Overview

* There are four fundamental tasks in the creation of any C program
* Editing
* Compiling
* Linking
* Executing
* These tasks will become second nature to you because you will be doing it so often
* The processes of editing, compiling, linking and executing are essentially the same for developing programs in any environment with any compiled language
* Editing is the process of creating and modifying your C source code
* Source code is inside a file and contains the program instructions you write
* choose a wise name for your base file name (all source files end in the .c extension)
* We will use an IDE (code blocks) for this class, but you can use any editor (notepad, etc) to create your source code
* a compiler converts your source code into machine language and detects and reports errors in your code
* the input to the compiler is the file you produce during your editing (source file)
* compilation is a two-stage process
* the first stage is called the preprocessing phase, during which your code may modified or added to
* the second stage is the actual compilation that generates the object code
* the compiler examines each program statements and checks it to ensure that it conforms to the syntactical and semantics of the language
* can also recognize structural errors (dead code)
* does not find logic errors
* typical errors reported might be due to an expression that has unbalanced parentheses (syntactical error), or due to the use of a variable that is not “defined” (semantics error)
* After all errors are fixed, the compiler will then take each statement of the program and translate it into assembly language
* the compiler will then translate the assembly language staements into actual machine instructions
* the output from the compuler is known as object code and it is stored in files called object files (same name as source file with a.obj or .o extension)
* The standard command to comple your C programs will be cc (or the GNU compiler, which is .gcc)
* cc - c myprog.c or gcc -c myprog.c
* if you omit the -c flag, your program will automatically be linked as well
* After the program has been translated into object code, it is ready to be linked
* the purpose of the linking phase is to get the program into a final form for execution on the computer
* linking usually occurs automatically when compiling depending on what system you are on, but can sometimes be a separate command
* The linker combines the object modules generated by the compiler with additional libraries needed by the program to create the whole executable
* also detects and reports errors
* if part of your program is missing or a nonexistent library component is referenced
* Program libraries suport and extend the C language by providing routines to carry out operations that are not part of the language
* input and output libraries, mathematical libraries, string manipulation libraries
* A failure furing the linking phase means that once again you have to go back and edit your source code
* Success will produce an executable file
* In a windows environment, the executable file will have a .exe extension
* in UNIX/Linux, there will be no such extension (a.out by default)
* Many IDEs have a build option, which will compile and link your program in a single operation to produce the executable
* A program of any significant size will consist of several source code files

Executing

Workflow

**Edit, compile, link, execute**