XZ data compression in Linux

Introduction

XZ is a general purpose data compression format with high compression ratio and relatively fast decompression. The primary compression algorithm (filter) is LZMA2. Additional filters can be used to improve compression ratio even further. E.g. Branch/Call/Jump (BCJ) filters improve compression ratio of executable data.

The XZ decompressor in Linux is called XZ Embedded. It supports the LZMA2 filter and optionally also BCJ filters. CRC32 is supported for integrity checking. The home page of XZ Embedded is at <https://tukaani.org/xz/embedded.html>, where you can find the latest version and also information about using the code outside the Linux kernel.

For userspace, XZ Utils provide a zlib-like compression library and a gzip-like command line tool. XZ Utils can be downloaded from <https://tukaani.org/xz/>.

XZ related components in the kernel

The xz_dec module provides XZ decompressor with single-call (buffer to buffer) and multi-call (stateful) APIs. The usage of the xz_dec module is documented in include/linux/xz.h.

The xz_dec_test module is for testing xz_dec. xz_dec_test is not useful unless you are hacking the XZ decompressor. xz_dec_test allocates a char device major dynamically to which one can write .xz files from userspace. The decompressed output is thrown away. Keep an eye on dmesg to see diagnostics printed by xz_dec_test. See the xz_dec_test source code for the details.

For decompressing the kernel image, initramfs, and initrd, there is a wrapper function in lib/decompress_unxz.c. Its API is the same as in other decompress_*.c files, which is defined in include/linux/decompress/generic.h.

scripts/xz_wrap.sh is a wrapper for the xz command line tool found from XZ Utils. The wrapper sets compression options to values suitable for compressing the kernel image.

For kernel makefiles, two commands are provided for use with $(call if_needed). The kernel image should be compressed with $(call if_needed,xzkern) which will use a BCJ filter and a big LZMA2 dictionary. It will also append a four-byte trailer containing the uncompressed size of the file, which is needed by the boot code. Other things should be compressed with $(call if_needed,xzmisc) which will use no BCJ filter and 1 MiB LZMA2 dictionary.

Notes on compression options

Since the XZ Embedded supports only streams with no integrity check or CRC32, make sure that you don't use some other integrity check type when encoding files that are supposed to be decoded by the kernel. With liblzma, you need to use either LZMA_CHECK_NONE or LZMA_CHECK_CRC32 when encoding. With the xz command line tool, use --check=none or --check=crc32.

Using CRC32 is strongly recommended unless there is some other layer which will verify the integrity of the uncompressed data anyway. Double checking the integrity would probably be waste of CPU cycles. Note that the headers will always have a CRC32 which will be validated by the decoder; you can only change the integrity check type (or disable it) for the actual uncompressed data.

In userspace, LZMA2 is typically used with dictionary sizes of several megabytes. The decoder needs to have the dictionary in RAM, thus big dictionaries cannot be used for files that are intended to be decoded by the kernel. 1 MiB is probably the maximum reasonable dictionary size for in-kernel use (maybe more is OK for initramfs). The presets in XZ Utils may not be optimal when creating files for the kernel, so don't hesitate to use custom settings. Example:

```
xz --check=crc32 --lzma2=dict=512KiB inputfile
```

An exception to above dictionary size limitation is when the decoder is used in single-call mode. Decompressing the kernel itself is an example of this situation. In single-call mode, the memory usage doesn't depend on the dictionary size, and it is perfectly fine to use a big dictionary: for maximum compression, the dictionary should be at least as big as the uncompressed data itself.

Future plans

Creating a limited XZ encoder may be considered if people think it is useful. LZMA2 is slower to compress than e.g. Deflate or LZO even at the fastest settings, so it isn't clear if LZMA2 encoder is wanted into the kernel.

Support for limited random-access reading is planned for the decompression code. I don't know if it could have any use in the kernel, but I know that it would be useful in some embedded projects outside the Linux kernel.

Conformance to the .xz file format specification

There are a couple of corner cases where things have been simplified at expense of detecting errors as early as possible. These should not matter in practice all, since they don't cause security issues. But it is good to know this if testing the code e.g. with the test

files from XZ Utils.

Reporting bugs

Before reporting a bug, please check that it's not fixed already at upstream. See <https://tukaani.org/xz/embedded.html> to get the latest code.

Report bugs to <lasse.collin@tukaani.org> or visit #tukaani on Freenode and talk to Larhzu. I don't actively read LKML or other kernel-related mailing lists, so if there's something I should know, you should email to me personally or use IRC.

Don't bother Igor Pavlov with questions about the XZ implementation in the kernel or about XZ Utils. While these two implementations include essential code that is directly based on Igor Pavlov's code, these implementations aren't maintained nor supported by him.