Command Line Arguments

What are they?

- A program can be executed by directly typing a command with parameters at the prompt

```
$ cc –o test test.c
$ ./a.out in.dat out.dat
$ prog_name param_1 param_2 param_3
  ..
```

  - The individual items specified are separated from one another by spaces
    - First item is the program name

What do they mean?

- Recall that main() is also a function
- It can also take parameters, just like other C function
- The items in the command line are passed as parameters to main
- Parameters argc and argv in main keeps track of the items specified in the command line

How to access them?

```
int main (int argc, char *argv[]);
```

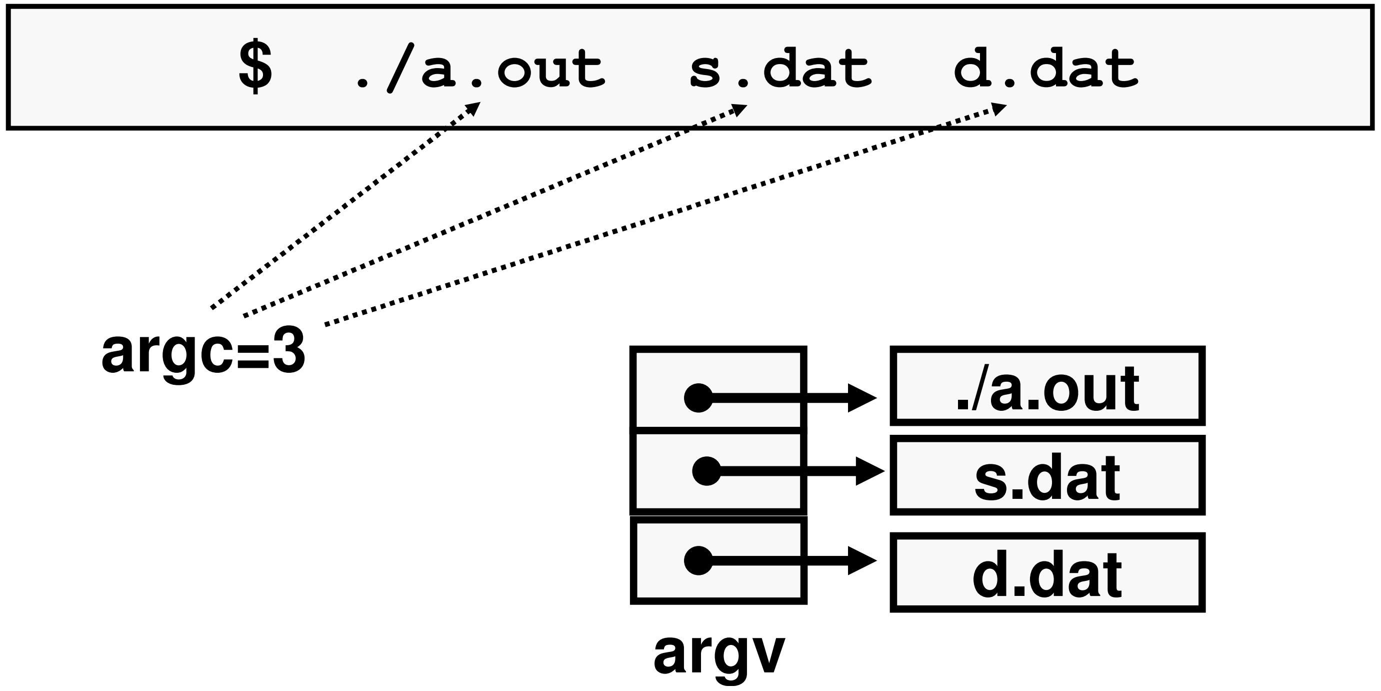
Argument Count

Array of strings as command line arguments including the command itself.

The parameters are filled up with the command line arguments typed when the program is run

They can now be accessed inside main just like any other variable

Example: Contd.



argv[0] = "./a.out"   argv[1] = "s.dat"   argv[2] = "d.dat"

Contd.

- Still there is a problem
  - All the arguments are passed as strings in argv[ ]
  - But the intention may have been to pass an int/float etc.
- Solution: Use sscanf()
  - Exactly same as scanf, just reads from a string (char *) instead of from the keyboard
  - The first parameter is the string pointer, the next two parameters are exactly the same as scanf

Example

- Write a program that takes as command line arguments 2 integers, and prints their sum

```
int main(int argc, char *argv[ ])
{
    int i, n1, n2;
    printf("No. of arg is %d\n", argc);
    for (i=0; i<argc; ++i)
        printf("%s\n", argv[i]);
    sscanf(argv[1], "%d", &n1);
    sscanf(argv[2], "%d", &n2);
    printf("Sum is %d\n", n1 + n2);
    return 0;
}
```

```
$ ./a.out 32  54
No. of arg is 3
./a.out
32
54
Sum is 86
```