Splitting up a Program into Multiple Files

Handout 1.1

Consider the following simple program which is all in one file, namely `prog.c`.

File: `prog.c`

```
#include <stdio.h>
int  main(void) {
  int   max(int, int);       /* Returns the larger of the two inputs. */
  float average(int, int);   /* Returns the average of the two inputs. */
  int i = 17, j = -20;
  printf("%d\n", max(i, j));
  printf("%f\n", average(i, j)); return 0;
}

int max (int a, int b) {
   if (a >= b)
     return a;
   else return b;
}

float average (int a, int b) {
   return (a+b)/2.0;
}
```

You can compile the above file (`prog.c`) to create an executable (`a.out`) using the Unix command

```
            gcc  prog.c
```

There are at least two difficulties with this approach. First, if the source file is large, it is harder to fix syntax errors. Second, even if all the errors are in just one function, you must recompile the whole file.

To overcome the above difficulties, it is more common to write programs in C in such a way that the source code is in several small files. As an illustration, let us split the above program into three files (say, `main.c`, `max.c` and `avg.c`) as follows.

File: `main.c`

```
#include <stdio.h>
int  main(void) {
  int   max(int, int);       /* Returns the larger of the two inputs. */
  float average(int, int);   /* Returns the average of the two inputs. */
  int i = 17, j = -20;
  printf("%d\n", max(i, j));
  printf("%f\n", average(i, j)); return 0;
}
```

```
int max (int a, int b) {
   if (a >= b)
      return a;
   else return b;
}
```

```
float average (int a, int b) {
   return (a+b)/2.0;
}
```

You can now generate the executable version of the program using the Unix command

```
gcc  main.c  max.c  avg.c
```

However, the more common method is to *separately compile* each file and eliminate syntax errors. Once each file has been compiled without any syntax errors, we can create the executable version of the program by *linking* together the compiled versions, also using the `gcc` command. This is done as follows.

To separately compile `main.c`, you would type

```
gcc  -c main.c
```

The "`-c`" option indicates to the `gcc` compiler that you want just the compilation of the file `main.c`. If the compiler reports syntax errors, you would edit `main.c` to eliminate them and use the separate compilation command again. You would repeat this process until the compiler completes its task without any syntax errors. At that time, the compiler stores the compiled version of the file (the **object code** for the source) in the file called `main.o`.

In a similar manner, you would separately compile `max.c` and `avg.c` files. At the end of this process, you would have eliminated syntax errors from all the files and have the files `main.o`. `max.o` and `avg.o` in your directory. (Of course, you will also have the source files `main.c`, `max.c` and `avg.c`.)

After having generated the three object files, you can produce an executable version (`a.out`) of the program using the following Unix command:

```
gcc  main.o  max.o   avg.o
```

When all the files specified in the command line have the "`.o`" extension, `gcc` realizes that you are trying to create an executable version by linking the necessary object files and performs that task.

If you encounter run-time errors while executing `a.out`, you will need to edit the corresponding source files and separately compile them. You would then re-create the executable file `a.out` by linking together the object files.