

Make

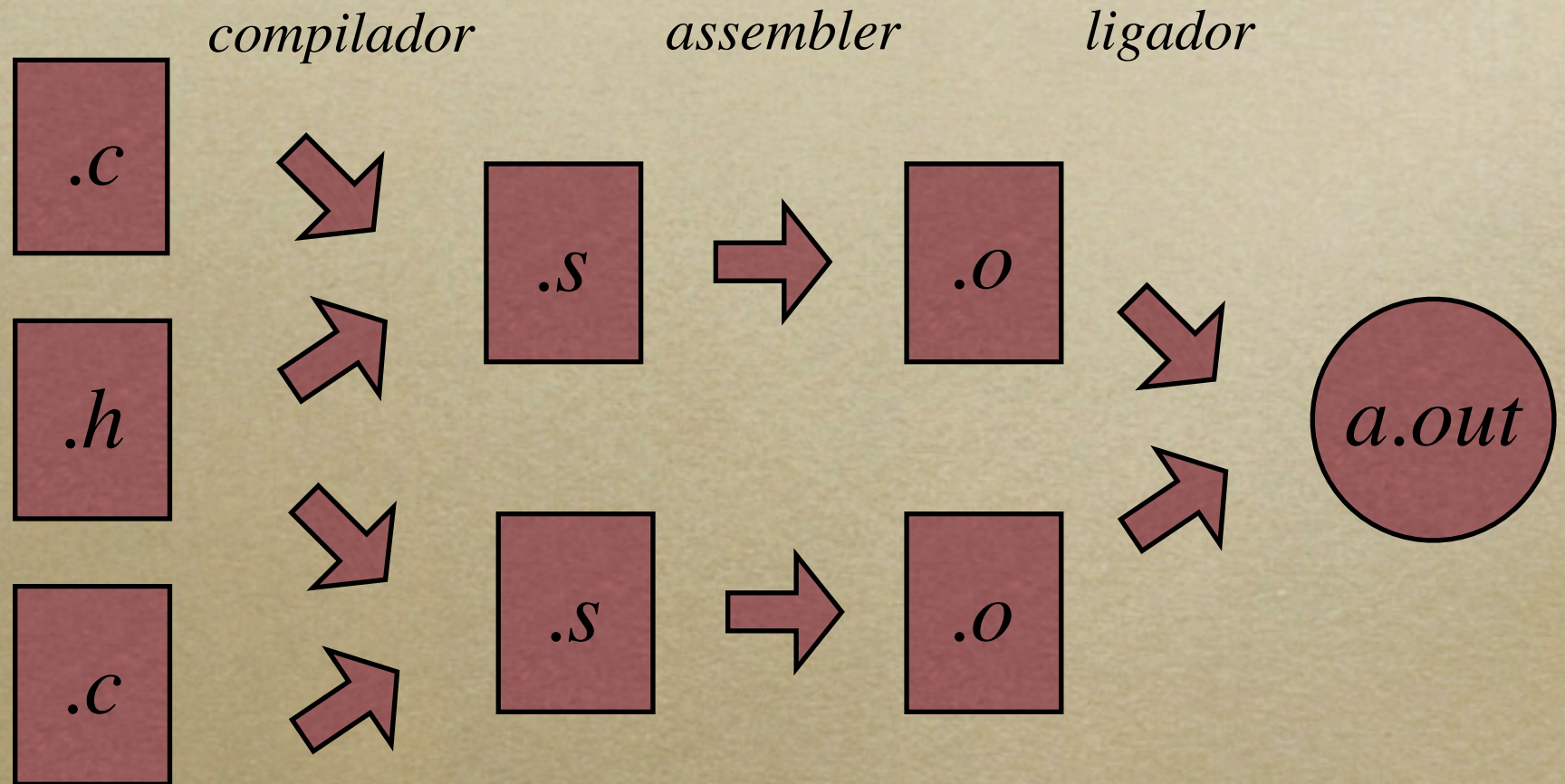
MC626

Ferramentas de Desenvolvimento

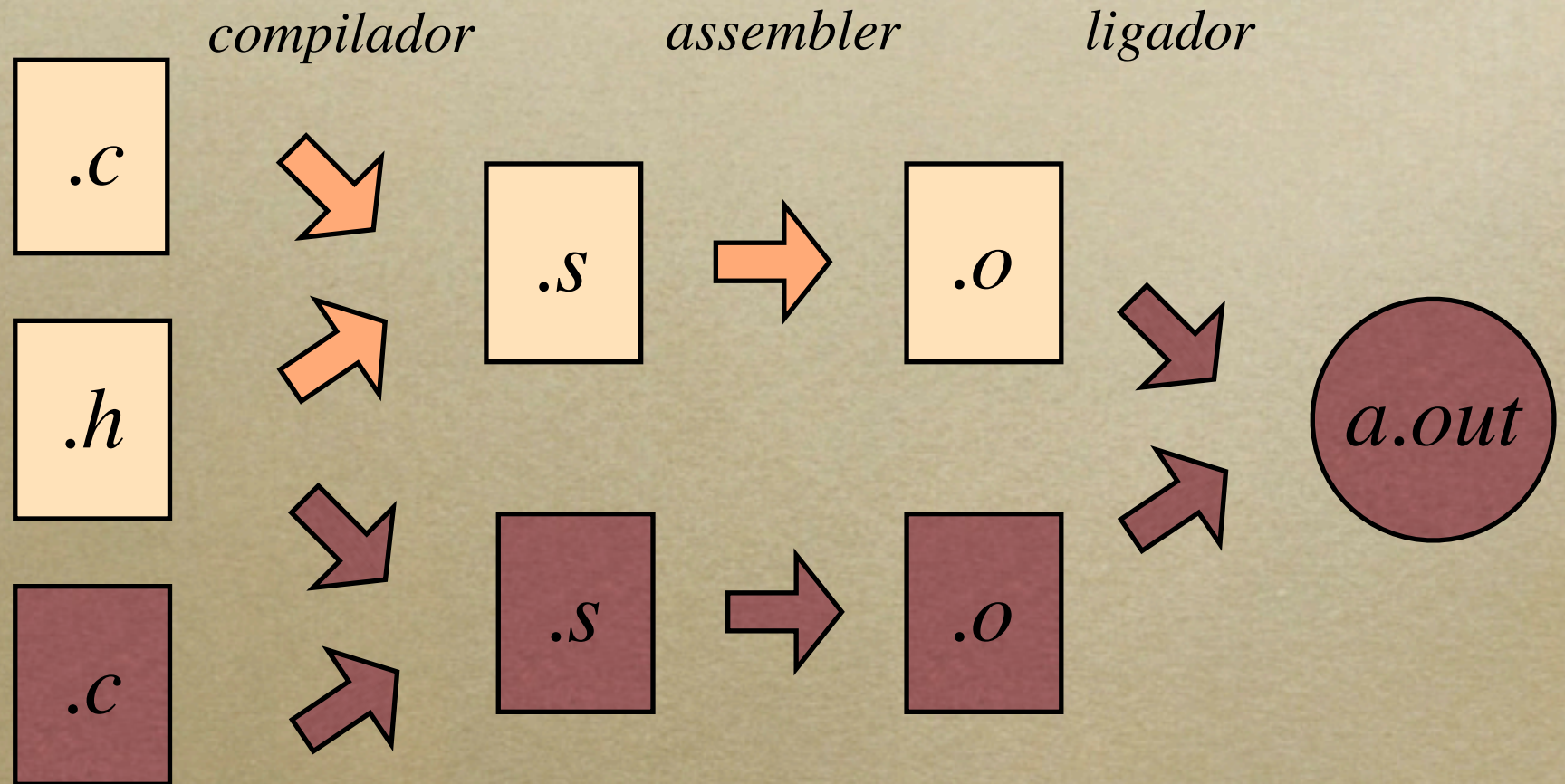
O que Make faz

-
- *Make é uma ferramenta que rastreia dependências temporais entre módulos (de programas, de texto, etc.)*
- *Permite gerenciar grandes programas ou grupos de programas*
 - *Permite a recompilação apenas dos módulos que necessitam ser recomilados (porque foram alterados ou porque dependem de módulos que foram alterados)*

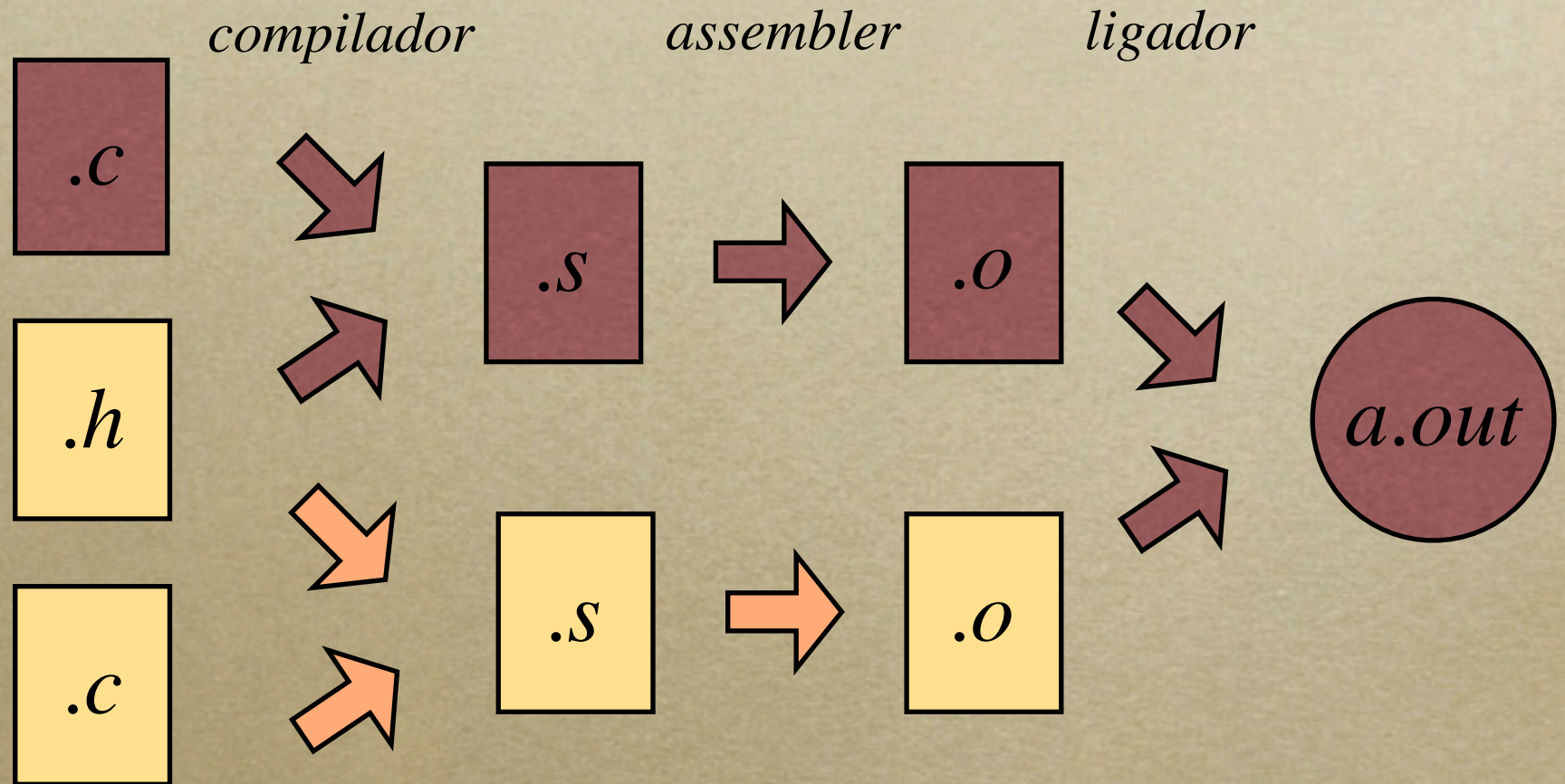
Exemplo



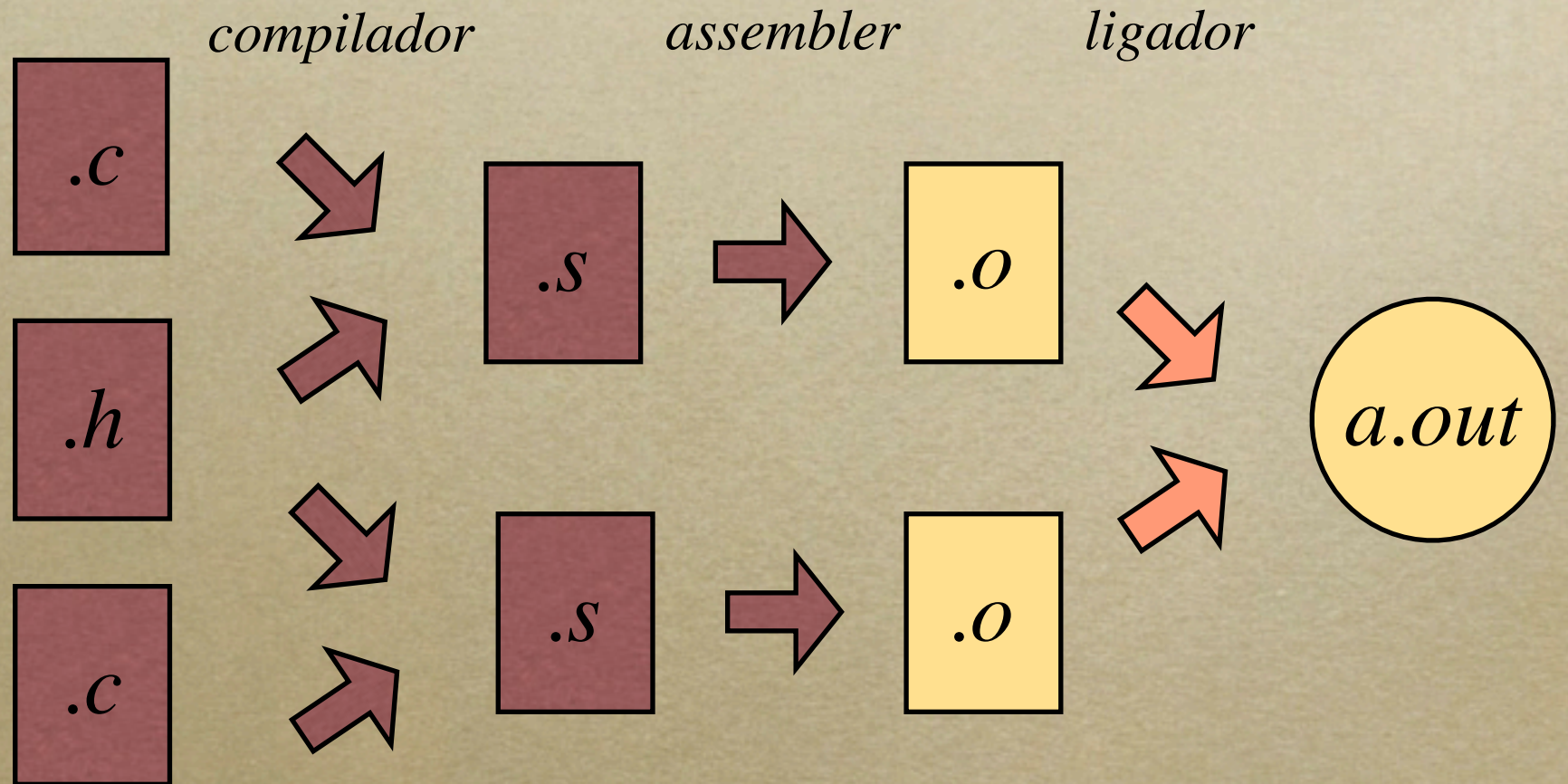
Exemplo



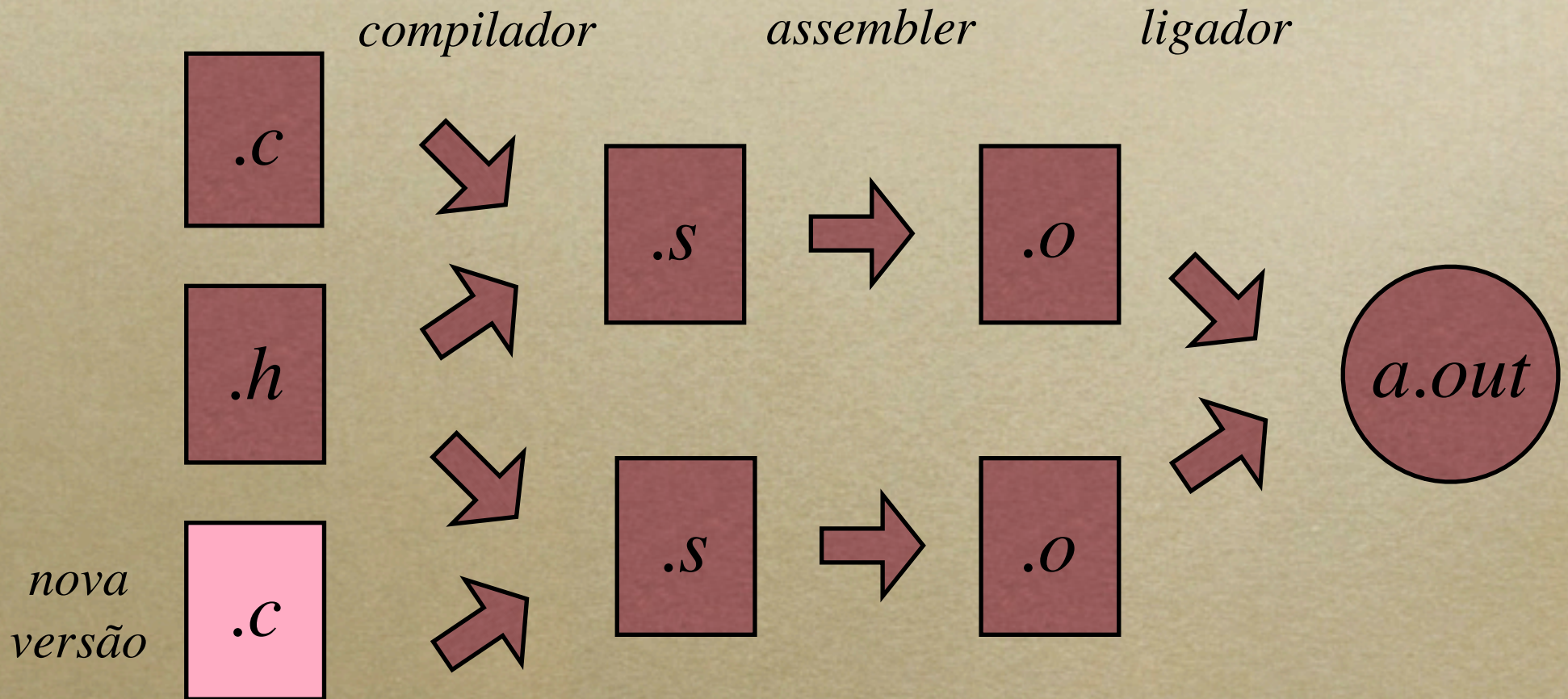
Exemplo



Exemplo

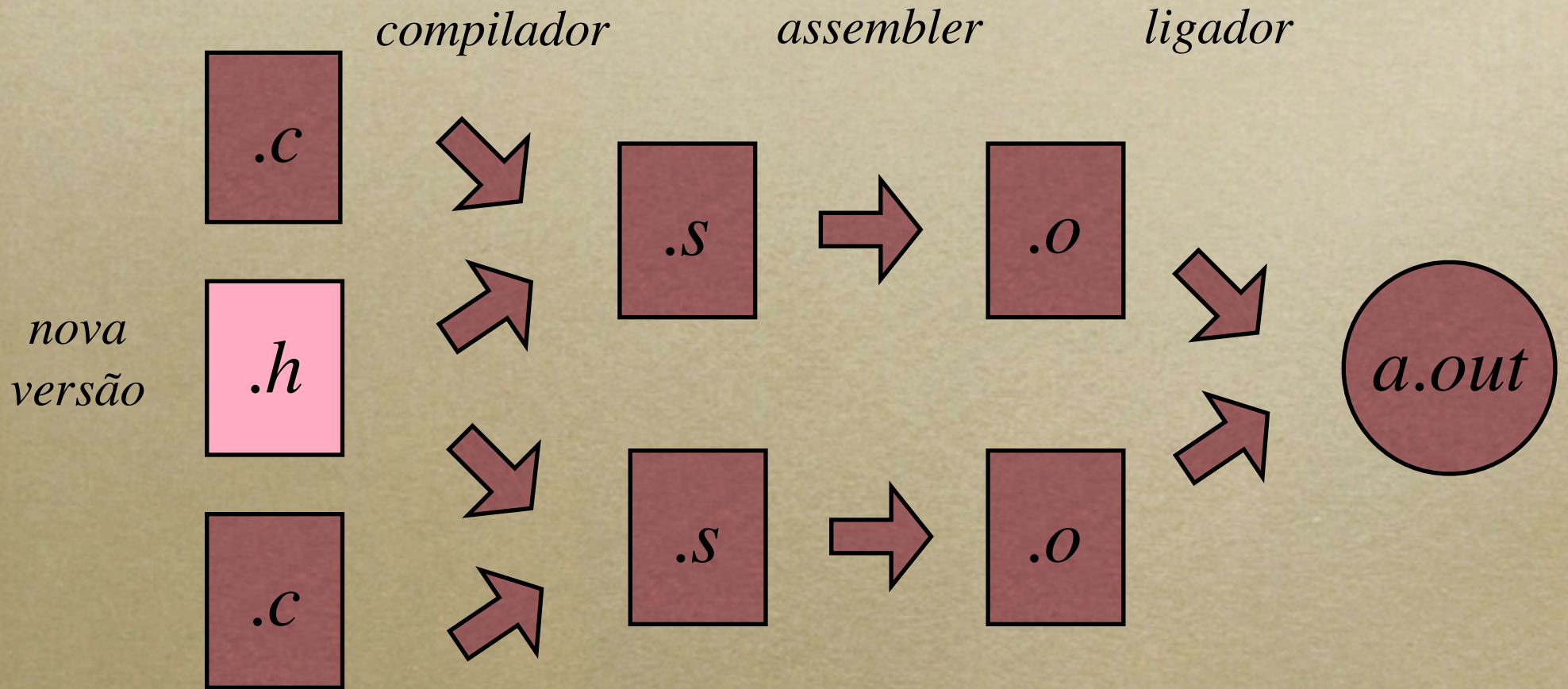


Exemplo



O que deve ser recompilado?

Exemplo

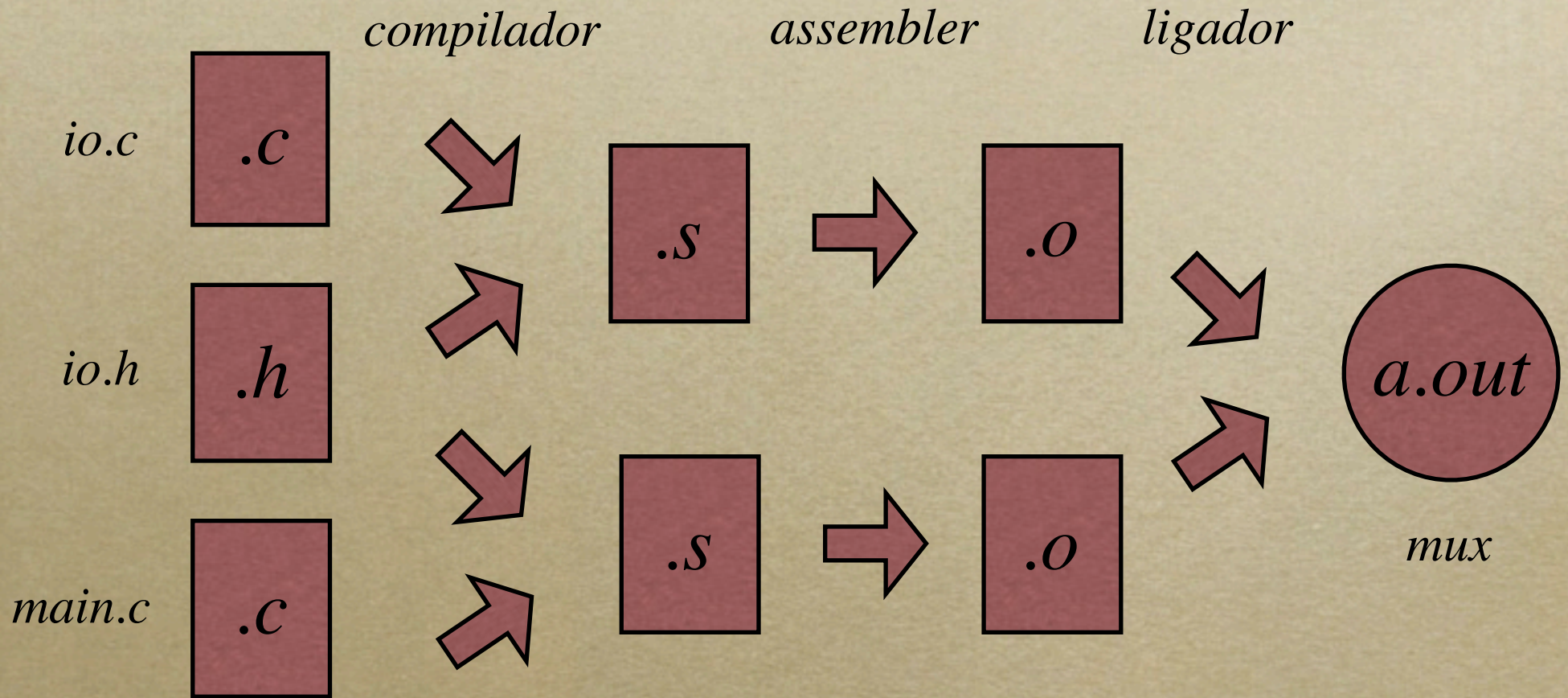


O que deve ser recompilado?

Como Make funciona

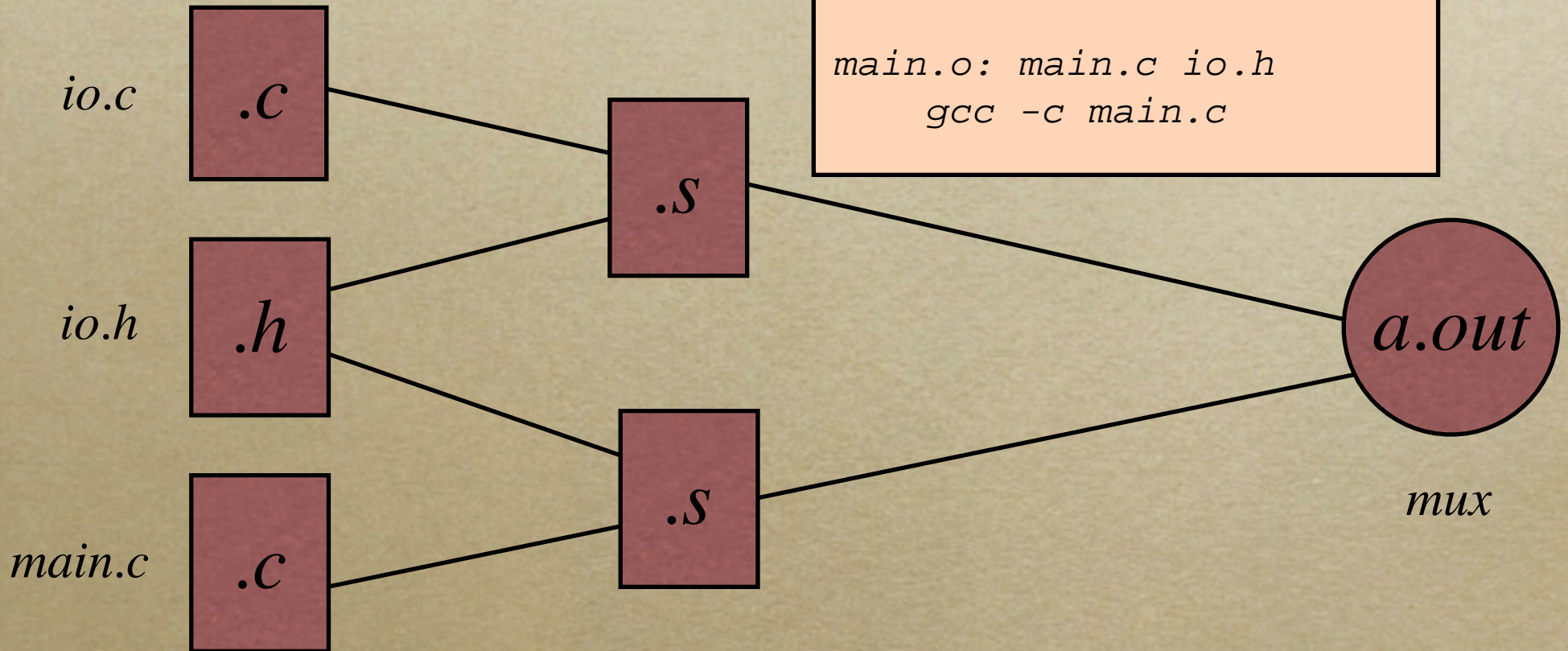
- *Make rastreia dependências entre arquivos e executa os comandos definidos para manter a consistência*
- *Make trabalha com grafos de dependências*
- *Dependências são descritas em um arquivo texto, geralmente chamado de Makefile, contendo diretivas (ou comandos) make*

Exemplo



Grafo de de


```
# exemplo de Makefile  
  
mux: io.o mux.o  
    gcc -o mux io.o mux.o  
  
io.o: io.c io.h  
    gcc -c io.c  
  
main.o: main.c io.h  
    gcc -c main.c
```



Makefile

- *Dependências são descritas na forma*

alvo: dependências...

 *comandos*

 *...*



tab

```
# exemplo de Makefile

mux: io.o mux.o
    gcc -o mux io.o mux.o

io.o: io.c io.h
    gcc -c io.h

main.o: main.c io.h
    gcc -c main.c
```

Usando Make

```
# exemplo de Makefile
```

```
mux: io.o mux.o  
    gcc -o mux io.o mux.o
```

```
io.o: io.c io.h  
    gcc -c io.h
```

```
main.o: main.c io.h  
    gcc -c main.c
```

```
$ make -n  
gcc -c io.c  
gcc -c main.c  
gcc -o mux io.o mux.o  
$ make  
gcc -c io.c  
gcc -c main.c  
gcc -o mux io.o mux.o
```

- o *make* executa os comandos, baseado nas dependências definidas no arquivo de nome 'Makefile', ou 'makefile'
- o use 'make -f meu_makefile' se arquivo de descrição tiver outro nome

Variáveis

- *É possível definir variáveis (macros) no makefile*

- *Nome de variável pode ser uma sequência que não inclua*

OBJECTS = main.o io.o

- *Para usar a variável (exp*

\$(OBJECTS)

```
# exemplo de Makefile
```

```
CC = gcc
```

```
OBJECTS = io.o mux.o
```

```
mux: $(OBJECTS)
```

```
$(CC) -o mux $(OBJECTS)
```

```
io.o: io.c io.h
```

```
$(CC) -c io.h
```

```
main.o: main.c io.h
```

```
$(CC) -c main.c
```

Variáveis especiais

- \$@
 - *nome completo do alvo corrente*
- \$?
 - *lista de arquivos da dependência corrente que estão desatualizados*
- \$<
 - *nome do arquivo fonte da dependência corrente*

Sufixos e regras especiais

```
# exemplo de Makefile
```

```
CC = gcc
```

```
OBJECTS = io.o mux.o
```

```
mux: $(OBJECTS)
```

```
    $(CC) -o mux $(OBJECTS)
```

```
io.o: io.c io.h
```

```
    $(CC) -c io.h
```

```
main.o: main.c io.h
```

```
    $(CC) -c main.c
```

```
.C.O:
```

```
    $(CC) $(CFLAGS) -c $<
```

```
# exemplo de Makefile
```

```
.SUFFIXES
```

```
.SUFFIXES .o .c
```

```
CC = gcc
```

```
OBJECTS = io.o mux.o
```

```
mux: $(OBJECTS)
```

```
    $(CC) -o mux $(OBJECTS)
```

```
io.o: io.h
```

```
main.o: io.h
```

```
.c.o:
```

```
    $(CC) -c $<
```

Alvos

```
$ make -n clean
rm mux io.o main.o
$ make clean
$ make
gcc -c io.c
gcc -c main.c
gcc -o mux io.o main.o
```

- *Por default, make tenta o primeiro alvo que aparece*
- *Podemos definir mais alvos no nome do alvo no comando exemplo 'make passo1'*

```
# exemplo de Makefile

.SUFFIXES
.SUFFIXES .o .c

CC = gcc
OBJECTS = io.o mux.o

mux: $(OBJECTS)
    $(CC) -o mux $(OBJECTS)

clean:
    rm mux $(OBJECTS)

io.o: io.h
main.o: io.h

.c.o:
    $(CC) -c $<
```

Condições

```
foo: $(objects)
      $(CC) -o foo $(objects) $(normal_libs
```

```
libs_for_gcc = -lgnu
normal_libs =
```

```
foo: $(objects)
ifeq ($(CC),gcc)
      $(CC) -o foo $(objects) $(libs_for_gcc)
else
      $(CC) -o foo $(objects) $(normal_libs)
endif
```

Condições

- *ifeq (arg1, arg2)*
- *ifneq (arg1, arg2)*
- *ifdef nome-variável*

Outro estilo de Makefile

```
objects = main.o kbd.o command.o display.o \  
         insert.o search.o files.o utils.o  
  
edit : $(objects)  
      cc -o edit $(objects)  
  
$(objects) : defs.h  
kbd.o command.o files.o : command.h  
display.o insert.o search.o files.o : buffer.h
```

Incluindo outros Makefile

- *A diretiva 'include' determina que make suspenda a leitura do makefile corrente e leia outros makefile antes de continuar*

Para fazer com que make ignore o makefile que não existe (e que não pode ser regenerado), utilize a diretiva

```
-include nomes_de_arquivos...
```

Uso de Wildcards

- *Os caracteres são os mesmos da Bourne Shell ('?' e '*')*

```
clean:  
  rm -f *.o
```

```
print: *.c  
  lpr -p $?  
  touch print
```

```
objects = *.o  
  
foo: $(objects)  
  cc -o foo $(objects)
```

Fim