

Introduction to 8086 Assembly

Lecture 11

Modular Programming
in C and Assembly

Modular Programming



K. N. Toosi
University of Technology

test.c

```
#include <stdio.h>

extern int fact(int);

extern int maxval;

int main() {
    int x = 8;
    printf("mv=%d\n", maxval);
    printf("x!=%d\n", fact(x));

    return 0;
}
```

fact.c

```
int maxval = 2;
static int flag = 1;

int fact(int n) {
    return n==0 ? 1 : n*fact(n-1);
}

static int condmax(int a, int b) {
    return (a > b && flag) ? a : b;
}
```

Modular Programming



K. N. Toosi
University of Technology

test.c

```
#include <stdio.h>

int fact(int);

extern int maxval;

int main() {
    int x = 8;
    printf("mv=%d\n", maxval);
    printf("x!=%d\n", fact(x));

    return 0;
}
```

fact.c

```
int maxval = 2;
static int flag = 1;

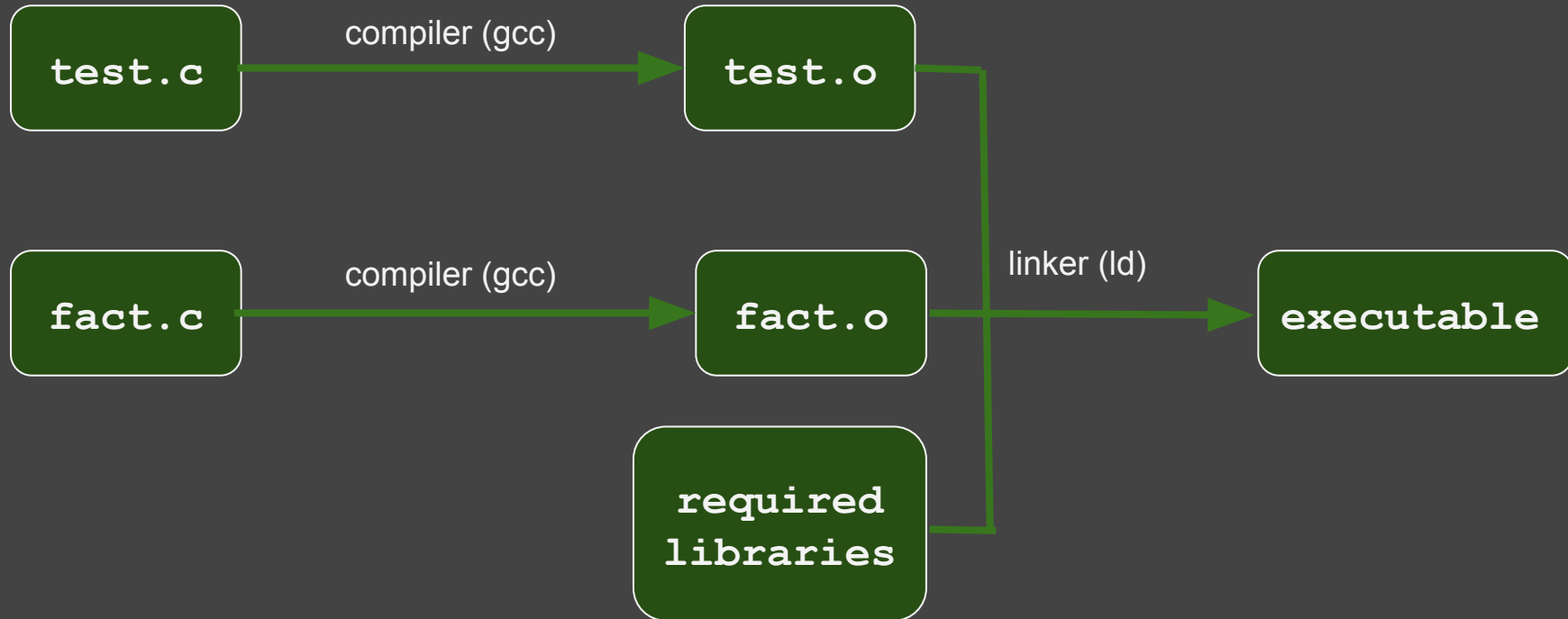
int fact(int n) {
    return n==0 ? 1 : n*fact(n-1);
}

static int condmax(int a, int b) {
    return (a > b && flag) ? a : b;
}
```

Remember: Compiling and linking C files



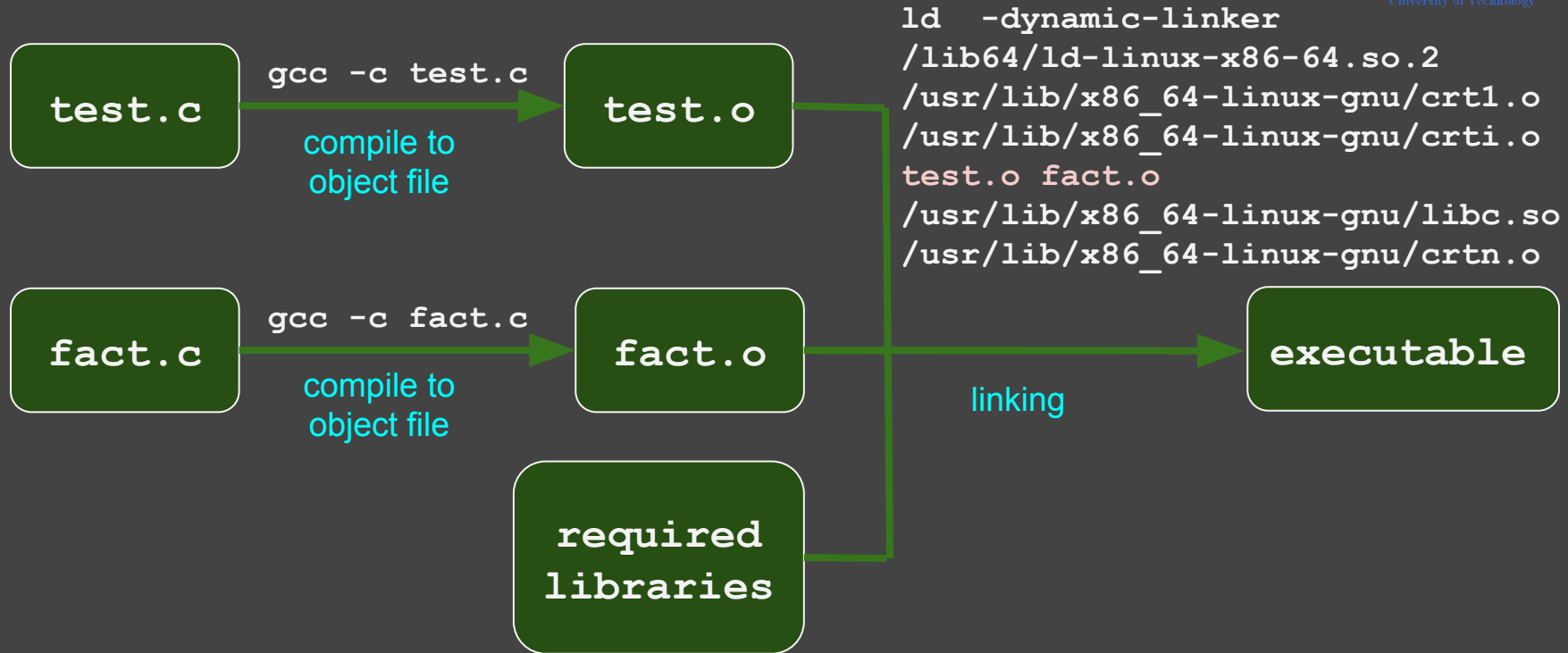
K. N. Toosi
University of Technology



Remember: Compiling and linking C files



K. N. Toosi
University of Technology



Remember: Compiling and linking C files



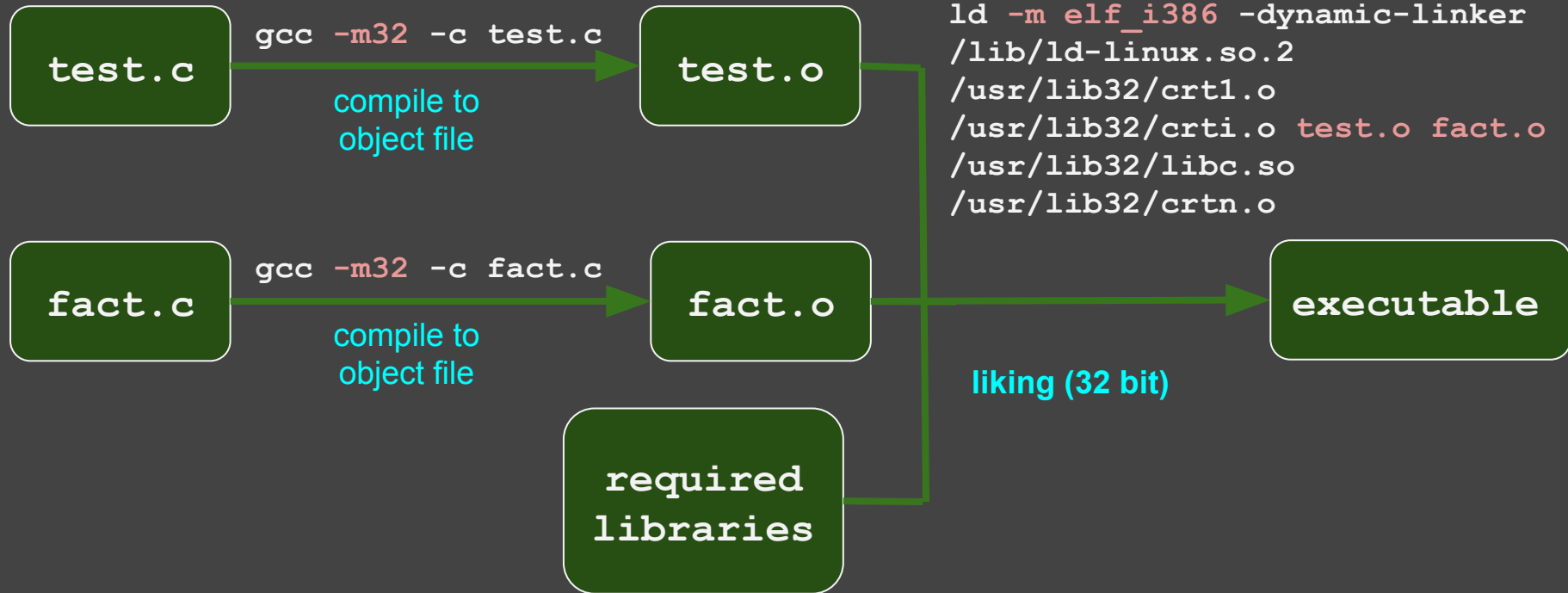
K. N. Toosi
University of Technology



32 bit Compiling and linking C files



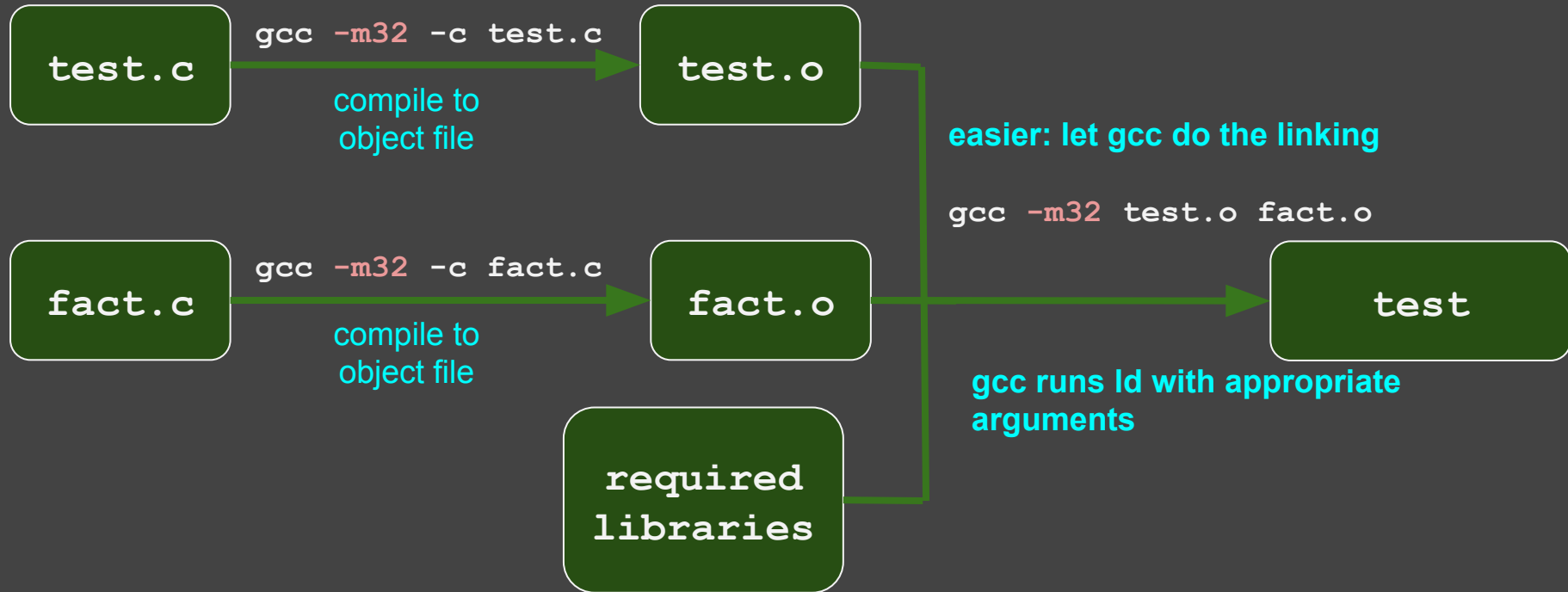
K. N. Toosi
University of Technology



32 bit Compiling and linking C files



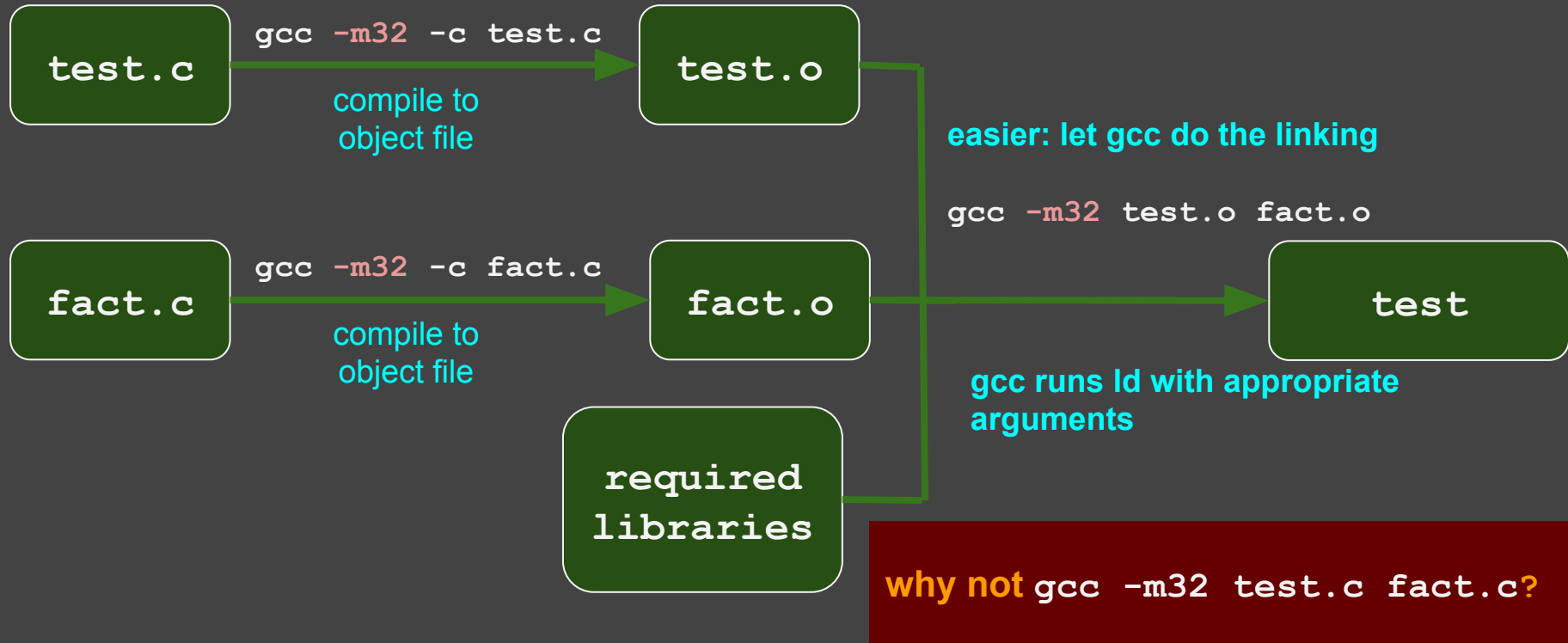
K. N. Toosi
University of Technology



32 bit Compiling and linking C files



K. N. Toosi
University of Technology



Building large projects

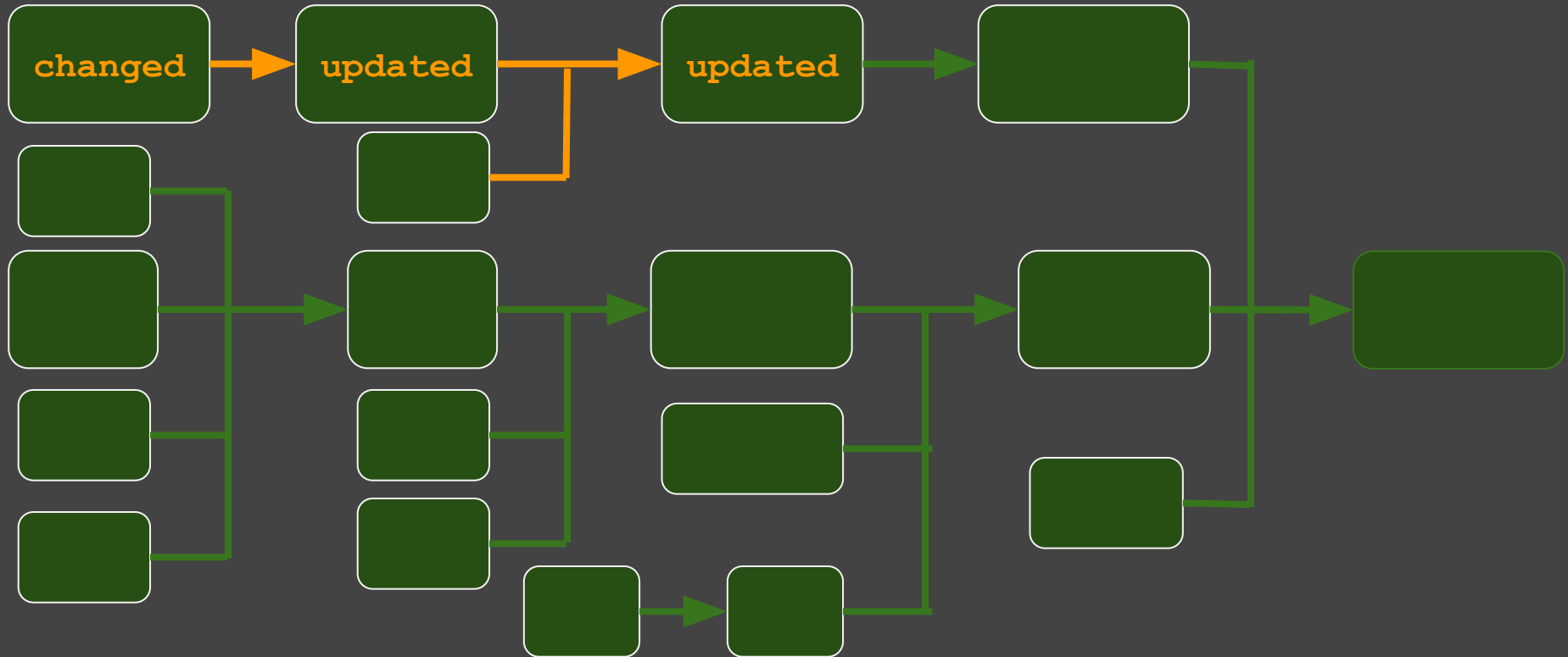
Building large projects

Building large projects

Building large projects



K. N. Toosi
University of Technology

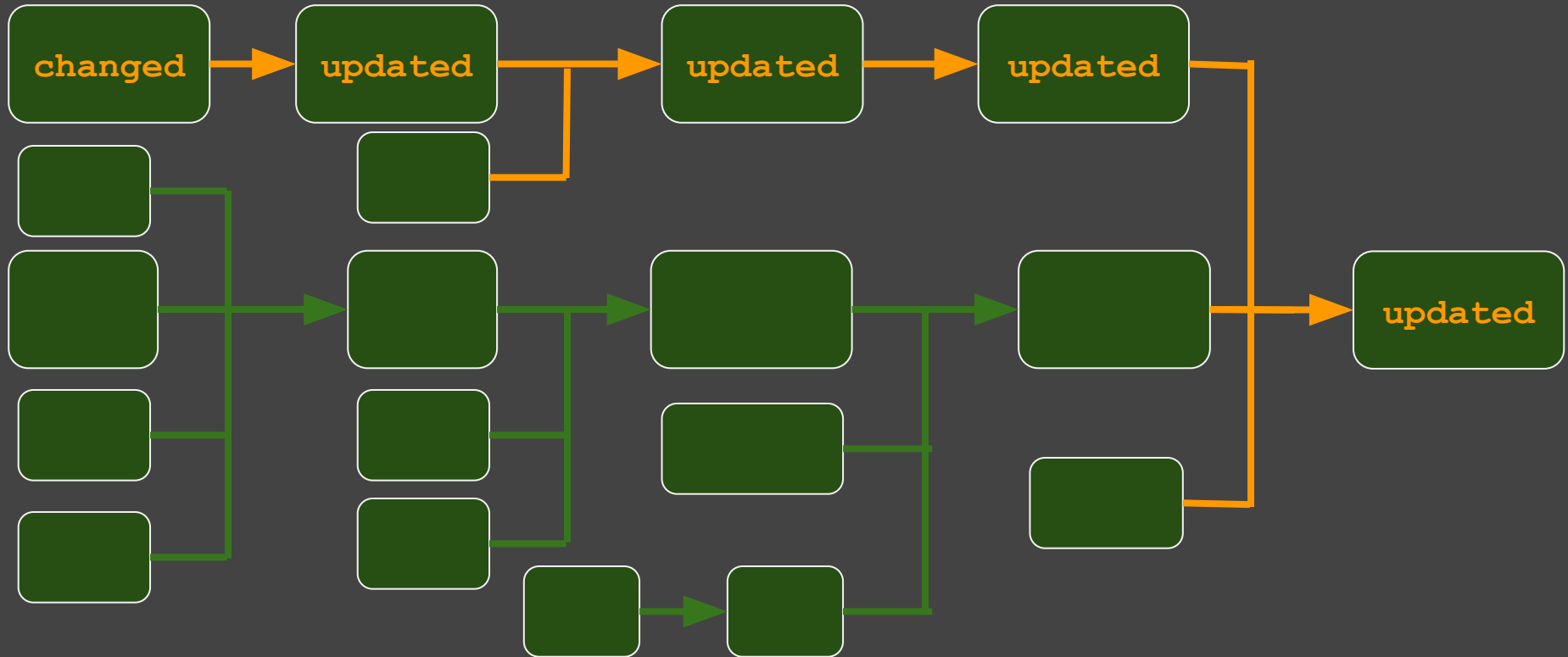


Building large projects

Building large projects



K. N. Toosi
University of Technology

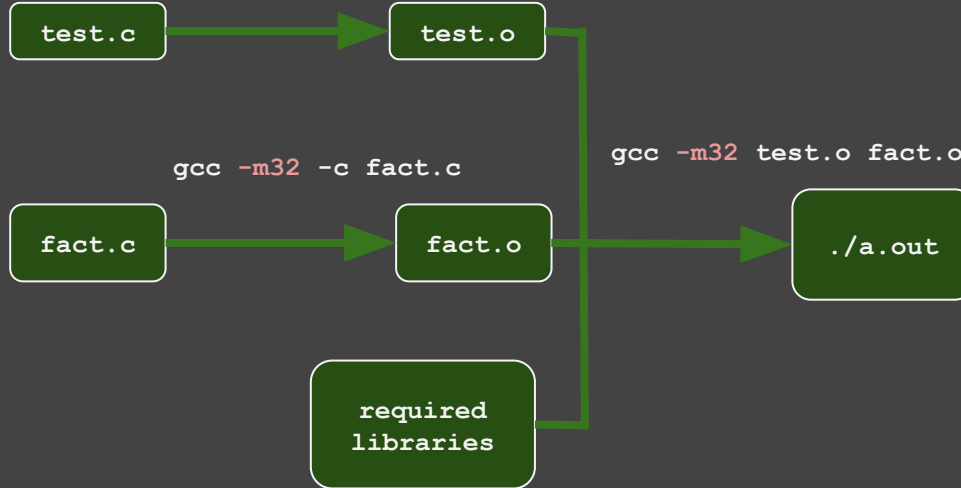


Makefile



K. N. Toosi
University of Technology

`gcc -m32 -c test.c`



target

```
a.out: test.o fact.o
    gcc -m32 test.o fact.o

test.o: test.c
    gcc -m32 -c test.c

fact.o: fact.c
    gcc -m32 -c fact.c
```

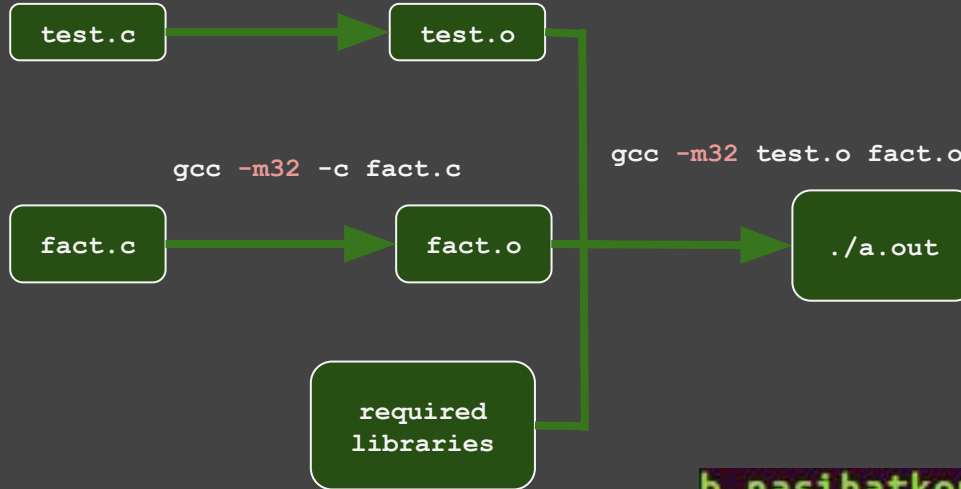
Makefile

Makefile



K. N. Toosi
University of Technology

`gcc -m32 -c test.c`



a.out: test.o fact.o

`gcc -m32 test.o fact.o`

test.o: test.c

`gcc -m32 -c test.c`

fact.o: fact.c

`gcc -m32 -c fact.c`

Makefile

`b.nasihatkon@kntu:modular_c$ make`

`gcc -m32 -c test.c`

`gcc -m32 -c fact.c`

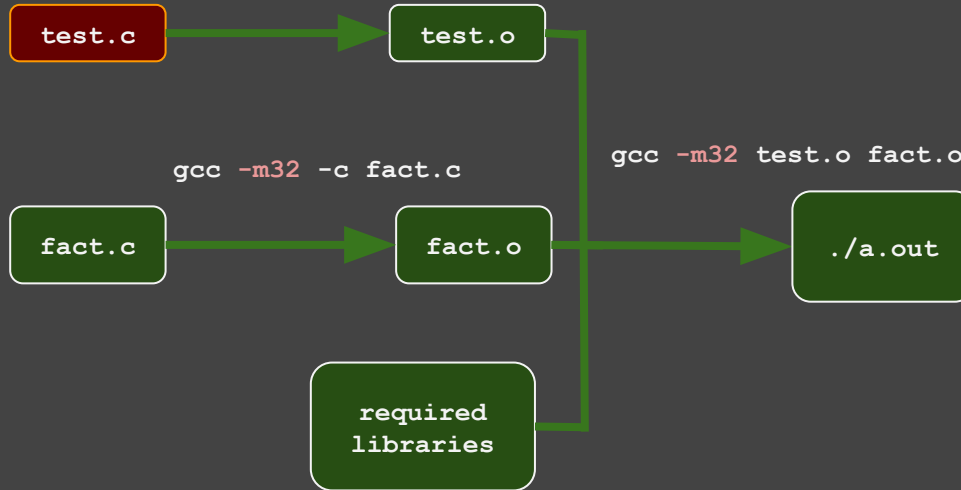
`gcc -m32 test.o fact.o`

Makefile



K. N. Toosi
University of Technology

`gcc -m32 -c test.c`



change **test.c** and run
make again:

```
a.out: test.o fact.o  
    gcc -m32 test.o fact.o
```

```
test.o: test.c  
    gcc -m32 -c test.c
```

```
fact.o: fact.c  
    gcc -m32 -c fact.c
```

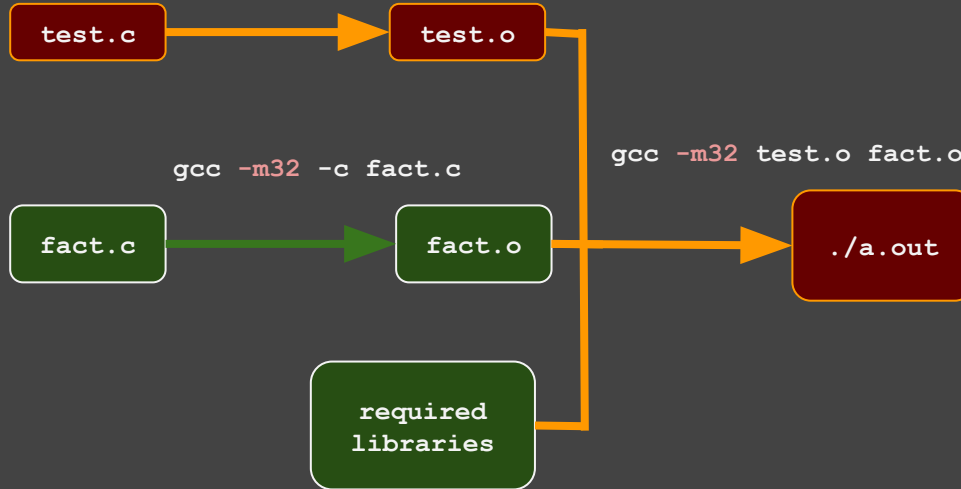
Makefile

Makefile



K. N. Toosi
University of Technology

`gcc -m32 -c test.c`



change `test.c` and run
make again:

```
a.out: test.o fact.o  
    gcc -m32 test.o fact.o
```

```
test.o: test.c  
    gcc -m32 -c test.c
```

```
fact.o: fact.c  
    gcc -m32 -c fact.c
```

Makefile

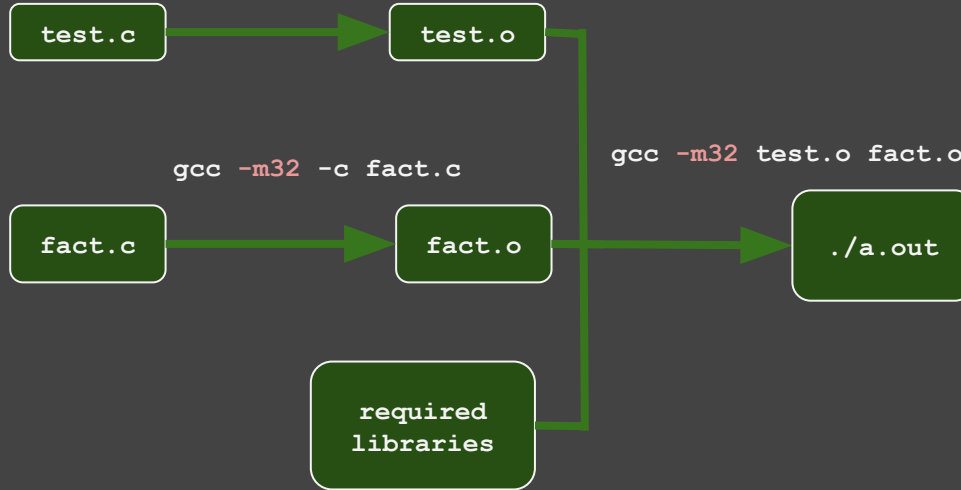
```
b.nasihatkon@kntu:modular_c$ make  
gcc -m32 -c test.c  
gcc -m32 test.o fact.o
```

Makefile



K. N. Toosi
University of Technology

`gcc -m32 -c test.c`



change nothing and
rerun make:

a.out: test.o fact.o

`gcc -m32 test.o fact.o`

test.o: test.c

`gcc -m32 -c test.c`

fact.o: fact.c

`gcc -m32 -c fact.c`

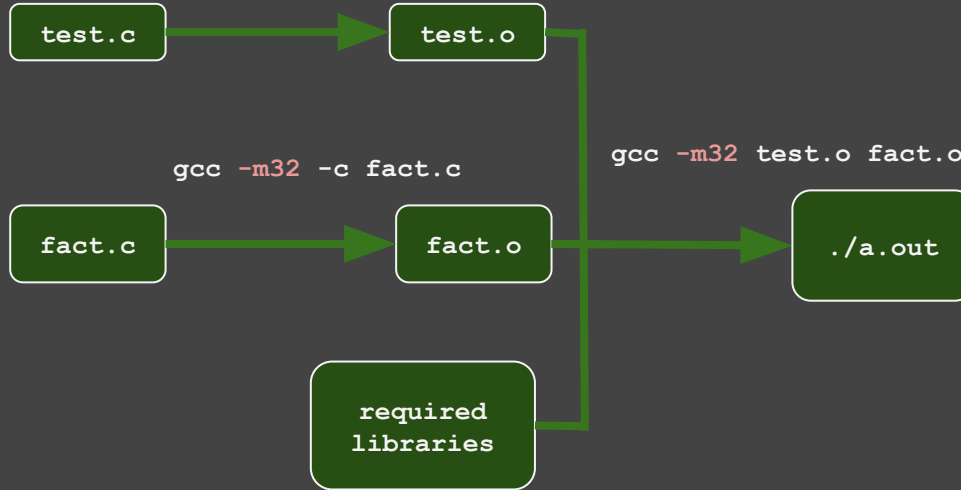
Makefile

Makefile



K. N. Toosi
University of Technology

```
gcc -m32 -c test.c
```



change nothing and
rerun make:

```
a.out: test.o fact.o  
    gcc -m32 test.o fact.o
```

```
test.o: test.c  
    gcc -m32 -c test.c
```

```
fact.o: fact.c  
    gcc -m32 -c fact.c
```

Makefile

```
b.nasihatkon@kntu:modular_c$ make  
make: 'a.out' is up to date.
```

Makefile



K. N. Toosi
University of Technology

- More on Makefile

- <http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/>
- <https://www.tutorialspoint.com/makefile/>

Modular Programming in assembly



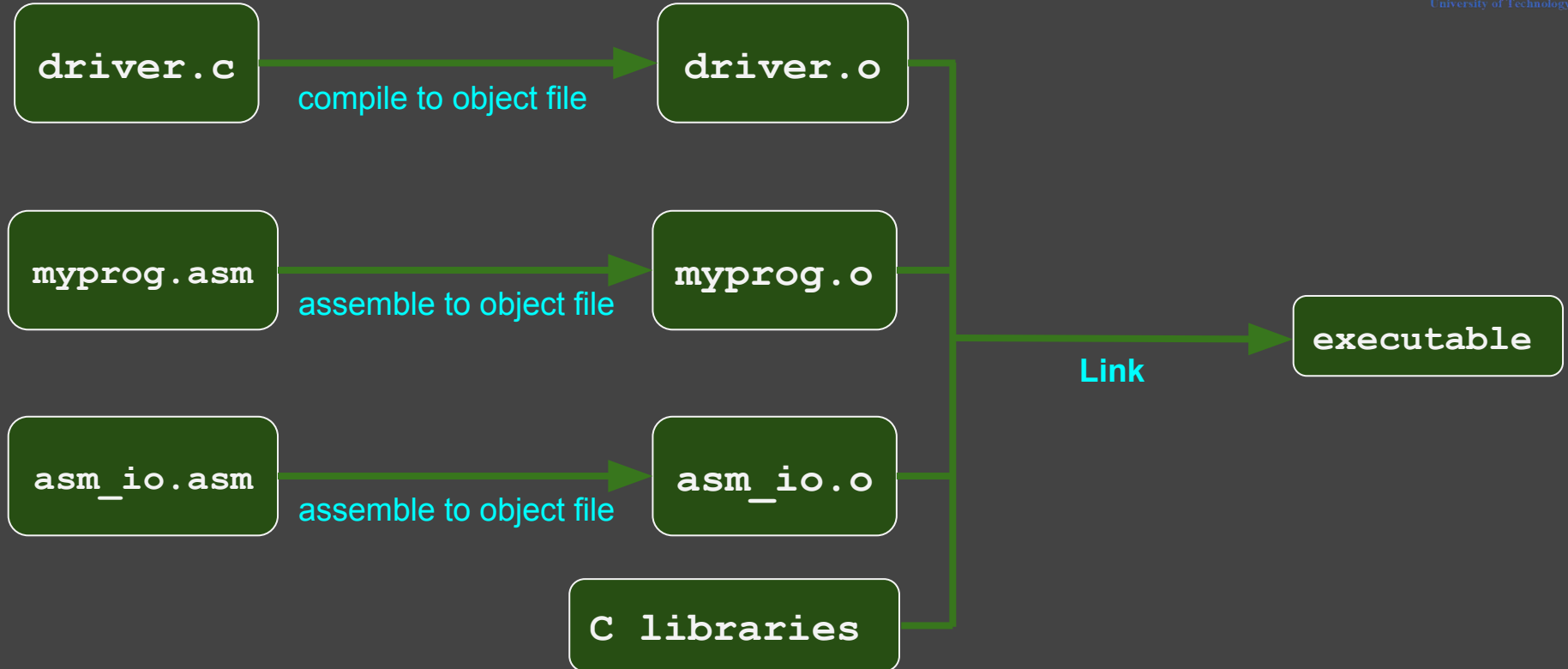
K. N. Toosi
University of Technology

- Multiple object files
- We have already done it!

We have already done it!



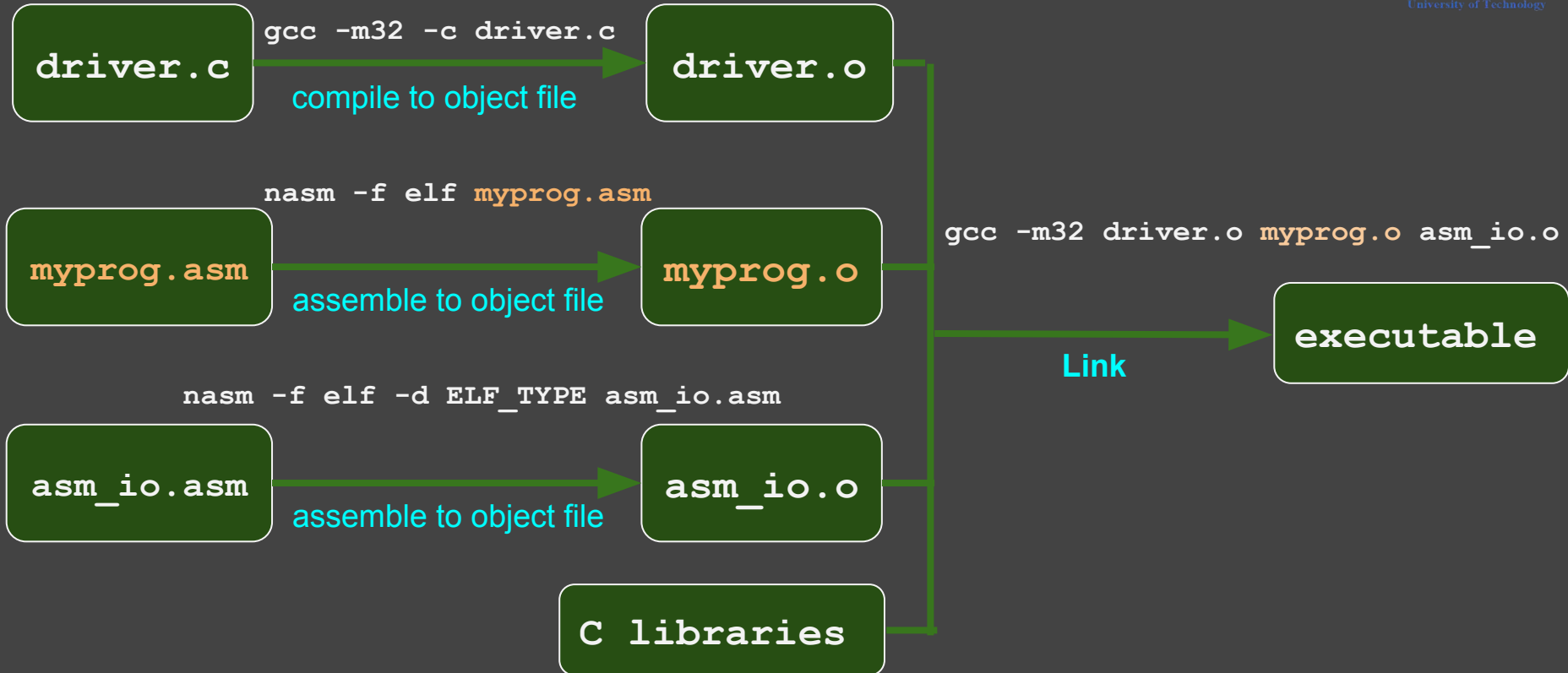
K. N. Toosi
University of Technology



We have already done it!



K. N. Toosi
University of Technology



Modular Programming in Assembly



K. N. Toosi
University of Technology

first.asm

```
extern fact, var1

segment .text

    mov eax, [var1]

    ;; compute fact(6)
    push 6
    call fact
    add esp, 4
```

second.asm

```
global fact, var1

segment .data

var1:    dd    22

segment .text

fact:
    ;; factorial function
```

Modular Programming in Assembly



K. N. Toosi
University of Technology

first.asm

```
extern fact, var1

segment .text

    mov eax, [var1]

    ;; compute fact(6)
    push 6
    call fact
    add esp, 4
```

second.asm

```
global fact, var1

segment .data

var1:    dd    22

segment .text

fact:
    ;; factorial function
```

Practice:

```
%include "asm_io.inc"
```

```
segment .text
```

```
global asm_main
```

```
extern fact, var1
```

```
asm_main:
```

```
    pusha
```

```
    mov eax, [var1]
```

```
    call print_int
```

```
    call print_nl
```

```
    ;; compute fact(6)
```

```
    push 6
```

```
    call fact
```

```
    add esp, 4
```

```
    call print_int
```

```
    call print_nl
```

```
    popa
```

```
    ret
```

first.asm

```
global fact, var1
```

```
segment .data
```

```
var1: dd 22
```

```
segment .text
```

```
fact:
```

```
    enter 0,0
```

```
    mov eax, [ebp+8]
```

```
    cmp eax, 0
```

```
    jg recur
```

```
    mov eax, 1
```

```
    jmp endfact
```

```
recur:
```

```
    dec eax
```

```
    push eax
```

```
    call fact
```

```
    add esp, 4
```

```
    imul dword [ebp+8]
```

```
endfact:
```

```
    leave
```

```
    ret
```

second.asm



K. N. Toosi
University of Technology

Practice:

```
%include "asm_io.inc"
```

first.asm

```
segment .text
```

```
global asm_main
```

```
extern fact, var1
```

```
asm_main:
```

```
pusha
```

```
mov eax, [var1]
```

```
call print_int
```

```
call print_nl
```

```
;; compute fact(6)
```

```
push 6
```

```
call fact
```

```
add esp, 4
```

```
call print_int
```

```
call print_nl
```

```
popa
```

```
ret
```

```
global fact, var1
```

second.asm

```
segment .data
```

```
var1: dd 22
```

```
segment .text
```

```
fact:
```

```
enter 0,0
```

```
mov eax, [ebp+8]
```

```
cmp eax, 0
```

```
jg recur
```

```
mov eax, 1
```

```
jmp endfact
```

```
recur:
```

```
dec eax
```

```
push eax
```

```
call fact
```

```
add esp, 4
```

```
imul dword [ebp+8]
```

```
endfact:
```

```
leave
```

```
ret
```



K. N. Toosi
University of Technology

Practice:

```
%include "asm_io.inc"
```

```
segment .text
```

```
global asm_main
```

```
extern fact, var1
```

```
asm_main:
```

```
    pusha
```

```
    mov eax, [var1]
```

```
    call print_int
```

```
    call print_nl
```

```
    ;; compute fact(6)
```

```
    push 6
```

```
    call fact
```

```
    add esp, 4
```

```
    call print_int
```

```
    call print_nl
```

```
    popa
```

```
    ret
```

first.asm

```
global fact, var1
```

```
segment .data
```

```
var1: dd 22
```

```
segment .text
```

```
fact:
```

```
    enter 0,0
```

```
    mov eax, [ebp+8]
```

```
    cmp eax, 0
```

```
    jg recur
```

```
    mov eax, 1
```

```
    jmp endfact
```

```
recur:
```

```
    dec eax
```

```
    push eax
```

```
    call fact
```

```
    add esp, 4
```

```
    imul dword [ebp+8]
```

```
endfact:
```

```
    leave
```

```
    ret
```

second.asm



K. N. Toosi
University of Technology

Practice:



K. N. Toosi
University of Technology

```
%include "asm_io.inc"  
segment .text
```

first.asm

```
global fact, var1  
segment .data  
var1: dd 22
```

second.asm

```
segment .text  
fact:
```

How to assemble & link?

```
$ nasm -f elf first.asm
```

```
$ nasm -f elf second.asm
```

```
$ gcc -m32 -o first driver.c first.o second.o asm_io.o
```

How to run?

```
$ ./first
```

```
call print_nl
```

```
popa  
ret
```

```
endfact:
```

```
leave  
ret
```

Using Makefile



K. N. Toosi
University of Technology

Makefile

```
GCC_OPTIONS= -m32
NASM_OPTIONS= -f elf

first: driver.o first.o second.o asm_io.o
    gcc $(GCC_OPTIONS) -o first driver.o first.o second.o asm_io.o

first.o: first.asm asm_io.inc
    nasm $(NASM_OPTIONS) first.asm

second.o: second.asm asm_io.inc
    nasm $(NASM_OPTIONS) second.asm

asm_io.o: asm_io.asm
    nasm $(NASM_OPTIONS) -d ELF_TYPE asm_io.asm

driver.o: driver.c
    gcc $(GCC_OPTIONS) -c driver.c
```


Using Makefile



K. N. Toosi
University of Technology

Makefile

```
GCC_OPTIONS= -m32
```

```
NASM_OPTIONS= -f elf
```

```
first: driver.o first.o second.o asm_io.o
```

```
gcc $(GCC_OPTIONS) -o first driver.o first.o second.o asm_io.o
```

```
first.o: first.asm asm_io.o
```

```
nasm $(NASM_OPTIONS) first.asm
```

```
second.o: second.asm asm_io.o
```

```
nasm $(NASM_OPTIONS) second.asm
```

```
asm_io.o: asm_io.asm
```

```
nasm $(NASM_OPTIONS) asm_io.asm
```

```
driver.o: driver.c
```

```
gcc $(GCC_OPTIONS) driver.c
```

```
b.nasihatkon@kntu:lecture11$ ls
```

1.html	driver.c	Makefile	second.asm
asm_io.asm	first	modular_c	second.asm.html
asm_io.asm.html	first.asm	README	template.asm
asm_io.inc	first.asm.html	run.sh	template.c

```
b.nasihatkon@kntu:lecture11$ make
```

```
gcc -m32 -c driver.c
```

```
nasm -f elf first.asm
```

```
nasm -f elf second.asm
```

```
nasm -f elf -d ELF_TYPE asm_io.asm
```

```
gcc -m32 -o first driver.o first.o second.o asm_io.o
```

```
b.nasihatkon@kntu:lecture11$ ./first
```

```
22
```

```
720
```

```
b.nasihatkon@kntu:lecture11$ make
```

```
make: 'first' is up to date.
```

```
b.nasihatkon@kntu:lecture11$
```

Practice:



K. N. Toosi
University of Technology

```
%include "asm_io.inc"
```

first.asm

```
segment .text
```

```
global asm_main
```

```
extern fact, var1
```

```
asm_main:
```

```
    pusha
```

```
    mov eax, [var1]
```

```
    call print_int
```

```
    call print_nl
```

```
;; compute fact(6)
```

```
    push 6
```

```
    call fact
```

```
    add esp, 4
```

```
    call print_int
```

```
    call print_nl
```

```
    popa
```

```
    ret
```

```
global fact, var1
```

second.asm

```
segment .data
```

```
var1: dd 22
```

```
segment .text
```

```
fact:
```

Why have not `print_int` and `print_nl` been defined as `extern`?

```
endfact:
```

```
    leave
```

```
    ret
```

Practice:



K. N. Toosi
University of Technology

```
%include "asm_io.inc"
```

```
segment .text
```

```
global asm_main
```

```
extern fact, var1
```

```
asm_main:
```

```
pusha
```

```
mov eax, [var1]
```

```
call print_int
```

```
call print_nl
```

```
;; compute fact(6)
```

```
push 6
```

```
call fact
```

```
add esp, 4
```

```
call print_int
```

```
call print_nl
```

```
popa
```

```
ret
```

first.asm

```
global fact, var1
```

```
segment .data
```

```
var1: dd 22
```

```
segment .text
```

```
fact:
```

second.asm

Why have not `print_int` and `print_nl` been defined as `extern`?

- Look at `asm_io.inc`

```
endfact:
```

```
leave
```

```
ret
```

Practice:



```
%include "asm_io.inc"
```

first.asm

```
segment .text
```

```
global asm_main
```

```
extern fact, var1
```

```
asm_main:
```

```
pusha
```

```
mov eax, [var1]
```

```
call print_int
```

```
call print_nl
```

```
;; compute fact(6)
```

```
push 6
```

```
call fact
```

```
add esp, 4
```

```
call print_int
```

```
call print_nl
```

```
popa
```

```
ret
```

asm_io.inc

```
extern read_int, print_int, print_uint, print_string
```

```
extern read_char, print_char, print_nl
```

```
extern sub_dump_regs, sub_dump_mem, sub_dump_math, sub_dump_stack
```

```
%macro dump_regs 1
```

```
    push    dword %1
```

```
    call    sub_dump_regs
```

```
%endmacro
```

```
; usage: dump_mem label, start-address, # paragraphs
```

```
%macro dump_mem 3
```

```
    push    dword %1
```

```
    push    dword %2
```

```
    push    dword %3
```

```
    call    sub_dump_mem
```

```
%endmacro
```

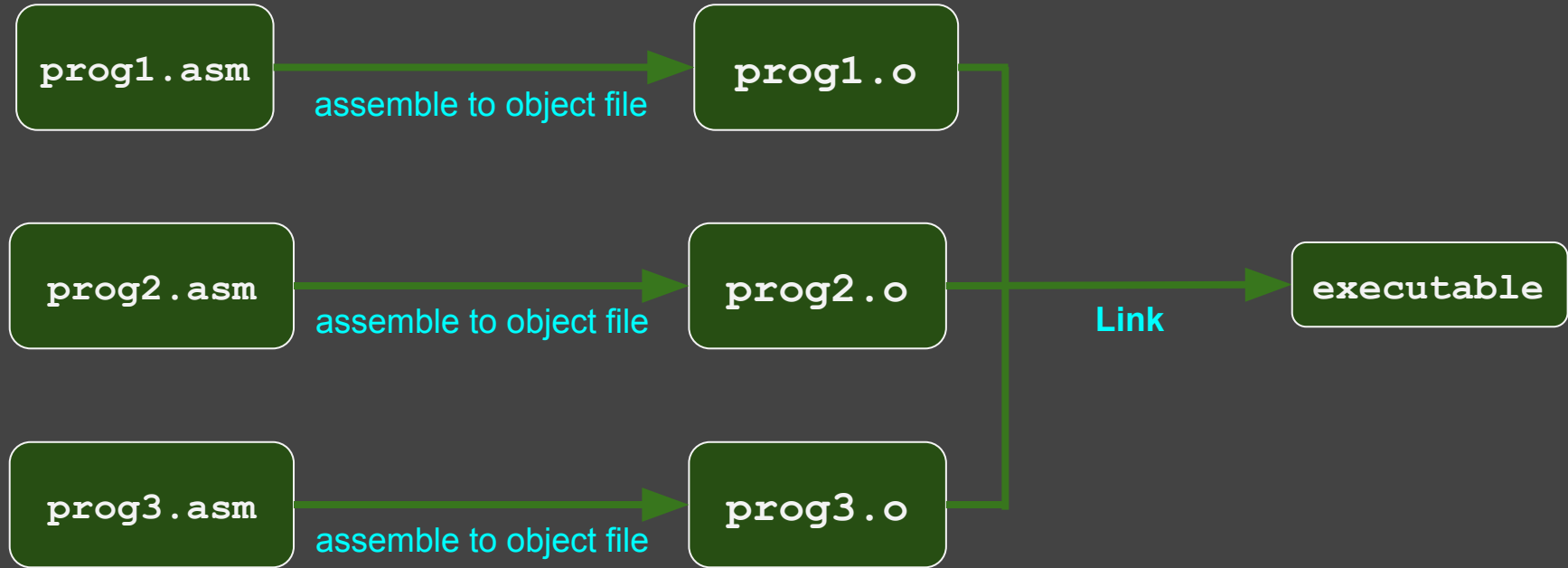
```
:
```

```
:
```

Standalone assembly programs (32-bit)



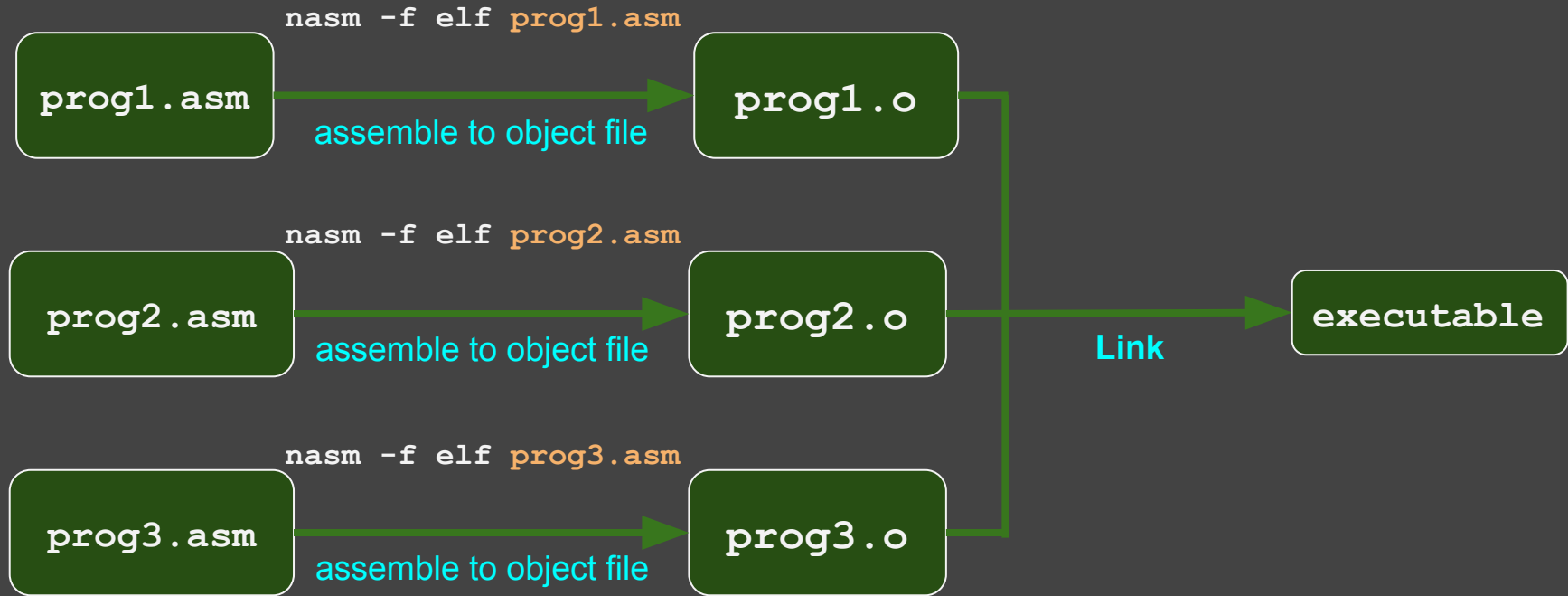
K. N. Toosi
University of Technology



Standalone assembly programs (32-bit)



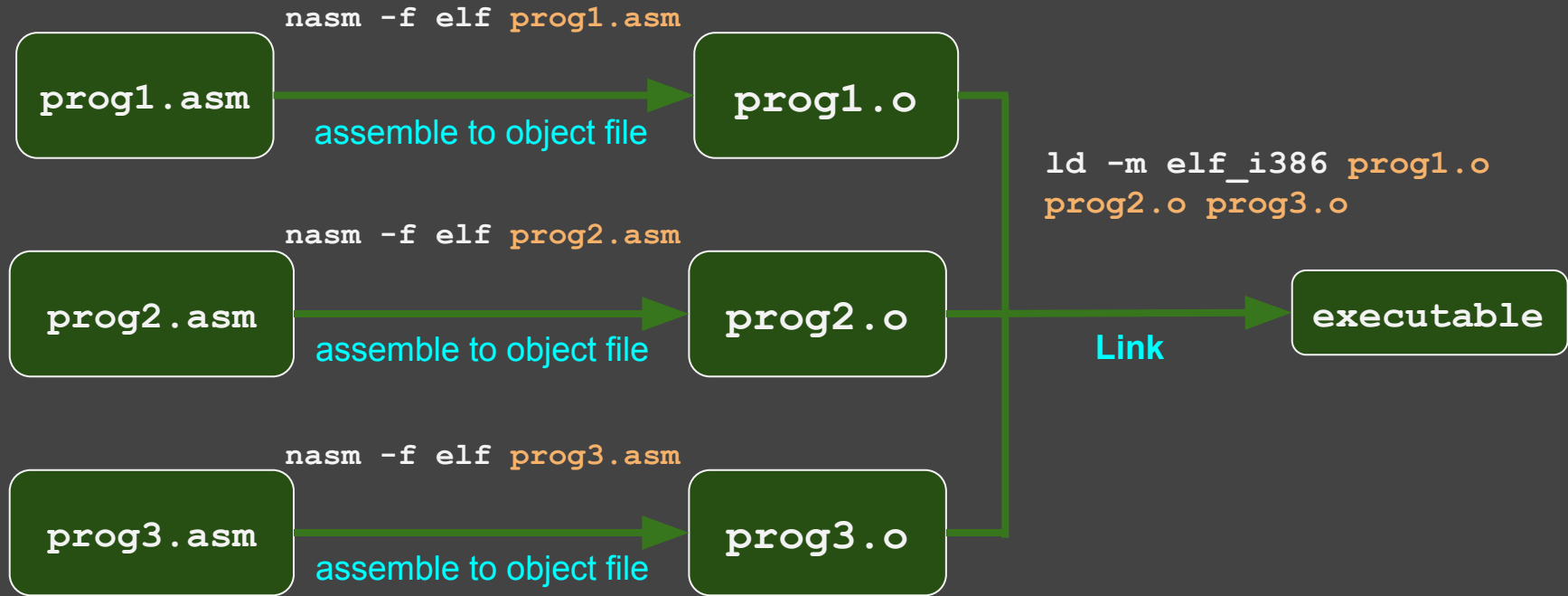
K. N. Toosi
University of Technology



Standalone assembly programs (32-bit)



K. N. Toosi
University of Technology



Standalone assembly programs (64-bit)



K. N. Toosi
University of Technology

