Command line args in C++

- many programs allow the user to enter arguments as part of the command to start the program, e.g.

```
ls csci160
```

- the program ls detects if the user added extra arguments specifying which directories to look at
- these are referred to as command line arguments, and are actually passed as parameters to the main routine
- we can detect/access these parameters if we set main up correctly
- many linux utilities are written as C programs (e.g. cp, ls, mv, rm, etc) and you can see sample source code here:

```
git.savannah.gnu.org/cgit/coreutils.git/tree/src
```

argc, argv as main's parameters

- for compatibility with the way the command line arguments are sent to the program, we must use the following parameters for main

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

- `argc` contains a count of the number of arguments the user typed on the command line, including the executable name
- `argv` is an array referencing 1 or more null-terminated character arrays (more-or-less an array of arrays), each representing one of the command line arguments

Example: argc, argv

- suppose our program is named myprog, and the user invokes it with the following command

```
./myprog blah foo 42!
```

- assuming we have declared argc and argv correctly:
  - argc is 4
  - argv[0] is “./myprog”
  - argv[1] is “blah”
  - argv[2] is “foo”
  - argv[3] is “42!”

Sample program

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{
    // display each of the args
    for (int i = 0; i < argc; i++) {
        cout << i << “: “;
        cout << argv[i] << endl;
    }
}
```

suppose user runs the program as

```
./myprog 1.234  ab.cde  x
```

the resulting output would be

```
0: ./myprog
1: 1.234
2: ab.cde
3: x
```

Each entry of argv is text

- the arguments are always passed to the program as text, e.g. `./myprog 123` would pass “123” as argv[1]
- within the program we can use the arguments accordingly, e.g. doing different things with the i'th argument:

```
char text[N]; // for some const N
strncpy(text, argv[i], N);
string s = argv[i];
cout << argv[i];
int num = atoi(argv[i]); // get int equivalent of i'th arg
```

Example: argument checking

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{
   // check correct #args were passed
   if (argc != 3) {
      cout << “Incorrect num args, run “;
      cout << “with two positive numbers”;
      cout << endl;
   }
```

```
   else {
      // get float equivalents of the two args
      float N1 = atof(argv[1]);
      float N2 = atof(argv[2]);

      // check they're both greater than 0
      if ((N1 <= 0) || (N2 <= 0)) {
         cout << “Numbers must be positive;
         cout << endl;
      } else {
         cout << N1 << “+” << N2;
         cout << “=” << (N1+N2) << endl;
      }
   }
}
```