NAME

prelink − prelink ELF shared libraries and binaries to speed up startup time

SYNOPSIS

prelink [OPTION...] [FILES]

DESCRIPTION

prelink is a program that modifies ELF shared libraries and ELF dynamically linked binaries in such a way that the time needed for the dynamic linker to perform relocations at startup significantly decreases. Due to fewer relocations, the run-time memory consumption decreases as well (especially the number of unshareable pages). The prelinking information is only used at startup time if none of the dependent libraries have changed since prelinking; otherwise programs are relocated normally.

prelink first collects ELF binaries to be prelinked and all the ELF shared libraries they depend on. Then it assigns a unique virtual address space slot to each library and relinks the shared library to that base address. When the dynamic linker attempts to load such a library, unless that virtual address space slot is already occupied, it maps the library into the given slot. After this is done, **prelink**, with the help of dynamic linker, resolves all relocations in the binary or library against its dependent libraries and stores the relocations into the ELF object. It also stores a list of all dependent libraries together with their checksums into the binary or library. For binaries, it also computes a list of *conflicts* (relocations that resolve differently in the binary's symbol search scope than in the smaller search scope in which the dependent library was resolved) and stores it into a special ELF section.

At runtime, the dynamic linker first checks whether all dependent libraries were successfully mapped into their designated address space slots, and whether they have not changed since the prelinking was done. If all checks are successful, the dynamic linker just replays the list of conflicts (which is usually significantly shorter than total number of relocations) instead of relocating each library.

OPTIONS

−v −−verbose

Verbose mode. Print the virtual address slots assigned to libraries and what binary or library is currently being prelinked.

−n −−dry−run

Don't actually prelink anything; just collect the binaries/libraries, assign them addresses, and with **−v** print what would be prelinked.

−a −−all

Prelink all binaries and dependent libraries found in directory hierarchies specified in */etc/prelink.conf*. Normally, only binaries specified on the command line and their dependent libraries are prelinked.

−m −−conserve−memory

When assigning addresses to libraries, allow overlap of address space slots provided that the two libraries are not present together in any of the binaries or libraries. This results in a smaller virtual address space range used for libraries. On the other hand, if **prelink** sees a binary during incremental prelinking which puts together two libraries which were not present together in any other binary and were given the same virtual address space slots, then the binary cannot be prelinked. Without this option, each library is assigned a unique virtual address space slot.

−R −−random

When assigning addresses to libraries, start with a random address within the architecture-dependent virtual address space range. This can make some buffer overflow attacks slightly harder to exploit, because libraries are not present on the same addresses across different machines. Normally, assigning virtual addresses starts at the bottom of the architecture-dependent range.

−r −−reloc−only=ADDRESS

Instead of prelinking, just relink given shared libraries to the specified base address.

−N −−no−update−cache
Don't save the cache file after prelinking. Normally, the list of libraries (and with **−m** binaries also) is stored into the */etc/prelink.cache* file together with their given address space slots and dependencies, so the cache can be used during incremental prelinking (prelinking without **−a** option).

−c −−config−file=CONFIG
Specify an alternate config file instead of default */etc/prelink.conf* .

−C −−cache−file=CACHE
Specify an alternate cache file instead of default */etc/prelink.cache*.

−f −−force
Force re-prelinking even for already prelinked objects whose dependencies are unchanged. This option causes new virtual address space slots to be assigned to all libraries. Normally, only binaries or libraries which are either not prelinked yet, or whose dependencies have changed, are prelinked.

−q −−quick
Run prelink in quick mode. This mode checks just mtime and ctime timestamps of libraries and binaries stored in the cache file. If they are unchanged from the last prelink run, it is assumed that the library in question did not change, without parsing or verifying its ELF headers.

−p −−print−cache
Print the contents of the cache file (normally */etc/prelink.cache*) and exit.

−−dynamic−linker=LDSO
Specify an alternate dynamic linker instead of the default.

−−ld−library−path=PATH
Specify a special *LD_LIBRARY_PATH* to be used when **prelink** queries the dynamic linker about symbol resolution details.

−−libs−only
Only prelink ELF shared libraries, don't prelink any binaries.

−h −−dereference
When processing command line directory arguments, follow symbolic links when walking directory hierarchies.

−l −−one−file−system
When processing command line directory arguments, limit directory tree walk to a single file system.

−−root=PATHNAME
In a cross-compilation environment, this specifies the path to the root of the target's file system. All other pathnames processed by **prelink** are relative to this root, including the name of the configuration file as well as its contents and pathnames supplied on the command line.

−u −−undo
Revert binaries and libraries to their original content before they were prelinked. Without the *−a* option, this causes only the binaries and libraries specified on the command line to be reverted to their original state (and e.g. not their dependencies). If used together with the *−a* option, all binaries and libraries from command line, all their dependencies, all binaries found in directories specified on command line and in the config file, and all their dependencies are undone.

−y −−verify
Verifies a prelinked binary or library. This option can be used only on a single binary or library. It first applies an *−−undo* operation on the file, then prelinks just that file again and compares this with the original file. If both are identical, it prints the file after *−−undo* operation on standard output and exits with zero status. Otherwise it exits with error status. Thus if *−−verify* operation returns zero exit status and its standard output is equal to the content of the binary or library before

prelinking, you can be sure that nobody modified the binaries or libraries after prelinking. Similarly with message digests and checksums (unless you trigger the improbable case of modified file and original file having the same digest or checksum).

−−md5 This is similar to *−−verify* option, except instead of outputting the content of the binary or library before prelinking to standard output, MD5 digest is printed. See **md5sum**(1).

−−sha This is similar to *−−verify* option, except instead of outputting the content of the binary or library before prelinking to standard output, SHA1 digest is printed. See **sha1sum**(1).

−−exec−shield −−no−exec−shield
On IA-32, if the kernel supports Exec-Shield, prelink attempts to lay libraries out similarly to how the kernel places them (i.e. if possible below the binary, most widely used into the ASCII armor zone). These switches allow overriding prelink detection of whether Exec-Shield is supported or not.

−b −−black−list=PATH
This option allows blacklisting certain paths, libraries or binaries. Prelink will not touch them during prelinking.

−o −−undo−output=FILE
When performing an **−−undo** operation, don't overwrite the prelinked binary or library with its original content (before it was prelinked), but save that into the specified file.

−V −−version
Print version and exit.

−? −−help
Print short help and exit.

ARGUMENTS

Command-line arguments should be either directory hierarchies (in which case *−l* and *−h* options apply), or particular ELF binaries or shared libraries. Specifying a shared library explicitly on the command line causes it to be prelinked even if no binary is linked against it. Otherwise, binaries are collected together and only the libraries they depend on are prelinked with them.

EXAMPLES

/usr/sbin/prelink -avmR

prelinks all binaries found in directories specified in */etc/prelink.conf* and all their dependent libraries, assigning libraries unique virtual address space slots only if they ever appear together, and starts assigning libraries at a random address.

/usr/sbin/prelink -vm ˜/bin/progx

prelinks ˜/bin/progx program and all its dependent libraries (unless they were prelinked already e.g. during *prelink −a* invocation).

/usr/sbin/prelink -au

reverts all binaries and libraries to their original content.

/usr/sbin/prelink -y /bin/prelinked_prog > /tmp/original_prog; echo $? verifies whether /bin/prelinked_prog is unchanged.

FILES

/etc/prelink.cache Binary file containing a list of prelinked libraries and/or binaries together with their assigned virtual address space slots and dependencies. You can run */usr/sbin/prelink -p* to see what is stored in there.

/etc/prelink.conf Configuration file containing a list of directory hierarchies that contain ELF shared libraries or binaries which should be prelinked. This configuration file is used in **−a** mode to find binaries which should be prelinked and also, no matter whether **−a** is given or not, to limit which dependent shared libraries should be prelinked. If **prelink** finds a dependent library of some binary or other library which is not present in any of the directories specified either in **/etc/prelink.conf** or on the command line, then it cannot be prelinked. Each line of the config file should be

either a comment starting with **#**, or a directory name, or a blacklist specification. Directory names can be prefixed by the **−l** switch, meaning the tree walk of the given directory is only limited to one file system; or the **−h** switch, meaning the tree walk of the given directory follows symbolic links. A blacklist specification should be prefixed by **−b** and optionally also **−l** or **−h** if needed. A blacklist entry can be either an absolute directory name (in that case all files in that directory hierarchy are ignored by the prelinker); an absolute filename (then that particular library or binary is skipped); or a glob pattern without a **/** character in it (then all files matching that glob in any directory are ignored).

SEE ALSO

ldd(1), **ld.so**(8).

BUGS

prelink Some architectures, including HPPA, are not yet supported.

AUTHORS

Jakub Jelinek <jakub@redhat.com>.