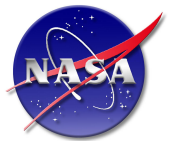
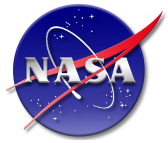


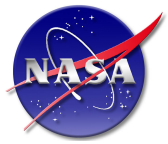
# *Compiling and Linking Overview*

***John Penn (L-3Com/ER7)***



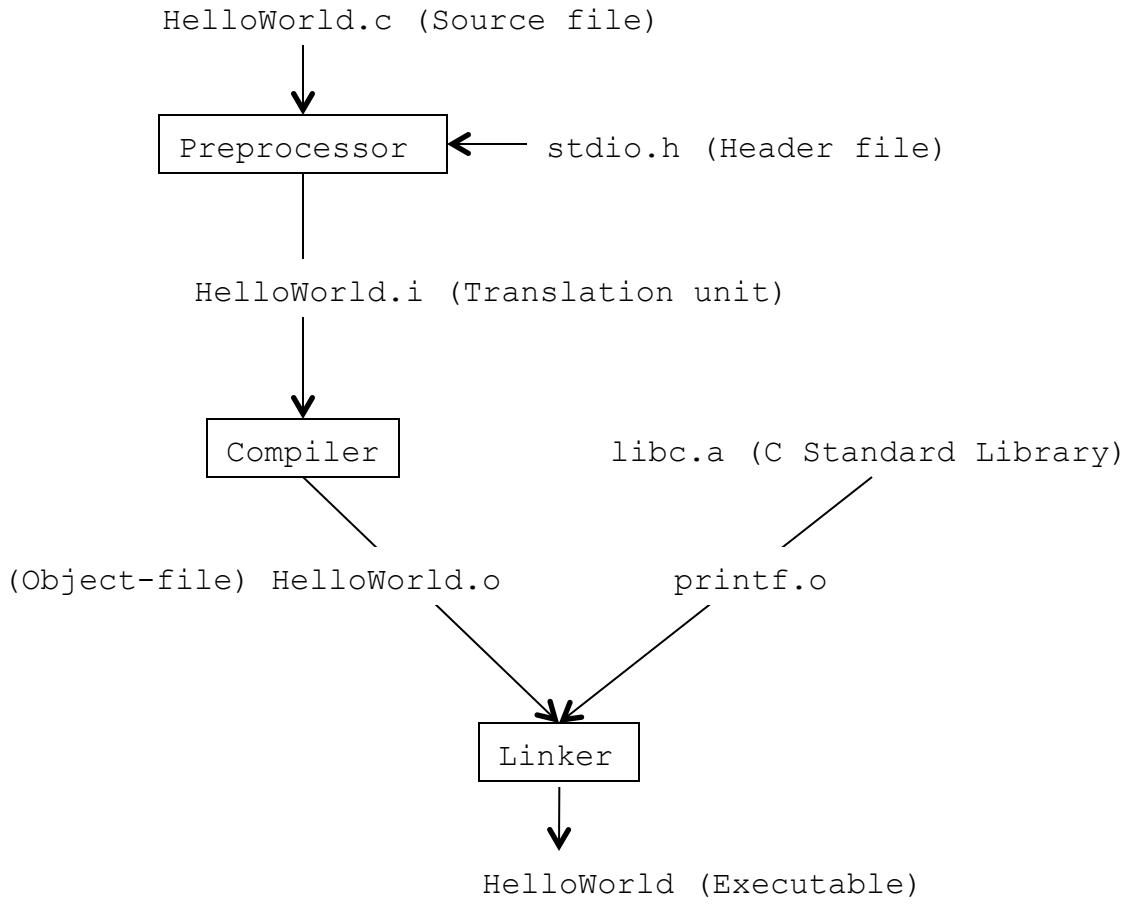
# HelloWorld.c

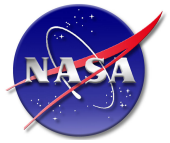
```
#include <stdio.h>

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

   printf("Hello World\n");

   return 0;

}
```



# Building a Program

**cc HelloWorld.c -o HelloWorld**





# Building a Program

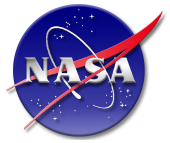
**The Preprocessor** is the first thing to run in a C compilation. It processes directives such as #include, #ifdef, #ifndef, #define, etc. It's just a text-processor.

**The Compiler** takes the output of the preprocessor and generates an object (.o) file which contains:

- Executable machine code.
- Global and static variables
- Constants and string literals
- Symbol table ( <name, address> )
- Relocation records (symbolic references to external variables and functions)

**The Linker** combines multiple object files(.o's) and/or libraries (.a's) to produce an executable file. It does so by:

- Resolving references to external symbols,
- Assigning final addresses to functions and variables,
- Updating code and data to reflect new addresses (a process called relocation).

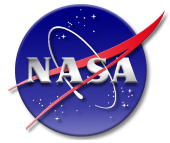


# Object files (.o)

HelloWorld.o

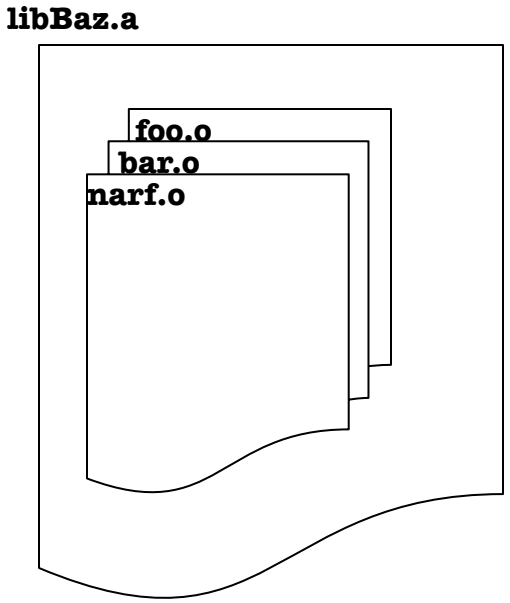
| | |
|---|---|
| •Executable machine code. | 01001010100101010110010010 10101001010101110101010101 |
| •Global and static variables | none |
| Constants and string literals | "Hello World\n" |
| •Symbol table | <main, addr of main> |
| •Relocation records | ? printf |

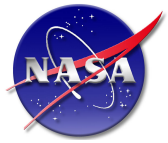
**Relocation records are the "blanks that need to be filled in".**



# *Libraries (.a files)*

**Libraries are just files that contain collections of .o files.**

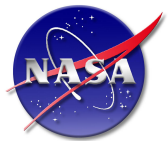




# *Linking*

**The linker looks at all of the symbol tables and the relocation record tables of a collection of object files and libraries, combines them and fills in the blanks. If all of the relocation records can be resolved then the link is successful.**

**In order for the resulting linked file to be executable, it's symbol table must define "main".**

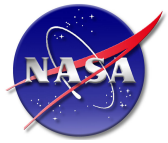


# *A difference between C and C++ linking*

**C++ function-call relocation records need to know whether they are calling a C-functions or a C++ functions. The default is to assume that a call is to a C++ function. If a function is in fact a C-function then the user needs to indicate that it's a C-function using "extern C". For example:**

```
#ifdef __cplusplus
extern "C" {
#endif

My_super_duper_C_function(void* all_the_worlds_knowledge);

#ifdef __cplusplus
}
#endif
```

**Surrounding your C-function prototypes with "#ifdef __cplusplus" stuff is generally a good idea so that it can be used from both C or C++.**



# The End