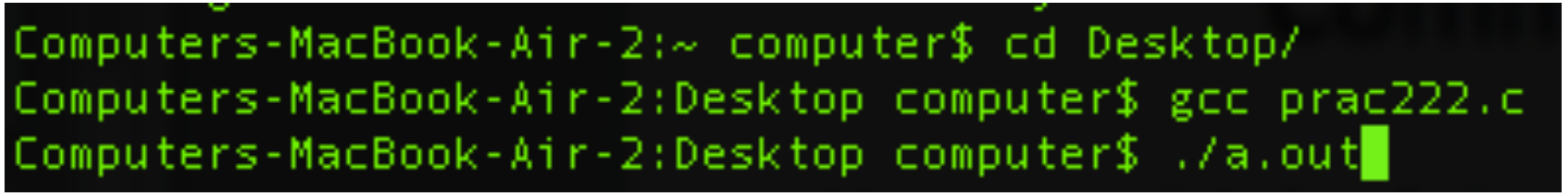
Note: these slides are taken from my 1320 lectures and the programs given are in C, not C++

Command Line

Main Function Parameters

Command Line

- You can think of the command line as a way for you to interact with your computer
- We've been typing into the command line for the past few weeks
- When we want to run a program, we can type it into the command line



Command Line

- Remember that functions have parameters as part of the definition
  - When we use a function, we can can pass arguments
- **main is a function** (with the parameters int argc and char ** argv)
  - We can supply arguments to the main function on the command line

```
Computers-MacBook-Air-2:Desktop computer$ ./a.out 3 4 8
```

```
Computers-MacBook-Air-2:Desktop computer$ ./a.out 3 4 8
Program name: ./a.out
Add the 3 nums: 15
Computers-MacBook-Air-2:Desktop computer$
```

Main is a FUNCTION. It has two parameters. When we use the function (i.e. run the program), we can supply it with arguments. The arguments are kept in argv.

```
#include <stdio.h>

int main(int argc, char **argv)
{
        if(argc !=4)
        {
                printf("Not the right amount of arguments!\n");
        }

        else
        {
                printf("Program name: %s\n", argv[0]);
                int a=atoi(argv[1]);
                int b=atoi(argv[2]);
                int c=atoi(argv[3]);

                printf("Add the 3 nums: %d\n", (a+b+c));

        }

}
```

```
Computers-MacBook-Air-2:Desktop computer$ ./a.out 3 4 8
Program name: ./a.out
Add the 3 nums: 15
Computers-MacBook-Air-2:Desktop computer$
```

```
#include <stdio.h>

int main(int argc, char **argv)
{
        if(argc !=4)
        {
                printf("Not the right amount of arguments!\n");
        }

        else
        {
                printf("Program name: %s\n", argv[0]);
                int a=atoi(argv[1]);
                int b=atoi(argv[2]);
                int c=atoi(argv[3]);

                printf("Add the 3 nums: %d\n", (a+b+c));

        }

}
```

argc *is the NUMBER of arguments (4 in my example)*
argv *contains the actual arguments (./a.out, 3, 4, 8)*
Notice it is a double pointer.

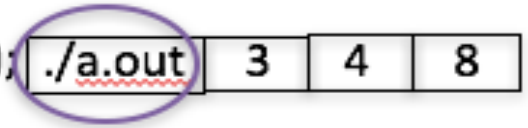
I'm taking each argument (in argv) and converting it to an int (arguments are ALWAYS represented as strings-even numbers)

```
#include <stdio.h>


int main(int argc, char **argv)
{
        if(argc !=4)
        {
                printf("Not the right amount of arguments!\n");
        }

        else
        {
                printf("Program name: %s\n", argv[0]);

                int a=atoi(argv[1]);

                int b=atoi(argv[2]);

                int c=atoi(argv[3]);

                printf("Add the 3 nums: %d\n", (a+b+c));

        }
}
```

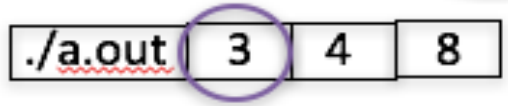
Pointer to array of char arrays (i.e. array of pointers)-each element is a char array

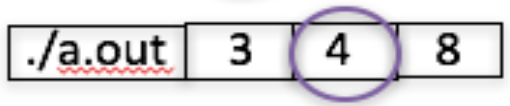
Command line:
computer$ ./a.out 3 4 8

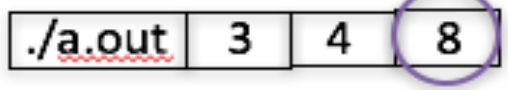
4 arguments:
1. a.out (the program name)
2. 3
3. 4
5. 8



atoi function changes a string into an int







Output:
Program name: ./a.out
Add the 3 nums: 15

Command Line

Main Function Parameters

Main Function Parameters

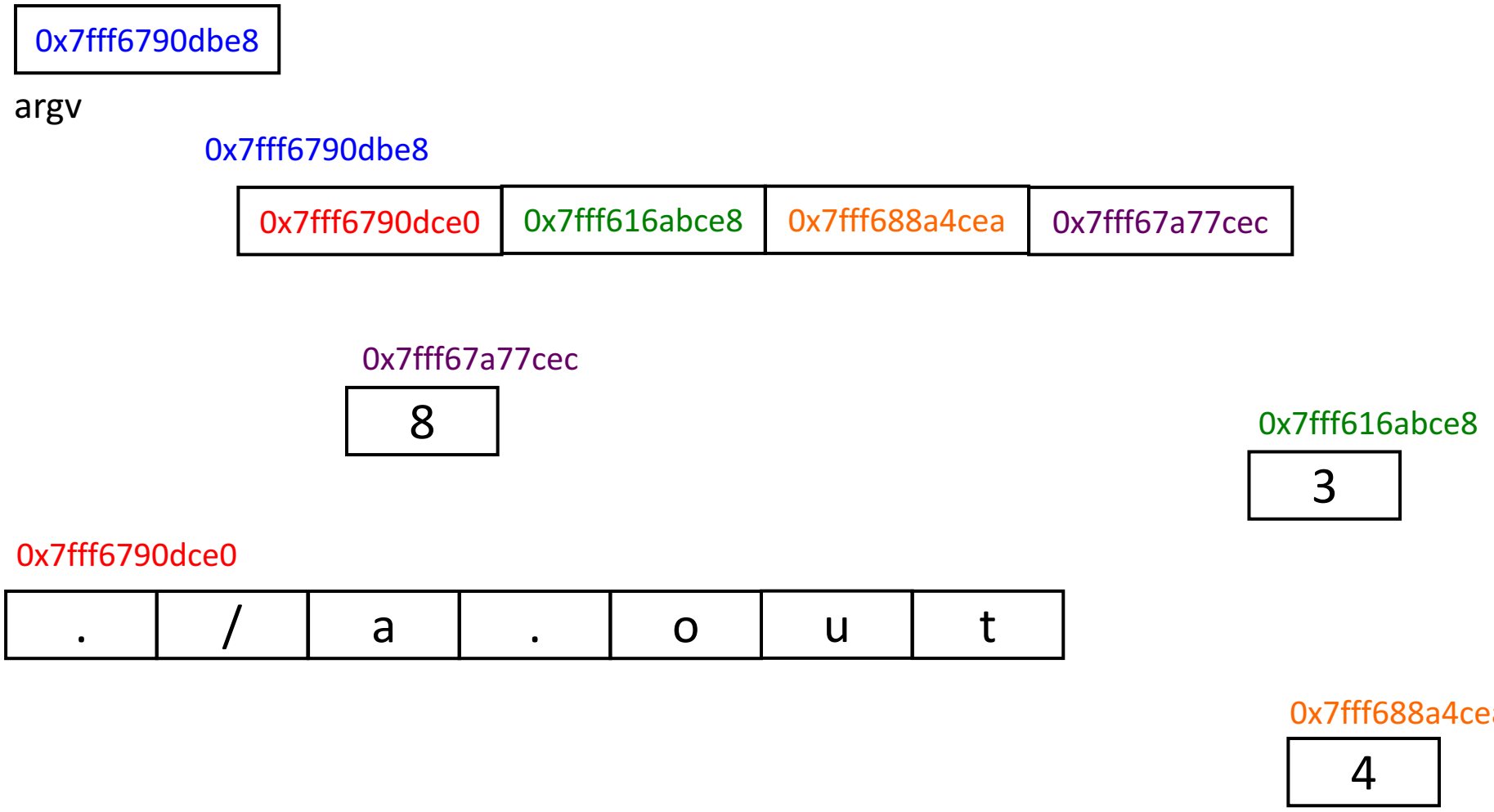
- So now we know that ***int argc*** and ***char ** argv*** are parameters to our main function
- So how do they work in our program?
  - You should be familiar by now with the double pointer format char **
  - I will show you an example of how it works using our previous command line example:

```
Computers-MacBook-Air-2:Desktop computer$ ./a.out 3 4 8
```

int main(int argc, char **argv)

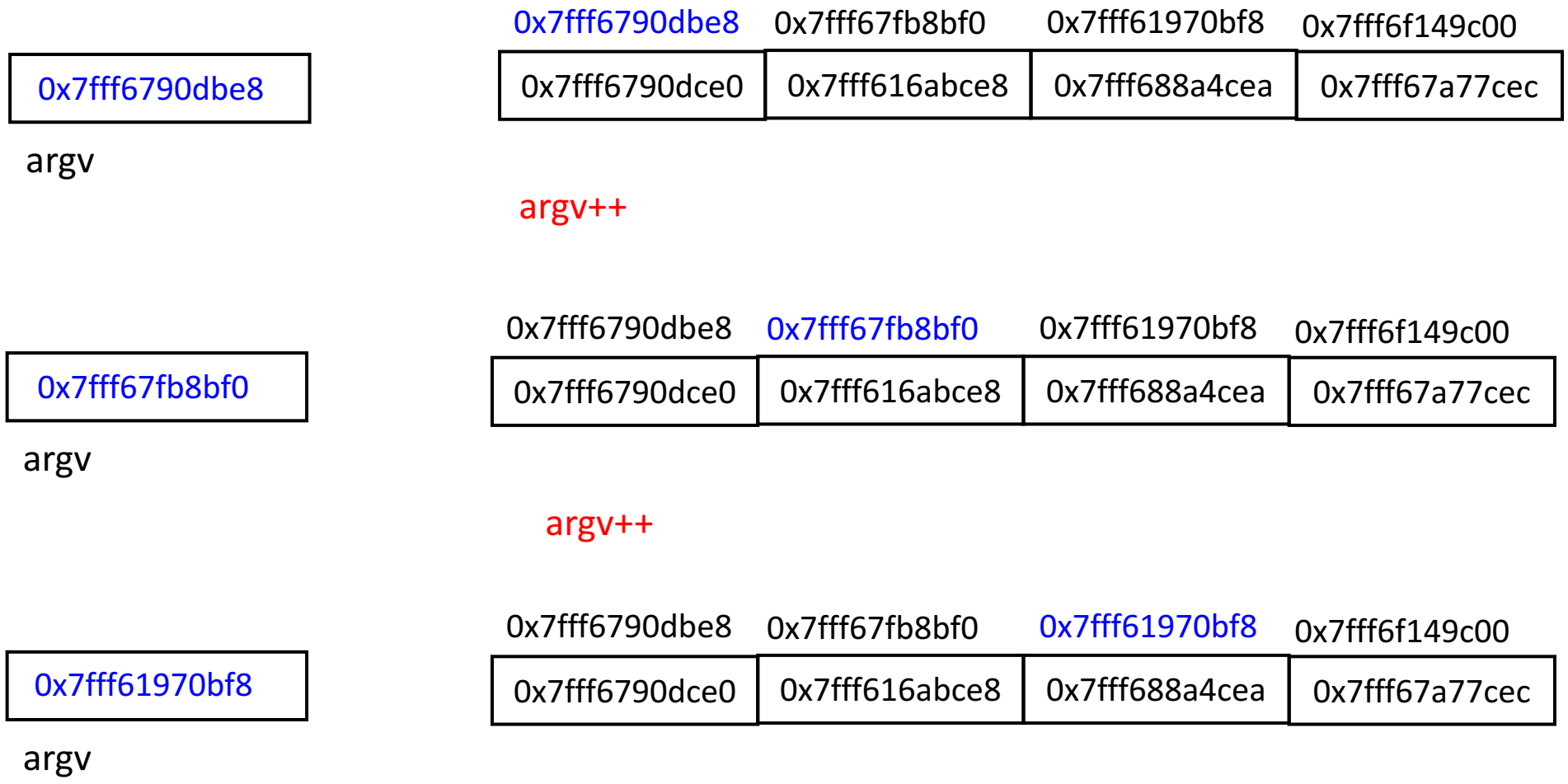
argv is a double pointer-meaning it is pointing to another pointer (an array of pointers in this case-remember arrays are really represented by the address of the first element)

```
:Desktop computer$ ./a.out 3 4 8
```

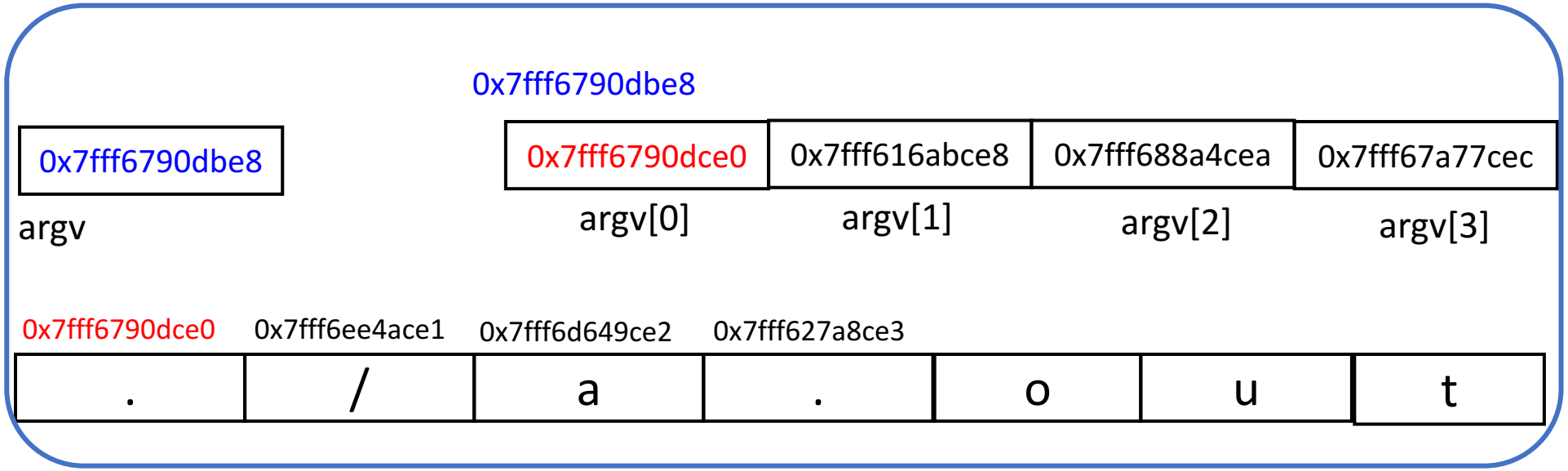


int main(int argc, char **argv)

We can increment argv (using pointer arithmetic) to get to the next address in our array of pointers:



We can also increment the pointer pointing at a specific argument argv[0]++



argv[0]++