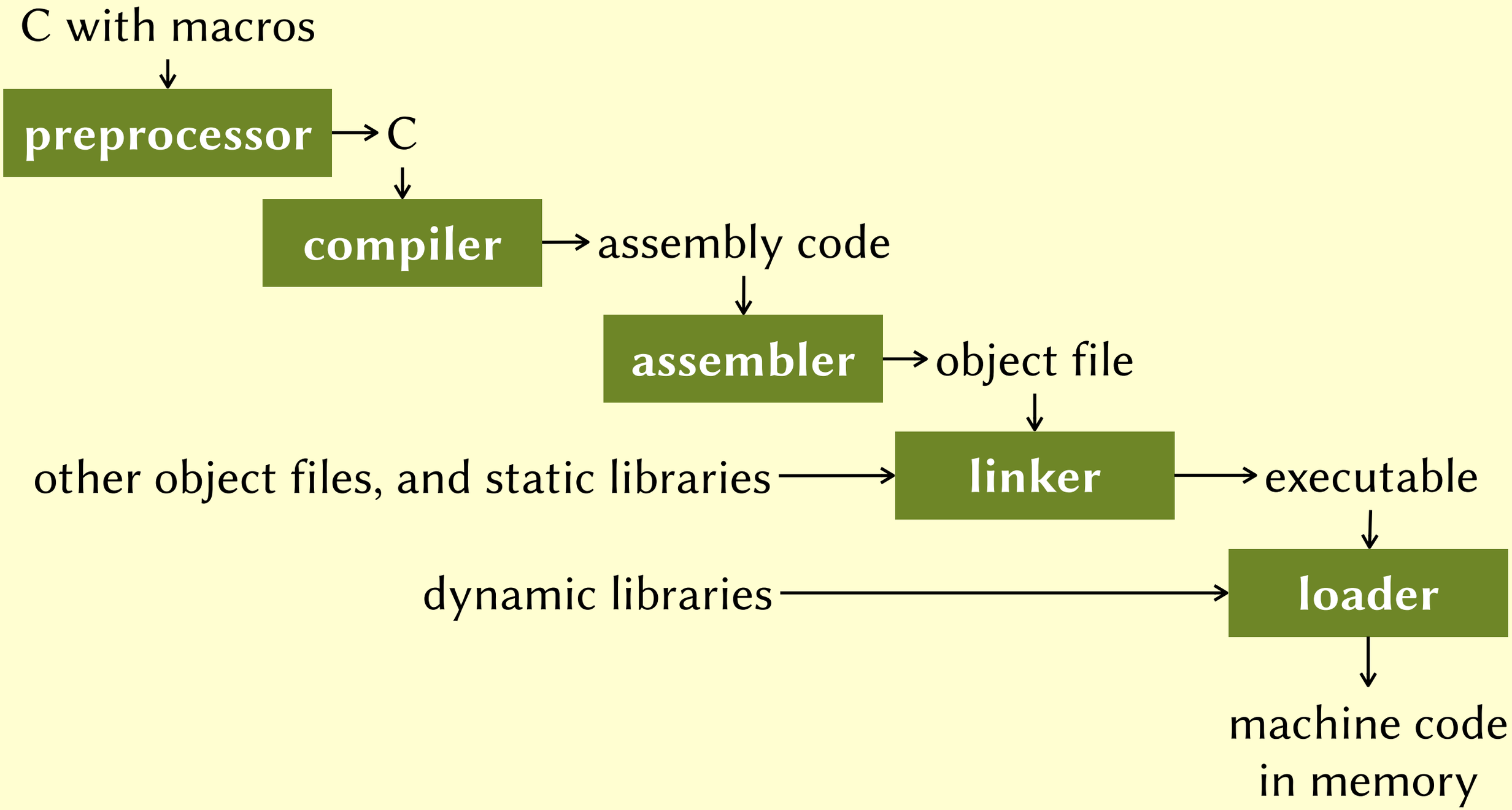
# Lecture 11: Building

John Wickerson

Compilers

# The compilation ecosystem



# 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

## 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 -c

## average.o

```
01001010101000010010101
00101010101001010100001
00101010100101010101001
01011101100111010001011 11
11011001010010110010001 01
00010100101011010010101 01
00001001010101001010101 01
01001010111011001110001 0
01111110101010011001000
10100101100111010001011 11
11011001010010110100101 01
01000010010101010010101 01
01010010101110110011101 00
01011111101100101001011 00
10001010000100010100011 01
```

ar rcs

## libaverage.a

```
01001010101000010010101
00101010101001010100001
00101010100101010101001
01011101100111010001011 11
11011001010010110010001 01
00010100101011010010101 01
00001001010101001010101 01
01001010111011001110001 0
01111110101010011001000
10100101100111010001011 11
11011001010010110100101 01
01000010010101010010101 01
01010010101110110011101 00
01011111101100101001011 00
10001010000100010100011 01
```

## main.c

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

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

gcc -static -L. -laverage

## a.out

```
01001010101000010010101
00101010101001010100001
00101010100101010101001
01011101100111010001011 11
11011001010010110010001 01
00010100101011010010101 01
00001001010101001010101 01
01001010111011001110001 0
01111110101010011001000
10100101100111010001011 11
11011001010010110100101 01
01000010010101010010101 01
01010010101110110011101 00
01011111101100101001011 00
10001010000100010100011 01
```

# 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

```
01001010101000010010101010
00101010101001010101000010
00101010100101010101010010
01011101100111010001011110
11011001010010110010001010
00010100101011010010101010
00001001010101001010101010
01001010111011001110100010
10100101100111010001011110
11011001010010110100101010
01000010010101010010101010
01010010101110110011101000
01011111101100101001011000
10001010000100010100011010
```

gcc -shared

## libaverage.so

```
01001010101000010010101010
00101010101001010101000010
00101010100101010101010010
01011101100111010001011110
11011001010010110010001010
00010100101011010010101010
00001001010101001010101010
01001010111011001110100010
01111110101010010110010000
10100101100111010001011110
11011001010010110100101010
01000010010101010010101010
01010010101110110011101000
01011111101100101001011000
10001010000100010100011010
```

## main.c

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

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

gcc -Wl,R. -L. -laverage

## a.out

```
01001010101000010010101010
00101010101001010101000010
00101010100101010101010010
01011101100111010001011110
11011001010010110010001010
00010100101011010010101010
00001001010101001010101010
01001010111011001110100010
01111110101010010110010000
10100101100111010001011110
11011001010010110100101010
01000010010101010010101010
01010010101110110011101000
01011111101100101001011000
10001010000100010100011010
```

# 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

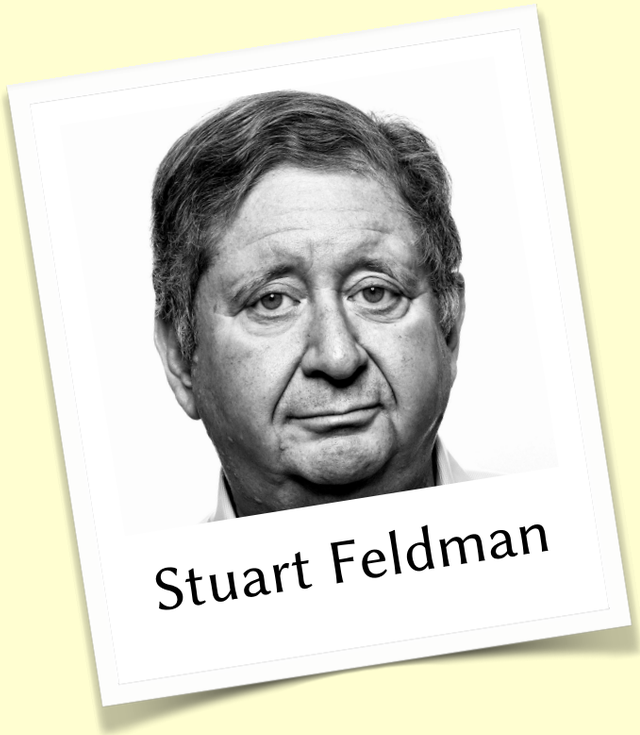
make a shared library

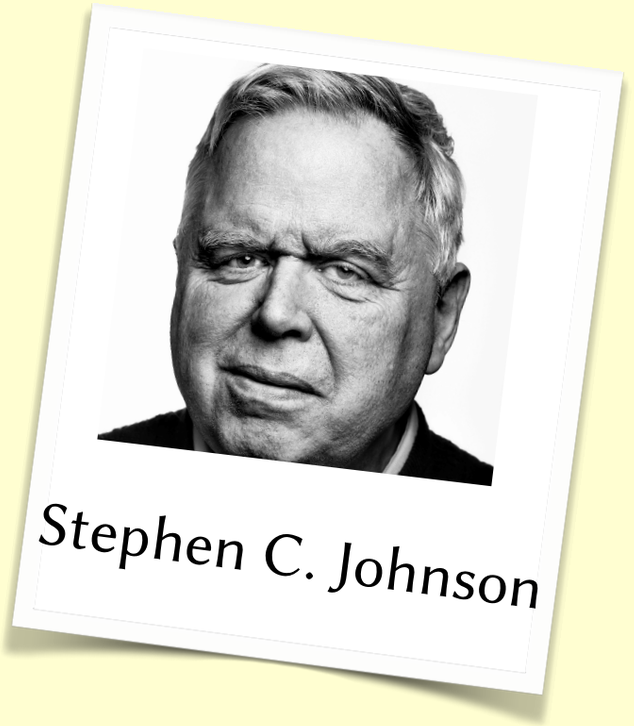
compile main.c and tell it where to find library at runtime

- Then later on:
  - `./a.out`

# Building automatically

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


Stuart Feldman


Stephen C. Johnson

# 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. -laverage -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. -laverage -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. -laverage -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. -laverage -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. -laverage -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. -laverage -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. -laverage -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. -laverage -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. -laverage -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. -laverage -o $@
```

# Cleaning up

```
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. -laverage -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. -laverage -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
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. -laverage -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. -laverage -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.