

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

Système Bibliothèques

Gilles Roussel

Gilles.Roussel@univ-mlv.fr

<http://igm.univ-mlv.fr/~roussel/L22/>

Licence 2

27 mars 2011

Créer une bibliothèque statique

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- Extension **.a**
- Ensemble de fichiers objets regroupés dans une archive contenant un index des fonctions de la l'archive (`nm -s` affiche l'index)
- Archivage avec la commande **ar**
 - **c** pour créer l'archive (elle est créée par défaut s'il y a un ajout)
 - **r** pour ajouter (remplacer) un fichier dans l'archive ;
 - **d** pour supprimer un fichier de l'archive ;
 - **x** pour extraire un fichier de l'archive ;
 - **t** pour lister le contenu de l'archive
- **Ordre** des fichiers dans l'archive important (les fonctions sont recherchées dans l'ordre des fichiers). Modificateurs permettent de préciser la position d'insertion.

Créer un bibliothèque statique

Système

Gilles Roussel

Bibliothèque

Liaison

statique

Liaison

dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

```
# ar cr libf.a f.o
# ar l libf.a
f.o
# nm -s libf.a
Index de l'archive:
f in f.o
f.o:
00000000 T f
# gcc -o main main.o -L. -lf
# nm main | grep 'T '
080483ac T f
08048478 T _fini
08048278 T _init
08048410 T __libc_csu_fini
080483bc T __libc_csu_init
08048368 T main
080482c0 T _start
```

Créer une bibliothèque dynamique

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- Fichiers objets relogeables créés avec gcc -c et l'option **-fpic** ou **-fPIC** ;
 - code indépendant de sa position dans la mémoire (*Position Independent Code*)
 - adresses absolues accédées *via* une table d'indirection (*Global Offset Table*)
 - table remplie au chargement de la bibliothèque par le chargeur dynamique (ld-linux.so ou ld.so)
- Bibliothèque avec extension **.so** (.dll sous Win)
- Création *via* gcc
 - **-shared** indique qu'il faut générer la bibliothèque ;
 - **-fpic** indique le format utilisé pour fichiers objets ;
 - **-o libname.so**

Créer une bibliothèque dynamique

Système

Gilles Roussel

Bibliothèque

Liaison

statique

Liaison

dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de

ld-linux.so

```
# gcc -c -fpic f.c
# gcc -shared -fpic -o libf.so f.o
# file libf.so
libf.so: ELF 32-bit LSB shared object, Intel 80386,
version 1 (SYSV), not stripped
# gcc -o main -L. -lf main.o
# nm main | grep 'T '
0804857c T _fini
0804837c T _init
08048514 T __libc_csu_fini
080484c0 T __libc_csu_init
0804847c T main
080483d4 T _start
# ./main
./main: error while loading shared libraries:
libf.so: cannot open shared object file:
No such file or directory
# ldd main
libf.so => not found
libc.so.6 => /lib/tls/libc.so.6 (0x009f4000)
/lib/ld-linux.so.2 (0x009db000)
```

Où sont recherchées les bibliothèques ?

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- Dans un ensemble de répertoires prédéfinis (/lib et /usr/lib)
- Dans un ensemble de répertoires précisés par **root** dans le fichier **/etc/ld.so.conf**
- Liste des bibliothèques placées dans un cache par **ldconfig** (avec option **-p** liste le cache)
- Dans un ensemble de répertoires précisés par l'utilisateur dans la variable d'environnement **LD_LIBRARY_PATH**

```
# LD_LIBRARY_PATH=. ./main
```

18

Comment se fait l'appel statique de fonction ?

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

static_link.c

```
#include <stdio.h>

int f (int i) {
    return 2*i;
}

int main(int argc, char *argv[]) {
    int (*g)(int) = (int (*)(int)) 134513604;
    int (*h) (int, char**) = (int (*)(int, char**)) 134513614;
    printf("%d\n",g(2));
    h(argc,argv);
    printf("%d\n",f(3));
    return 0;
}
```

```
# nm -t d static_link | grep 'T f'
134513512 T f
```

Comment se fait l'appel statique de fonction ?

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- À la création du fichier objet les fonctions définies n'ont pas d'adresse absolue

```
# nm f.o | grep 'T f'
00000000 T f
```

- À la création du fichier objet pour chaque appel de fonction ont ajoute une entrée dans la table réallocation

```
# objdump -r main.o
...
00000025 R_386_PC32 f
...
```

- Les appels des fonctions utilisent comme adresse leur position dans le code

```
# objdump -d main.o
...
22 : 6a 03                push    $0x3
24 : e8 fc ff ff ff      call   25 <main+0x25>
...
```

Comment se fait l'appel statique de fonction ?

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- Au moment de l'édition de liens, l'adresse absolue des fonctions est calculée

```
# nm main | grep 'T f'
08048368 T f
```

- Les adresses absolues sont placées dans le code

```
# objdump -d src/main
...
804839a :      6a 03                push    $0x3
804839c :      e8 c7 ff ff ff       call    8048368 <f>
...
```

- Les adresses absolues sont les adresses, dans la mémoire du processus, où sont placées les instructions du code de la fonction
- Les fonctions sont donc toujours placées **au même endroit**

Comment se fait l'appel dynamique de fonction ?

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- Tout n'est pas chargé à partir du même fichier
- Les adresses absolues des fonctions de bibliothèque ne sont pas connues avant leur chargement
- Il n'est pas possible de les placer toujours au même endroit
- Les adresses ne peuvent pas être résolues directement
- Il y a besoin d'une indirection

Comment se fait l'appel dynamique de fonction ?

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- Appel à travers une *Procedure Linkage Table* (PLT) ;
- À l'origine remplie avec l'adresse de l'éditeur de liens dynamique (ld-linux.so)
- Après le premier appel, l'adresse de l'éditeur dynamique de liens est remplacée par adresse absolue de la fonction (sauf si LD_BIND_NOW affectée)
- *Global Offset Table* (GOT) permet de trouver l'adresse absolue des fonctions

Charger soi-même une bibliothèque dynamique

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- Permet d'utiliser des fonctions qui ne sont pas écrites au moment de la compilation
- Utilisé pour étendre simplement un logiciel (*plugin*)
- Nécessite de connaître les noms de fonctions à appeler et leur prototype
 - Mécanisme utilisé par `ld-linux.so` pour les fonctions dont les noms sont connus au moment de la compilation
 - Utilisé par exemple par mozilla : les bibliothèques sont recherchées dans des répertoires, associées à un type MIME, manipulées par des fonctions prédéfinies (`NP_Initialize`, `NPP_New`, etc.)

L'Application Programming Interface (API)

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

Fournie par la bibliothèque `libdl`

`dlopen()` chargement de la bibliothèque ;

`dlsym()` récupération de l'adresse du symbole recherché ;

`dlclose()` fermeture de la bibliothèque ;

`dlerror()` gestion des erreurs.

Un exemple

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

acroread_plugin.c

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>

void check_error(char* error,int exit_value);

int main(int argc, char *argv[]) {
    void *handle;
    char *plugin = "/usr/local/Acrobat5/Browsers/intellinux/nppdf.so";
    char *function = "NP_GetMIMEDescription";
    char* (*mime_type) ();
    dlerror(); /* Effacer les erreurs */
    handle = dlopen(plugin, RTLD_LAZY);
    check_error(dlerror(),1);
    mime_type = dlsym(handle,function);
    check_error(dlerror(),2);
    fprintf (stdout,"%s\n", mime_type());
    dlclose(handle);
    check_error(dlerror(),3);
    return 0;
}

void check_error(char *error, int exit_value) {
    if (NULL!= error) {
        fprintf (stderr, "%s\n", error);
        exit(exit_value);
    }
}
```

Compilation et exécution

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

Makefile

```
CFLAGS=-ansi -Wall -rdynamic
```

```
acroread_plugin:
```

```
    $(CC) $(CFLAGS) -o acroread_plugin \  
    acroread_plugin.c -L/usr/X11R6/lib -lXt -ldl
```

```
# make
```

```
cc -ansi -Wall -rdynamic -o acroread_plugin  
acroread_plugin.c -L/usr/X11R6/lib -lXt -ldl
```

```
# ./acroread_plugin
```

```
application/pdf :pdf :Portable Document Format
```

dlopen()

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- Recherche le fichier dont le nom est passé en premier argument
- Charge récursivement en mémoire le code des bibliothèques nécessaires
- Met à jour la GOT
 - Si le deuxième argument est `RTLD_NOW` (ou `LD_BIND_NOW` affectée) tous les symboles sont résolus
 - Si le deuxième argument est `RTLD_LAZY` les symboles sont résolus lorsqu'ils sont utilisés
- Retourne un pointeur d'accès à la bibliothèque ou `NULL` en cas d'erreur
- Possibilité d'appeler des fonctions à la première ouverture

dlsym()

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- Prend en argument un pointeur d'accès à une bibliothèque
- Recherche l'adresse de chargement d'un symbole dont le nom est passé en deuxième argument
- Retourne l'adresse de la fonction recherchée et NULL en cas d'erreur

dlclose() et dlerror()

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- dlclosure()

- Ferme la bibliothèque dont le pointeur d'accès est passé en argument
- Si cette bibliothèque n'est plus ouverte par personne, libère la mémoire associée
- Retourne 0 en cas de succès

- dlerror()

- Retourne une chaîne de caractères décrivant la dernière erreur qui s'est produite
- Si elle est appelée deux fois le deuxième appel retourne NULL

Préchargement de bibliothèque

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

- Il est possible de précharger des bibliothèques
- Variable d'environnement `LD_PRELOAD` permet d'indiquer une bibliothèque à précharger
- Permet de masquer les fonctions d'autres bibliothèques
- Constante `RTLD_NEXT`, pseudo pointeur d'accès à une bibliothèque, permet d'aller chercher les fonctions masquées (défini dans `dlfcn.h` si la constante `_GNU_SOURCE` est définie)

Préchargement de bibliothèque

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

malloc.c

```
#define _GNU_SOURCE
#include <stdlib.h>
#include <stdio.h>
#include <dlfcn.h>
extern void init_lib() __attribute__((constructor));

static void check_error(char *error, int exit_value);
static void error();

static void* (*real_malloc)(size_t, size_t) = (void * (*)(size_t, size_t)) error;
static void* (*real_free)(void *) = (void * (*)(void *)) error;
static void* (*real_realloc)(void *, size_t) = (void * (*)(void *, size_t)) error;

void init_lib() {
    fprintf(stderr, "libmalloc init begin\n");
    dlerror(); /* Effacer les erreurs */
    real_malloc = dlsym(RTLD_NEXT, "malloc");
    check_error(dlerror(), 1);
    real_free = dlsym(RTLD_NEXT, "free");
    check_error(dlerror(), 2);
    real_realloc = dlsym(RTLD_NEXT, "realloc");
    check_error(dlerror(), 3);
    fprintf(stderr, "libmalloc init end\n");
}
```

Préchargement de bibliothèque

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

malloc.c

```
void *calloc(size_t nmemb, size_t size) {
    fprintf(stderr,"calloc\n");
    return real_calloc(nmemb,size);
}

void *malloc(size_t size) {
    fprintf(stderr,"malloc\n");
    return real_malloc(size);
}

void free(void *ptr) {
    fprintf(stderr,"free\n");
    real_free(ptr);
}

void *realloc(void *ptr, size_t size) {
    fprintf(stderr,"realloc\n");
    return real_realloc(ptr,size);
}

void check_error(char *error, int exit_value) {
    if (NULL != error) {
        fprintf (stderr, "%s\n", error);
        exit(exit_value);
    }
}

void error() {
    fprintf(stderr,"ERROR: called before init_lib\n");
    exit(EXIT_FAILURE);
}
```

Compilation et exécution

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

Makefile

```
CFLAGS=-ansi -Wall -fpic

libmalloc.so: malloc.o
    gcc -shared -fpic -o libmalloc.so malloc.o -ldl
```

test_malloc.c

```
#include<stdlib.h>
int main(int argc, char* argv[]) {
    void *ptr = malloc(12);
    return 0;
}
```

```
# LD_LIBRARY_PATH=. LD_PRELOAD=libmalloc.so ./test_malloc
libmalloc init begin
libmalloc init end
malloc
```

Simulation de ld-linux.so

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

dynamic.c

```
#include <stdlib.h>
#include <stdio.h>
#include <dlfcn.h>
void check_error(char* error,int exit_value);
int ld_linux_f(int);

void * GOT[] = {ld_linux_f};

int PLT_F (int v) {
    return ((int (*)(int)) GOT[0])(v);
}

int main(int argc, char* argv[]) {
    printf("program start\n");
    printf("f(4)=%d\n",PLT_F(4));
    printf("f(2)=%d\n",PLT_F(2));
    return 0;
}
```

Simulation de ld-linux.so

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

dynamic.c

```
int ld_linux_f (int v) {
    void *handle;
    fprintf(stderr,"dynamic loading start\n");
    dlerror();    /* Nettoyer les erreurs */
    handle = dlopen("libf.so", RTLD_LAZY);
    check_error(dlerror(),1);
    fprintf(stderr,"libf.so loaded\n");
    GOT[0] = dlsym(handle, "f");
    check_error(dlerror(),1);
    fprintf(stderr,"f found\n");
    fprintf(stderr,"dynamic loading end\n");
    return ((int (*) (int)) GOT[0])(v);
}

void check_error(char *error, int exit_value) {
    if (NULL != error) {
        fprintf (stderr, "%s\n", error);
        exit(exit_value);
    }
}
```

Compilation et exécution

Système

Gilles Roussel

Bibliothèque

Liaison
statique

Liaison
dynamique

Chargement à
la volée

ld-linux.so

Préchargement

Simulation de
ld-linux.so

Makefile

```
CFLAGS=-ansi -Wall -rdynamic
```

```
dynamic:  
    $(CC) $(CFLAGS) -o dynamic \  
    dynamic.c -ldl
```

```
# make  
cc -ansi -Wall -rdynamic -o dynamic \  
dynamic.c -ldl  
# LD_LIBRARY_PATH=. ./dynamic  
program start  
dynamic loading start  
libf.so loaded  
f found  
dynamic loading end  
f(4)=32  
f(2)=8
```