



```
$ ./args hello
```


argc
2
argv[0]
0xfbc01234
8 bytes
argv[1]
'.'
'/'
'a'
'r'
'g'
's'
'\0'
0xfbc01234
'h'
'e'
'l'
'l'
'o'
'\0'

strcmp(a, b)

char *, or null-term. strings

returns 0 – if a string identical to b string

>0 – if a comes later "alphabetically" than b

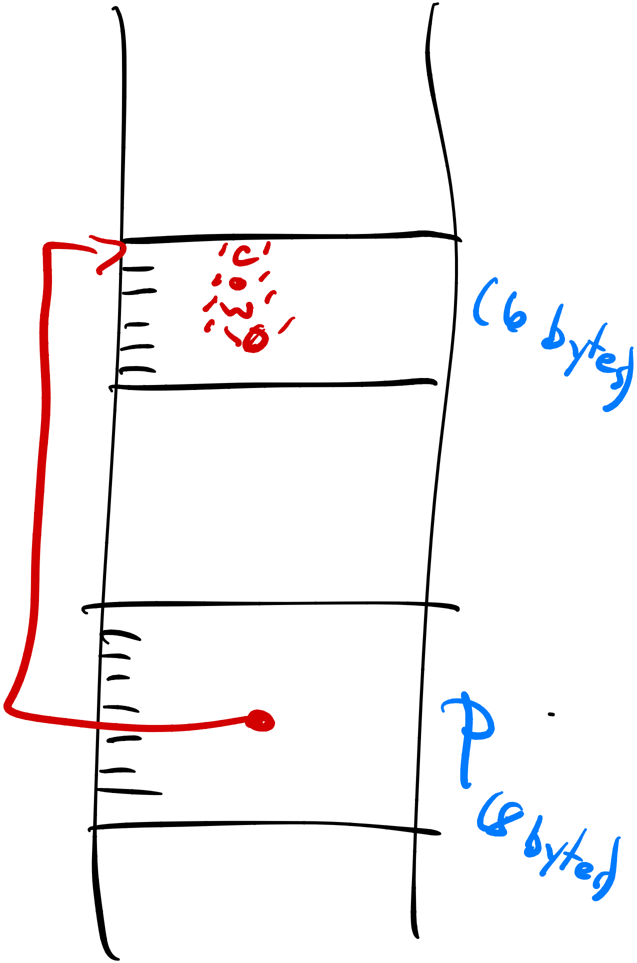
< – if a comes earlier

malloc

```
char *p = malloc(6);
```

asks the OS for 6 bytes if OS has 6 bytes of RAM to give, it marks these 6 bytes as "in use" + returns an address (which we assign to p)

```
strcpy(p, "cow");
```



```
p = malloc(   );

  ⋮  do stuff
     w/ p

free(p)
```

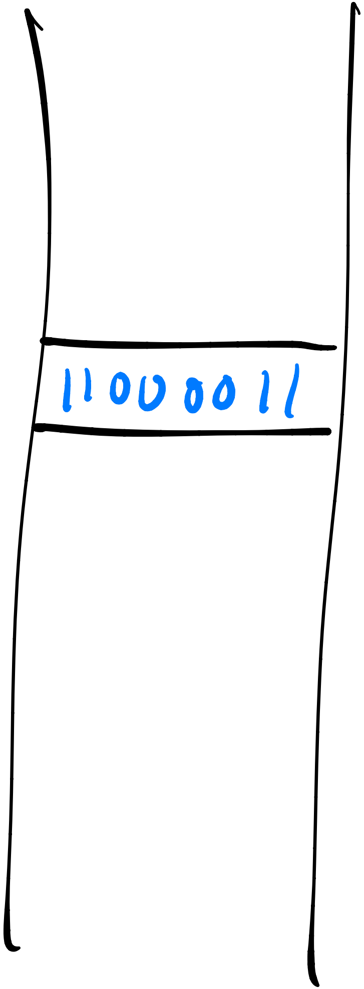
→ Hey OS, you can mark those bytes as "unused"

```
char ch = 0xC3;

int n = ch;
```

hey, this
a char!

But but - - - -



C integer types

1 byte	char	unsigned char
2bytes	short	unsigned
4	int	
8	long	

2 ops that depend on leftmost bit

```
int n = 0xcdef1234
n = n >> 5 ;
```

1100 1101 - - -

11111 1100 1101 - - - .