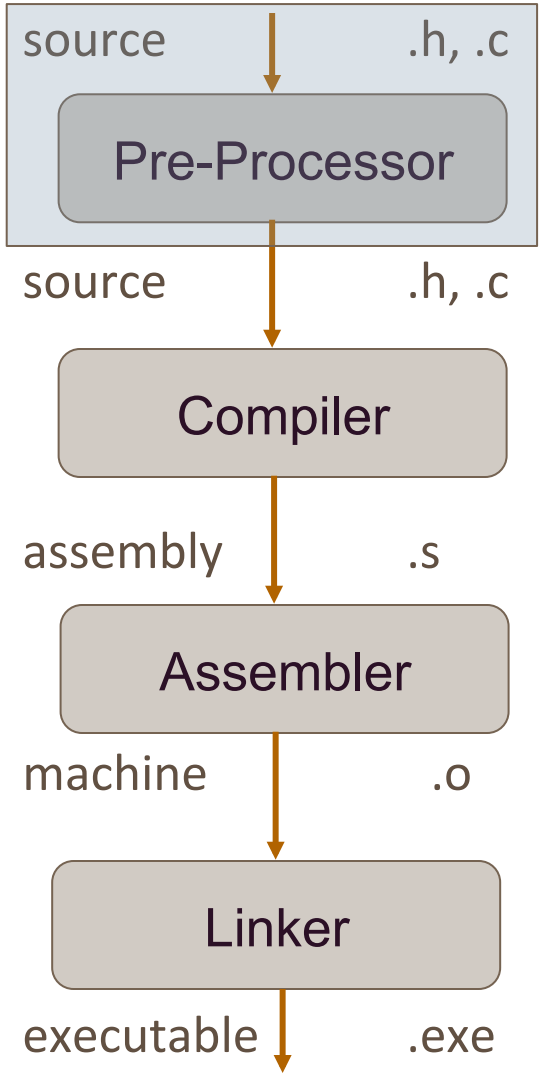
CS 261 – Data Structures

C – Compilation Process

Oregon State
UNIVERSITY

Compilation Process



Remove Comments

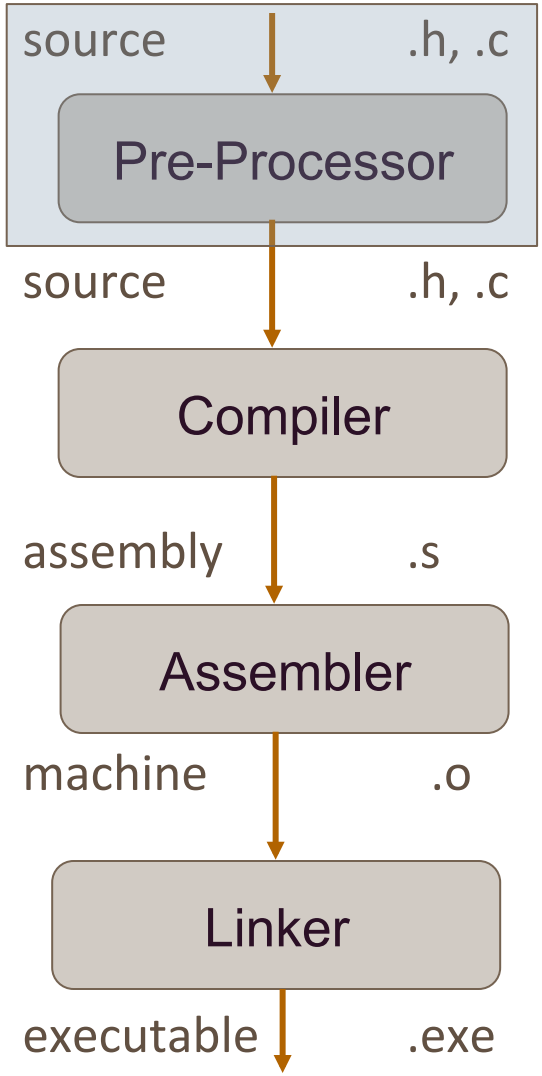
Replace Pre-processor directives (#). For example, #include copies the header file

Conditional compilation

```
#include<stdio.h>
int main(void) {
    printf("Hello world!\n");
}
```

Oregon State UNIVERSITY

Compilation Process



Take resulting source code and compiles to assembly code (.s)

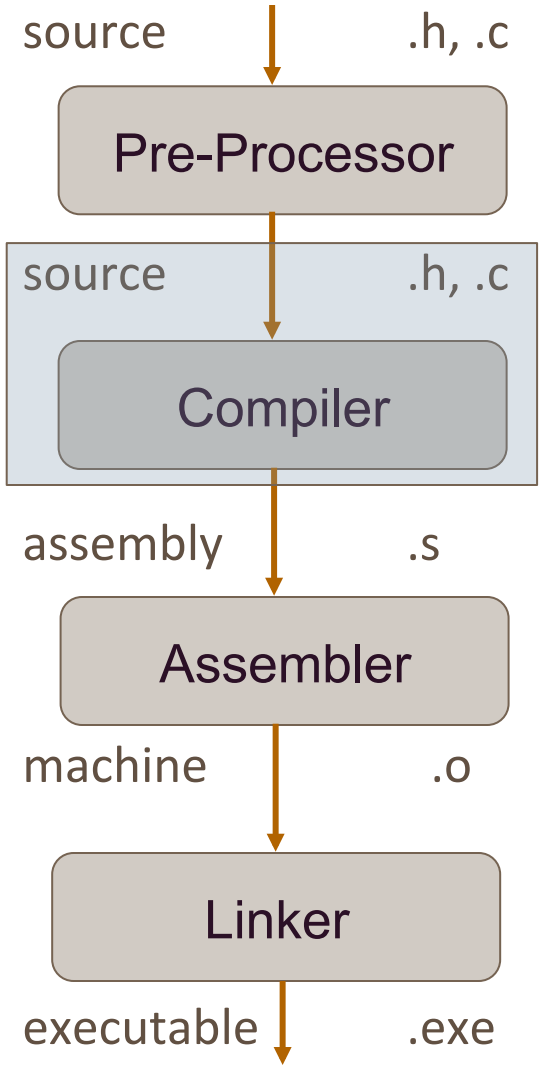
```
gcc -g -S -Wall -std=c99 -c main.c
```

Machine level code that manipulates memory and processor

```
.section    __TEXT,__text,regular,pure_instructions
        .globl      _main
        .align      4, 0x90
_main:
Leh_func_begin1:
        pushq %rbp
Ltmp0:
        movq  %rsp, %rbp
Ltmp1:
        subq  $16, %rsp
Ltmp2:
        movl  %edi, %eax
        movl  %eax, -4(%rbp)
        movq  %rsi, -16(%rbp)
        leaq  L_.str(%rip), %rax
        movq  %rax, %rdi
        callq _puts
        addq  $16, %rsp
        popq  %rbp
        ret
```

Oregon State UNIVERSITY

Compilation Process



Assembler generates object code (.o) from the assembly code.

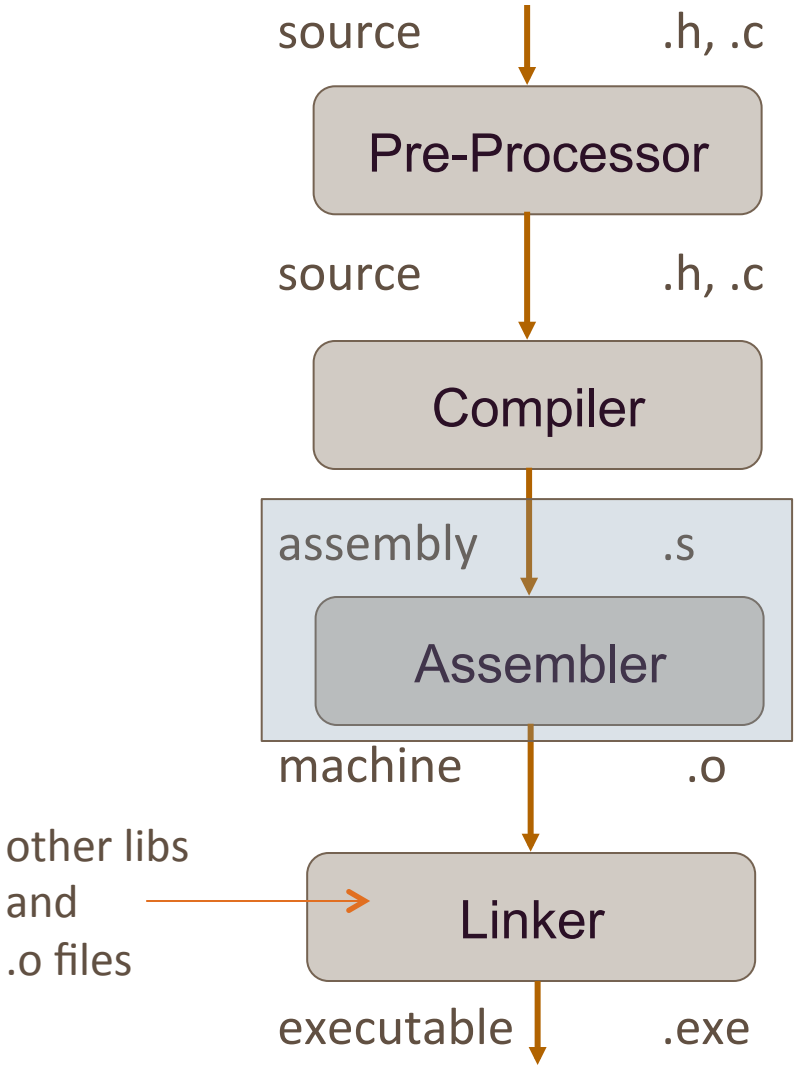
Object code is in binary and can't be viewed with a text reader

```
gcc –c main.c
gcc – c dynArr.c
```

Pre-process, compile, assemble

Oregon State UNIVERSITY

Compilation Process



The Linker pulls together your object code with libraries that you're using in your program.

In this case, we use 'printf', so it will pull in the c standard library

The result is the executable

```
#include<stdio.h>

int main(void) {

    printf("Hello world!\n");

}
```

gcc –o prog main.o dynArr.o

Oregon State UNIVERSITY

Separation of Interface and Implementation

Header files (*.h) have declarations and function prototypes (ie. interfaces)

Implementation files (*.c) have implementation source

A prototype is function header but no body (promise that the code will be linked in later)

```
int max(int a, int b);/* Function prototype */
```

Function prototypes must be terminated with a semicolon.

Allows us to 'hide' implementation in pre-compiled .o files

Oregon State UNIVERSITY

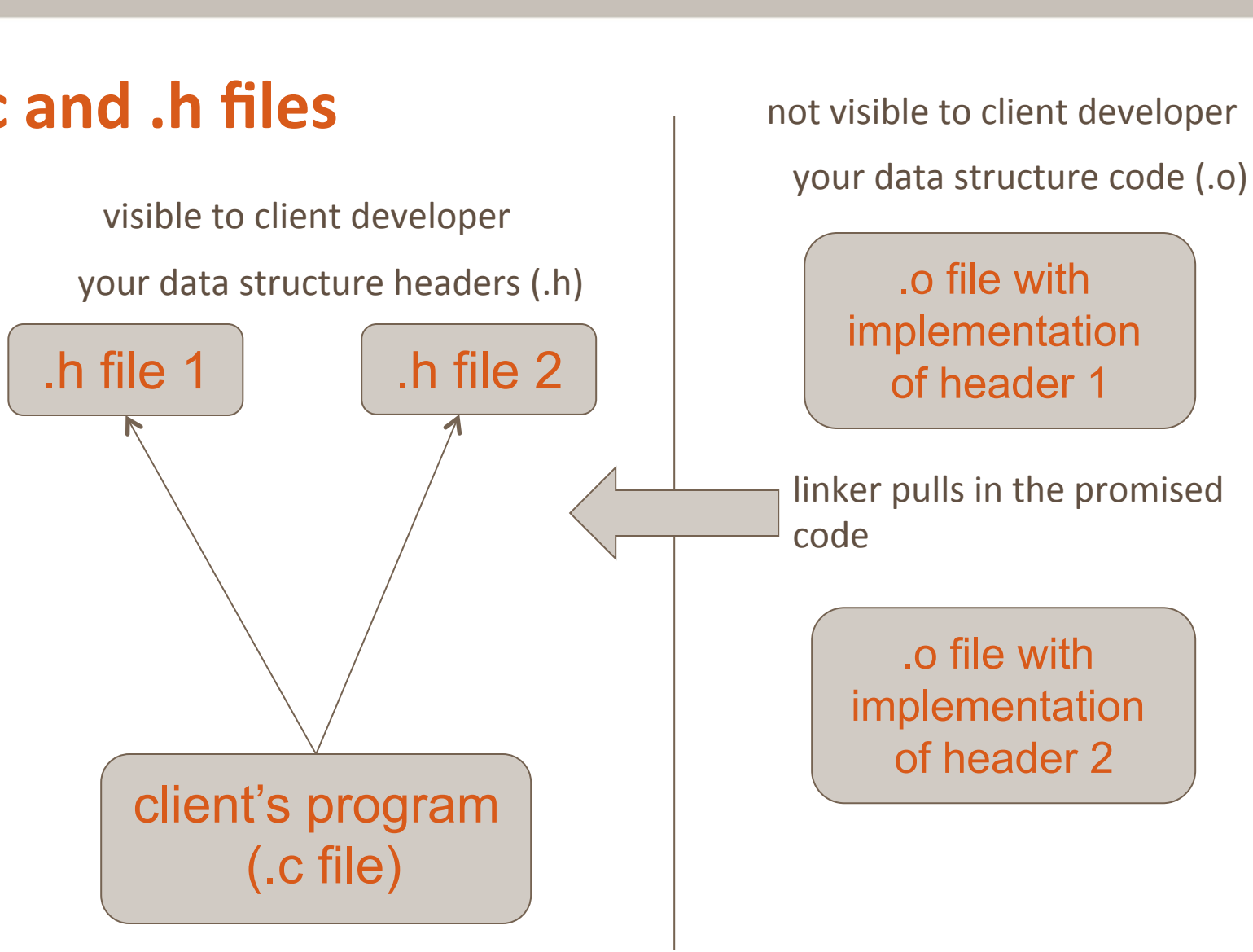
Source Files - arrayBagStack.h

```
# ifndef ArrayBagStack
# define ArrayBagStack

# define TYPE int
# define EQ(a, b) (a == b)

/* prototype for the struct, which is hidden away in the .c file */
struct arrayBagStack;

/* Bag Interface */
struct arrayBagStack *createArray();
void initArray (struct arrayBagStack * b);
void addArray (struct arrayBagStack * b, TYPE v);
int containsArray (struct arrayBagStack * b, TYPE v);
void removeArray (struct arrayBagStack * b, TYPE v);
int sizeArray (struct arrayBagStack * b);

/* Stack Interface */
void pushArray (struct arrayBagStack * b, TYPE v);
TYPE topArray (struct arrayBagStack * b);
void popArray (struct arrayBagStack * b);
int isEmptyArray (struct arrayBagStack * b);
```

Source Files - arrayBagStack.c

```c
#include "arrayBagStack.h"
#include <assert.h>
#include <stdlib.h>
#include <stdio.h>

struct arrayBagStack
{
    TYPE data[100];
    int count;
};


/* Bag Interface */

/* This function allocates space for an arrayBagStack structure! */
struct arrayBagStack * createArray()
{
    struct arrayBagStack * b = malloc(sizeof(struct arrayBagStack));

    return b;
}

void initArray (struct arrayBagStack *b)
{
```

.c and .h files



Oregon State UNIVERSITY

Tips: Ensuring Declarations Seen only Once

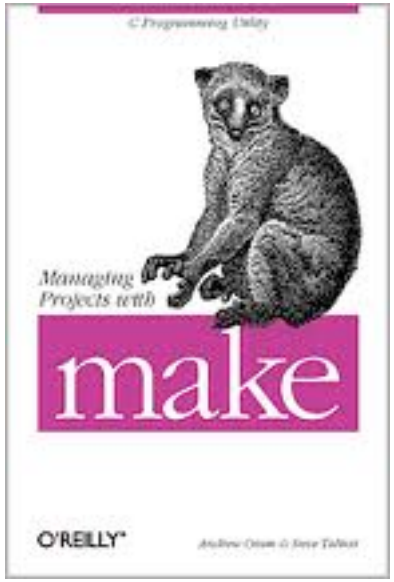
```
/* Header file for foo.h */
#ifndef BIG_COMPLEX_NAME
#define BIG_COMPLEX_NAME
…
/* Rest of foo.h file. */
#endif
```

Conditionally compiled code!!!

If `foo.h` is included more than once (e.g., through other included files), it only gets inserted into the source (.c) file once

Oregon State UNIVERSITY

GCC And Make



Managing Projects with make, 2nd Edition
The Power of GNU make for Building Anything
By Andy Oram, Steve Talbott
Publisher: O'Reilly Media
Released: October 1991
Pages: 168

Also, see the class website for much more on gcc, make and c-programming in general

Oregon State
UNIVERSITY