

```
#!/bin/sh
```

```
# Default PATH differs between shells, and is not automatically exported
# by klibc dash. Make it consistent.
# Furthermore, this PATH ends up being used by the init, set it to the
# Standard PATH, without /snap/bin as documented in
```

```
# https://wiki.ubuntu.com/PATH
```

```
# This also matches /etc/environment, but without games path
```

```
export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

```
[ -d /dev ] || mkdir -m 0755 /dev
```

```
[ -d /root ] || mkdir -m 0700 /root
```

```
[ -d /sys ] || mkdir /sys
```

```
[ -d /proc ] || mkdir /proc
```

```
[ -d /tmp ] || mkdir /tmp
```

```
mkdir -p /var/lock
```

```
mount -t sysfs -o nodev,noexec,nosuid sysfs /sys
```

```
mount -t proc -o nodev,noexec,nosuid proc /proc
```

```
# shellcheck disable=SC2013
```

```
for x in $(cat /proc/cmdline); do
```

```
    case $x in
```

```
        initramfs.clear)
```

```
            clear
```

```
            ;;
```

```
        quiet)
```

```
            quiet=y
```

```
            ;;
```

```
    esac
```

```
done
```

```
if [ "$quiet" != "y" ]; then
```

```
    quiet=n
```

```
    echo "Loading, please wait..."
```

```
fi
```

```
export quiet
```

```
# Note that this only becomes /dev on the real filesystem if udev's scripts
```

```
# are used; which they will be, but it's worth pointing out
```

```
test -x /usr/sbin/v86d && dev_exec="exec" || dev_exec="noexec"
```

```
mount -t devtmpfs -o $dev_exec,nosuid,mode=0755 udev /dev
```

```
mkdir /dev/pts
```

```
mount -t devpts -o noexec,nosuid,gid=5,mode=0620 devpts /dev/pts || true
```

```
# Export the dpkg architecture
```

```
export DPKG_ARCH=
```

```
. /conf/arch.conf
```

```
# Set modprobe env
```

```
export MODPROBE_OPTIONS="-qb"
```

```
# Export relevant variables
```

```
export ROOT=
```

```
export ROOTDELAY=
```

```
export ROOTFLAGS=
```

```
export ROOTFSTYPE=
```

```
export IP=
```

```
export IP6=
```

```
export VLAN=
```

```
export DEVICE=
```

```
export BOOT=
```

```
export BOOTIF=
```

```
export UBIMTD=
```

```
export NETWORK_SKIP_ENSLAVED=
```

```
export break=
```

```
export init=/sbin/init
```

```

export readonly=y
export rootmnt=/root
export debug=
export panic=
export blacklist=
export resume=
export resume_offset=
export noresume=
export drop_caps=
export fastboot=n
export forcefsck=n
export fsckfix=
export recovery=

```

```

export KLOOP=
export KROOT=
export KLVM=
export HOSTFSTYPE=
export KLOOPFSTYPE=
export SQUASHFS=
export UPPERDIR=
export LOWERDIR=
export WORKDIR=
export QEMUNBD=

```

1.

```

# mdadm needs hostname to be set. This has to be done before the udev rules are
called!

```

```

if [ -f "/etc/hostname" ]; then
    /bin/hostname -F /etc/hostname >/dev/null 2>&1
fi

```

```

# Bring in the main config
. /conf/initramfs.conf
for conf in conf/conf.d/*; do
    [ -f "${conf}" ] && . "${conf}"
done
. /scripts/functions

```

```

# Parse command line options
# shellcheck disable=SC2013
for x in $(cat /proc/cmdline); do
    case $x in
        init=*)
            init=${x#init=}
            ;;
        root=*)
            ROOT=${x#root=}
            if [ -z "${BROOT}" ] && [ "$ROOT" = "/dev/nfs" ]; then
                BROOT=nfs
            fi
            ;;
        rootflags=*)
            ROOTFLAGS="-o ${x#rootflags=}"
            ;;
        rootfstype=*)
            ROOTFSTYPE="${x#rootfstype=}"
            ;;
        rootdelay=*)
            ROOTDELAY="${x#rootdelay=}"
            case ${ROOTDELAY} in
                *[:digit:].*)
                    ROOTDELAY=
                ;;
            esac
            ;;
        roottimeout=*)

```

```
ROOTDELAY="${x#roottimeout=}"
case ${ROOTDELAY} in
*[[[:digit:]]]*)
    ROOTDELAY=
    ;;
esac
;;
loop=*)
    # shellcheck disable=SC2034
    LOOP="${x#loop=}"
    ;;
loopflags=*)
    # shellcheck disable=SC2034
    LOOPFLAGS="-o ${x#loopflags=}"
    ;;
loopfstype=*)
    # shellcheck disable=SC2034
    LOOPFSTYPE="${x#loopfstype=}"
    ;;
nfsroot=*)
    # shellcheck disable=SC2034
    NFSROOT="${x#nfsroot=}"
    ;;
initramfs.runsize=*)
    RUNSIZE="${x#initramfs.runsize=}"
    ;;
ip=*)
    IP="${x#ip=}"
    ;;
ip6=*)
    IP6="${x#ip6=}"
    ;;
vlan=*)
    VLAN="${x#vlan=}"
    ;;
boot=*)
    BOOT=${x#boot=}
    ;;
ubi.mtd=*)
    UBIMTD=${x#ubi.mtd=}
    ;;
resume=*)
    RESUME="${x#resume=}"
    case $RESUME in
    UUID=*)
        RESUME="/dev/disk/by-uuid/${RESUME#UUID=}"
        ;;
    esac
    ;;
resume_offset=*)
    resume_offset="${x#resume_offset=}"
    ;;
noresume)
    noresume=y
    ;;
drop_capabilities=*)
    drop_caps="-d ${x#drop_capabilities=}"
    ;;
panic=*)
    panic="${x#panic=}"
    case ${panic} in
    -1) ;;
    *[[[:digit:]]]*)
        panic=
        ;;
    esac
    ;;
```

```
ro)
    readonly=y
;;
rw)
    readonly=n
;;
debug)
    debug=y
    quiet=n
    if [ -n "${netconsole}" ]; then
        log_output=/dev/kmsg
    else
        log_output=/run/initramfs/initramfs.debug
    fi
    set -x
;;
debug=*)
    debug=y
    quiet=n
    set -x
;;
break=*)
    break=${x#break=}
;;
break)
    break=premount
;;
blacklist=*)
    blacklist=${x#blacklist=}
;;
netconsole=*)
    netconsole=${x#netconsole=}
    [ "x$debug" = "xy" ] && log_output=/dev/kmsg
;;
BOOTIF=*)
    BOOTIF=${x#BOOTIF=}
;;
hwaddr=*)
    BOOTIF=${x#hwaddr=}
;;
fastboot|fsck.mode=skip)
    fastboot=y
;;
forcefsck|fsck.mode=force)
    forcefsck=y
;;
fsckfix|fsck.repair=yes)
    fsckfix=y
;;
fsck.repair=no)
    fsckfix=n
;;
kloop=*)
    KLOOP=${x#kloop=}
;;
kroot=*)
    KROOT=${x#kroot=}
;;
klvm=*)
    KLVM=${x#klvm=}
;;
hostfstype=*)
    HOSTFSTYPE=${x#hostfstype=}
;;
kloopfstype=*)
    KLOOPFSTYPE=${x#kloopfstype=}
```

```

        ;;
        squashfs=*)
            SQUASHFS=${x#squashfs=}
        ;;
        upperdir=*)
            UPPERDIR=${x#upperdir=}
        ;;
        lowerdir=*)
            LOWERDIR=${x#lowerdir=}
        ;;
        workdir=*)
            WORKDIR=${x#workdir=}
        ;;
        qemunbd=*)
            QEMUNBD=${x#qemunbd=}
        ;;
    esac
done

# Default to B00T=local if no boot script defined.
if [ -z "${B00T}" ]; then
    B00T=local
fi

if [ -n "${noresume}" ] || [ "$RESUME" = none ]; then
    noresume=y
else
    resume=${RESUME:-}
fi

mount -t tmpfs -o "nodev,noexec,size=50%,mode=0755" tmpfs /run
mkdir -m 0700 /run/initramfs

if [ -n "$log_output" ]; then
    exec >$log_output 2>&1
    unset log_output
fi

maybe_break top

# export B00T variable value for compcache,
# so we know if we run from casper
export B00T

# Don't do log messages here to avoid confusing graphical boots
run_scripts /scripts/init-top

maybe_break modules
[ "$quiet" != "y" ] && log_begin_msg "Loading essential drivers"
[ -n "${netconsole}" ] && modprobe netconsole netconsole="${netconsole}"
load_modules
[ "$quiet" != "y" ] && log_end_msg

maybe_break premount
[ "$quiet" != "y" ] && log_begin_msg "Running /scripts/init-premount"
run_scripts /scripts/init-premount
[ "$quiet" != "y" ] && log_end_msg

maybe_break mount
log_begin_msg "Mounting root file system"
# Always load local and nfs (since these might be needed for /etc or
# /usr, irrespective of the boot script used to mount the rootfs).
. /scripts/local
. /scripts/nfs
. /scripts/${B00T}
parse_numeric "${ROOT}"

```

2.

UBUNTU 20.04 LTS
/usr/share/
initramfs-tools
/init
删除这一句里的nosuid

3.

```
maybe_break mountroot
mount_top
mount_premount
mountroot
log_end_msg

if read_fstab_entry /usr; then
    log_begin_msg "Mounting /usr file system"
    mountfs /usr
    log_end_msg
fi

# Mount cleanup
mount_bottom
nfs_bottom
local_bottom

maybe_break bottom
[ "$quiet" != "y" ] && log_begin_msg "Running /scripts/init-bottom"
# We expect udev's init-bottom script to move /dev to ${rootmnt}/dev
run_scripts /scripts/init-bottom
[ "$quiet" != "y" ] && log_end_msg

# Move /run to the root
mount -n -o move /run ${rootmnt}/run

validate_init() {
    run-init -n "${rootmnt}" "${1}"
}

# Check init is really there
if ! validate_init "$init"; then
    echo "Target filesystem doesn't have requested ${init}."
    init=
    for inittest in /sbin/init /etc/init /bin/init /bin/sh; do
        if validate_init "${inittest}"; then
            init="$inittest"
            break
        fi
    done
fi

# No init on rootmount
if ! validate_init "${init}" ; then
    panic "No init found. Try passing init= bootarg."
fi

maybe_break init

# don't leak too much of env - some init(8) don't clear it
# (keep init, rootmnt, drop_caps)
unset debug
unset MODPROBE_OPTIONS
unset DPKG_ARCH
unset ROOTFLAGS
unset ROOTFSTYPE
unset ROOTDELAY
unset ROOT
unset IP
unset IP6
unset VLAN
unset BOOT
unset BOOTIF
unset DEVICE
unset UBIMTD
unset blacklist
```

```
unset break
unset noresume
unset panic
unset quiet
unset readonly
unset resume
unset resume_offset
unset noresume
unset fastboot
unset forcefsck
unset fsckfix
```

```
unset KLOOP=
unset KROOT=
unset KLVM=
unset HOSTFSTYPE=
unset KLOPPFSTYPE=
unset SQUASHFS=
unset UPPERDIR=
unset LOWERDIR=
unset WORKDIR=
unset QEMUNBD=
```

4.

修改或添加合计四处

```
# Move virtual filesystems over to the real filesystem
mount -n -o move /sys ${rootmnt}/sys
mount -n -o move /proc ${rootmnt}/proc

# Chain to real filesystem
# shellcheck disable=SC2086,SC2094
exec run-init ${drop_caps} "${rootmnt}" "${init}" "$@" <"${rootmnt}/dev/console"
>"${rootmnt}/dev/console" 2>&1
echo "Something went badly wrong in the initramfs."
panic "Please file a bug on initramfs-tools."
```