6 Command line arguments

Command-line processing uses text rather than point-and-click. For example, to edit a file `commandline.c` using the vi or vim visual editor:

```
vi commandline.c
```

This invokes the `vi` editor to create or edit a file named `commandline.c`

This was done, and the result is:

```
        // file commandline.c
#include <stdio.h>
#include <stdlib.h> // needed for atoi below


int main ( int argc, char * argv[] )
{
  int day, month, year;
  day = atoi ( argv[1] );   // atoi converts string to integer
  month = atoi ( argv[2] );
  year = atoi ( argv[3] );

  printf("The date is %d/%d/%d\n", day,month,year);
}
```

Compile and execute:

```
% gcc commandline.c
% a.out 27 9 21
The date is 27/9/21
%
```

Rationale. When

```
%a.out 27 9 21
```

is run, the numbers 27, 9, 21, are *command-line arguments.* They are copied to the program *as character strings.* So

```
The function atoi() converts string to int.

argc is 4, the number of arguments.
argv[0] is "a.out" --- argv[0] is the name  of the program being run.

argv[1] is "27" and atoi(argv[1]) is 27, assigned to the variable ‘day.’
argv[2] is "9" and atoi(argv[1]) is 9, assigned to the variable ‘month.’
argv[3] is "21" and atoi(argv[1]) is 21, assigned to the variable ‘year.’
```