```
# Local filesystem mounting                     -*- shell-script -*-

local_top()
{
	if [ "${local_top_used}" != "yes" ]; then
		[ "$quiet" != "y" ] && log_begin_msg "Running /scripts/local-top"
		run_scripts /scripts/local-top
		[ "$quiet" != "y" ] && log_end_msg
	fi
	local_top_used=yes

	# Start time for measuring elapsed time in local_device_setup
	if [ -z "${local_top_time}" ]; then
		local_top_time="$(cat /proc/uptime)"
		local_top_time="${local_top_time%%[. ]*}"
		local_top_time=$((local_top_time + 1)) # round up
		export local_top_time
	fi
}

local_block()
{
	[ "$quiet" != "y" ] && log_begin_msg "Running /scripts/local-block"
	run_scripts /scripts/local-block "$@"
	[ "$quiet" != "y" ] && log_end_msg
}

local_premount()
{
	if [ "${local_premount_used}" != "yes" ]; then
		[ "$quiet" != "y" ] && log_begin_msg "Running /scripts/local-
premount"
		run_scripts /scripts/local-premount
		[ "$quiet" != "y" ] && log_end_msg
	fi
	local_premount_used=yes
}

local_bottom()
{
	if [ "${local_premount_used}" = "yes" ] || [ "${local_top_used}" =
"yes" ]; then
		[ "$quiet" != "y" ] && log_begin_msg "Running /scripts/local-
bottom"
		run_scripts /scripts/local-bottom
		[ "$quiet" != "y" ] && log_end_msg
	fi
	local_premount_used=no
	local_top_used=no
	unset local_top_time
}

# $1=device ID to mount
# $2=optionname (for root and etc)
# $3=panic if device is missing (true or false, default: true)
# Sets $DEV to the resolved device node
local_device_setup()
{
	local dev_id="$1"
	local name="$2"
	local may_panic="${3:-true}"
	local real_dev
	local time_elapsed
	local count

	# If wait-for-root understands this prefix, then use it to wait for
	# the device rather than settling the whole of udev.

	# Timeout is max(30, rootdelay) seconds (approximately)
	local slumber=30
	case $DPKG_ARCH in
		powerpc|ppc64|ppc64el)
			slumber=180
			;;
		*)
			slumber=30
			;;
	esac
	if [ ${ROOTDELAY:-0} -gt $slumber ]; then
		slumber=$ROOTDELAY
	fi

	case "$dev_id" in
	UUID=*|LABEL=*|/dev/*)
		FSTYPE=$( wait-for-root "$dev_id" $slumber )
		;;
	*)
		wait_for_udev 10
		;;
	esac

	# Load ubi with the correct MTD partition and return since fstype
	# doesn't work with a char device like ubi.
	if [ -n "$UBIMTD" ]; then
		modprobe ubi mtd=$UBIMTD
		DEV="${dev_id}"
		return
	fi

	# Don't wait for a device that doesn't have a corresponding
	# device in /dev and isn't resolvable by blkid (e.g. mtd0)
	if [ "${dev_id#/dev}" = "${dev_id}" ] &&
	   [ "${dev_id#*=}" = "${dev_id}" ]; then
		DEV="${dev_id}"
		return
	fi

	# If the root device hasn't shown up yet, give it a little while
	# to allow for asynchronous device discovery (e.g. USB).  We
	# also need to keep invoking the local-block scripts in case
	# there are devices stacked on top of those.
	#
	# in ubuntu, we should never actually enter this loop as wait-for-root
	# above should waited until the device appeared.
	if ! real_dev=$(resolve_device "${dev_id}") ||
	   ! get_fstype "${real_dev}" >/dev/null; then
		log_begin_msg "Waiting for ${name}"

		while true; do
			sleep 1
			time_elapsed="$(cat /proc/uptime)"
			time_elapsed="${time_elapsed%%[. ]*}"
			time_elapsed=$((time_elapsed - local_top_time))

			local_block "${dev_id}"

			# If mdadm's local-block script counts the
			# number of times it is run, make sure to
			# run it the expected number of times.
			while true; do
				if [ -f /run/count.mdadm.initrd ]; then
					count="$(cat /run/count.mdadm.initrd)"
```

```
                                elif [ -n "${count}" ]; then
                                        # mdadm script deleted it; put it back
                                        count=$((count + 1))
                                        echo "${count}" >/run/count.mdadm.initrd
                                else
                                        break
                                fi
                                if [ ${count} -ge ${time_elapsed} ]; then
                                        break;
                                fi
                                /scripts/local-block/mdadm "${dev_id}"
                        done

                        if real_dev=$(resolve_device "${dev_id}") &&
                           get_fstype "${real_dev}" >/dev/null; then
                                wait_for_udev 10
                                log_end_msg 0
                                break
                        fi
                        if [ ${time_elapsed} -ge ${slumber} ]; then
                                log_end_msg 1 || true
                                break
                        fi
                done
        fi

        # We've given up, but we'll let the user fix matters if they can
        while ! real_dev=$(resolve_device "${dev_id}") ||
              ! get_fstype "${real_dev}" >/dev/null; do
                if ! $may_panic; then
                        echo "Gave up waiting for ${name}"
                        return 1
                fi
                echo "Gave up waiting for ${name} device.  Common problems:"
                echo " - Boot args (cat /proc/cmdline)"
                echo "   - Check rootdelay= (did the system wait long enough?)"
                if [ "${name}" = root ]; then
                        echo "   - Check root= (did the system wait for the right
device?)"
                fi
                echo " - Missing modules (cat /proc/modules; ls /dev)"
                panic "ALERT!  ${dev_id} does not exist.  Dropping to a shell!"
        done

        DEV="${real_dev}"
}

local_mount_root()
{
        local_top
        local_device_setup "${ROOT}" "root file system"
        ROOT="${DEV}"

        # Get the root filesystem type if not set
        if [ -z "${ROOTFSTYPE}" ]; then
                FSTYPE=$(get_fstype "${ROOT}")
        else
                FSTYPE=${ROOTFSTYPE}
        fi

        local_premount
if [ -z "$KLOOP" ] && [ -z "$SQUASHFS" ] && [ -z "$UPPERDIR" ] && [ -z
"$QEMUNBD" ] ; then
        if [ "${readonly}" = "y" ] && \
```

找到这个段落,代码加在这里面

找到这一句,把这句后面花括号以前的所有代码用if---fi包围,在其后面再加入新代码

加入的if句

1.

```
           [ -z "$LOOP" ]; then
                roflag=-r
        else
                roflag=-w
        fi

        # FIXME This has no error checking
        [ -n "${FSTYPE}" ] && modprobe ${FSTYPE}

        checkfs ${ROOT} root "${FSTYPE}"

        # FIXME This has no error checking
        # Mount root
        mount ${roflag} ${FSTYPE:+-t ${FSTYPE} }${ROOTFLAGS} ${ROOT} ${rootmnt}
        mountroot_status="$?"
        if [ "$LOOP" ]; then
                if [ "$mountroot_status" != 0 ]; then
                        if [ ${FSTYPE} = ntfs ] || [ ${FSTYPE} = vfat ]; then
                                panic "
Could not mount the partition ${ROOT}.
This could also happen if the file system is not clean because of an operating
system crash, an interrupted boot process, an improper shutdown, or unplugging
of a removable device without first unmounting or ejecting it.  To fix this,
simply reboot into Windows, let it fully start, log in, run 'chkdsk /r', then
gracefully shut down and reboot back into Windows. After this you should be
able to reboot again and resume the installation.
(filesystem = ${FSTYPE}, error code = $mountroot_status)
"
                        fi
                fi

                mkdir -p /host
                mount -o move ${rootmnt} /host

                while [ ! -e "/host/${LOOP#/}" ]; do
                        panic "ALERT!  /host/${LOOP#/} does not exist.  Dropping
to a shell!"
                done

                # Get the loop filesystem type if not set
                if [ -z "${LOOPFSTYPE}" ]; then
                        eval $(fstype < "/host/${LOOP#/}")
                else
                        FSTYPE="${LOOPFSTYPE}"
                fi
                if [ "$FSTYPE" = "unknown" ] && [ -x /sbin/blkid ]; then
                        FSTYPE=$(/sbin/blkid -s TYPE -o value "/host/${LOOP#/}")
                        [ -z "$FSTYPE" ] && FSTYPE="unknown"
                fi

                if [ ${readonly} = y ]; then
                        roflag=-r
                else
                        roflag=-w
                fi

                # FIXME This has no error checking
                modprobe loop
                modprobe ${FSTYPE}

                # FIXME This has no error checking
                mount ${roflag} -o loop -t ${FSTYPE} ${LOOPFLAGS} "/host/$
{LOOP#/}" ${rootmnt}

                if [ -d ${rootmnt}/host ]; then
                        mount -o move /host ${rootmnt}/host
                fi
        fi
fi
```

加入的fi语句

2.

```
        ##########################################################
        #                        kloop by niumao                  #
        ##########################################################

if [ -n "$KLOOP" ]; then

        ### reset the value of the root variable
        HOSTDEV="${ROOT}"
        NEWROOT="${rootmnt}"
        [ -n "$KROOT" ] && ROOT="$KROOT"
        [ -n "$KROOT" ] || ROOT="/dev/loop0"
        export ROOT
        realroot="$ROOT"

        ###  auto probe the fs-type of the partition in which vhd-file live and
mount it  /host
        mkdir -p /host
        if [ -e ${NEWROOT}${KLOOP} ]; then
                mount --move $NEWROOT /host
        else
                if [ -z "$HOSTFSTYPE" ]; then
                        HOSTFSTYPE="$(blkid -s TYPE -o value "${HOSTDEV}")"
                        [ -z "$HOSTFSTYPE"  -o "${HOSTFSTYPE}" = "ntfs" ] &&
HOSTFSTYPE="ntfs-3g"
                fi
                [ "${HOSTFSTYPE}" = "ntfs-3g" ] || modprobe ${HOSTFSTYPE}
                mount -t ${HOSTFSTYPE} -o rw  ${HOSTDEV}  /host
        fi

        ### mount the vhd-file on a loop-device
        if [ "${KLOOP#/}" !=  "${KLOOP}" ]; then
                modprobe  loop
                kpartx -av /host${KLOOP}
                [ -e "$realroot" ] || sleep 3
        fi

        ### probe lvm on vhd-file
        if [ -n "$KLVM" ];  then
                modprobe dm-mod
                vgscan
                vgchange  -ay  ${KLVM}
                [ -e "$realroot" ] ||  sleep 3
        fi

        if [ "${readonly}" = "y" ] ; then
                roflag="-r"
        else
                roflag="-w"
        fi

        ### mount the realroot / in vhd-file on $NEWROOT
        if [ -z "${KLOOPFSTYPE}" ]; then
                KLOOPFSTYPE="$(blkid -s TYPE -o value "$realroot")"
                [ -z "${KLOOPFSTYPE}" ] && KLOOPFSTYPE="ext4"
        fi
        [ -e "$realroot" ] || sleep 3
        mount    ${roflag} -t "${KLOOPFSTYPE}"  $realroot $NEWROOT

        ### mount /host in initrd to /host of the realrootfs
        [ -d  ${NEWROOT}/host ] || mkdir -p ${NEWROOT}/host
        mount --move /host   ${NEWROOT}/host
fi
```

```
if [ -n "$SQUASHFS" ];  then

        ### reset the value of the root variable
        HOSTDEV="${ROOT}"
        NEWROOT="${rootmnt}"

        ###  auto probe the fs-type of the partition in which vhd-file live and
mount it  /host
        mkdir -p /host
        if [ -e ${NEWROOT}${SQUASHFS} ]; then
                mount --move $NEWROOT /host
        else
                if [ -z "$HOSTFSTYPE" ]; then
                        HOSTFSTYPE="$(blkid -s TYPE -o value "${HOSTDEV}")"
                        [ -z "$HOSTFSTYPE"  -o "${HOSTFSTYPE}" = "ntfs" ] &&
HOSTFSTYPE="ntfs-3g"
                fi
                [ "${HOSTFSTYPE}" = "ntfs-3g" ] || modprobe ${HOSTFSTYPE}
                mount -t ${HOSTFSTYPE} -o rw  ${HOSTDEV}  /host
        fi

        ###try to boot from squashfs
        modprobe overlay
        mkdir  -p /run/lowerdir /run/upperdir  /run/workdir
        mount  /host$SQUASHFS  /run/lowerdir
        mount  -t overlay overlay -o lowerdir=/run/lowerdir,upperdir=/run/
upperdir,workdir=/run/workdir    $NEWROOT

        ### mount /host in initrd to /host of the realrootfs
        [ -d  ${NEWROOT}/host ] || mkdir -p ${NEWROOT}/host
        mount --move /host   ${NEWROOT}/host
fi

if [ -n "$UPPERDIR" ] && [ -n "$WORKDIR" ];  then

        ### reset the value of the root variable
        HOSTDEV="${ROOT}"
        NEWROOT="${rootmnt}"

        ###  auto probe the fs-type of the partition in which vhd-file live and
mount it  /host
        mkdir -p /host
        if [ -e ${NEWROOT}${UPPERDIR} ]; then
                mount --move $NEWROOT /host
        else
                if [ -z "$HOSTFSTYPE" ]; then
                        HOSTFSTYPE="$(blkid -s TYPE -o value "${HOSTDEV}")"
                        [ -z "$HOSTFSTYPE"  -o "${HOSTFSTYPE}" = "ntfs" ] &&
HOSTFSTYPE="ntfs-3g"
                fi
                [ "${HOSTFSTYPE}" = "ntfs-3g" ] || modprobe ${HOSTFSTYPE}
                mount -t ${HOSTFSTYPE} -o rw  ${HOSTDEV}  /host
        fi

        ###try to boot from dir
        modprobe overlay
        mkdir  /run/lowerdir
        mount  -t overlay overlay -o lowerdir=/run/lowerdir,upperdir=/
host$UPPERDIR,workdir=/host$WORKDIR  $NEWROOT

        ### mount /host in initrd to /host of the realrootfs
        [ -d  ${NEWROOT}/host ] || mkdir -p ${NEWROOT}/host
        mount --move /host   ${NEWROOT}/host
fi
```

```
if  [ -n "$QEMUNBD" ] ; then

        ### reset the value of the root variable
        HOSTDEV="${ROOT}"
        NEWROOT="${rootmnt}"
        [ -n "$KROOT" ] && ROOT="$KROOT"
        [ -n "$KROOT" ] || ROOT="/dev/loop0"
        export ROOT
        realroot="$ROOT"

        ###  auto probe the fs-type of the partition in which vhd-file live and
mount it  /host
        mkdir -p /host
        if [ -e $NEWROOT$QEMUNBD ]; then
                mount --move $NEWROOT /host
        else
                if [ -z "$HOSTFSTYPE" ]; then
                        HOSTFSTYPE="$(blkid -s TYPE -o value "${HOSTDEV}")"
                        [ -z "$HOSTFSTYPE"  -o "${HOSTFSTYPE}" = "ntfs" ] &&
HOSTFSTYPE="ntfs-3g"
                fi
                [ "${HOSTFSTYPE}" = "ntfs-3g" ] || modprobe ${HOSTFSTYPE}
                mount -t ${HOSTFSTYPE} -o rw  ${HOSTDEV}  /host
        fi

        ### mount the vhd-file on a loop-device
        if [ "${QEMUNBD#/}" !=  "${QEMUNBD}" ]; then
                modprobe  nbd max_part=8
                modprobe  loop
                [ -e  /dev/nbd0 ] || sleep 3
                qemu-nbd  -c /dev/nbd0  /host${QEMUNBD}
                kpartx -av /dev/nbd0
                [ -e "$realroot" ] || sleep 3
        fi

        if [ "${readonly}" = "y" ] ; then
                roflag="-r"
        else
                roflag="-w"
        fi

        ### mount the realroot / in vhd-file on $NEWROOT
        if [ -z "${KLOOPFSTYPE}" ]; then
                KLOOPFSTYPE="$(blkid -s TYPE -o value "$realroot")"
                [ -z "${KLOOPFSTYPE}" ] && KLOOPFSTYPE="ext4"
        fi
        [ -e "$realroot" ] || sleep 3
        mount    ${roflag} -t "${KLOOPFSTYPE}"  $realroot $NEWROOT

        ### mount /host in initrd to /host of the realrootfs
        [ -d  ${NEWROOT}/host ] || mkdir -p ${NEWROOT}/host
        mount --move /host   ${NEWROOT}/host
fi

        #########################################################
        #                        kloop by niumao                #
        #########################################################
```

3.

总共改动3处

```
}

local_mount_fs()
{
        read_fstab_entry "$1"
```

主要代码,其后面就是函数结尾的花括号了.

```
	local_device_setup "$MNT_FSNAME" "$1 file system"
	MNT_FSNAME="${DEV}"

	local_premount

	if [ "${readonly}" = "y" ]; then
		roflag=-r
	else
		roflag=-w
	fi

	# FIXME This has no error checking
	modprobe "${MNT_TYPE}"

	if [ "$MNT_PASS" != 0 ]; then
		checkfs "$MNT_FSNAME" "$MNT_DIR" "${MNT_TYPE}"
	fi

	# FIXME This has no error checking
	# Mount filesystem
	mount ${roflag} -t "${MNT_TYPE}" -o "${MNT_OPTS}" "$MNT_FSNAME" "${rootmnt}
${MNT_DIR}"



}

mountroot()
{
	local_mount_root
}

mount_top()
{
	# Note, also called directly in case it's overridden.
	local_top
}

mount_premount()
{
	# Note, also called directly in case it's overridden.
	local_premount
}

mount_bottom()
{
	# Note, also called directly in case it's overridden.
	local_bottom
}
```