

Lecture 10: Building

John Wickerson

Compilers

The compilation ecosystem

C with macros



preprocessor → C



compiler

→ assembly code



assembler

→ object file



other object files, and static libraries

linker

→ executable



dynamic libraries

loader



machine code
in memory

Writing library code

average.c

```
int mean(int a, int b, int c) {  
    return (a + b + c) / 3;  
}  
  
int median(int a, int b, int c) {  
    if (a <= b && b <= c) return b;  
    if (c <= b && b <= a) return b;  
    if (a <= c && c <= b) return c;  
    if (b <= c && c <= a) return c;  
    return a;  
}  
  
int mode(int a, int b, int c) {  
    if (a == b) return a;  
    if (b == c) return b;  
    if (c == a) return c;  
    return a;  
}
```

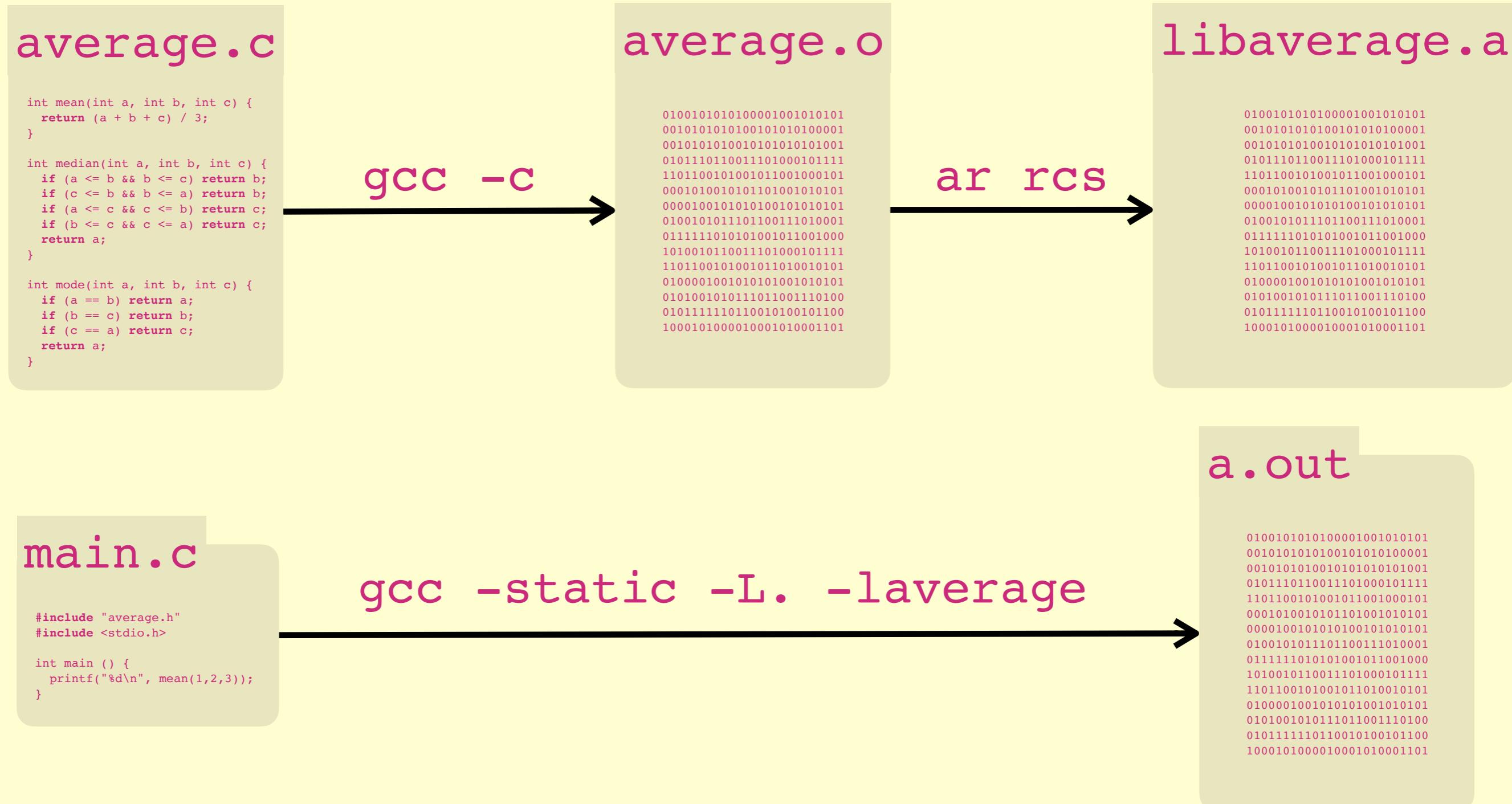
average.h

```
// Mean of three numbers  
int mean(int a, int b, int c);  
  
// Median of three numbers  
int median(int a, int b, int c);  
  
// Mode of three numbers  
int mode(int a, int b, int c);
```

main.c

```
#include "average.h"  
#include <stdio.h>  
  
int main () {  
    printf("%d\n", mean(1,2,3));  
}
```

Building a static library



Building a dynamic library

average.c

```
int mean(int a, int b, int c) {
    return (a + b + c) / 3;
}

int median(int a, int b, int c) {
    if (a <= b && b <= c) return b;
    if (c <= b && b <= a) return b;
    if (a <= c && c <= b) return c;
    if (b <= c && c <= a) return c;
    return a;
}

int mode(int a, int b, int c) {
    if (a == b) return a;
    if (b == c) return b;
    if (c == a) return c;
    return a;
}
```

gcc -fPIC -c

average.o

```
0100101010100001001010101  
0010101010100101010100001  
001010101001010101010001  
0101110110011101000101111  
110110010101011001000101  
0001010010101101001010101  
0000100101010100101010101  
0100101011101100111010001  
011111010101001011001000  
1010010110011101000101111  
1101100101001011010010101  
0100001001010100101010101  
0101001010111011001110100  
010111110110010100101100  
1000101000010001010001101
```

gcc -shared

libaverage.so

```
0100101010100001001010101  
0010101010100101010100001  
001010101001010101010001  
0101110110011101000101111  
1101100101001011001000101  
0001010010101101001010101  
0000100101010100101010101  
0100101011101100111010001  
0100101011101100111010001  
011111010101001011001000  
1010010110011101000101111  
1101100101001011010010101  
0100001001010100101010101  
0101001010111011001110100  
010111110110010100101100  
1000101000010001010001101
```

main.c

```
#include "average.h"
#include <stdio.h>

int main () {
    printf("%d\n", mean(1,2,3));
}
```

gcc -Wl,R. -L. -lavrage

a.out

```
0100101010100001001010101  
0010101010100101010100001  
001010101001010101010001  
0101110110011101000101111  
1101100101001011001000101  
0001010010101101001010101  
0000100101010010101010101  
0100101011101100111010001  
011111010101001011001000  
1010010110011101000101111  
1101100101001011010010101  
0100001001010100101010101  
0101001010111011001110100  
010111110110010100101100  
1000101000010001010001101
```

Building manually

- With static linking:

- gcc -c average.c -o average.o
 - ar rcs libaverage.a average.o
 - gcc main.c -static -L. -laverage
- compile average.c
make a static Library
compile main.c and link it with library

- With dynamic linking:

- gcc -c -fPIC average.c -o average.o
 - gcc -shared average.o -o libaverage.so
 - gcc main.c -Wl,-R. -L. -laverage
- compile average.c
make a shared library

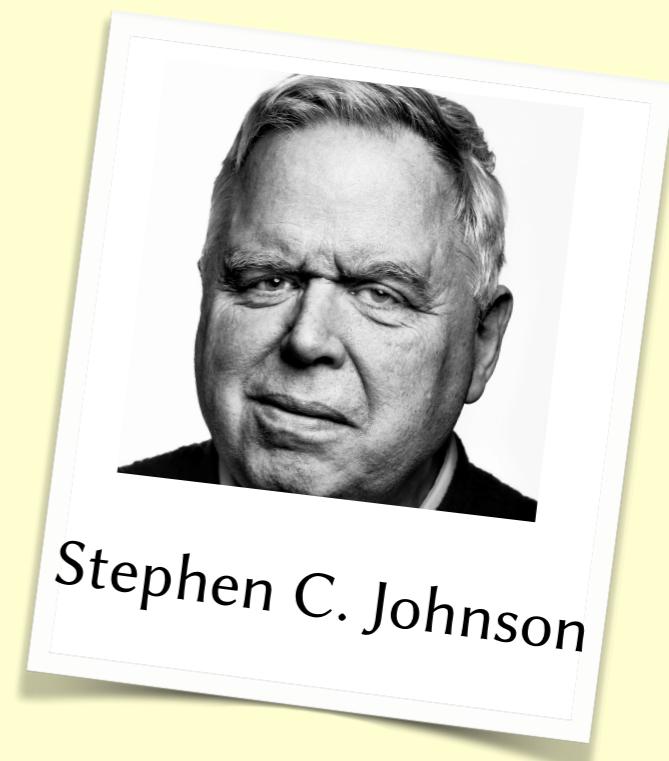
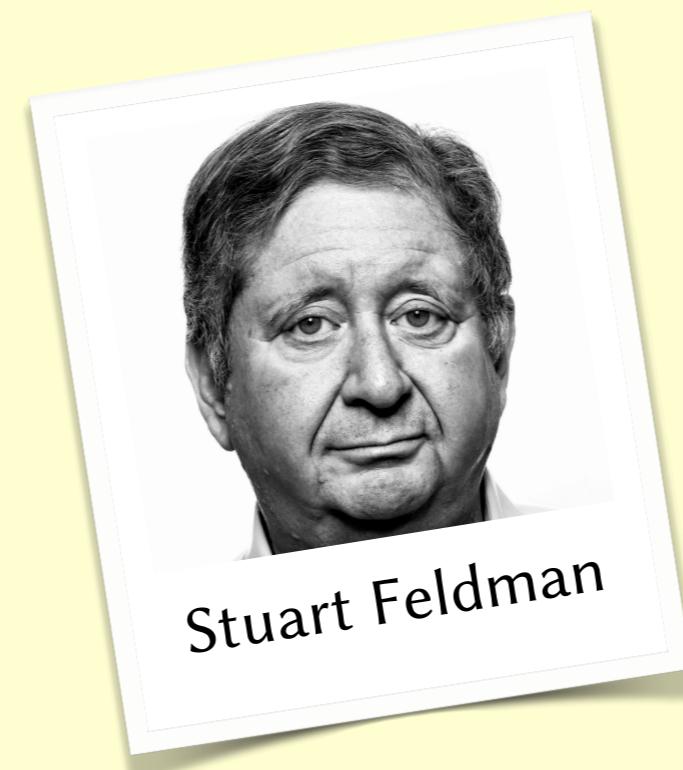
- Then later on:

- ./a.out

compile main.c and tell it where to find library at runtime

Building automatically

- Make was written in 1976 at AT&T Labs by Stuart Feldman.



Turn into a Makefile

```
average.o:  
    gcc -c average.c -o average.o  
  
libaverage.a:  
    ar rcs libaverage.a average.o  
  
statically_linked:  
    gcc main.c -static -L. -lavrage -o statically_linked  
  
average_pic.o:  
    gcc -c -fPIC average.c -o average_pic.o  
  
libaverage.so:  
    gcc -shared average_pic.o -o libaverage.so  
  
dynamically_linked:  
    gcc main.c -Wl,-R. -L. -lavrage -o dynamically_linked
```

Turn into a Makefile

```
average.o: average.c
```

```
    gcc -c average.c -o average.o
```

```
libaverage.a: average.o
```

```
    ar rcs libaverage.a average.o
```

```
statically_linked: main.c libaverage.a
```

```
    gcc main.c -static -L. -lavrage -o statically_linked
```

```
average_pic.o: average.c
```

```
    gcc -c -fPIC average.c -o average_pic.o
```

```
libaverage.so: average_pic.o
```

```
    gcc -shared average_pic.o -o libaverage.so
```

```
dynamically_linked: main.c libaverage.so
```

```
    gcc main.c -Wl,-R. -L. -lavrage -o dynamically_linked
```

Turn into a Makefile

```
static: statically_linked
dynamic: dynamically_linked

average.o: average.c
    gcc -c average.c -o average.o

libaverage.a: average.o
    ar rcs libaverage.a average.o

statically_linked: main.c libaverage.a
    gcc main.c -static -L. -lavrage -o statically_linked

average_pic.o: average.c
    gcc -c -fPIC average.c -o average_pic.o

libaverage.so: average_pic.o
    gcc -shared average_pic.o -o libaverage.so

dynamically_linked: main.c libaverage.so
    gcc main.c -Wl,-R. -L. -lavrage -o dynamically_linked
```

Make it shorter

```
static: statically_linked  
dynamic: dynamically_linked
```

```
average.o: average.c  
    gcc -c $< -o $@
```

```
libaverage.a: average.o  
    ar rcs $@ $<
```

```
statically_linked: main.c libaverage.a  
    gcc main.c -static -L. -lavrage -o $@
```

```
average_pic.o: average.c  
    gcc -c -fPIC $< -o $@
```

```
libaverage.so: average_pic.o  
    gcc -shared $< -o $@
```

```
dynamically_linked: main.c libaverage.so  
    gcc main.c -Wl,-R. -L. -lavrage -o $@
```

Add macros

```
CC=gcc
```

```
CCFLAGS=-g
```

```
static: statically_linked
```

```
dynamic: dynamically_linked
```

```
average.o: average.c
```

```
$(CC) $(CCFLAGS) -c $< -o $@
```

```
libaverage.a: average.o
```

```
ar rcs $@ $<
```

```
statically_linked: main.c libaverage.a
```

```
$(CC) $(CCFLAGS) main.c -static -L. -lavrage -o $@
```

```
average_pic.o: average.c
```

```
$(CC) $(CCFLAGS) -c -fPIC $< -o $@
```

```
libaverage.so: average_pic.o
```

```
$(CC) $(CCFLAGS) -shared $< -o $@
```

```
dynamically_linked: main.c libaverage.so
```

```
$(CC) $(CCFLAGS) main.c -Wl,-R. -L. -lavrage -o $@
```

Cleaning up

```
CC=gcc  
CFLAGS=-g
```

```
static: statically_linked  
dynamic: dynamically_linked  
  
average.o: average.c  
    $(CC) $(CFLAGS) -c $< -o $@
```

```
libaverage.a: average.o  
    ar rcs $@ $<
```

```
statically_linked: main.c libaverage.a  
    $(CC) $(CFLAGS) main.c -static -L. -lavrage -o $@
```

```
average_pic.o: average.c  
    $(CC) $(CFLAGS) -c -fPIC $< -o $@
```

```
libaverage.so: average_pic.o  
    $(CC) $(CFLAGS) -shared $< -o $@
```

```
dynamically_linked: main.c libaverage.so  
    $(CC) $(CFLAGS) main.c -Wl,-R. -L. -lavrage -o $@
```

```
.PHONY: clean  
  
clean:  
    rm -f average.o  
    rm -f average_pic.o  
    rm -f libaverage.a  
    rm -f libaverage.so  
    rm -f dynamically_linked  
    rm -f statically_linked
```

Cleaning up

```
CC=gcc  
CFLAGS=-g
```

```
static: statically_linked  
dynamic: dynamically_linked  
  
average.o: average.c  
    $(CC) $(CFLAGS) -c $< -o $@
```

```
libaverage.a: average.o  
ar rcs $@ $<
```

```
statically_linked: main.c libaverage.a  
    $(CC) $(CFLAGS) main.c -static -L. -lavrage -o $@
```

```
average_pic.o: average.c  
    $(CC) $(CFLAGS) -c -fPIC $< -o $@
```

```
libaverage.so: average_pic.o  
    $(CC) $(CFLAGS) -shared $< -o $@
```

```
dynamically_linked: main.c libaverage.so  
    $(CC) $(CFLAGS) main.c -Wl,-R. -L. -lavrage -o $@
```

```
.PHONY: clean  
  
clean:  
    rm -f *.o  
    rm -f libaverage.a  
    rm -f libaverage.so  
    rm -f dynamically_linked  
    rm -f statically_linked
```

Aside

- Microsoft Excel is a build-automation system in disguise!

	A	B	C
1	42		36
2		=A1+17	
3			=C1-B2
4	=C3*A1	92	

Top tip

- Don't put files that can be generated by your Makefile into your GitHub repository.