Early userspace support
==========

Last update: 2004-12-20 tlh

"Early userspace" is a set of libraries and programs that provide various pieces of functionality that are important enough to be available while a Linux kernel is coming up, but that don't need to be run inside the kernel itself.

It consists of several major infrastructure components:

- gen_init_cpio, a program that builds a cpio-format archive containing a root filesystem image. This archive is compressed, and the compressed image is linked into the kernel image.
- initramfs, a chunk of code that unpacks the compressed cpio image midway through the kernel boot process.
- klibc, a userspace C library, currently packaged separately, that is optimized for correctness and small size.

The cpio file format used by initramfs is the "newc" (aka "cpio -H newc") format, and is documented in the file "buffer-format.txt". There are two ways to add an early userspace image: specify an existing cpio archive to be used as the image or have the kernel build process build the image from specifications.

CPIO ARCHIVE method

You can create a cpio archive that contains the early userspace image. Your cpio archive should be specified in CONFIG_INITRAMFS_SOURCE and it will be used directly. Only a single cpio file may be specified in CONFIG_INITRAMFS_SOURCE and directory and file names are not allowed in combination with a cpio archive.

IMAGE BUILDING method

The kernel build process can also build an early userspace image from source parts rather than supplying a cpio archive. This method provides a way to create images with root-owned files even though the image was built by an unprivileged user.

The image is specified as one or more sources in CONFIG_INITRAMFS_SOURCE. Sources can be either directories or files - cpio archives are *not* allowed when building from sources.

A source directory will have it and all of its contents packaged. The specified directory name will be mapped to '/'. When packaging a directory, limited user and group ID translation can be performed. INITRAMFS_ROOT_UID can be set to a user ID that needs to be mapped to user root (0). INITRAMFS_ROOT_GID can be set to a group ID that needs to be mapped to group root (0).

A source file must be directives in the format required by the usr/gen_init_cpio utility (run 'usr/gen_init_cpio --help' to get the file format). The directives in the file will be passed directly to

usr/gen_init_cpio.

When a combination of directories and files are specified then the initramfs image will be an aggregate of all of them. In this way a user can create a 'root-image' directory and install all files into it. Because device-special files cannot be created by a unprivileged user, special files can be listed in a 'root-files' file. Both 'root-image' and 'root-files' can be listed in CONFIG_INITRAMFS_SOURCE and a complete early userspace image can be built by an unprivileged user.

As a technical note, when directories and files are specified, the entire CONFIG_INITRAMFS_SOURCE is passed to scripts/gen_initramfs_list.sh. This means that CONFIG_INITRAMFS_SOURCE can really be interpreted as any legal argument to gen_initramfs_list.sh. If a directory is specified as an argument then the contents are scanned, uid/gid translation is performed, and usr/gen_init_cpio file directives are output. If a directory is specified as an arugemnt to scripts/gen_initramfs_list.sh then the contents of the file are simply copied to the output. All of the output directives from directory scanning and file contents copying are processed by usr/gen_init_cpio.

See also 'scripts/gen_initramfs_list.sh -h'.

Where's this all leading?
==========

The klibc distribution contains some of the necessary software to make early userspace useful. The klibc distribution is currently maintained separately from the kernel, but this may change early in the 2.7 era (it missed the boat for 2.5).

You can obtain somewhat infrequent snapshots of klibc from ftp://ftp.kernel.org/pub/linux/libs/klibc/

For active users, you are better off using the klibc git repository, at http://git.kernel.org/?p=libs/klibc/klibc.git

The standalone klibc distribution currently provides three components, in addition to the klibc library:

- ipconfig, a program that configures network interfaces. It can configure them statically, or use DHCP to obtain information dynamically (aka "IP autoconfiguration").
- nfsmount, a program that can mount an NFS filesystem.
- kinit, the "glue" that uses ipconfig and nfsmount to replace the old support for IP autoconfig, mount a filesystem over NFS, and continue system boot using that filesystem as root.

kinit is built as a single statically linked binary to save space.

Eventually, several more chunks of kernel functionality will hopefully move to early userspace:

- Almost all of init/do_mounts* (the beginning of this is already in place)

- ACPI table parsing
- Insert unwieldy subsystem that doesn't really need to be in kernel
  space here

If kinit doesn't meet your current needs and you've got bytes to burn,
the klibc distribution includes a small Bourne-compatible shell (ash)
and a number of other utilities, so you can replace kinit and build
custom initramfs images that meet your needs exactly.

For questions and help, you can sign up for the early userspace
mailing list at http://www.zytor.com/mailman/listinfo/klibc

How does it work?
==========

The kernel has currently 3 ways to mount the root filesystem:

a) all required device and filesystem drivers compiled into the kernel, no
   initrd. init/main.c:init() will call prepare_namespace() to mount the
   final root filesystem, based on the root= option and optional init= to run
   some other init binary than listed at the end of init/main.c:init().

b) some device and filesystem drivers built as modules and stored in an
   initrd. The initrd must contain a binary '/linuxrc' which is supposed to
   load these driver modules. It is also possible to mount the final root
   filesystem via linuxrc and use the pivot_root syscall. The initrd is
   mounted and executed via prepare_namespace().

c) using initramfs. The call to prepare_namespace() must be skipped.
   This means that a binary must do all the work. Said binary can be stored
   into initramfs either via modifying usr/gen_init_cpio.c or via the new
   initrd format, an cpio archive. It must be called "/init". This binary
   is responsible to do all the things prepare_namespace() would do.

   To maintain backwards compatibility, the /init binary will only run if it
   comes via an initramfs cpio archive. If this is not the case,
   init/main.c:init() will run prepare_namespace() to mount the final root
   and exec one of the predefined init binaries.

Bryan O'Sullivan <bos@serpentine.com>