

Biblioteki dzielone w UNIXie (na przykładzie FreeBSD)

Rafał Jaworowski
raj@semihalf.com

MeetBSD 2005, Kraków 17-19 czerwca 2005

Wstęp

- Zakres i profil wykładu
- Czym są biblioteki dzielone
- Skąd
 - SVR4 ABI – ELF
 - implementacja SunOS (~87)
- Gdzie
 - /lib/
 - /usr/lib/
 - /usr/local/lib/

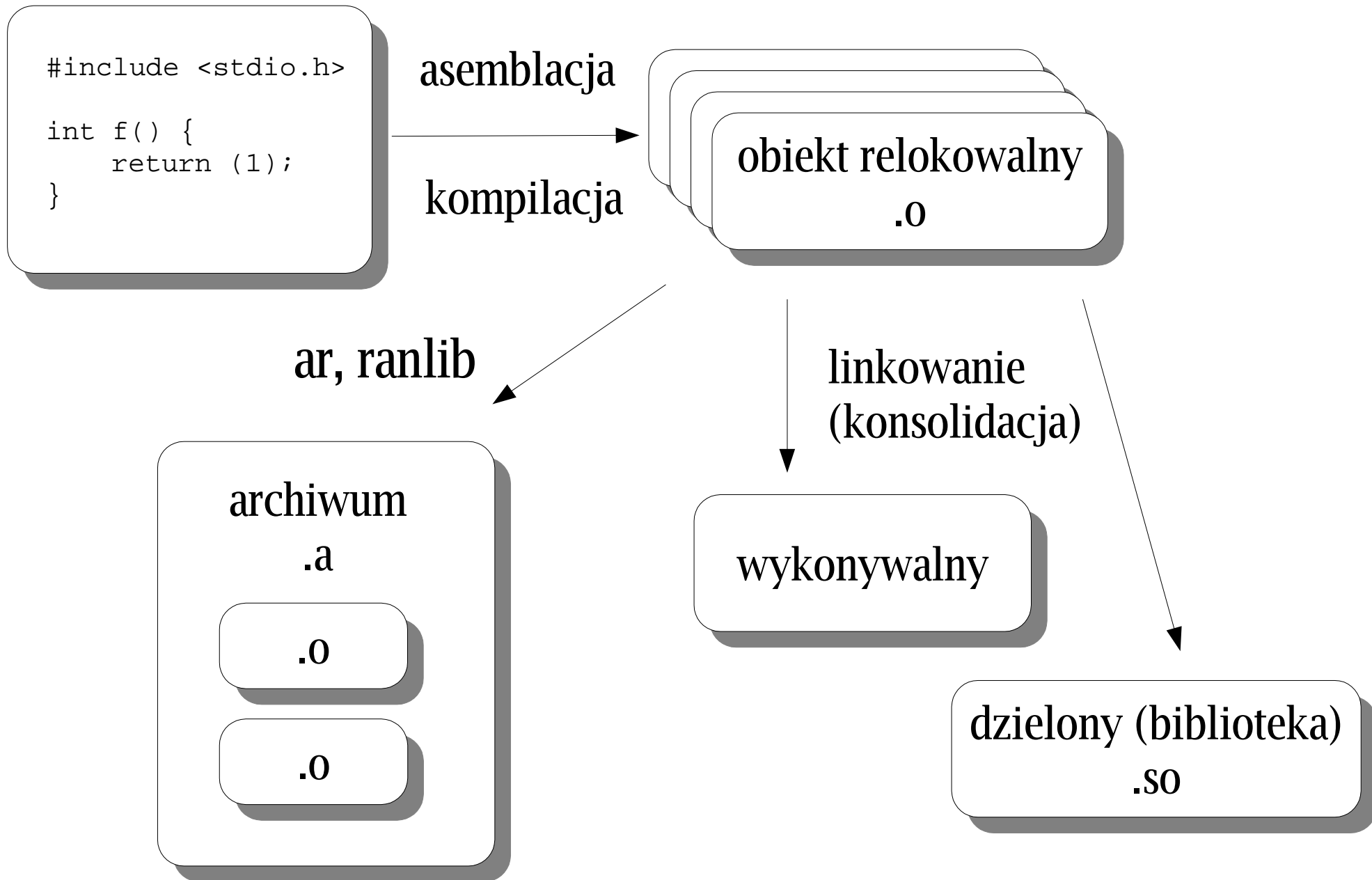
Elementy układanki

- Binaria ELF
 - wykonywalne, obiekty dzielone (.so), obiekty relokowalne (.o)
- Obiekty relokowalne – podstawowe cegiełki (kod wykonywalny i dane) w uzgodnionym opakowaniu
- Biblioteki
 - archiwa .a – zbiór wielu prostych obiektów relokowalnych (sklejone .o)
 - obiekty (biblioteki) dzielone (Shared Object, Dynamic Shared Object) - dynamicznie linkowane

Link editing – podstawy

- Dwa aspekty
 - linkowanie obiektów (konsolidacja) – końcowa faza kompilacji
 - run time linking (dla dynamicznie linkowanych)
- Formy linkowania obiektu wykonywalnego
 - statyczne – powstaje samodzielny program wykonywalny (w kontekście OS)
 - dynamiczne - powstaje "niekompletny" program, uzależniony od dzielonego obiektu (biblioteki)
- Dynamiczne
 - w istocie ten sam cel i efekt końcowy, pewne operacje są odłożone w czasie (run-time vs. compile time)

Tworzenie binariów



Jak zbudować bibliotekę dzieloną

- Implementacja funkcji bibliotecznych
 - `gcc -fPIC -c example.c`
 - `%gcc -shared -Wl,-soname,mylib.so.1 -o libmylib.so.1 example.o`
 - `%file libmylib.so.1`
`libmylib.so.1: ELF 32-bit LSB shared object, Intel 80386, version 1 (FreeBSD), not stripped`
- “niekompletny” obiekt, m.in. nie zawiera właściwego punktu wejścia
- wyjątek *ld-elf.so.1* jest jednocześnie SO i de facto wykonywalne
- Position Independent Code (PIC)

Jak używać biblioteki

- Automatycznie
 - Instalacja do katalogu lub przez `LD_LIBRARY_PATH`
 - `ldconfig` – buduje cache `/var/run/ld-elf.so.hints`
 - potencjalnie niebezpieczne! (`/etc/rc.d/ldconfig`)
 - `%gcc main.c -lmylib`
- Interfejs “`dlopen()`”
 - jawne odwołanie do biblioteki

Dynamiczny linker

- początek w `execve()`

- dynamiczny linker

```
- ll /libexec/ld-elf.so.1
```

```
-r-xr-xr-x 1 root wheel 141604 5 Lis 2004 /libexec/ld-elf.so.1
```

```
- ldd /bin/ls
```

```
/bin/ls:
```

```
libutil.so.4 => /lib/libutil.so.4 (0x28082000)
```

```
libncurses.so.5 => /lib/libncurses.so.5 (0x2808e000)
```

```
libc.so.5 => /lib/libc.so.5 (0x280cd000)
```

```
- readelf -a /bin/ls
```

```
[...]
```

```
Program Headers:
```

Type	Offset	VirtAddr	PhysAddr	FileSiz	MemSiz	Flg	Align
PHDR	0x000034	0x08048034	0x08048034	0x000c0	0x000c0	R E	0x4
INTERP	0x0000f4	0x080480f4	0x080480f4	0x00015	0x00015	R	0x1
[Requesting program interpreter: <code>/libexec/ld-elf.so.1</code>]							
LOAD	0x000000	0x08048000	0x08048000	0x04fb0	0x04fb0	R E	0x1000

0,5 sek. z życia dynamicznego linkera

- mapowanie bibliotek do przestrzeni adresowej procesu
- inicjalizacja
- rozwiązywanie symboli
- binding (powiązanie symbolu z jego adresem)
 - domyślnie podczas pierwszej referencji (lazy bound)
 - LD_BIND_NOW – w momencie załadowania
- zmienne LD_
 - LD_LIBRARY_PATH etc.

0,5 sek. z życia dynamicznego linkera

- punkt wejścia – symbol *.rtld_start*
- implementacja
np. `libexec/rtld-elf/i386/rtld_start.S`
- główna funkcja niezależna od sprzętu *_rtld()*
`libexec/rtld-elf/rtld.c`
- po wyjściu z *_rtld()* otrzymujemy adres f. `main()`
głównego programu
- skok do `main()`

_rtld()

- relokacja kodu samego linkera dynamicznego
- załadowanie obrazu głównego programu
- załadowanie wszystkich potrzebnych obiektów, od których mamy zależności
- relokacja załadowanych obiektów
 - w tym rozwiązanie odwołań symbolicznych
- zawołanie funkcji inicjalizujących dla załadowanych obiektów (sekcje `.init`)
- powrót (oddaje adres *main()*)

Interfejs “dlopen()”

- dlopen(), dlsym(), dlfunc(), dlerror(), dlclose()
- jawne otwieranie i zamykanie biblioteki, wyszukanie symbolu, użycie itp.
- ta sama infrastruktura

...

```
handle = dlopen("/libm.so.3", RTLD_LAZY);  
if (!handle) {  
    /* obsługa błędu z dlerror(); */  
}
```

```
sine = dlsym(handle, "sin");  
if ((error = dlerror()) != NULL) {  
    /* obsługa błędu */  
}
```

```
res = (*sine)(1.2);  
dlclose(handle);
```

Podsumowanie

- Jak debugować bibliotekę dzieloną?
 - specyficzne efekty np. chwilowo nierozwiązane symbole
- Kiedy uratuje nas `/rescue`?
 - co zrobić po skasowaniu `ld-elf.so.1`, `libc` itp.
- Czy biblioteka dzielona może być zależna od innego SO?
- Referencje
 - `man 1 ld`
 - `man 1 rtld, ld-elf.so.1`
 - `man 3 dlopen, dlsym` itd.

Podsumowanie

- Wady
 - gorsza wydajność (prebinding – MacOS X)
 - trudniejsze w utrzymaniu, więcej plików-komponentów
 - wymagana spora infrastruktura
- Zalety
 - redukcja rozmiaru binariów
 - numeracja wersji, pozwala zarządzać ABI
- FreeBSD
 - nie używamy minor w nazwie pliku, SONAME
 - libmap.conf

Biblioteki dzielone w UNIXie (na przykładzie FreeBSD)

Rafał Jaworowski
raj@semihalf.com

Dziękuję za uwagę

http://www.semihalf.com/pub/meetbsd/2005_biblioteki.pdf

MeetBSD 2005, Kraków 17-19 czerwca 2005