ld.so, ld-linux.so

loader/linker dynamique

Le linker dynamique peut être lancé soit indirectement par un programme lié dynamiquement ou une librairie, ou directement en exécutant :

/lib/ld-linux.so.* [OPTIONS] [PROGRAM [ARGUMENTS]]

les programmes ld.so et ld-linux.so* trouvent et chargent les librairie nécessaires à un programme, préparent le programme et le lance.

Les binaires linux nécessitent une liaison dynamique sauf si l'option -static a été donné à ld(1) durant la compilation.

Le programme ld.so manipule les binaires a.out, un format utilisé depuis longtemps, ld-linux.so* manipulent les formats ELF (/lib/ld-linux.so.1 pour libc5, /lib/ld-linux.so.2 pour glibc2), qui sont utilisés depuis de nombreuses années. Sinon, les 2 ont le même comportement, et utilisent les même fichiers et programmes ldd(1), ldconfig(8), et /etc/ld.so.conf

En résolvant les dépendances de librairies, le linker inspecte d'abord chaque chaîne de dépendance pour voir s'il contient un slash (cela se produit si un chemin de librairie contenant des '/' a été spécifié au moment du lien). Si un slash est trouvé, la chaîne de dépendance est interprété comme chemin (relatif ou absolu), et la librairie est chargée en utilisant ce chemin.

Si une dépendance de librairie ne contient pas de '/', elle recherché dans l'ordre suivant :

- (ELF uniquement) Utiliser les répertoires spécifiés dans l'attribut de section dynamique DT_RPATH du binaire si présent et que l'attribut DT_RUNPATH n'existe pas. L'utilisation de DT_RPATH est déprécié
- Utilise la variable d'environnement LD_LIBRARY_PATH. Excepté si l'exécutable est un binaire set-user-ID/set-group-ID, auquel cas il est ignoré
- (ELF uniquement) Utilise les répertoires spécifiés dans l'attribut de section dynamique DT_RUNPATH du binaire si présent
- depuis le fichier de cache /etc/ld.so.cache, qui contient une liste compilée de librairies candidat précédemment trouvé. Si, cependant, le binaire a été lié avec l'option du linker -z nodeflib, les librairies dans les chemins par défaut sont sautés.
- Dans le chemin par défaut /lib, puis /usr/lib. Si le binaire a été lié avec -z nodeflib, cette étape est sauté

Expansion de token rpath

ld.so comprend certaines chaîne dans une spécification rpath (DT_RPATH ou DT_RUNPATH) ; ces chaînes sont substituées comme suit :

$ORIGIN ou ${ORIGIN}** Étend au répertoire contenant l'exécutable de l'application. Donc, une application localisée dans somedir/app pourrait être compilé avec **gcc -Wl,-rpath,'$ORIGIN/../lib' pour qu'il trouve une librairie partagée dans somedir/lib peut importe où se trouve somedir.

$LIB ou ${LIB} S'étends à lib ou lib64 en fonction de l'architecture.

$PLATFORM ou ${PLATFORM} s'étends à la chaîne correspondant au type de processeur du système hôte (ex : X86_64). Dans certaines architectures, le kernel linux ne fournis pas un telle chaîne.

OPTIONS

–list Liste toutes les dépendances et comment elles sont résolues

–verify Vérifie que le programme est lié dynamiquement et que ce linker peut le gérer.

–library-path PATH Utilise le PATH spécifié au lieu de la variable LD_LIBRARY_PATH

–inhibit-rpath LIST Ignore les information RPATH et RUNPATH dans les noms d'objet dans LISTE.

–audit LIST Utilise les objets nommés dans LIST comme auditeurs.

Capacités hardware

Certaines librairies sont compilées en utilisant des instructions spécifiques au hardware qui n'existe pas sur tous les CPU. De telles librairies devraient être installée dans des répertoires dont le nom définis les capacités hardware, tel que /usr/lib/sse2/. Le linker dynamique vérifie ces répertoires avec le hardware de la machine et sélectionne la version la plus appropriée de la librairie donnée. Les répertoire de capacité hardware peuvent être hiérarchisés pour combiner les fonctionnalités CPU. La liste des noms de capacités hardware supportée dépend du CPU. Les noms suivant sont actuellement reconnus :

Alpha ev4, ev5, ev56, ev6, ev67

MIPS loongson2e, loongson2f, octeon, octeon2

PowerPC 4xxmac, altivec, arch_2_05, arch_2_06, booke, cellbe, dfp, efpdouble, efpsingle, fpu, ic_snoop, mmu, notb, pa6t, power4, power5, power5+, power6x, ppc32, ppc601, ppc64, smt, spe, ucache, vsx

SPARC flush, muldiv, stbar, swap, ultra3, v9, v9v, v9v2

s390 dfp, eimm, esan3, etf3enh, g5, highgprs, hpage, ldisp, msa, stfle, z900, z990, z9-109, z10, zarch

x86 (32-bit only) acpi, apic, clflush, cmov, cx8, dts, fxsr, ht, i386, i486, i586, i686, mca, mmx, mtrr, pat, pbe, pge, pn, pse36, sep, ss, sse, sse2, tm

Variables d'environnement

LD_ASSUME_KERNEL glibc >= 2.2.3. Force le linker dynamique à assumer qu'il fonctionne sur un système avec une version ABI kernel différent.

LD_BIND_NOT glibc >=2.2. Ne met pas à jour le Global Offset Table et la Procedure Linkage Table en résolvant un symbole.

LD_BIND_NOW glibc >=2.1.1. Si non vide, le linker dynamique résous tous les symboles au démarrage au lieu de déférer les appels de fonction au moment ou il sont référencé la première fois.

LD_LIBRARY_PATH Liste de répertoire dans lequel rechercher les librairies ELF au moment de l'exécution. Ignoré pour les programmes set-user-ID et set-group-ID

LD_PRELOAD Une liste de librairies partagées ELF additionnels à charger avant toutes les autres.

LD_TRACE_LOADED_OBJECTS ELF uniquement. non vide, force le programme à lister ses dépendance comme lancé par ldd, au lieu de se lancer normalement

LD_AOUT_LIBRARY_PATH (libc5) Version de LD_LIBRARY_PATH pour les binaires a.out

LD_AOUT_PRELOAD (libc5) Version de LD_PRELOAD pour les binaires a.out

LD_AUDIT glibc >=2.4. Liste d'objets partagé à charger avec tous les autres dans un espace de nom de linker séparé.

LD_BIND_NOT glibc >=2.1.95. ne met pas à jours GOT et PLT après avoir résolus un symbole

LD_DEBUG glibc >=2.1. Mode verbeux du linker dynamique. (all ou help).

LD_DEBUG_OUTPUT glibc >=2.1. Fichier dans lequel écrit la sortie LD_DEBUG (défaut : l'erreur standard)

LD_DYNAMIC_WEAK glibc >=2.1.91. Autorise à écraser les symboles faibles (revenir à l'ancien comportement glibc)

LD_HWCAP_MASK glibc >=2.1. Masque pour les capacités hardware.

LD_KEEPDIR (libc5). N'ignore pas le répertoire dans les noms des librairies a.out à charger.

LD_NOWARN (libc5).Supprime les alertes sur les librairie a.out avec des numéros de version mineur incompatible.

LD_ORIGIN_PATH glibc >=2.1. Chemin œu le binaire est trouvé

LD_POINTER_GUARD glibc >=2.4. À 0, désactive le pointer guarding, sinon l'active.

LD_PROFILE glibc >=2.1. Le nom d'un simple objet partagé à profiler.

LD_PROFILE_OUTPUT glibc >=2.1. Répertoire où la sortie LD_PROFILE doit être écrite.

LD_SHOW_AUXV glibc >=2.1. Affiche un tableau auxiliaire passé par le kernel.

LD_USE_LOAD_BIAS Non définis, les exécutables et objets partagés pré-liés honorent les adresses de base de leur librairies et les PIE et autres objets partagé ne les honore pas. Si définis, tous les honorent. À 0 aucun de les honorent.

LD_VERBOSE glibc >=2.1. Si non vide, affiche les informations de versionning de symbole sur le programme si LD_TRACE_LOADED_OBJECTS a été mis.

LD_WARN ELF, glibc >=2.1.3. non vide, alerte sur les symboles non-résolus

LDD_ARGV0 libc5. argv[0] est utilisé par ldd quand aucun n'est présent.

Fichiers

/lib/ld.so linker/loader dynamique a.out

/lib/ld-linux.so.{1,2} linker/loader dynamique ELF

/etc/ld.so.cache Fichier contenant une liste compilée de répertoires dans lesquels rechercher les librairies et une liste ordonnées de librairies candidates.

/etc/ld.so.preload Fichier contenant une liste de librairies partagées ELF à charger avant le programme.

lib*.so* Librairies partagées.