NAME

initramfs-tools − an introduction to writing scripts for mkinitramfs

DESCRIPTION

initramfs-tools has one main script and two different sets of subscripts which will be used during different phases of execution. Each of these will be discussed separately below with the help of an imaginary tool which performs a frobnication of a lvm partition prior to mounting the root partition.

Kernel Command Line

The root filesystem used by the kernel is specified by the boot loader as always. The traditional **root=/dev/sda1** style device specification is allowed. If a label is used, as in **root=LABEL=rootPart** the initrd will search all available devices for a filesystem with the appropriate label, and mount that device as the root filesystem. **root=UUID=uuidnumber** will mount the partition with that UUID as the root filesystem.

Standard

init= "<path to real init>"
: the binary to hand over execution to on the root fs after the initramfs scripts are done.

initramfs.clear
: clear screen at the beginning

initramfs.runsize
: The size of the */run* tmpfs mount point in bytes (suffixes are supported) or as percentage of your physical RAM. This parameter is used as the value of the size mount option to tmpfs. See **https://www.kernel.org/doc/Documentation/filesystems/tmpfs.txt** for details. The default is 10%.

root= "<path to blockdevice>"
: the device node to mount as the root file system. The recommended usage is to specify the UUID as followed "root=UUID=xxx".

rootfstype
: set the root file system type.

roottimeout
: set timeout in seconds. Determines how long mountroot waits for root to appear. The default is 30 seconds.

rootdelay
: alias for roottimeout.

rootflags
: set the file system mount option string.

loop= "<path to image>"
: path within the original root file system to loop-mount and use as the real root file system.

loopfstype
set the loop file system type, if applicable.

loopflags
set the loop file system mount option string, if applicable.

nfsroot can be either "auto" to try to get the relevant information from DHCP or a string of the form NFSSERVER:NFSPATH or NFSSERVER:NFSPATH:NFSOPTS. Use root=/dev/nfs for NFS to kick to in. NFSOPTS can be looked up in *nfs(5)*.

ip tells how to configure the ip address. Allows one to specify an different NFS server than the DHCP server. See Documentation/filesystems/nfsroot.txt in any recent Linux source for details. Optional parameter for NFS root.

vlan tells to create a VLAN tagged device. Allows one to configure one or multiple VLAN tagged devices using the "vlan=$name.$id:$parent" syntax. E.g. "vlan=eth0.1:eth0" Optional parameter for NFS root.

BOOTIF
is a mac address in pxelinux format with leading "01-" and "-" as separations. pxelinux passes mac address of network card used to PXE boot on with this bootarg.

boot either local or NFS (affects which initramfs scripts are run, see the "Subdirectories" section under boot scripts).

resume
The resume hook tries to autodetect the resume partition and uses the first swap partition as valid guess. It is possible to set the RESUME variable in /etc/initramfs-tools/conf.d/resume. The boot variable noresume overrides it.

resume_offset
Specify the offset from the partition given by "resume=" at which the swap header of the swap file is located.

quiet reduces the amount of text output to the console during boot.

ro mounts the rootfs read-only.

rw mounts the rootfs read-write.

blacklist
disables load of specific modules. Use blacklist=module1,module2,module3 bootparameter.

Debug

panic sets an timeout on panic. panic=<sec> is a documented security feature: it disables the debug shell.

debug generates lots of output. It writes a log to /run/initramfs/initramfs.debug. Instead when invoked with an arbitrary argument output is written to console. Use for example "debug=vc".

break spawns a shell in the initramfs image at the chosen phase (top, modules, premount, mount, mountroot, bottom, init) before actually executing the corresponding scripts (see the "Boot scripts" section) or action. Multiple phases may be specified, delimited by commas. The default, if no phase is specified, is "premount". Beware that if both "panic" and "break" are present, initramfs will not spawn any shells but reboot instead.

netconsole
loads netconsole linux modules with the chosen args.

all_generic_ide
loads generic IDE/ATA chipset support on boot.

SCRIPTS

Valid boot and hook scripts names consist solely of alphabetics, numerics, dashes and underscores. Other scripts are discarded.

Configuration hook scripts

These are used to override the user configuration where necessary, for example to force use of busybox instead of klibc utilities.

Hook scripts

These are used when an initramfs image is created and not included in the image itself. They can however cause files to be included in the image. Hook scripts are executed under errexit. Thus a hook script can abort the mkinitramfs build on possible errors (exitcode != 0).

Boot scripts

These are included in the initramfs image and normally executed during kernel boot in the early user-space before the root partition has been mounted.

CONFIGURATION HOOK SCRIPTS

Configuration hook scripts can be found in /usr/share/initramfs-tools/conf-hooks.d. They are sourced by mkinitramfs after the configuration files in /etc and before running any hook scripts. They can override any of the variables documented in *initramfs.conf*(5), but this should be done only if absolutely necessary. For example, if a package's boot script requires commands not provided by klibc-utils, it should also install a configuration hook that sets **BUSYBOX=y**.

HOOK SCRIPTS

Hooks can be found in two places: /usr/share/initramfs-tools/hooks and /etc/initramfs-tools/hooks. They are executed during generation of the initramfs-image and are responsible for including all the necessary components in the image itself. No guarantees are made as to the order in which the different scripts are executed unless the prereqs are setup in the script. Please notice that PREREQ is only honored inside a single directory. So first the scripts in /usr/share/initramfs-tools are ordered according to their PREREQ values and executed. Then all scripts in /etc/initramfs-tools are ordered according to **their** PREREQ values and executed. This mean that currently there is no possibility to have a local script (/etc/initramfs-tools) get executed before one from the package (/usr/share/initramfs-tools).

If a hook script requires configuration beyond the exported variables listed below, it should read a private configuration file that is separate from the /etc/initramfs-tools directory. It *must not* read initramfs-tools

configuration files directly.

Header

In order to support prereqs, each script should begin with the following lines:

```
#!/bin/sh
PREREQ=""
prereqs()
{
        echo "$PREREQ"
}

case $1 in
prereqs)
        prereqs
        exit 0
        ;;
esac

. /usr/share/initramfs-tools/hook-functions
# Begin real processing below this line
```

For example, if you are writing a new hook script which relies on lvm, the line starting with PREREQ should be changed to PREREQ="lvm" which will ensure that the lvm hook script is run before your custom script.

Help functions

/usr/share/initramfs-tools/hook-functions contains a number of functions which deal with some common tasks in a hook script:

manual_add_modules adds a module (and any modules which it depends on) to the initramfs image.

Example: manual_add_modules isofs

add_modules_from_file reads a file containing a list of modules (one per line) to be added to the initramfs image. The file can contain comments (lines starting with #) and arguments to the modules by writing the arguments on the same line as the name of the module.

Example: add_modules_from_file /tmp/modlist

force_load adds a module (and its dependencies) to the initramfs image and also unconditionally loads the module during boot. Also supports passing arguments to the module by listing them after the module name.

Example: force_load cdrom debug=1

copy_modules_dir copies an entire module directory from /lib/modules/KERNELVERSION/ into the initramfs image.

Example: copy_modules_dir kernel/drivers/ata

Including binaries

If you need to copy binaries to the initramfs module, a command like this should be used:

copy_exec /sbin/mdadm /sbin

mkinitramfs will automatically detect which libraries the executable depends on and copy them to the initramfs. This means that most executables, unless compiled with klibc, will automatically include glibc in the image which will increase its size by several hundred kilobytes.

Including a system firmware preimage (early initramfs)

If you need to prepend data to the initramfs image, you need to prepare it in a file, and call the *prepend_earlyinitramfs* function. The file can be disposed of as soon as the function returns.

Example:

```
TEMP_FILE=$(mktemp ...)
 ...
prepend_earlyinitramfs ${TEMP_FILE}
rm -f ${TEMP_FILE}
```

Exported variables

mkinitramfs sets several variables for the hook scripts environment.

MODULESDIR
: corresponds to the linux modules dir.

version
: is the $(uname −r) linux version against mkinitramfs is run.

CONFDIR
: is the path of the used initramfs-tools configurations.

DESTDIR
: is the root path of the newly build initramfs.

DPKG_ARCH
: allows arch specific hook additions.

verbose
: corresponds to the verbosity of the update-initramfs run.

BUSYBOX, MODULES
: are as described in *initramfs.conf*(5).

BUSYBOXDIR
: is the directory where busybox utilities should be installed from, or empty if busybox is not being used.

BOOT SCRIPTS

Similarly to hook scripts, boot scripts can be found in two places /usr/share/initramfs-tools/scripts/ and /etc/initramfs-tools/scripts/. There are a number of subdirectories to these two directories which control the boot stage at which the scripts are executed.

Header

Like for hook scripts, there are no guarantees as to the order in which the different scripts in one subdirectory (see "Subdirectories" below) are executed. In order to define a certain order, a similar header as for hook scripts should be used:

```
#!/bin/sh
PREREQ=""
prereqs()
```

```
{
        echo "$PREREQ"
}

case $1 in
prereqs)
        prereqs
        exit 0
        ;;
esac
```

Where PREREQ is modified to list other scripts in the same subdirectory if necessary.

Help functions

A number of functions (mostly dealing with output) are provided to boot scripts in */scripts/functions* :

log_success_msg Logs a success message

Example: log_success_msg "Frobnication successful"

log_failure_msg Logs a failure message

Example: log_failure_msg "Frobnication component froobz missing"

log_warning_msg Logs a warning message

Example: log_warning_msg "Only partial frobnication possible"

log_begin_msg Logs a message that some processing step has begun

log_end_msg Logs a message that some processing step is finished

Example:

```
log_begin_msg "Frobnication begun"
# Do something
log_end_msg
```

panic Logs an error message and executes a shell in the initramfs image to allow the user to investigate the situation.

Example: panic "Frobnication failed"

add_mountroot_fail_hook NN-name Registers the script as able to provide possible further information, in the event that the root device cannot be found. See the example script in the initramfs-tools examples directory for more information.

Example: add_mountroot_fail_hook NN-name

Subdirectories

Both /usr/share/initramfs-tools/scripts and /etc/initramfs-tools/scripts contains the following subdirectories.

init-top the scripts in this directory are the first scripts to be executed after sysfs and procfs have been mounted. It also runs the udev hook for populating the /dev tree (udev will keep running until init-bottom).

init-premount happens after modules specified by hooks and /etc/initramfs-tools/modules have been loaded.

local-top OR nfs-top After these scripts have been executed, the root device node is expected to be present (local) or the network interface is expected to be usable (NFS).

local-block These scripts are called with the name of a local block device. After these scripts have been executed, that device node should be present. If the local-top or local-block scripts fail to create the wanted device node, the local-block scripts will be called periodically to try again.

local-premount OR nfs-premount are run after the sanity of the root device has been verified (local) or the network interface has been brought up (NFS), but before the actual root fs has been mounted.

local-bottom OR nfs-bottom are run after the rootfs has been mounted (local) or the NFS root share has been mounted.

init-bottom are the last scripts to be executed before procfs and sysfs are moved to the real rootfs and execution is turned over to the init binary which should now be found in the mounted rootfs. udev is stopped.

Boot parameters

/conf/param.conf allows boot scripts to change exported variables that are listed on top of init. Write the new values to it. It will be sourced after an boot script run if it exists.

EXAMPLES

Hook script

An example hook script would look something like this (and would usually be placed in /etc/initramfs-tools/hooks/frobnicate):

```
#!/bin/sh
# Example frobnication hook script

PREREQ="lvm"
prereqs()
{
        echo "$PREREQ"
}

case $1 in
prereqs)
        prereqs
        exit 0
        ;;
esac

. /usr/share/initramfs-tools/hook-functions
# Begin real processing below this line

if [ ! −x "/sbin/frobnicate" ]; then
        exit 0
```

```
fi

force_load frobnicator interval=10
copy_exec /sbin/frobnicate /sbin
exit 0
```

Boot script

An example boot script would look something like this (and would usually be placed in /etc/initramfs-tools/scripts/local-top/frobnicate):

```
#!/bin/sh
# Example frobnication boot script

PREREQ="lvm"
prereqs()
{
        echo "$PREREQ"
}

case $1 in
prereqs)
        prereqs
        exit 0
        ;;
esac

. /scripts/functions
# Begin real processing below this line
if [ ! -x "/sbin/frobnicate" ]; then
        panic "Frobnication executable not found"
fi

if [ ! -e "/dev/mapper/frobb" ]; then
        panic "Frobnication device not found"
fi

log_begin_msg "Starting frobnication"
/sbin/frobnicate "/dev/mapper/frobb" || panic "Frobnication failed"
log_end_msg

exit 0
```

Exported variables

init sets several variables for the boot scripts environment.

ROOT corresponds to the root boot option. Advanced boot scripts like cryptsetup or live-initramfs need to play tricks. Otherwise keep it alone.

ROOTDELAY, ROOTFLAGS, ROOTFSTYPE, IP
corresponds to the rootdelay, rootflags, rootfstype or ip boot option. Use of ROOTDELAY is deprecated; you should implement a *local-block* boot script rather than delaying or polling.

DPKG_ARCH
allows arch specific boot actions.

blacklist, panic, quiet, resume, noresume, resume_offset
set according relevant boot option.

break Useful for manual intervention during setup and coding an boot script.

REASON
Argument passed to the *panic* helper function. Use to find out why you landed in the initramfs shell.

init passes the path to init(8) usually /sbin/init.

readonly
is the default for mounting the root corresponds to the ro bootarg. Overridden by rw bootarg.

rootmnt
is the path where root gets mounted usually /root.

debug indicates that a debug log is captured for further investigation.

UPDATING THE INITRAMFS FROM ANOTHER PACKAGE

Package maintainer scripts should not run **update-initramfs** directly. A package that installs hooks for initramfs-tools should include a triggers file containing:

activate−noawait update−initramfs

Kernel packages must call the kernel hooks as documented in the Debian Kernel Handbook.

A package that requires an initramfs to function, but is not a kernel package, should include a triggers file containing:

activate−await update−initramfs

KERNEL HOOKS

initramfs-tools includes hook scripts that are called by kernel packages on installation and removal, so that an initramfs is automatically created, updated or deleted as necessary. The hook scripts do nothing if the environment variable **INITRD** is set to **No**. This will be the case for kernel packages built with **make deb-pkg** and with **CONFIG_BLK_DEV_INITRD** not set in the kernel config, or built with **make-kpkg** and not using the **--initrd** option.

DEBUG

It is easy to check the generated initramfs for its content. One may need to double-check if it contains the relevant binaries, libs or modules:

lsinitramfs /boot/initrd.img−3.16−3−amd64

FILES

/run/initramfs/fsck.log
Log of fsck commands run within the initramfs, with their output.

/run/initramfs/fsck-root
Exists only if fsck ran successfully for the root filesystem.

/run/initramfs/fsck-usr
Exists only if fsck ran successfully for the */usr* filesystem.

AUTHOR

The initramfs-tools are written by Maximilian Attems <maks@debian.org>, Jeff Bailey <jbailey@raspberryginger.com> and numerous others.

This manual was written by David Härdeman <david@hardeman.nu>, updated by Maximilian Attems <maks@debian.org>.

SEE ALSO

initramfs.conf (5), *mkinitramfs*(8), *update-initramfs*(8), *lsinitramfs*(8).