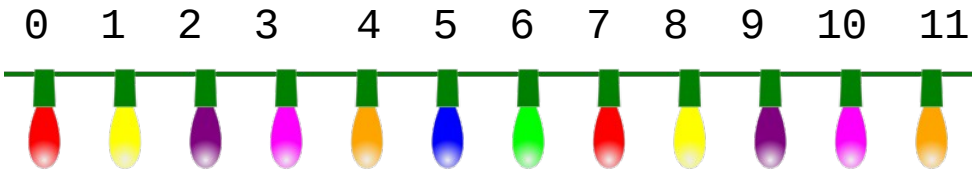
CS 261
Spring 2024

Mike Lam, Professor



https://openclipart.org/detail/188824/animated-christmas-lights-2

Arrays and Strings

Arrays and Pointers

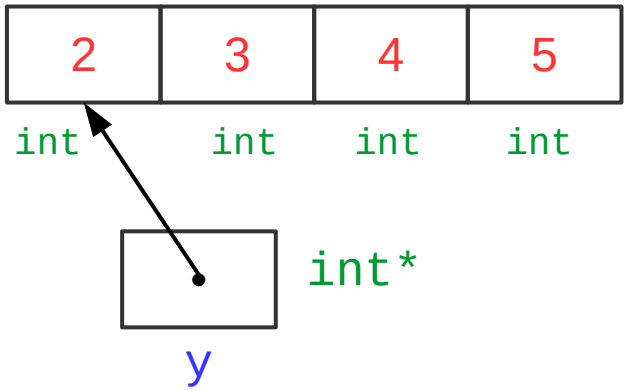
- In C, array names are just aliases that can be used as pointers

```
int  y[] = {2, 3, 4, 5};    // these two are
int *y   = {2, 3, 4, 5};    // roughly equivalent
```

- Indexing and dereferencing pointers are equivalent

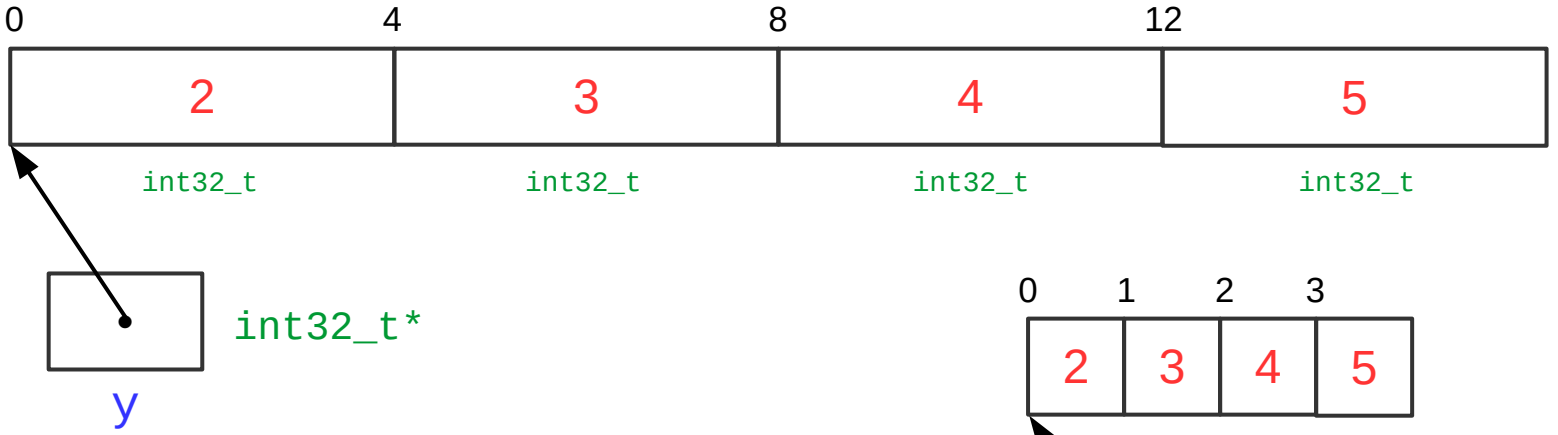
  Side note: you can do arithmetic with pointers!

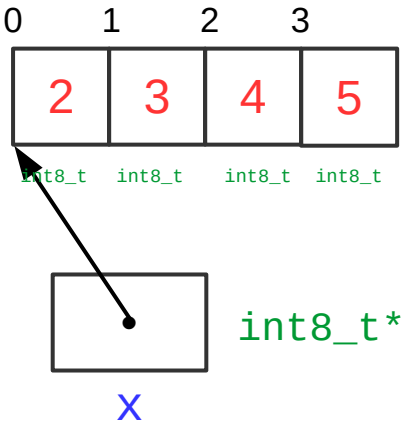
```
*y  ≡  y[0]          *(y+1)  ≡  y[1]
```



Arrays and Pointers

- Pointer types are important!
  - If `x` is an `int8_t*`, `x[3]` accesses element at byte offset 3×1 = 3
  - If `y` is an `int32_t*`, `y[3]` accesses element at byte offset 3×4 = 12





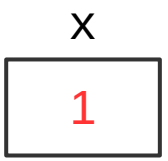
Pointers

```
int x = 1;
int y[4] = {2, 3, 4, 5};
int *p = &x;
*p = 6;
p = y;
*p = 7;
```

What are the values of x and y at the end?

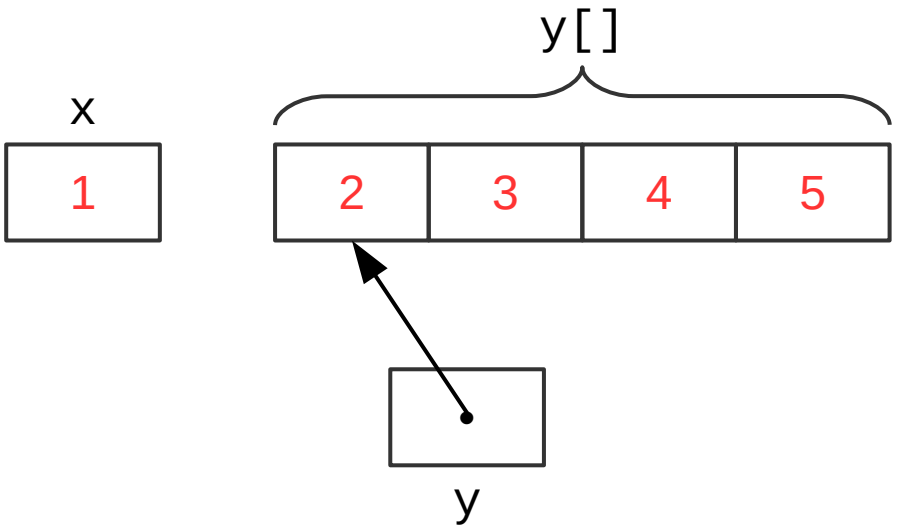
Pointers

```
int x = 1;
int y[4] = {2, 3, 4, 5};
int *p = &x;
*p = 6;
p = y;
*p = 7;
```



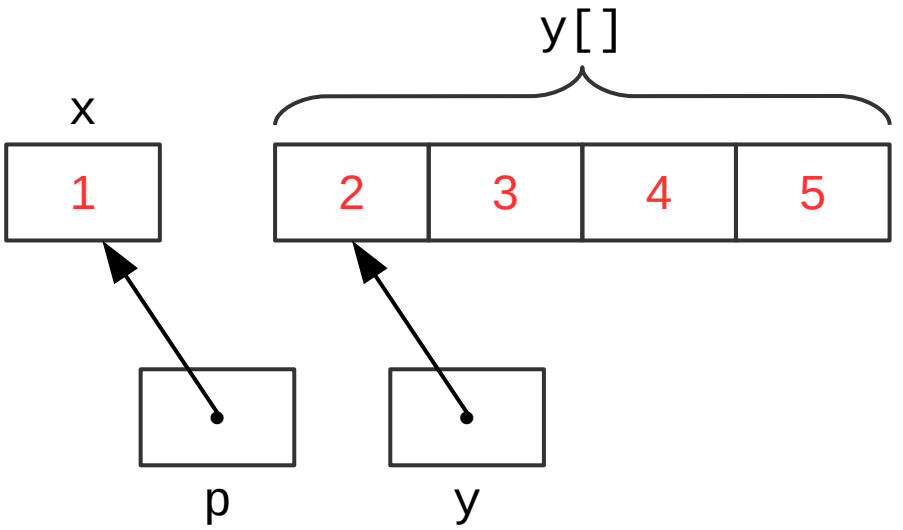
Pointers

```
int x = 1;
int y[4] = {2, 3, 4, 5};
int *p = &x;
*p = 6;
p = y;
*p = 7;
```



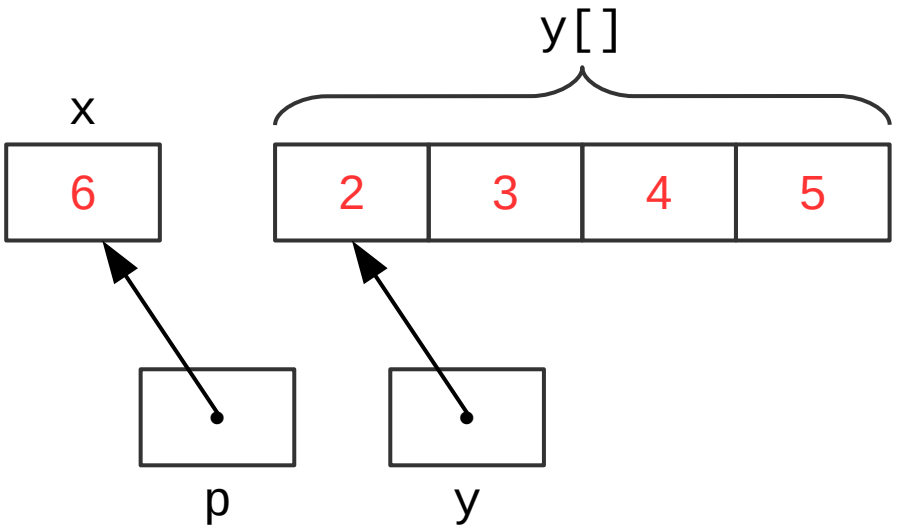
Pointers

```
int x = 1;
int y[4] = {2, 3, 4, 5};
int *p = &x;
*p = 6;
p = y;
*p = 7;
```



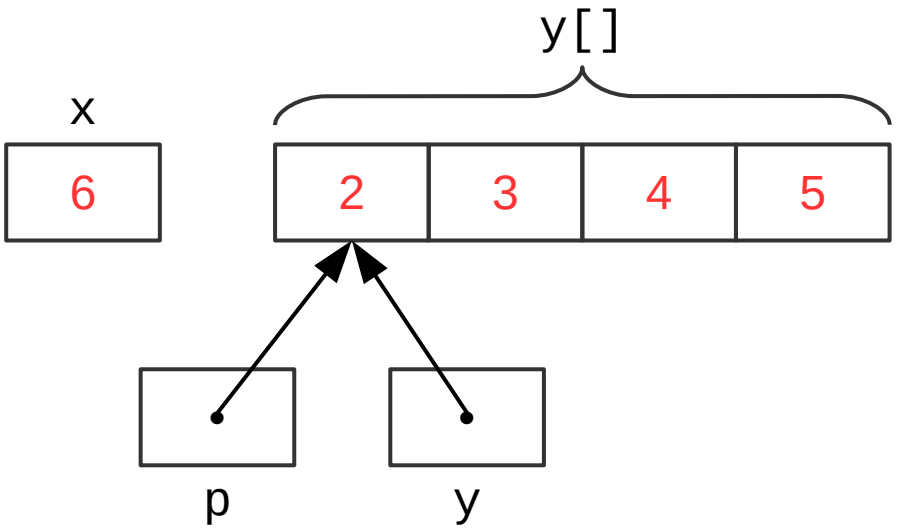
Pointers

```
int x = 1;
int y[4] = {2, 3, 4, 5};
int *p = &x;
*p = 6;
p = y;
*p = 7;
```



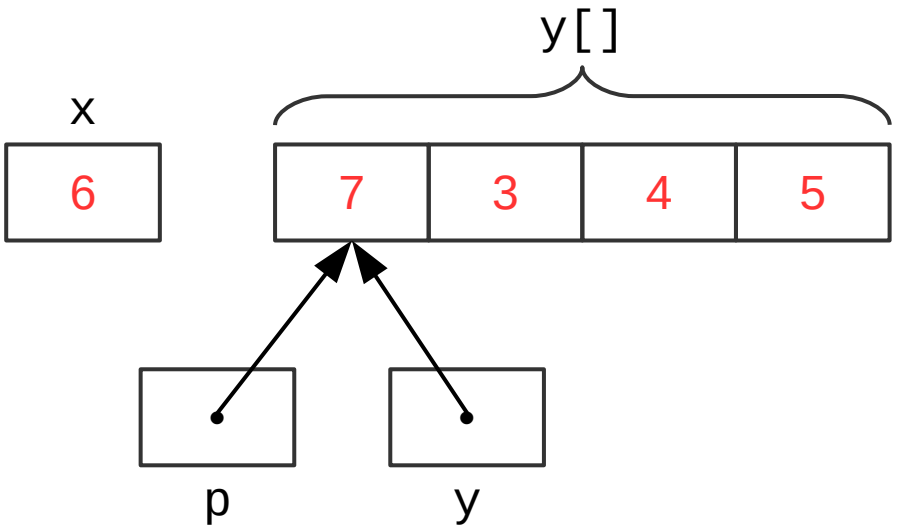
Pointers

```
int x = 1;
int y[4] = {2, 3, 4, 5};
int *p = &x;
*p = 6;
p = y;
*p = 7;
```



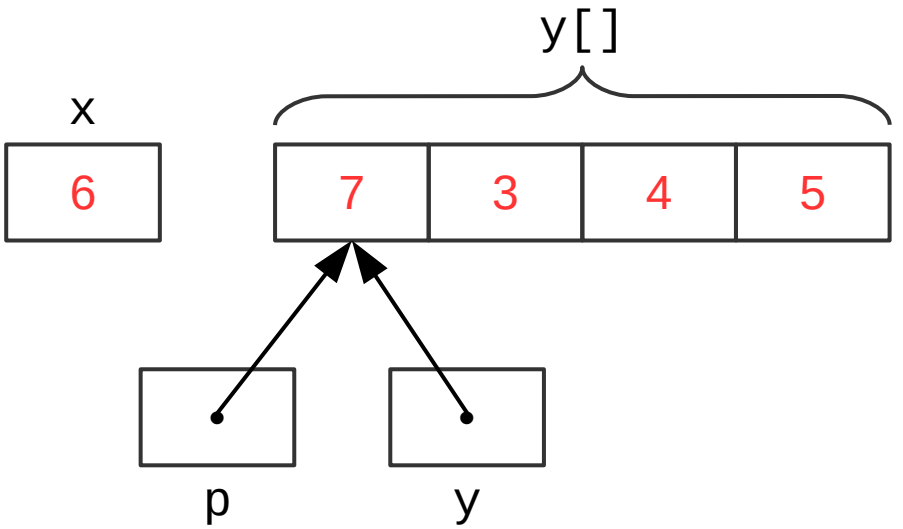
Pointers

```
int x = 1;
int y[4] = {2, 3, 4, 5};
int *p = &x;
*p = 6;
p = y;
*p = 7;
```



Pointers

```
int x = 1;
int y[4] = {2, 3, 4, 5};
int *p = &x;
*p = 6;
p = y;
*p = 7;
```



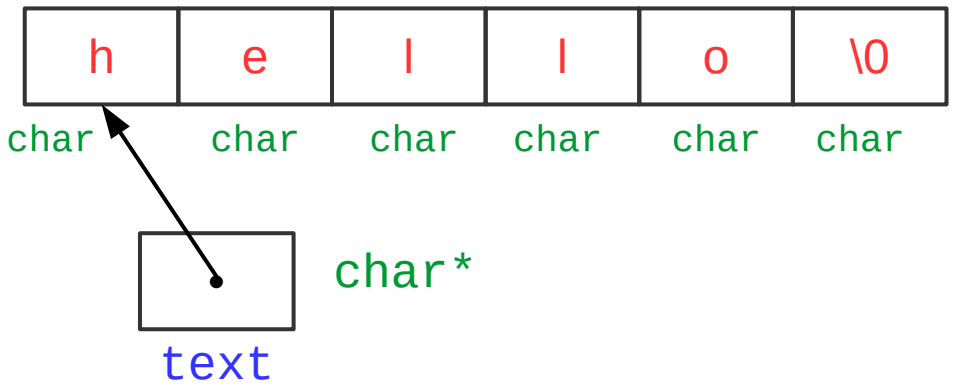
What about this?

```
p++;
*p = 9;
```

Arrays and Pointers

- The same is (roughly) true for C "strings" (arrays of `chars`)

```
char  text[] = "hello";      // read-write
char *text   = "hello";      // read-only
```



C Strings

- C strings are a sequence of ASCII chars **terminated with null char (‘\0’)**
  - Declare and initialize (static/stack, no explicit size needed):
    - `char *name   = "John Smith";`
    - `char  name[] = "John Smith";`
  - Declare only (static/stack, size needed):
    - `char name[11];`
  - Declare only (heap, size needed):
    - `char *name = (char*) malloc (sizeof(char) * 11);`

- Useful functions (need to `#include <string.h>`)
  - Find length: `strlen`
  - Copy string or convert / format data into string: `snprintf`
  - Convert to long / float: `strtol` / `strtof`
  - Compare strings: `strncmp`
  - Search for substring: `strstr`

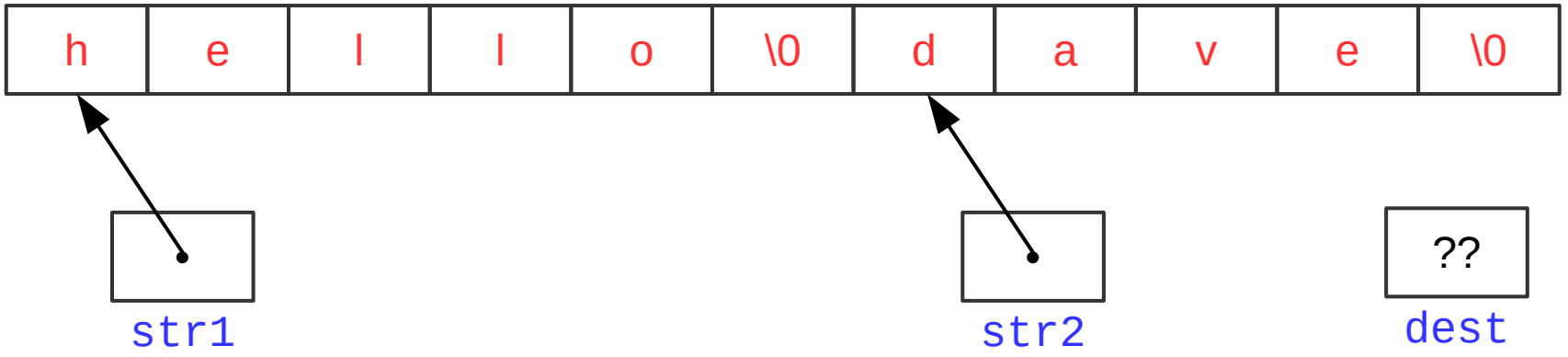
Information = Bits + Context

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Copying strings

- In Java: `dest = str1 + str2;  // copy strings`
  - What does this code do in C?



- Need to copy all characters from one string to another
  - First for `str1` and then for `str2`

Copying strings

- Old solution: `strcpy`
  - Basically:

```
void strcpy (char *dest, char *src)
{
    for (int i = 0; src[i] != '\0'; i++) {
        dest[i] = src[i];
    }
}
```

  - What happens if `src` isn't null-terminated?

OUT OF BOUNDS!!!

Copying strings

- Using `strcpy` is now considered **unsafe**
  - You are **not permitted** to use it in CS 261
- Solution: specify a maximum length that is safe to copy
  - This is usually the allocated length of the destination
- Older alternative: `strncpy`
  - Requires a maximum length
  - However, it does not guarantee the result is null-terminated
- Newer alternative: `strcpy_s`
  - However, it is not in the C99 standard
- **Better alternative: `snprintf`**
  - Safe, C99-standard, and more powerful than the other two

Output and string conversion

- `printf` and `snprintf` are conceptually similar
  - The former prints to standard out
  - The latter "prints" to a string (character array)
  - The latter can also copy strings and convert to strings
    - `snprintf(dest_str, max_size, "%s", src_str);  // copy string`
    - `snprintf(dest_str, max_size, "%d", int_var);  // int -> string`
    - `snprintf(dest_str, max_size, "%f", float_var); // float -> string`

```
int   printf (                              char *format, ...)

int snprintf (char *buffer, int  bufsize, char *format, ...)
```

destination string; must be at least "`bufsize`" bytes

maximum size

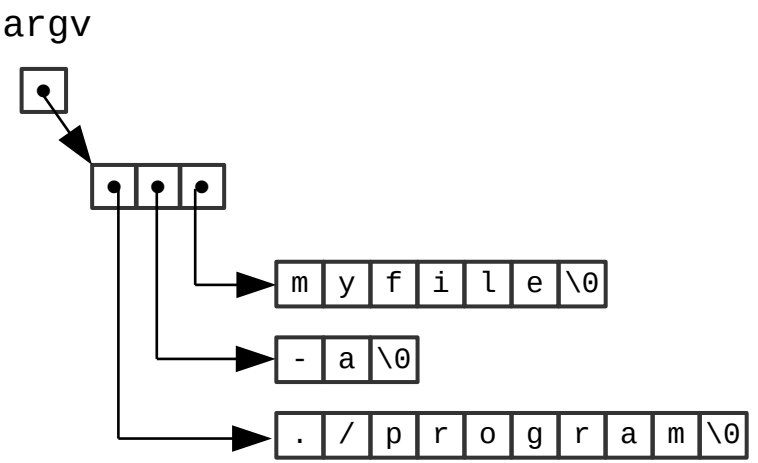
Question

- How do we declare an array of strings?

Arrays of arrays

- Array of string (`char*`) pointers
  - Two (roughly) equivalent syntax choices
    - `char *name[];`
    - `char **name;`
  - Must allocate/initialize each sub-array separately

- Command-line parameters
    - `int main (int argc, char *argv[])`
  - Example: "`./program -a myfile.txt`"
    - argc = 3
    - argv[0] = "`./program`"
    - argv[1] = "`-a`"
    - argv[2] = "`myfile.txt`"



Modified “Hello, World”

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define STR_LEN 8

int main(int argc, char **argv)
{
    // check parameters
    if (argc != 3) {
        fprintf(stderr, "Usage: ./hello2 <fname> <lname>\n");
        exit(EXIT_FAILURE);
    }

    // convert name to "First L." format
    char fullname[STR_LEN];
    snprintf(fullname, STR_LEN, "%s %c.", argv[1], argv[2][0]);

    // output new full name
    printf("Hello, %s!\n", fullname);

    return EXIT_SUCCESS;
}
```