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

```
Computers-MacBook-Air-2:~ computer$ cd Desktop/
Computers-MacBook-Air-2:Desktop computer$ gcc prac222.c
Computers-MacBook-Air-2:Desktop computer$ ./a.out
```

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

```
#include <stdio.h>
                                 Main is a FUNCTION. It has two parameters. When we use the
int main(int argc, char **argv)
                                 function (i.e. run the program), we can supply it with
                                 arguments. The arguments are kept in 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>
                                 argc is the NUMBER of arguments (4 in my example)
int main(int argc, char **argv)
                                 argy contains the actual arguments (./a.out, 3, 4, 8)
                                    Notice it is a double pointer.
          if(argc!=4)
                    printf("Not the right amount of arguments!\n");
          else
                    printf("Program name: %s\n", argv[0]);
                                                             I'm taking each argument (in
                    int a=atoi(argv[1]);
                                                             argv) and converting it to an int
                    int b=atoi(argv[2]);
                                                             (arguments are ALWAYS
                    int c=atoi(argv[3]);
                                                             represented as strings-even
                   printf("Add the 3 nums: %d\n", (a+b+c)); numbers)
```

```
#include <stdio.h>
                                 Pointer to array of char
                                                                    Command line:
                                 arrays (i.e. array of
                                                                    computer$ ./a.out 3 4 8
                                 pointers)-each element
 int main(int argc, char **argv)
                                 is a char array
                                                                     4 arguments:
                              ./a.out
      if(argc !=4)
                                                                     1. a.out (the program name)
                                                                     2.3
           printf("Not the right amount of arguments!\n");
                                                                     3.4
                                                                     5.8
      else
           printf("Program name: %s\n", argv[0]); ./a.out)
           int a=atoi(argv[1]);
                                ./a.out
atoi
function
           int b=atoi(argv[2]); ./a.out
changes a
string into
           int c=atoi(argv[3]);
                                ./a.out
an int
                                                                      Output:
           printf("Add the 3 nums: %d\n", (a+b+c));
                                                                       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:

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

0x7fff6790dbe8

argv

0x7fff6790dbe8

 0x7fff6790dce0
 0x7fff616abce8
 0x7fff688a4cea
 0x7fff67a77cec

0x7fff67a77cec

8

0x7fff616abce8

3

0x7fff6790dce0

. / a . o u t

0x7fff688a4cea

4

## int main(int argc, char \*\*argv)

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

0x7fff6790dbe8

argv

 0x7fff6790dbe8
 0x7fff67fb8bf0
 0x7fff61970bf8
 0x7fff6f149c00

 0x7fff6790dce0
 0x7fff616abce8
 0x7fff688a4cea
 0x7fff67a77cec

argv++

0x7fff67fb8bf0

argv

0x7fff6790dbe8	0x7fff67fb8bf0	0x7fff61970bf8	0x7fff6f149c00		
0x7fff6790dce0	0x7fff616abce8	0x7fff688a4cea	0x7fff67a77cec		

argv++

 0x7fff6790dbe8
 0x7fff67fb8bf0
 0x7fff61970bf8
 0x7fff6f149c00

 0x7fff6790dce0
 0x7fff616abce8
 0x7fff688a4cea
 0x7fff67a77cec

0x7fff61970bf8

argv

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

				3 1 3							
0x7fff6790dbe8											
0x7fff6790dbe8		0x7fff6790dd	ce0 0x7	0x7fff616abce8		0x7fff688a4cea		0x7fff67a77cec			
argv	rgv		] argv[		.]	argv[2]		argv[3]			
0x7fff6790dce0	0x7fff6ee4ace1	0x7fff6d649ce2	0x7fff627a	a8ce3							
•	/	а	•		(	0	u		t		

#### argv[0]++

