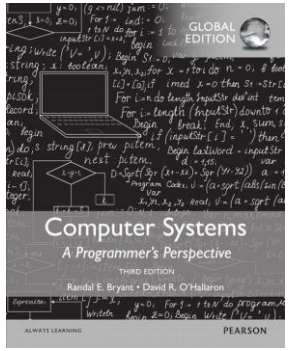


Bibliotecas estáticas e Makefile

Bib: Computer Systems: A Programmer's Perspective (cap. 7, .6.2)

Programação em Sistemas Computacionais

João Pedro Patriarca (jpatri@cc.isel.ipl.pt, joao.patriarca@isel.pt), Gabinete F.0.23 do edifício F

ISEL, ADEETC, LEIC

Agenda

- Motivação, utilização e criação
- Processo de ligação com bibliotecas estáticas
- Ferramenta **`make`** e ficheiro **`Makefile`**

Agenda

- Motivação, utilização e criação
- Processo de ligação com bibliotecas estáticas
- Ferramenta `make` e ficheiro `Makefile`

Motivação – considerando a biblioteca standard do C

- Facto: constituída por muitas funções (dimensão aproximada entre 5 e 6 Mbytes)
- Hipótese 1: definir todas as funções num único ficheiro **`.o`**

```
$ gcc –o main main.c /usr/lib/libc.o
```

  - ✗ O ficheiro executável inclui a biblioteca standard do C na totalidade
  - ✗ Qualquer alteração de uma função da biblioteca implica gerar novamente todos os executáveis que a utilizem

- Hipótese 2: definir cada função num ficheiro objeto independente

```
$ gcc –o main main.c /usr/lib/printf.o /usr/lib/strlen.o ...
```

  - ✓ O executável inclui apenas as funções da biblioteca de que depende
  - ✗ Delega no programador a responsabilidade de conhecer o nome de todos os módulos da biblioteca
  - ✗ Sofre do mesmo problema da hipótese 1: qualquer alteração de uma função da biblioteca usada no executável, implica novamente a geração do executável

Utilização de bibliotecas estáticas

- Uma biblioteca estática é um ficheiro arquivo constituído por ficheiros objeto
- Na ligação são usados apenas os ficheiros objeto que satisfaçam indefinições

```
$ gcc –o main main.c /usr/lib/libc.a /usr/lib/libm.a
```

- Pode-se usar as opções **-L** e **-l** para identificar o caminho para as bibliotecas e as próprias bibliotecas, respetivamente
  - Em diretorias pré-definidas pode-se omitir o caminho
  - O nome das bibliotecas tem de ter o prefixo **lib**
- Para usar as versões estáticas das bibliotecas é preciso usar a opção **-static**

```
$ gcc –static –o main main.c -L/usr/lib/ -lc -lm
```

- Pode-se omitir a biblioteca standard do C

```
$ gcc –static –o main main.c -L/usr/lib/ -lm
```

Exemplo base

Criação de biblioteca estática

asum.c

```c
int asum(int a[], int size) {
    int r = 0;
    for (int i = 0; i < size; i++)
        r += a[i];
    return r;
}
```

aprint.c

```c
#include <stdio.h>

void aprint(int a[], int size) {
    for (int i = 0; i < size-1; i++)
        printf("a[%d]=%d, ",
                i, a[i]);
    if (size > 0)
        printf("a[%d]=%d\n",
                size-1, a[size-1]);
}
```

Criação de biblioteca estática

```
$ ar crs libarray_utils.a asum.o aprint.o
```

- As bibliotecas estáticas têm extensão **.a**, em sistemas Linux
- Opções:
  - **c** – cria ficheiro
  - **r** – sobrepõe módulos repetidos
  - **s** – cria tabela com indexação para os módulos

```
$ ar t libarray_utils.a
asum.o
aprint.o
```

```
$ nm libarray_utils.a
asum.o:
0...0000 T asum
aprint.o:
         U _GLOBAL_OFFSET_TABLE_
0...0000 T aprint
         U printf
```

Agenda

- Motivação, utilização e criação
- Processo de ligação com bibliotecas estáticas
- Ferramenta `make` e ficheiro `Makefile`

Programa principal e geração do executável

Criação de biblioteca estática

array_utils.h

```
#ifndef _ARRAY_UTILS_H
#define _ARRAY_UTILS_H

void aprint(int a[], int size);
int asum(int a[], int size);

#endif/*_ARRAY_UTILS_H*/
```

main.c

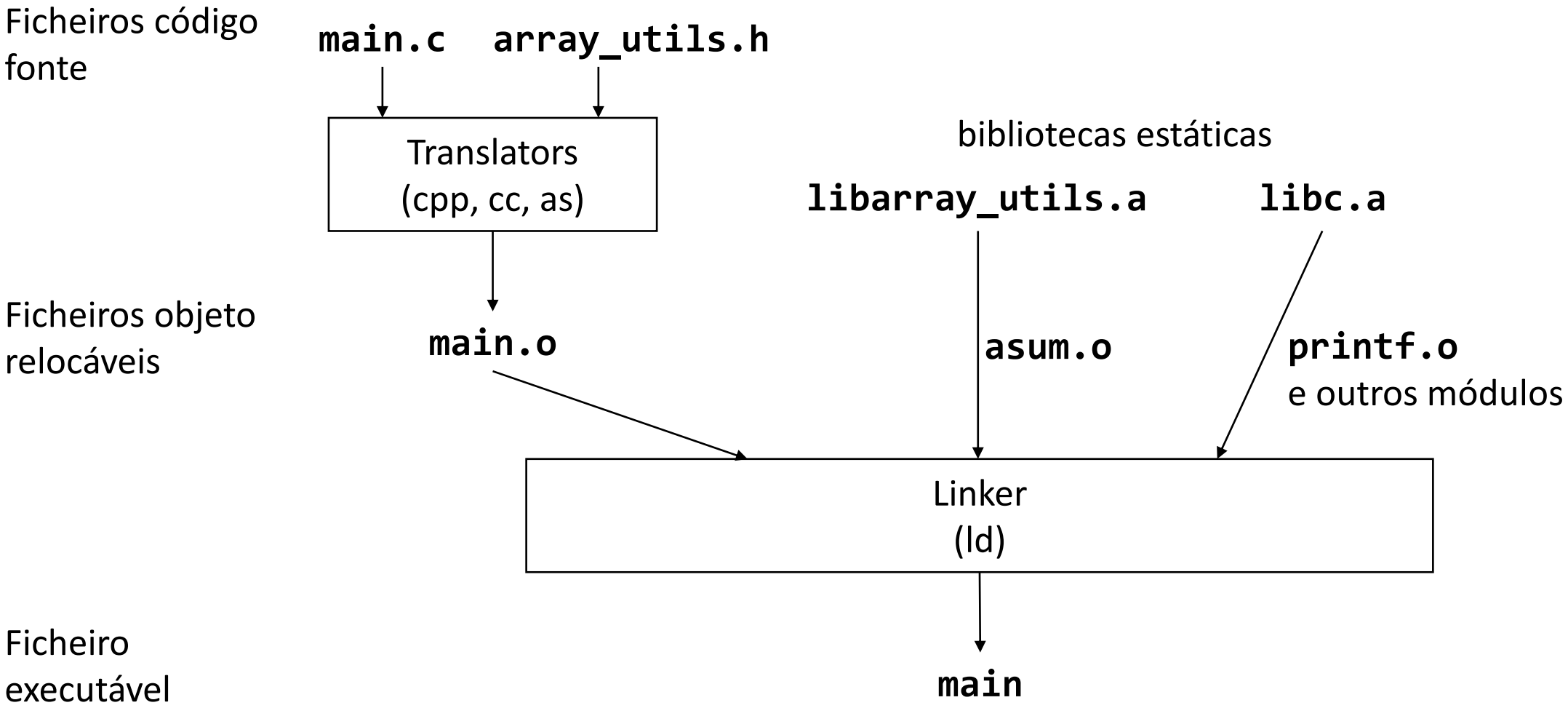
```
#include <stdio.h>
#include "../common.h"
#include "array_utils.h"

int v1[] = {1, 2, 3, 4};

int main(void) {
    PRINT_EXP(
        asum(v1, ARRAY_SIZE(v1))
    );
    return 0;
}
```

```
$ gcc -static -o main main.c -L. -larray_utils
```

Resumo da produção do executável



Processo de resolução de símbolos

```
$ gcc -static -o main main.c -L. -larray_utils
```

- Mantém o conjunto *E* de ficheiros objeto que produzirão o executável
- Mantém o conjunto *U* de símbolos indefinidos
- Mantém o conjunto *D* de símbolos definidos
- No início do processo estes conjuntos estão vazios
- Processa os ficheiros da esquerda para a direita da linha de comando
- Um ficheiro objeto relocável é adicionado a *E* e atualiza *U* e *D* baseado na contribuição da sua tabela de símbolos
- Para um ficheiro arquivo, um ficheiro objeto que resolva indefinições é adicionado a *E* e atualiza *U* e *D* baseado na contribuição da sua tabela de símbolos
  - São realizadas várias iterações sobre os membros do ficheiro arquivo até que *U* e *D* permaneçam constantes
  - No final das iterações, são descartados os ficheiros objeto que não resolveram indefinições
- No final do processamento de todos os ficheiros, gera erro se *U* não estiver vazio, caso contrário, passa à fase de relocação baseado nos módulos presentes no conjunto *E*

Comparação com e sem a opção **-static**

```
$ gcc -o main main.o -L. -larray_utils
$ gcc -static -o main_static main.o -L. -larray_utils
$ ls -l
-rwxr-xr-x 1 jpatri jpatri  19792 Nov 23 23:10 main
-rwxr-xr-x 1 jpatri jpatri 874904 Nov 23 23:10 main_static
...
$ nm main
...
000000000000117c T asum
0000000000001149 T main
                 U printf@@GLIBC_2.2.5
0000000000004010 D v1
$ nm main_static
...
0000000000401ce8 T asum
0000000000401cb5 T main
00000000004109a0 T printf
00000000004c00f0 D v1
```

Consequências do processo de resolução de símbolos

- Um ficheiro objeto da biblioteca estática é incluído apenas se resolver símbolos indefinidos
- É relevante a ordem da especificação dos ficheiros na linha de comando

```
$ gcc –static –o main –L. –larray_utils main.c
/usr/bin/ld: /tmp/ccB7enSa.o: in function `main':
main.c:(.text+0x15): undefined reference to `asum'
collect2: error: ld returned 1 exit status
```

- Considerando duas bibliotecas, **libx.a** e **liby.a**
  - **libx.a** inclui os módulos **m1.o** e **m2.o** e **liby.a** inclui o módulo **m3.o**
  - **m1.o** é incluído no conjunto *E* e depende do símbolo **s1** definido no módulo **m3.o** e, por sua vez, **m3.o** depende do símbolo **s2** definido no módulo **m2.o**

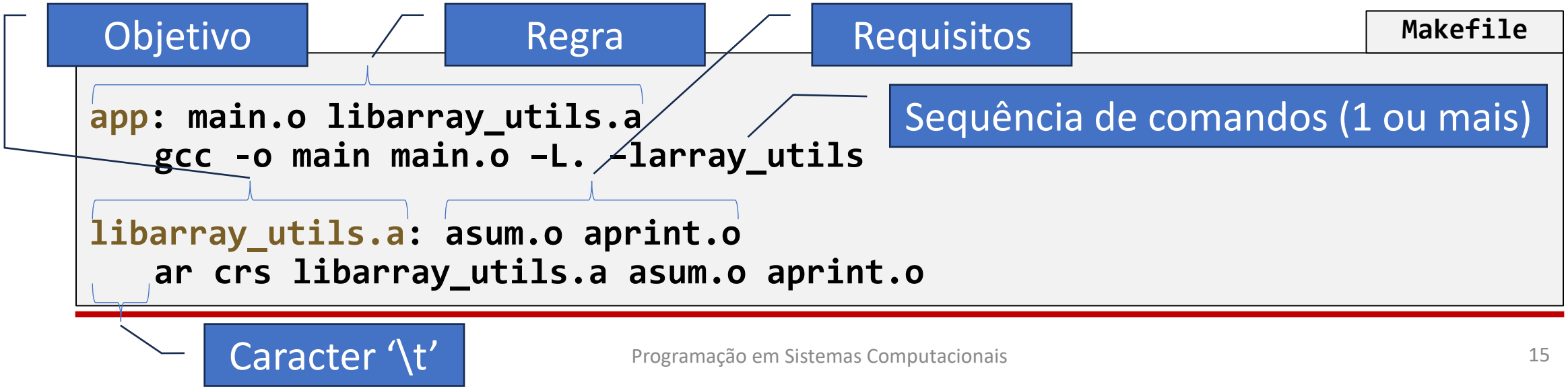
```
$ gcc <outros ficheiros e opções> -lx -ly -lx
```

Agenda

- Motivação, utilização e criação
- Processo de ligação com bibliotecas estáticas
- Ferramenta **`make`** e ficheiro **`Makefile`**

Ferramente **make** e ficheiro **Makefile**

- A ferramenta **make** é uma ferramenta de geração automática
  - Usada, tipicamente, para gerar ficheiros objeto, bibliotecas e executáveis
  - O processamento da ferramenta **make** depende de um ficheiro de texto com o nome **Makefile** (por omissão)
- O ficheiro **Makefile** define regras e respetivos comandos
  - Uma regra define um objetivo e um conjunto de requisitos
  - Os comandos associados à regra são executados apenas e somente se qualquer um dos requisitos tiver uma data posterior à data do objetivo



Exemplo aplicado a **asum** e **aprint**

Makefile

```
main: main.o libarray_utils.a
    gcc -o main main.o -L. -larray_utils

libarray_utils.a: asum.o aprint.o
    ar crs libarray_utils.a asum.o aprint.o

asum.o: asum.c
    gcc -c asum.c -Wall -pedantic -g
aprint.o: aprint.c
    gcc -c aprint.c -Wall -pedantic -g
main.o: main.c array_utils.h
    gcc -c main.c -Wall -pedantic -g
```

1ª Regra: regra por omissão

```
$ make main.o
gcc -c main.c -Wall -pedantic -g
$ make
gcc -c asum.c -Wall -pedantic -g
gcc -c aprint.c -Wall -pedantic -g
ar crs libarray_utils.a asum.o aprint.o
gcc -o main main.o -L. -larray_utils
```

Exemplo aplicado à biblioteca **xalloc** com variáveis automáticas (1 de 3)

Makefile

```
# Variáveis automáticas:
#    $@ – nome do objectivo
#    $^ – requisitos (todos)
#    $< – primeiro requisito
# $(CFLAGS) – variável definida como CFLAGS
CFLAGS = -Wall -pedantic -c -g
HEADER_FILES = dyn_alloc.h dyn_dbg.h list.h
ALL = app1 app2 app3 app4
all: $(ALL)

app1: app.o dyn_alloc.o list.o dyn_dbg.o
    gcc -o app1 app.o dyn_alloc.o list.o dyn_dbg.o
# Equivalente a app1 mas usando variáveis automáticas
app2: app.o dyn_alloc.o list.o dyn_dbg.o
    gcc -o $@ $^
# Ligação com biblioteca estática
app3: app.o liballoc.a
    gcc -o $@ $^
```

Exemplo aplicado à biblioteca **xalloc** com variáveis automáticas (2 de 3)

Makefile

```
# Ligação com biblioteca estática (inclui libc.a)
app4: app.o liballoc.a
    gcc -o $@ $^ -static

liballoc.a: dyn_alloc.o dyn_dbg.o list.o
    ar crs $@ $^

app.o: app.c $(HEADER_FILES)
    gcc $(CFLAGS) $<
dyn_alloc.o: dyn_alloc.c
    gcc $(CFLAGS) $<
dyn_dbg.o: dyn_dbg.c
    gcc $(CFLAGS) $<
list.o: list.c
    gcc $(CFLAGS) $<

# Uma regra sem requisitos executa sempre
clean:
    rm -f *.o $(ALL) *.a
```

Exemplo aplicado à biblioteca **xalloc** com variáveis automáticas (3 de 3)

```
$ make
gcc -Wall -pedantic -c -g app.c
gcc -Wall -pedantic -c -g dyn_alloc.c
gcc -Wall -pedantic -c -g list.c
gcc -Wall -pedantic -c -g dyn_dbg.c
gcc -o app1 app.o dyn_alloc.o list.o dyn_dbg.o
gcc -o app2 app.o dyn_alloc.o list.o dyn_dbg.o
ar crs liballoc.a dyn_alloc.o dyn_dbg.o list.o
gcc -o app3 app.o liballoc.a
gcc -o app4 app.o liballoc.a -static

$ ls -l
-rwxr-xr-x 1 jpatri jpatri  27024 Nov 23 23:30 app1
-rwxr-xr-x 1 jpatri jpatri  27024 Nov 23 23:30 app2
-rwxr-xr-x 1 jpatri jpatri  27024 Nov 23 23:30 app3
-rwxr-xr-x 1 jpatri jpatri 886328 Nov 23 23:30 app4
```