Biblioteci. Gestiunea bibliotecilor

Răzvan Deaconescu
razvan@rosedu.org

Întâlnirile lunare RLUG – Iulie 2011

14 iulie 2011

Outline

Fișiere obiect

- fișiere binare
- cod compilat și asamblat
- format ELF, COFF, PE

Biblioteci

- colecție de fișiere obiect
- funcționalitățile oferite de fișierele obiect sunt disponibile în bibliotecă
- link-area unei biblioteci pentru obținerea unui executabil

De ce biblioteci?

- modularizare, reutilizare
- *reinventing the wheel*

Biblioteci statice

- colecție “dumb” de fișiere obiect
- linkarea unei biblioteci statice înseamnă adăugarea de cod executabil
- se adaugă întreg codul modulului obiect din care provine funcția
- `ar rc libmylib.a a.o b.o c.o`

Biblioteci dinamice

- fișier cu format specializat în care sunt colectate fișierele obiect
- deține informații suplimentare despre obiecte, simboluri, funcții
- linkarea înseamnă doar marcarea unor referințe către bibliocă
- încărcarea codului din cadrul bibliotecii va fi realizată ulterior
- `.so` pe Unix, `.dll` pe Windows
- `gcc -shared -o libmylib.so a.o b.o c.o`
- fișierele obiect compilate cu `-fPIC`

Biblioteci statice vs. dinamice

- statice
  - cod portabil (independent de bibliotecile sistemului)
  - nu există overhead la load-time sau run-time
- dinamice
  - dimensiune mică a executabilului
  - eficiență în folosirea memoriei – bibliotecile sunt “partajate” de mai multe procese

Biblioteci vs. headere

- bibliotecile sunt colecții de fișiere obiect
- headerele sunt fișiere cu extensia `.h` ce conțin
  - declarații de funcții
  - macrodefiniții
  - definiții de structuri și tipuri de date
- un header este inclus de un fișier C sau alt fișier header
- o bibliotecă este linkată la un executabil
- un header este scris de programator
- o bibliotecă este obținută prin "comasarea" fișierelor obiect
- header – preprocesare
- bibliotecă – linking

Analiza unei biblioteci

- ldd – listarea dependențelor pentru bibliotei dinamice
- nm – listarea simbolurilor
- readelf – citește fișierele ELF

Outline

Linker-ul

- rezolvarea simbolurilor și unirea modulelor obiect
- un simbol: o variabilă, un nume de funcție
- la compilare, simbolurile folosite în modul, dar nedefinite, sunt marcate special (undefined)
- rezolvare = descoperirea modulului în care este definit simbolurile
- linker-ul este folosit pentru a obține executabile și biblioteci dinamice
- `LD_FLAGS` – flag-uri de linker
  - `-L.` – la linkare sunt căutate bibliotecile și în directorul curent
- `LD_LIBS` – biblioteci linkate
  - `-ltorrent`

Loader-ul

- încarcă programul în execuție și începe rularea procesului
- load time = la execuție
- loaderul are cunoștință de formatul executabilului
- traduce zonele din fișierul executabil în zone de memorie
- apelat prin intermediul `execve(2)`

Static linking

- folosit la legarea modulelor obiect și a bibliotecilor statice
- tot codul necesar este “colectat” în executabil
- executabilul poate fi portat pe un sistem ce nu deține bibliotecile folosite
- se poate folosi opțiunea `-static` la gcc

Dynamic linking

- folosit la legarea bibliotecilor dinamice
- se adaugă puțin cod în executabil
- aducerea codului necesar în memorie se realizează mai târziu

Load time dynamic linking

- codul necesar din cadrul bibliotecii este adus în memorie la load time (lansarea în execuție)
- se ocupă loaderul
- loaderul trebuie să știe unde să caute
  - opțiunea `-L.` este folosită la **link time** nu la **load time**
- se lansează procesul și se adaugă codul bibliotecii
- dacă biblioteca există în memorie (încărcată de alt proces), atunci este referită și partajată

Run time dynamic linking

- codul este adus în memorie la run time (în momentul în care procesul se execută)
- similaritate cu `malloc`
- dezvoltatorul precizează numele bibliotecii; căutarea se face similar ca la load time dynamic linking
- `dlopen` & friends
- `LoadLibrary` & friends
- De ce?
  - plugins
  - hooking (injection)

LD_PRELOAD

- nume de fișiere de tip bibliotecă ce sunt încărcate înaintea altora
  - **nu** căi către directoare cu biblioteci (cum se întâmplă la `LD_LIBRARY_PATH`
- scenariu tipic – hooking
  - se creează un modul ce implementează `malloc`
  - se creează o bibliotecă ce conține modulul
  - se folosește `LD_PRELOAD`
  - la un apel malloc se apelează biblioteca proprie
  - folosind `dlopen` & friends se apelează `malloc` din biblioteca standard C

LD_DEBUG

- depanarea interacțiunii cu biblioteca
- `export LD_DEBUG=help`
- `export LD_DEBUG=files`

Outline

Biblioteci statice

- se păstrează în directoare standard (`/usr/lib`, `/lib`)
- altfel, se precizează la link time calea către bibliotecă (`LD_FLAGS`, `-L.`)
- nu este nevoie de un management specializat

Biblioteci dinamice

- rezolvarea referințelor se face la link time, fără a încărca bucăți de cod în fișierul executabil
- calea către biblioteci trebuie rezolvată la load time (sau run time)
- trebuie configurat loader-ul, care nu este apelat explicit de utilizator (este apelat prin intermediul `execve`)
- `man ld.so`

What is this?

- `ld.so`
- `ld-linux.so`
- `/etc/ld.so.conf`
- `/etc/ld.so.conf.d/`
- `/etc/ld.so.cache`
- `/etc/ld.so.preload`

LD_LIBRARY_PATH

- precizează căi suplimentare unde să fie căutate bibliotecile dinamice
- similar cu `PATH` – separație prin caracterul "două puncte" / "colon" (`:`)
- `LD_LIBRARY_PATH=. ./exec`

ldconfig

- controlează cache-ul de de biblioteci dinamice
- cache-ul este stocat în fișierul `/etc/ld.so.cache`
- loader-ul caută bibliotecile urmând un set de pași dați (`man ld.so`)
- înainte de a căuta în directoarele implicite (`/lib`, `/usr/lib`), va căuta în cache – viteză sporită
- `ldconfig` este apelat în general la instalarea de aplicații de sistemul de gestiune a pachetelor (`apt`, `yum`)
- `ldconfig -p`
- `ldconfig -n .`

Outline

Resurse utile

- `http://www.dwheeler.com/program-library/`
- MSDNAA – Dynamic Link Libraries – `http://msdn.microsoft.com/en-us/library/ms682589(v=VS.85).aspx`

Keywords

- fișiere obiect
- biblioteci
- biblioteci statice
- biblioteci dinamice
- `-fPIC`
- header
- linker
- loader
- `ld.so`
- run-time
- load-time
- `LD_PRELOAD`
- `LD_DEBUG`
- `LD_LIBRARY_PATH`
- `/etc/ld.*`
- `ldconfig`