

# Compilação Avançada de Programas C no Linux

Professor Guilherme Abelha

# ANSI C

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

# Que assuntos serão abordados neste minicurso?

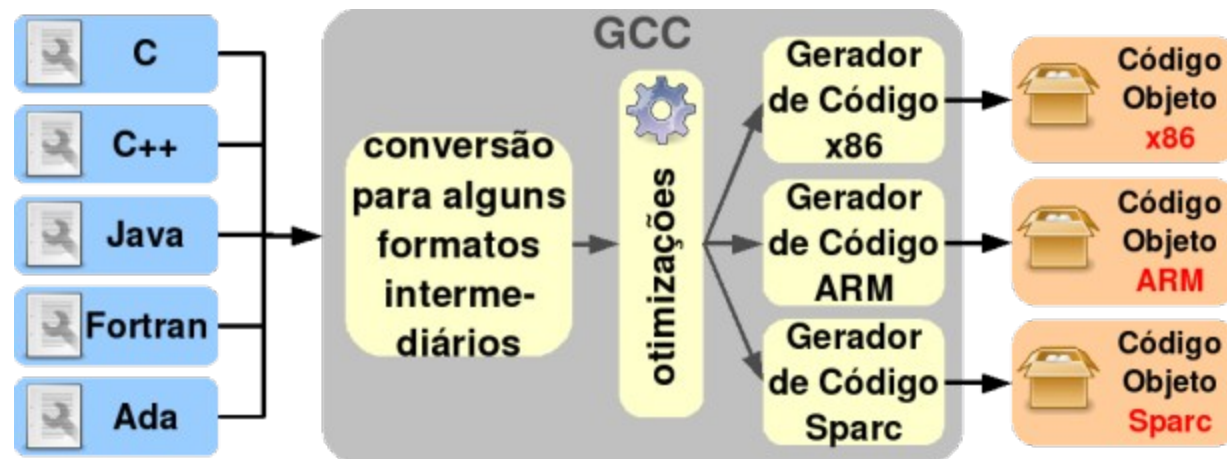
- GCC:
  - arquitetura
  - compilação
  - linkagem
  - makefile
  - debugger



# Introdução ao GCC

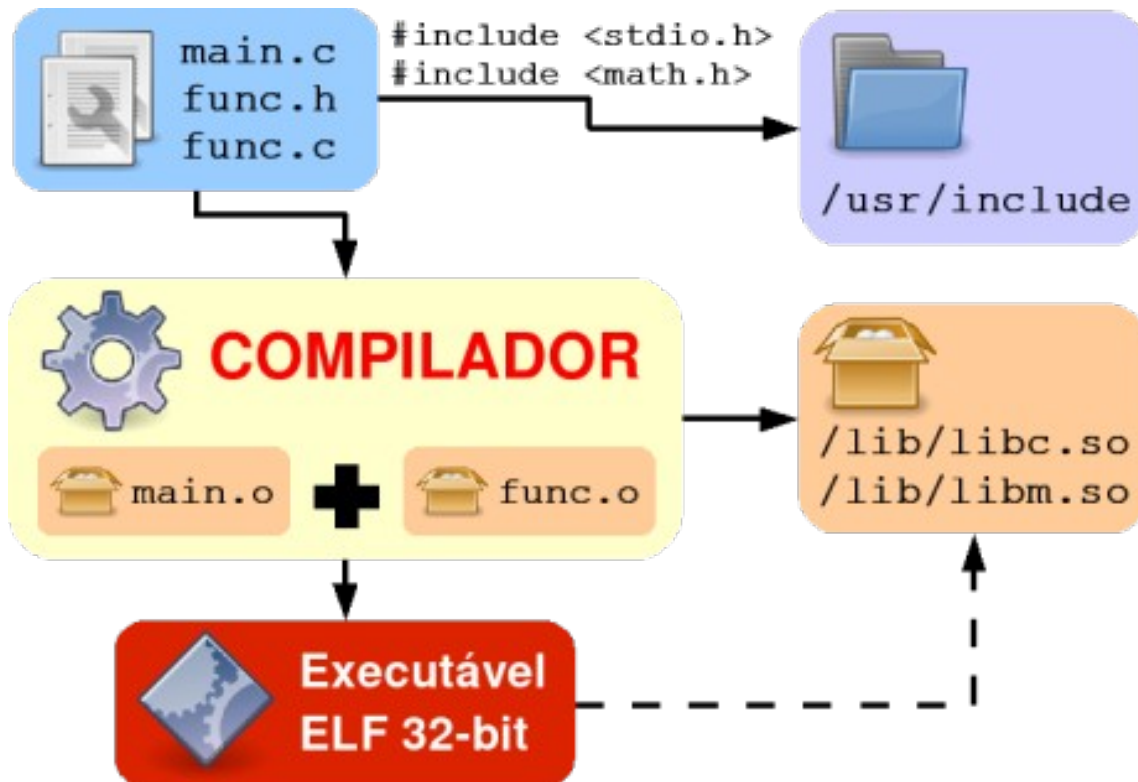
# O que é o GCC

- GNU Compiler Collection



# Visão Geral do Desenvolvimento com GCC

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

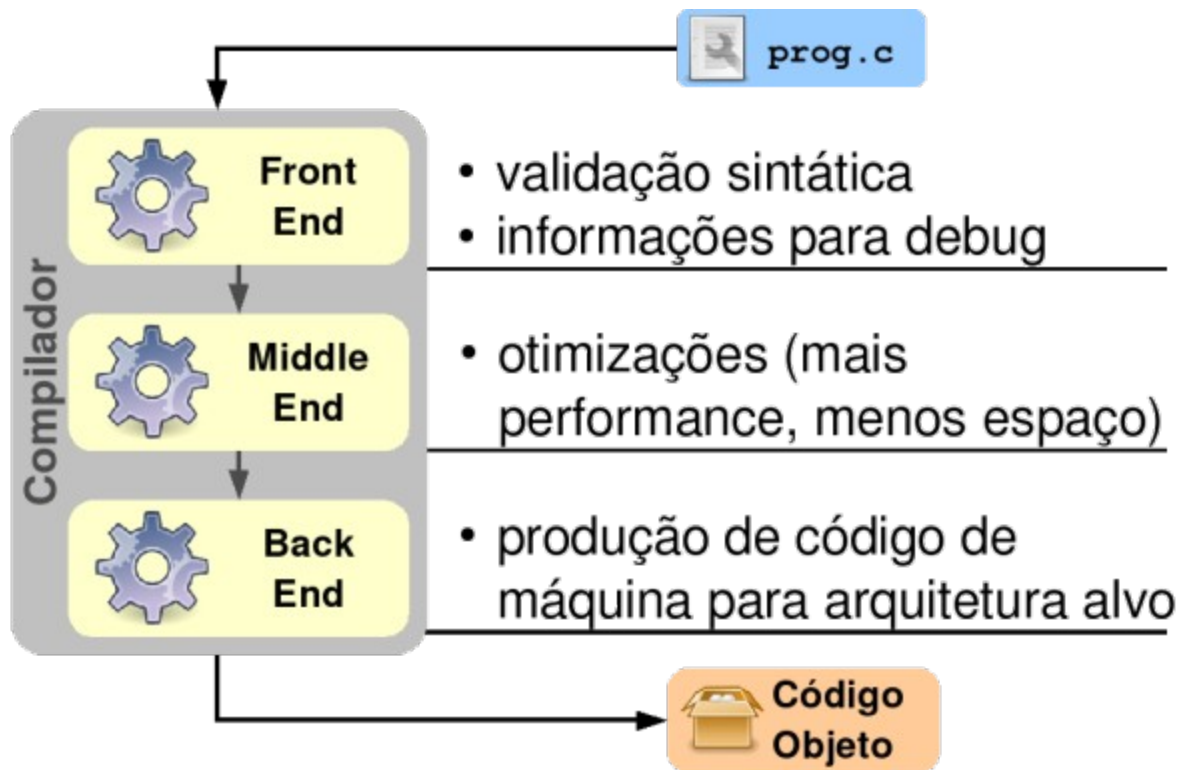


Fontes e APIs

Arquivos objeto e bibliotecas

Programa executável

# Estrutura de um Copilador do GCC



```
while (a!=EOF)
    i++;
```

```
a = 3
print a + 5 ↔ print 8
```

# **Compilação**

## **Passo a passo**

# Compilação Passo a Passo

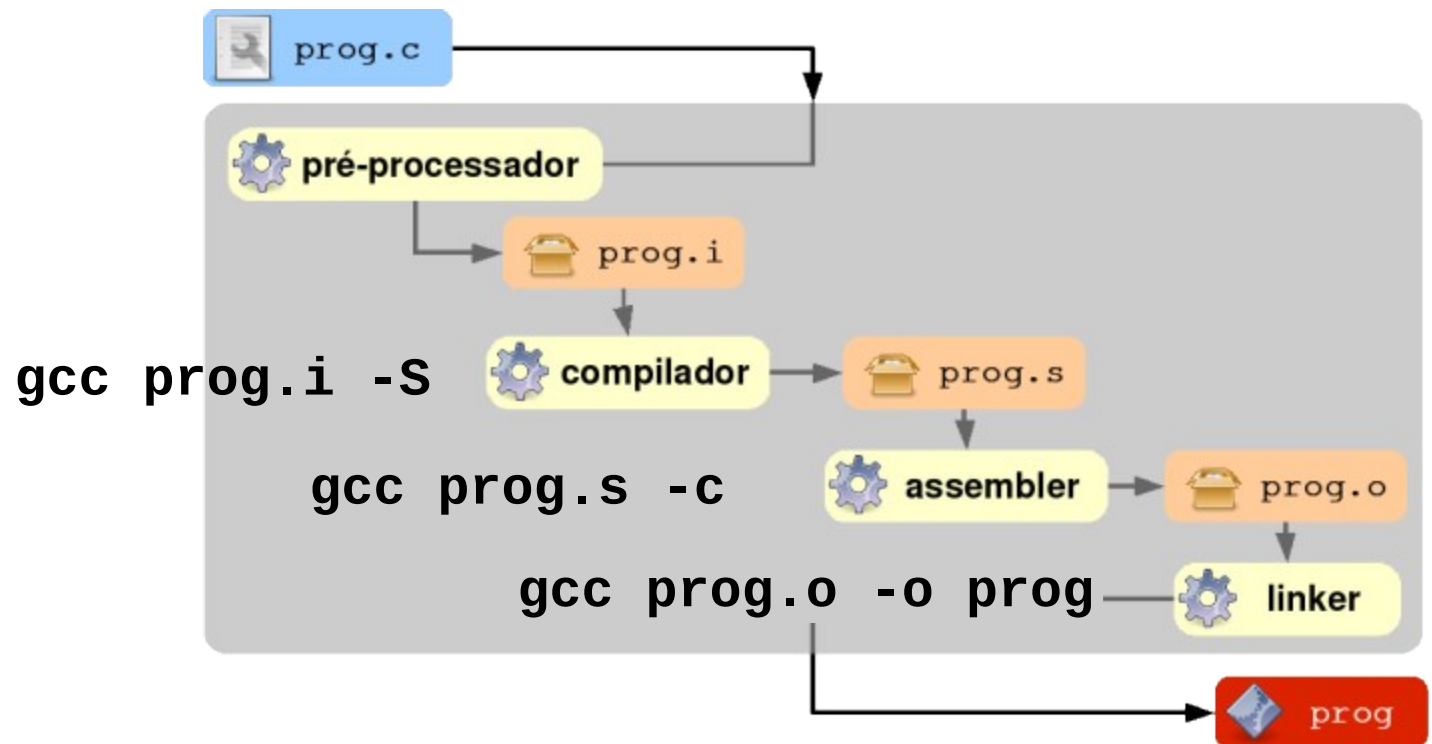
```
#include <stdio.h>
int main(void)
{
    printf("Hello World!\n");
}
```

`gcc prog.c -E -o prog.i`

`gcc prog.i -S`

`gcc prog.s -c`

`gcc prog.o -o prog`





# Hello World Assembly

```
.file    "hello.c"
.section .rodata
.LC0:
.string "Hello World!"
.text
.globl main
.type    main, @function
main:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $8, %esp
    andl    $-16, %esp
    movl    $0, %eax
    addl    $15, %eax
    addl    $15, %eax
    shrl    $4, %eax
    sall    $4, %eax
    subl    %eax, %esp
    movl    $.LC0, (%esp)
    call    puts
    movl    $0, %eax
    leave
    ret
    .size   main, .-main
    .ident  "GCC: (GNU) 4.0.3 (Ubuntu 4.0.3-
1ubuntu5)"
    .section .note.gnu-stack, "", @progbits
```

# Opções de Parada da Compilação

Parâmetro do GCC	Para após	Saída
-E	pré-processamento	código pré-processado (.i)
-S	compilação	código assembly AT&T (.s)
-c	assembler	código objeto (.o)

# Exercício U6.1 – Etapas da Compilação

1. Baixe o arquivo ExerciciosUD6.tar.gz
2. Descompacte-o em sua área de trabalho
3. Entre no diretório EX1
4. A cada comando digitado use o comando `ls -la`
5. `gcc hello.c -E -o hello.pre`
6. `gcc hello.c -S`
7. `gcc hello.c -c`
8. `gcc hello.o -o hello1`
9. `gcc hello.c -o hello2`

# **Forçando a ligação de bibliotecas**

# Forçando a ligação de bibliotecas

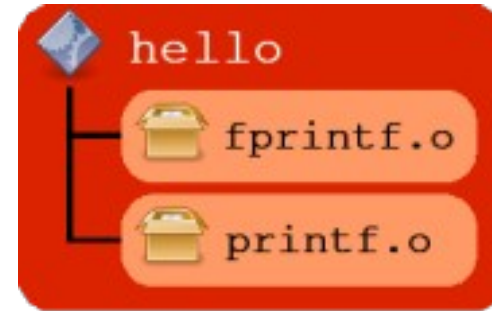
```
#include <stdio.h>
#include <math.h>
int main(void)
{
    double valor = 37.0;
    printf("O seno de %3.2f eh %3.2f\n", valor, sin(valor));
}
```

```
> gcc main.c -o contador
/tmp/cc8MZpoP.o: In function `main':main.c:(.text+0x5a): undefined
reference to `sin'
collect2: ld returned 1 exit status
```

```
> gcc main.c -o contador -lm
```

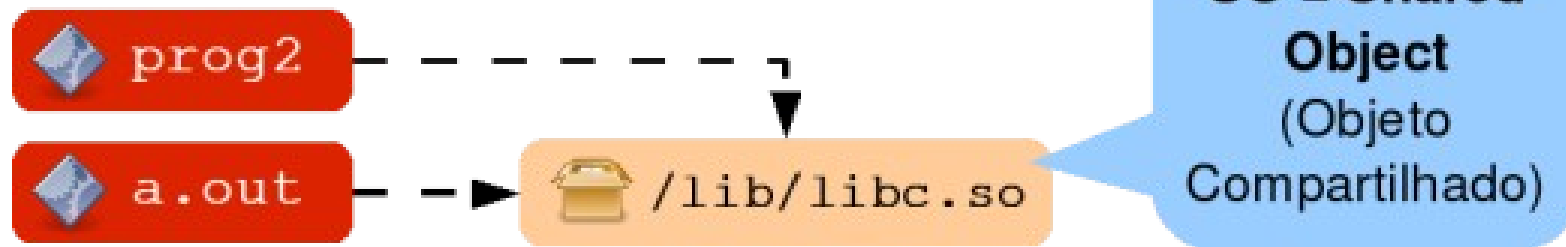
# Forçando a ligação de bibliotecas

- Ligação estática



```
> gcc hello.c -o hello /MyLibPath/libc.a  
> gcc hello.c -o hello --static -L/MyLibPath/ -lc  
490K
```

- Ligação dinâmica



```
> gcc hello.c -o hello /MyLibPath/libc.so  
> gcc hello.c -o hello -L/MyLibPath/ -lc  
6,8K
```

necessita mudança da variável de ambiente `LD_LIBRARY_PATH` ou  
[linux.die.net/man/8/ldconfig](http://linux.die.net/man/8/ldconfig)

## Exercício UD6.2 – Forçando a linkagem

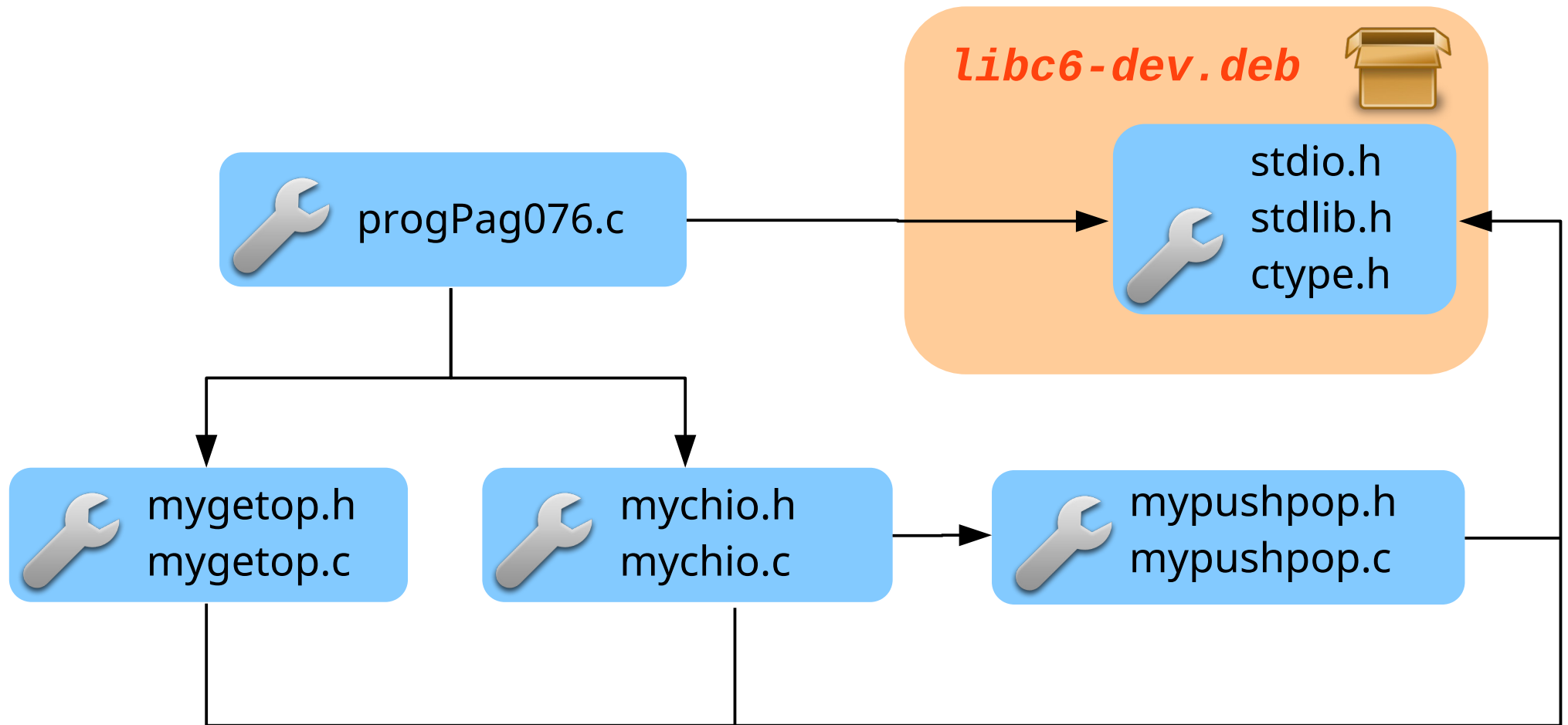
1. Entre no diretório EX2
2. Após digitar os comando a seguir use o comando `ls -la` e compare o tamanho dos executáveis.
3. `gcc seno.c -o seno`
4. `gcc seno.c -o senoDin1 -lm`
5. `gcc seno.c -o senoDin2 /usr/lib/x86_64-linux-gnu/libm.so`
6. `gcc seno.c -lm --static -o senoStat`
7. `gcc seno.c /usr/lib/x86_64-linux-gnu/libm.a -o senoStaDin`

# **Compilando Múltiplos Arquivos**



# Compilando Vários Arquivos Fonte de um Programa

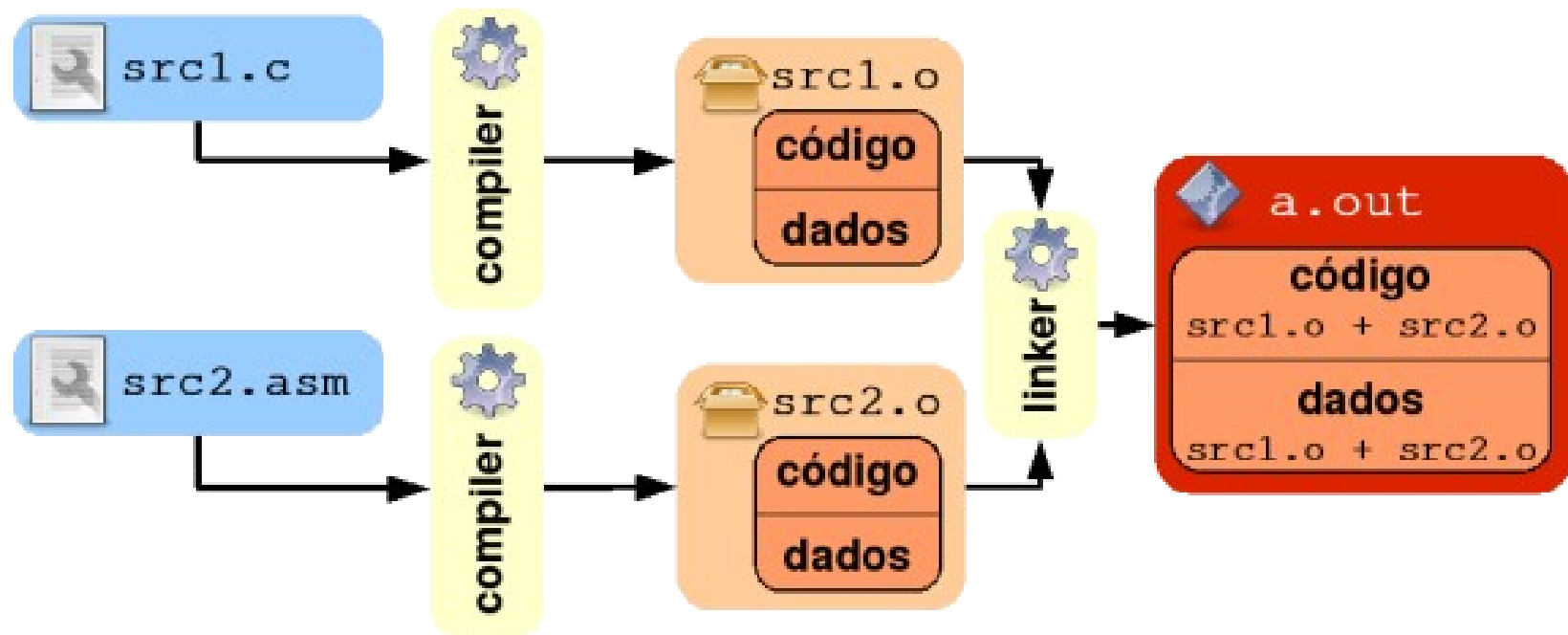
- Múltiplos arquivos fonte e compilação num só passo



```
> gcc mygetop.c mychio.c mypushpop.c progPag076.c -o progPag076
```

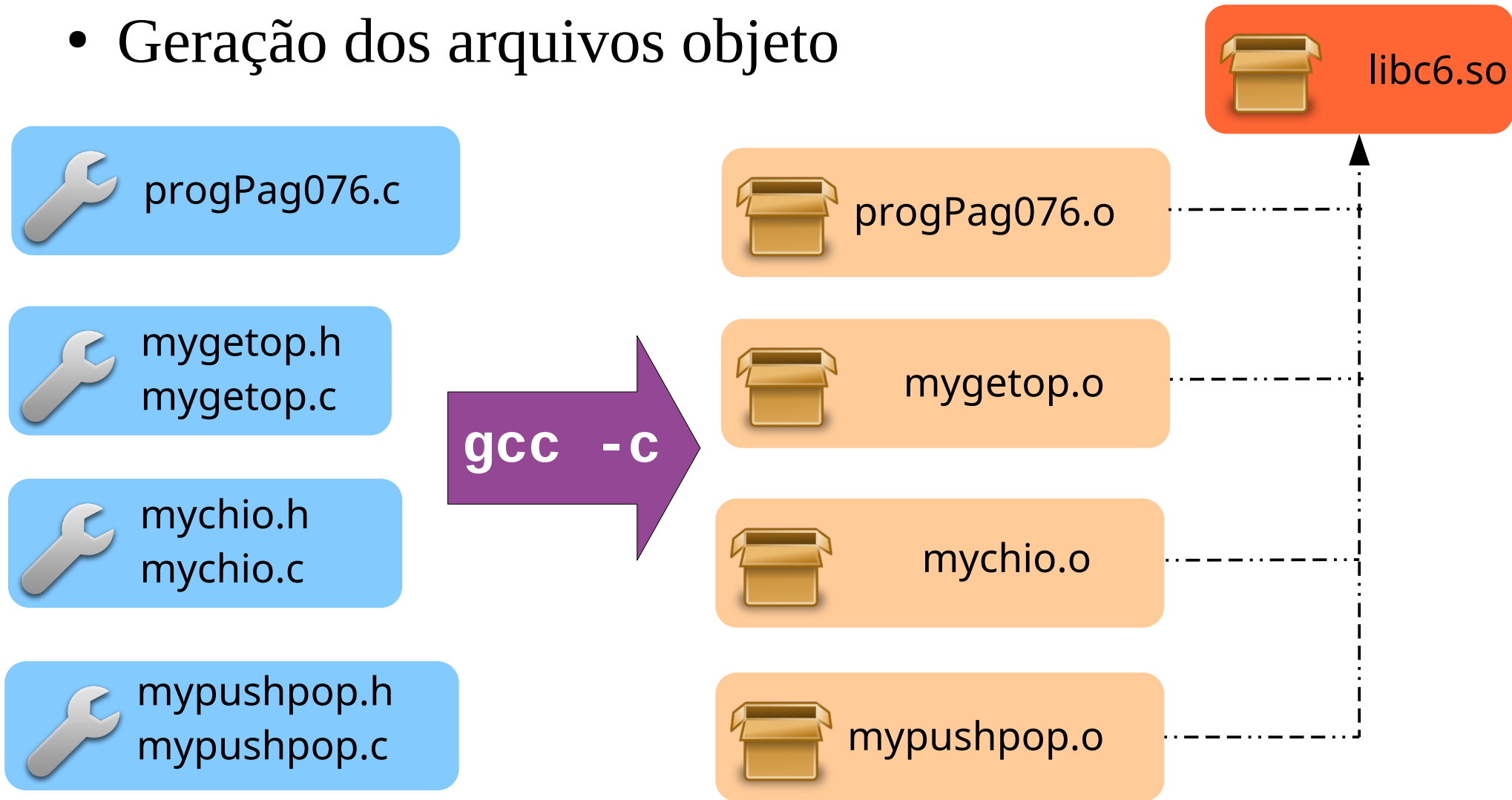
# Compilando Vários Arquivos Fonte de um Programa

- Visão geral



# Compilando Vários Arquivos Fonte de um Programa

- Geração dos arquivos objeto



```
> gcc mygetop.c mychio.c mypushpop.c progPag076.c -c
```

# Compilando Vários Arquivos Fonte de um Programa

- Arquivos objeto e bibliotecas compartilhadas



progPag076.o

```
main()
push()
pop()
printf()
getop()
atof()
```

*libc6-dev.deb*



```
libc6.so
printf()
atof()
getchar()
isdigit()
```



mygetop.o

```
getop()
isdigit()
getch()
ungetch()
```



mypushpop.o

```
push()
pop()
printf()
```



mychio.o

```
getch()
ungetch()
getchar()
printf()
```

# Compilando Vários Arquivos Fonte de um Programa

- Linking



progPag076.o



mygetop.o



mychio.o



mypushpop.o



gcc linker



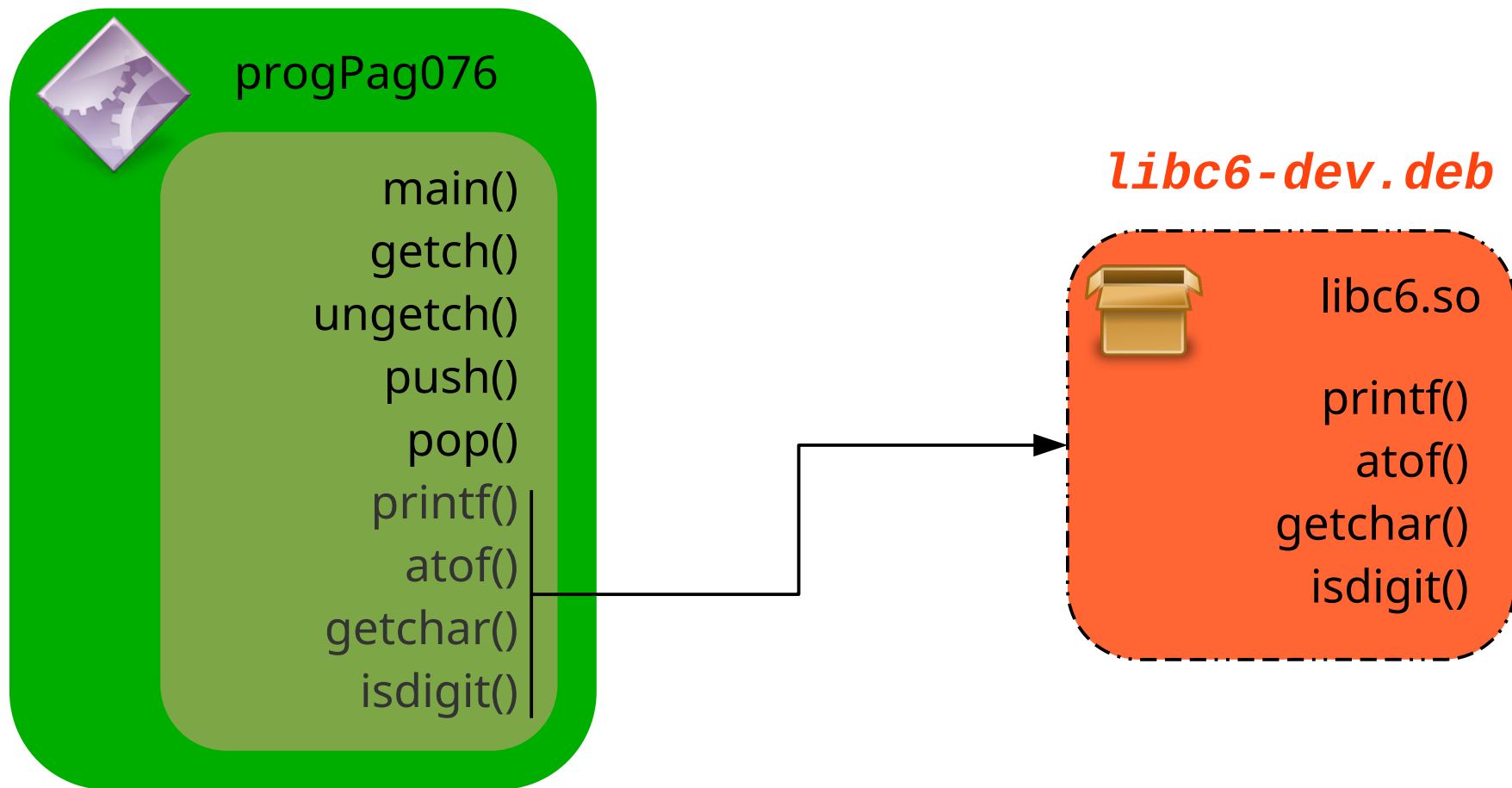
progPag076

```
main()
getch()
ungetch()
push()
pop()
printf()
atof()
getchar()
isdigit()
```

```
> gcc mygetop.o mychio.o mypushpop.o progPag076.o -o progPag076
```

# Compilando Vários Arquivos Fonte de um Programa

- Execução e ligação dinâmica



> ./progPag076

# Exercício UD6.3 – Compilando Múltiplos Arquivos

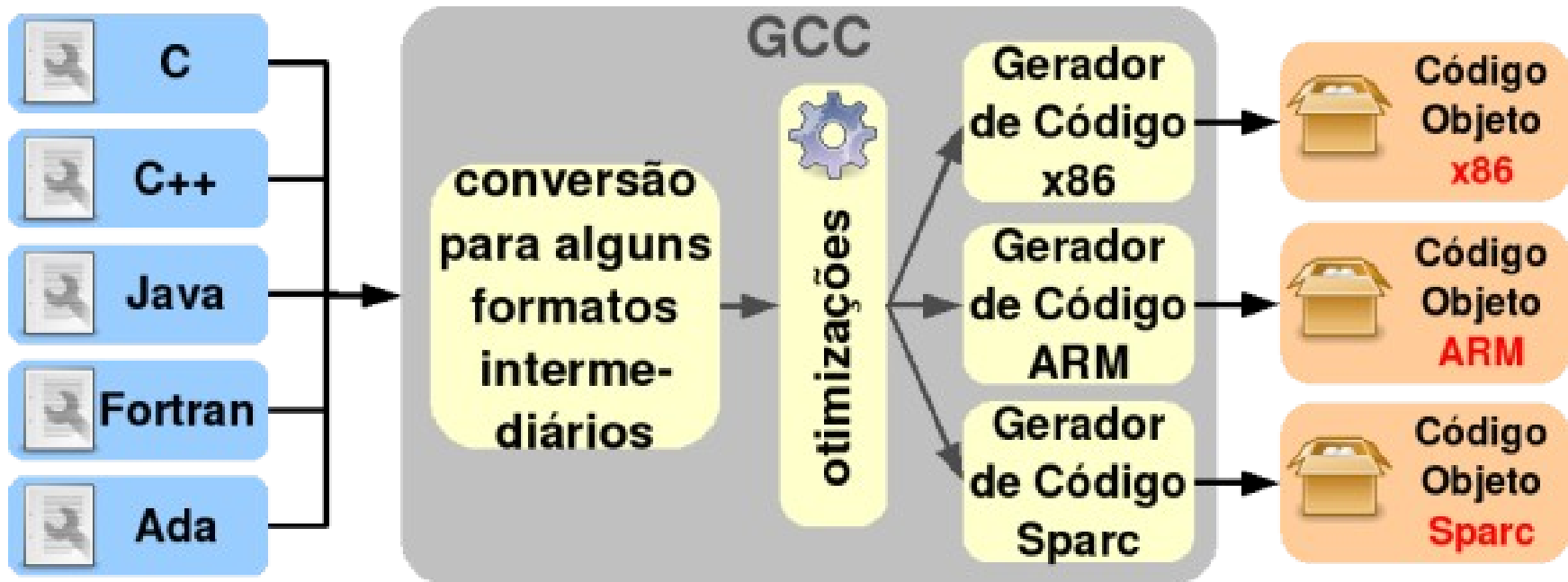
1. Entre no diretório EX3
2. Após digitar os comando a seguir use o comando `ls -la` e analise os arquivos gerados
3. `gcc mygetop.c -c`
4. `gcc mychio.c -c`
5. `gcc mypushpop.c -c`
6. `gcc progPag076.c -c`
7. `gcc mygetop.o mychio.o mypushpop.o  
progPag076.o -o progPag076`

# Compilação Cruzada

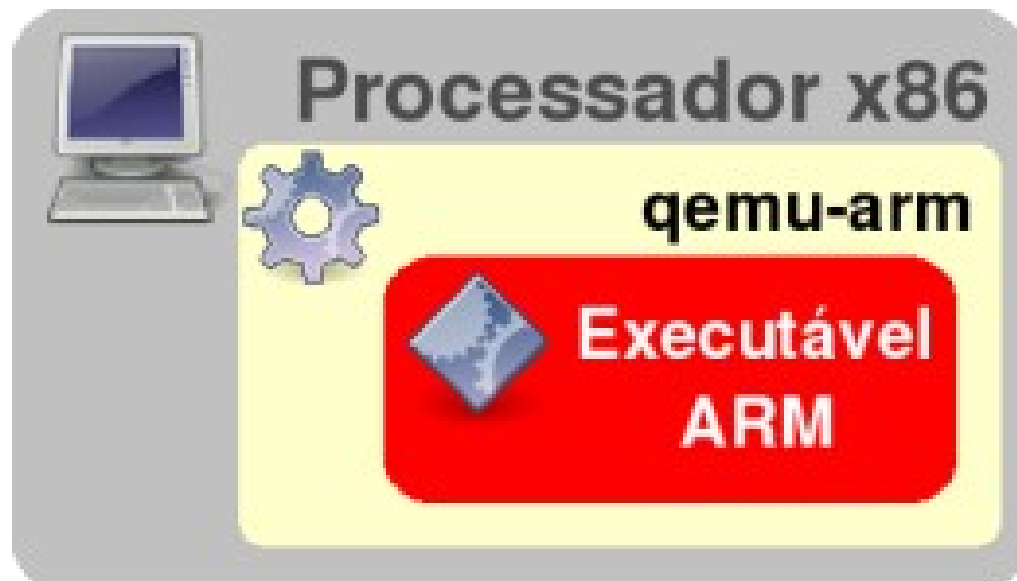
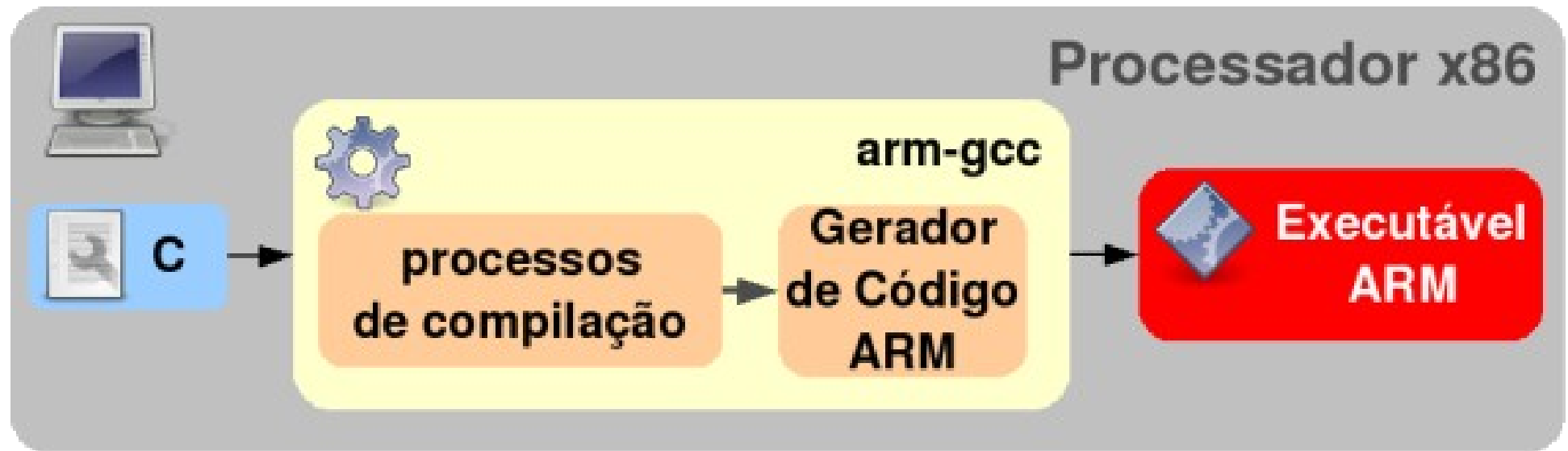


# Compilação Cruzada

- Plataformas alvo



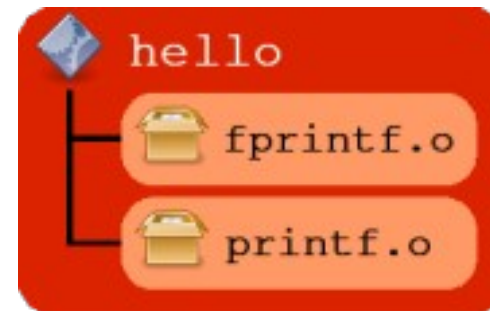
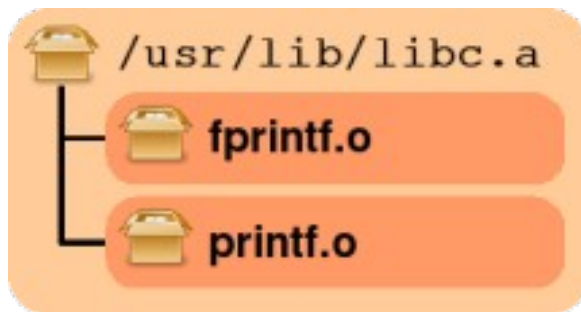
# Compilação Cruzada



# **Bibliotecas pré-compiladas**

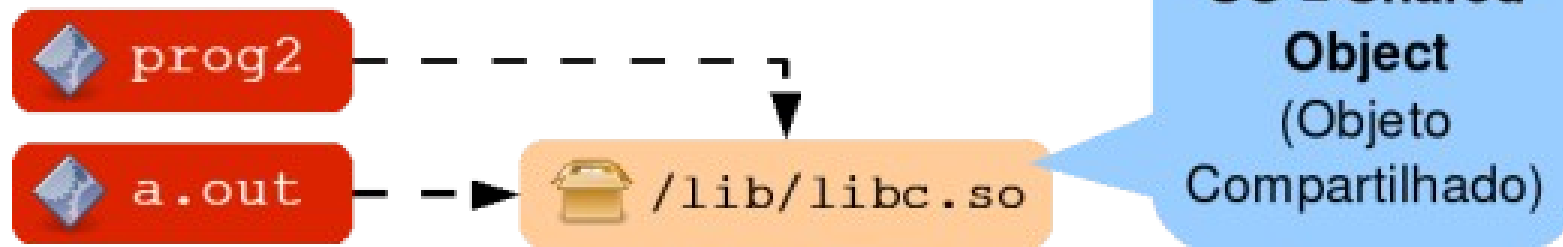
# Usando Bibliotecas Pré-compiladas

- Ligação estática



> gcc hello.c -o hello --static /usr/lib/x86\_64-linux-gnu/libc.a  
490K

- Ligação dinâmica



> gcc hello.c -o hello /usr/lib/x86\_64-linux-gnu/libc.so  
6,8K

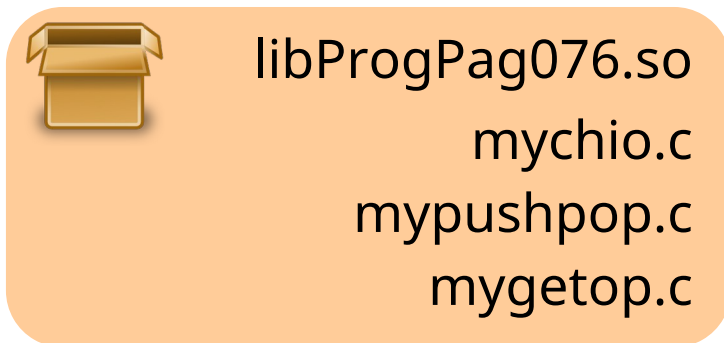
# Criando Bibliotecas Pré-compiladas

- Ligação estática



> **ar rcs libProgPag076.a mygetop.o mypushpop.o mychio.o**

- Ligação dinâmica



> **gcc mychio.c mypushpop.c mygetop.c -shared -fPIC -o libProgPag076.so**

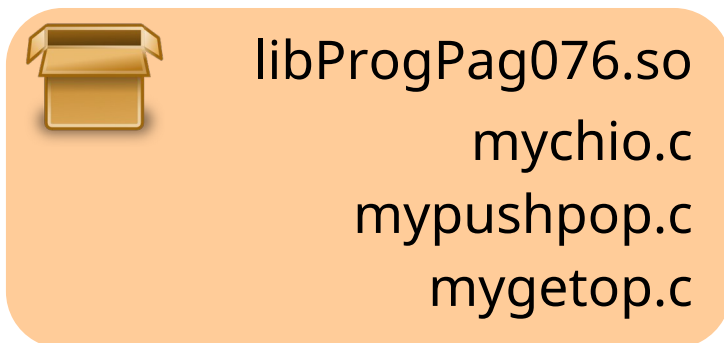
# Examinando Bibliotecas Pré-compiladas

- Ligação estática



> `objdump -a libProgPag076.a | grep mychio.o`

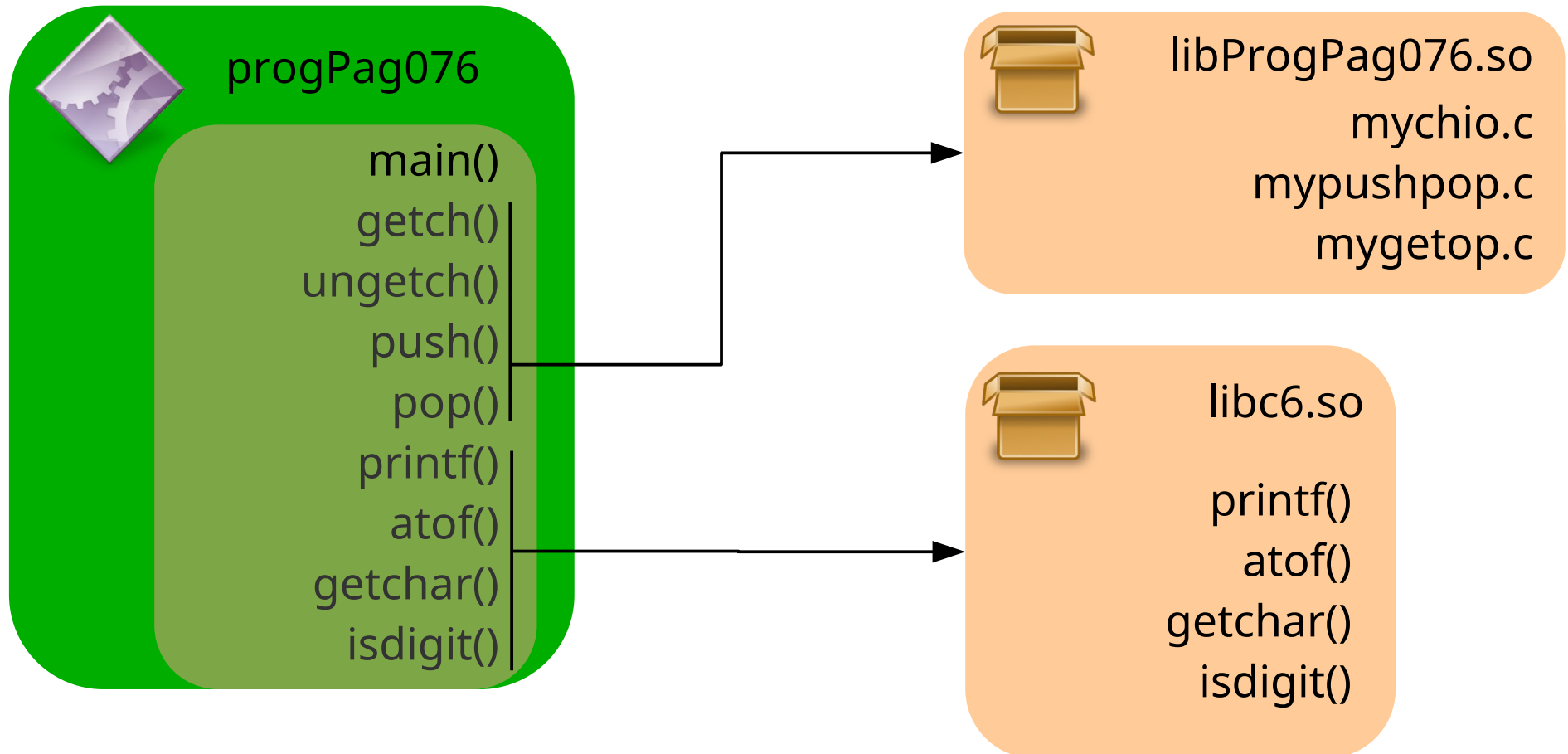
- Ligação dinâmica



> `objdump -tT libProgPag076.so`

# Dependências dinâmicas de um executável

- Varrendo o executável



```
> ldd progPag076
./libProgPag076.so (0x00007fbe968d6000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fbe964f4000)
/lib64/ld-linux-x86-64.so.2 (0x00007fbe96ada000)
```

## Exercício UD6.4 – Compilando bibliotecas .so e .a

1. Entre no diretório EX4
2. Após digitar os comando a seguir use o comando `ls -la` e analise os arquivos gerados quanto ao tipo e tamanho
3. `gcc mygetop.c mypushpop.c mychio.c -c`
4. `ar rcs libProgPag076.a mygetop.o mypushpop.o mychio.o`
5. `gcc mychio.c mypushpop.c mygetop.c -shared -fPIC -o libProgPag076.so`
6. `gcc progPag076.c libProgPag076.a -o progPag076Stat --static`
7. `./progPag076Stat`



## Exercício UD6.4 – Compilando bibliotecas .so e .a

8. `gcc progPag076.c ./libProgPag076.so -o progPag076Dina`
9. `./progPag076Dina`
10. `objdump -a libProgPag076.a`
11. `objdump -tT libProgPag076.so`
12. `ldd progPag076Dina`

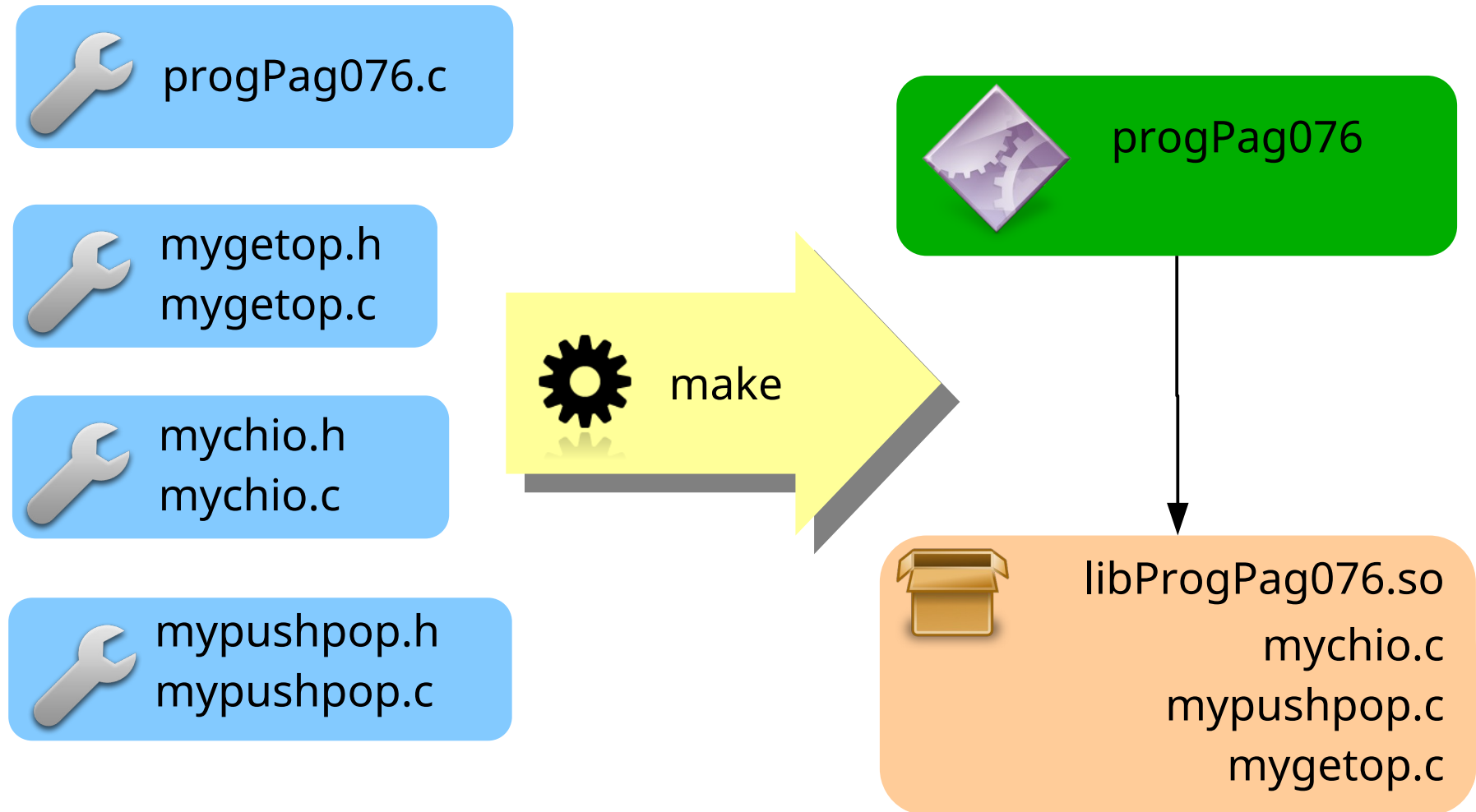
# Makefile

# Makefile

- Aplicativo `make` automatiza o processo de construção do software, respeitando a ordem de dependências entre os arquivos.
- O comando `make` procura no mesmo diretório o arquivo `Makefile` que contém as regras de construção para o código fonte.

```
# Comentários  
VARIABEL=valor  
target: dependencias  
    comando 1  
    comando n
```

# Exemplo Makefile



# Exemplo Makefile

```
LIBS=./libProgPag076.so
PROG=progPag076

progPag076: progPag076.o libProgPag076.so
    gcc progPag076.o ${LIBS} -o ${PROG}

progPag076.o: progPag076.c
    gcc progPag076.c -c

libProgPag076.so: mychio.h mychio.c mypushpop.c mypushpop.h mygetop.c
mygetop.h
    gcc mychio.c mypushpop.c mygetop.c -shared -fPIC -o libProgPag076.so

clean:
    rm *.o *.so ${PROG}
```



make



progPag076



libProgPag076.so

# Exercício UD6.5 – Adaptando um Makefile

1. Entre no diretório EX5
2. Crie a seção `libProgPag076.a` que ao ser evocada gere uma versão estática da biblioteca `libProgPag076`
3. Crie a seção `progPag076Stat` que ao evocada gere uma versão do executável original chamada `progPag076Stat`

OBS.: A linkagem deve ser estática e feita entre `libProgPag076.a` e `progPag076.o`

4. Modifique a seção `clean` para que sejam removidos também os arquivos `.a`
5. Após completar a tarefa use o comando `ls -la` e analise os resultados

# Debugger

# Debugger

- Compilando modo debugger

> `gcc primeiroProg.c -ggdb`

- Executando o debugger

> `gdb a.out`

- Criando um breakpoint

> `(gdb) break primeiroProg.c:13`

- Iniciando a execução

> `(gdb) run`



# Debugger

- Exibindo o conteúdo de uma variável

```
> (gdb) print a  
> (gdb) printf "%d", a
```

- Continuando

```
> (gdb) cont
```

- Saindo

```
> (gdb) quit
```

# Exercício UD6.6 – Debugando com gdb

1. Entre no diretório EX6
2. Compile o programa seno com opção de debug
3. Rode o programa através do gdb
4. Crie um breakpoint na linha 7
5. Verifique os valores impressos a cada passo do laço

**Fim**