pgfid3017900937 pgfid423920617900937 pgfid547841217900937 pgfid671761817900937 pgfid15017900937 pgfid1623920617900937 pgfid1747841217900937 pgfid24017900937 pgfid2523920617900937 pgfid2647841217900937 pgfid33017900937 pgfid3423920617900937 pgfid3547841217900937 pgfid42017900937 pgfid4323920617900937 pgfid4447841217900937 pgfid51017900937 pgfid5223920617900937 pgfid5347841217900937 pgfid60017900937 pgfid6123920617900937 pgfid6247841217900937 pgfid71017900937 pgfid7223920617900937 pgfid7347841217900937 pgfid80017900937 pgfid8123920617900937 pgfid8247841217900937 pgfid89017900937 pgfid9023920617900937 pgfid9147841217900937 pgfid98017900937 pgfid9923920617900937 pgfid10047841217900937 pgfid107017900937 pgfid10823920617900937 pgfid10947841217900937 pgfid116017900937 pgfid11723920617900937 pgfid11847841217900937 pgfid125017900937 pgfid12623920617900937 pgfid12747841217900937 pgfid140017900937 pgfid14123920617900937 pgfid14247841217900937 pgfid149017900937 pgfid15023920617900937 pgfid15147841217900937 pgfid158017900937 pgfid15923920617900937 pgfid16047841217900937 pgfid167017900937 pgfid16823920617900937 pgfid16947841217900937 pgfid176017900937 pgfid17723920617900937 pgfid17847841217900937 pgfid185017900937 pgfid18623920617900937 pgfid18747841217900937 pgfid194017900937 pgfid19523920617900937 pgfid19647841217900937

Spis treści

1 pliki obiektowe

1.1 format plików obiektowych

Nagłówki ELF

ELF

[7]Najważniejsze sekcje zawierają:

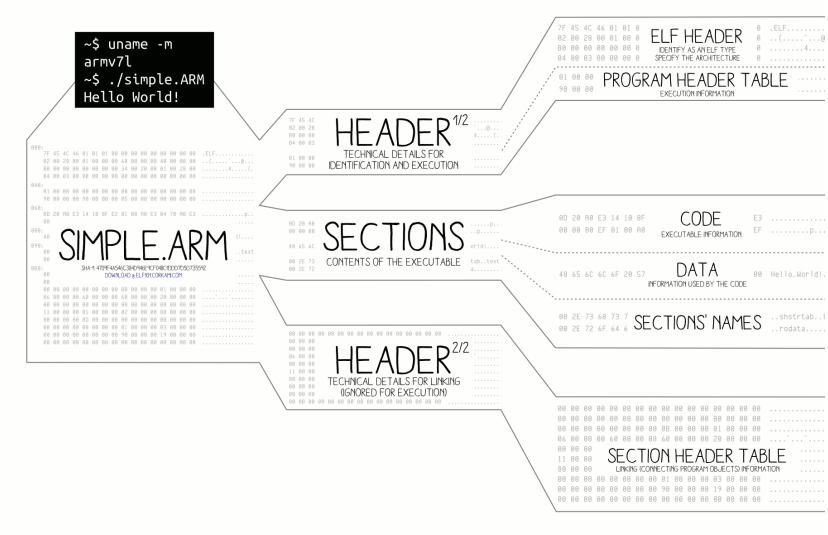
- .bss — niezainicjowane zmienne, sekcja **.bss** zajmuje nieznaczne ilości pamięci, ponieważ zawiera jedynie informacje o zmiennych, a nie przechowuje ich wartości
- .text — wykonywalne części programu
- .data — zainicjowane wartości

Sekcja vs Segment

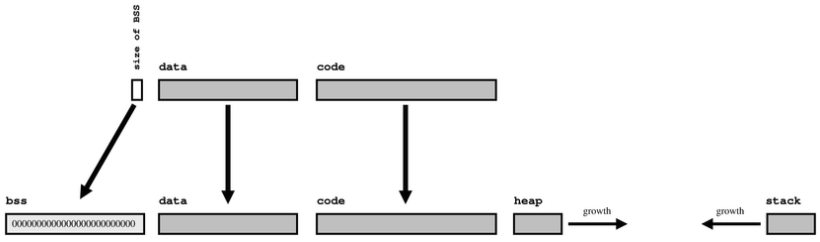
Sekcje występują przed procesem linkowania i dzielimy je na:

- "raw data" – np. .text, .data itd ...
- "metadata" – np. .symtab, .strtab itd ...

sekcje zawierające metadane nie są konieczne do uruchomienia programu, ale stanową źródło informacji dla debugerów, i programów analizujących pliki binarne [`examples/strip`]



Rysunek 1: budowa pliku ELF [4]



Rysunek 2: Proces ładowania programu do pamięci [3]

Segmenty występują po procesie linkowania i zawierają informacje jak system operacyjny ma je załadować do pamięci

1.2 Rodzaje plików obiektowych

- Pliki przesuwalne (Relocatable)
- Pliki wykonywalne (Executable)
- Pliki współdzielone (Shared Objects)

Przesuwalne (relocatable) pliki obiektowe

Przesuwalne pliki obiektowe to takie których funkcje i zmienne nie są przywiązane do adresu, ale do symboli. [5]

Plik źródłowy:

```
int add(int a, int b)
{
        return a+b;
}
```

[examples/00_relocatable]

1.2.1 Klasy symboli (nm)

Wybrane klasy symboli w pliku obiektowym [1]

- B - symbol jest w sekcji BSS (niezainicjowany)
- C - Wspólny symbol, nie zainicjowane
- D - zainicjowane symbol
- R - tylko do odczytu
- T - symbol jest w sekcji text (code)
- U - symbol niezdefiniowany

Przykład użycia programu nm na pliku obiektowym:

Wykonywalne (executable) pliki obiektowe

```
int add(int a, int b);

int main()
{
        return add(1,-1);
}
```

[examples/01_executable]

binarny plik wykonywalny zawiera adresy, natomiast przenośny plik obiektowy wszystkie adresy ma ustawione na 0

2 Biblioteki

Biblioteki statyczne

Biblioteka statyczna to archiwum plików "relocatable", więc korzysta się z niej analogicznie jak z tych plików. Nazwa takiej biblioteki zazwyczaj rozpoczyna się od 'lib' a kończy się rozszerzeniem ".a".

Aby zbudować bibliotekę statyczną korzystamy z polecenia `gcc -static -o libfoo.a foo.o`

Aby skorzystać z takiej biblioteki korzystamy z polecenia `gcc example.o -lfoo`

File	a.o	b.o	libx.a			liby.a		
Object	a.o	b.o	x1.o	x2.o	x3.o	y1.o	y2.o	y3.o
Definitions	a1, a2, a3	b1, b2	x11, x12, x13	x21, x22, x23	x31, x32	y11, y12	y21, y22	y31, y32
Undefined references	b2, x12	a3, y22	x23, y12	y11		y21		x31

Rysunek 3: zależności przy linkowaniu [3]

kolejność linkowania

Współdzielone (shared)

Aby stworzyć plik biblioteki współdzielonej musimy wywołać następującą komendę:

```
gcc -shared -o libfoo.so foo.o
```

jednak plik obiektowy foo.o musi być odpowiednio skompilowany (flaga -fPIC) ponieważ pliki typu .so muszą być „position independent”

[examples/02_shared]

2.1 Realocation and PIC

Realocation vs Position Independent Code

Re alokacja
polega na wyliczeniu poprawnych adresów obiektów z biblioteki współdzielonej przed uruchomieniem programu.

Position Independent Code
jest nowszą technologią która zakłada że poprzez dodanie warstwy abstrakcji sekcja .text ma uprawnienia tylko do odczytu, co zapewnia proste mapowanie adresów, zwiększa bezpieczeństwo i zapewnia współdzielenie kodu biblioteki przez wiele procesów

2.2 GOT

Global Offset Table

[examples/PIC_variable/]

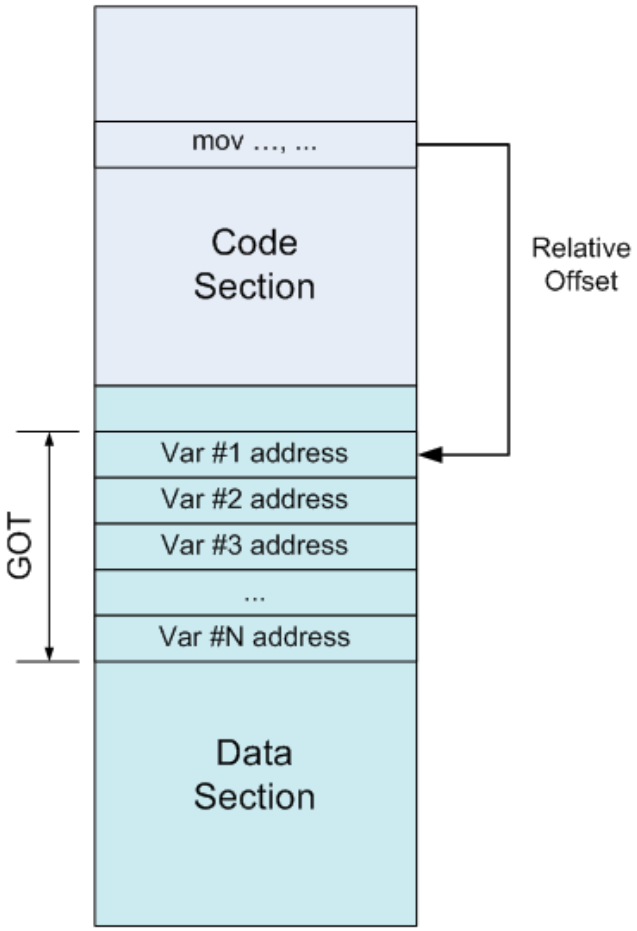
2.3 PLT

Procedure Link Table

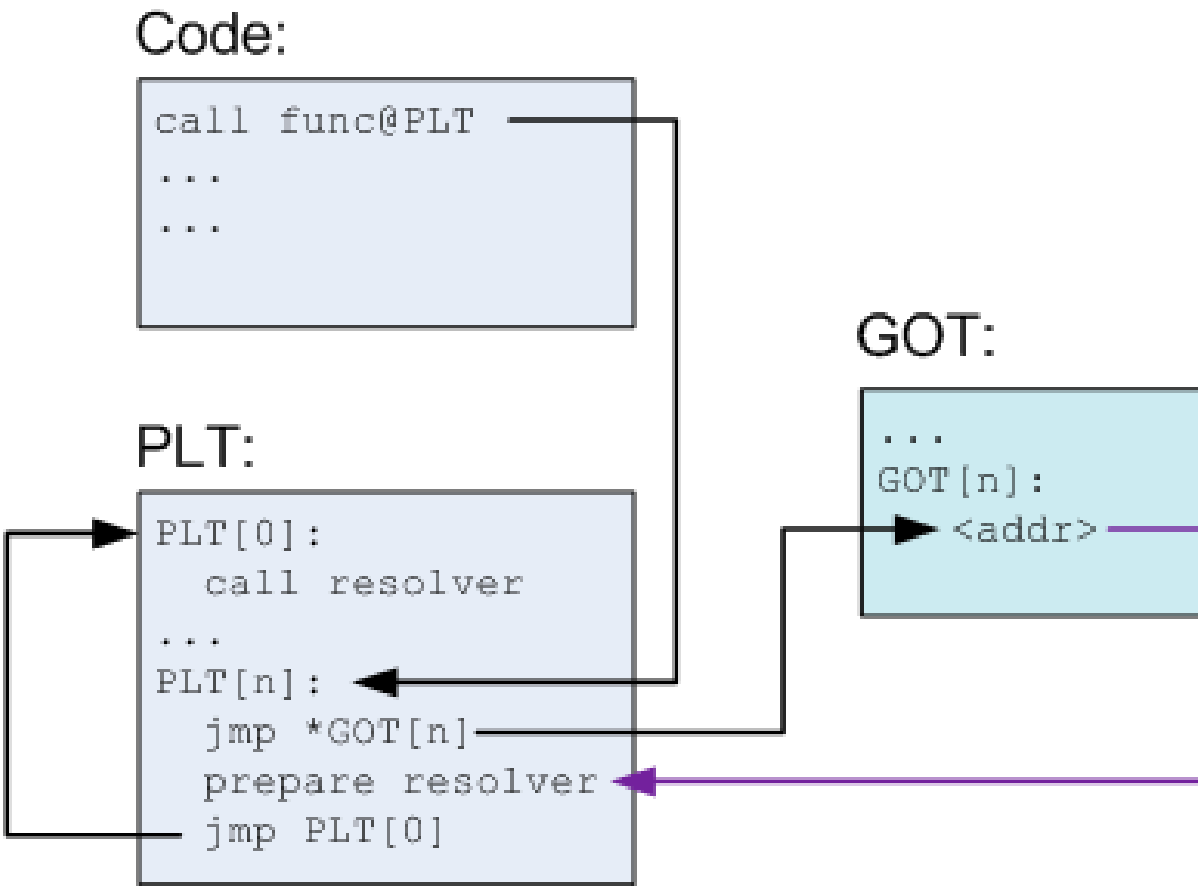
Procedure Link Table

[examples/PIC_functions]

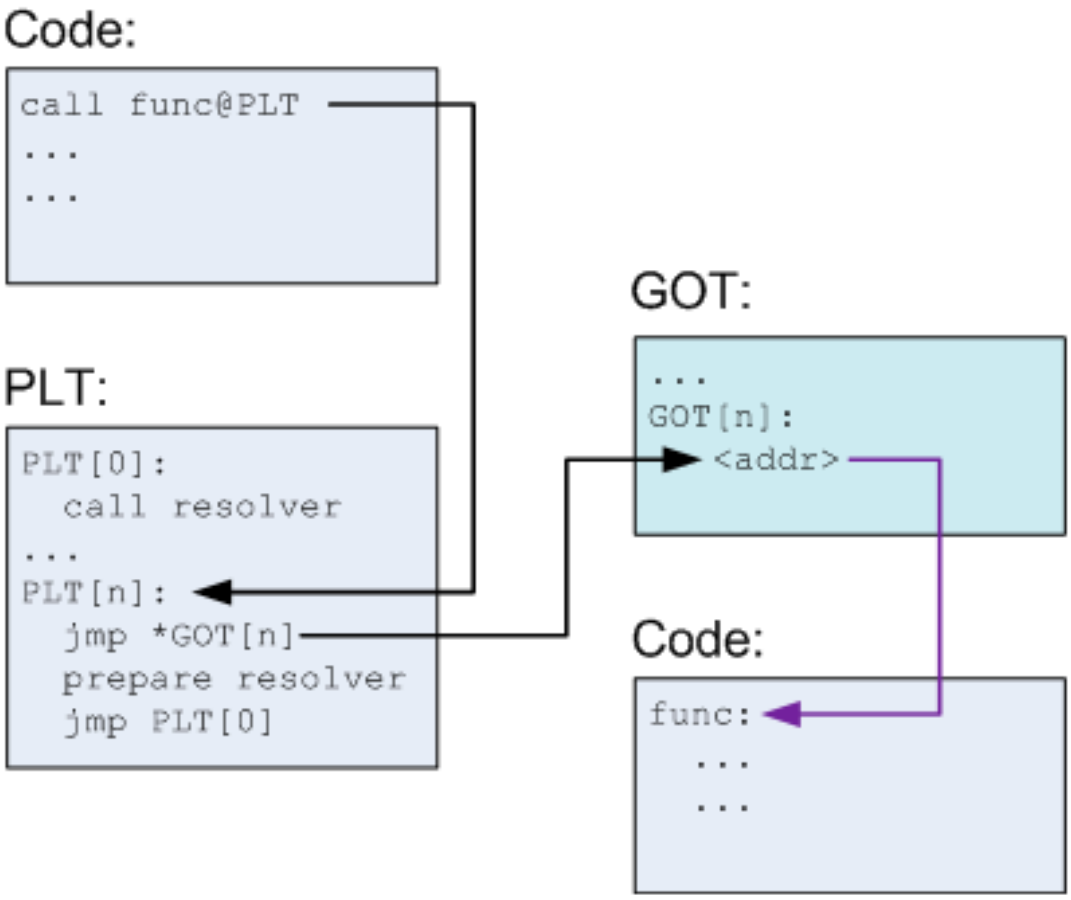
Wartość	Klasa	Nazwa
0000000000000000	D	global_initialized_variable
0000000000000004	C	global_variable
0000000000000013	t	local_function
0000000000000021	T	main
0000000000000000	T	not_implemented_function



Rysunek 4: Schemat tablicy GOT [2]



Rysunek 5: Schemat tablicy PLT przed pierwszym wywołaniem funkcji [2]



Rysunek 6: Schemat tablicy PLT po pierwszym wywołaniem funkcji [2]

2.4 Dynamiczne ładowanie bibliotek

dlfcn.h

[6] Biblioteka "dlfcn.h" umożliwia dynamiczne ładowanie bibliotek współdzielonych w trakcie działania programu

Do obiektów biblioteki mamy dostęp poprzez handler zwracany z funkcji `void * dlopen(const char *filename, in`

Do pierwszego argumentu można podać ścieżkę do pliku biblioteki współdzielonej, lub jej nazwę (wtedy będzie szukana w domyślnych katalogach lub w LD_LIBRARY_PATH)

Drugi argument określa czy linkowanie będzie leniwe, czy od razu z linkować wszystkie obiekty.

```
void * dlsym(void *handle, char *symbol);
```

Powyższa funkcja zwraca wskaźnik na obiekt którego żądamy w argumencie [`examples/DL_library`]

Literatura

[1] Manual programu nm.

[2] Eli Bendersky. Position independent code (pic) in shared libraries. [online]. [dostęp: 2018-04-04].

[3] David Drysdale. Beginner's guide to linkers. [online], Kwiecień 2018. [dostęp: 2018-04-04].

[4] Ciro Santilli. Elf hello world tutorial. [online]. [dostęp: 2018-04-04].

[5] Carson Tang. Guide to object file linking. [online], June 2013. [dostęp: 2018-04-04].

[6] David A. Wheeler. Program library howto. [online], April 2013. [dostęp: 2018-04-04].

[7] Eric Youngdale. The elf object file format: Introduction. [online], April 1995. [dostęp: 2018-04-04].