V. Приложения

Содержание:

D. Скрипты загрузки и sysconfig версии-20240825

Скрипты в этом приложении перечислены по каталогу, в котором они обычно находятся. Порядок `/etc/rc.d/init.d`, `/etc/sysconfig`, `/etc/sysconfig/network-devices`, и `/etc/sysconfig/network-devices/services`. В каждом разделе файлы перечислены в порядке их обычного вызова.

D.1. /etc/rc.d/init.d/rc

Скрипт `rc` является первым скриптом, вызываемым init и инициирующим процесс загрузки.

#!/bin/bash
Begin rc # # Description : Main Run Level Control Script # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # : DJ Lucas - dj AT linuxfromscratch D0T org # Updates : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # : Pierre Labastie - pierre AT linuxfromscratch D0T org # # Version : LFS 7.0 # # Notes : Updates March 24th, 2022: new semantics of S/K files # - Instead of testing that S scripts were K scripts in the # previous runlevel, test that they were not S scripts # - Instead of testing that K scripts were S scripts in the # previous runlevel, test that they were not K scripts # - S scripts in runlevel 0 or 6 are now run with # «script start» (was «script stop» previously).
##

. /lib/lsb/init-functions

print_error_msg() {

```
log_failure_msg
# $i is set when called
MSG="FAILURE:\n\nYou should not be reading this error message.\n\n"
MSG="${MSG}It means that an unforeseen error took place in\n"
MSG="${MSG}${i},\n"
MSG="${MSG}which exited with a return value of ${error_value}.\n"
```

```
MSG="${MSG}If you're able to track this error down to a bug in one of\n"
```

```
MSG="${MSG}the files provided by the ${DISTRO_MINI} book,\n"
MSG="${MSG}please be so kind to inform us at ${DISTRO_CONTACT}.\n"
log_failure_msg "${MSG}"
```

```
log_info_msg "Press Enter to continue..."
wait_for_user
```

```
}
```

check_script_status() {

```
# $i is set when called
if [ ! -f ${i} ]; then
   log_warning_msg "${i} is not a valid symlink."
   SCRIPT_STAT="1"
fi
```

```
if [ ! -x ${i} ]; then
   log_warning_msg "${i} is not executable, skipping."
   SCRIPT_STAT="1"
fi
```

```
}
```

run() {

```
if [ -z $interactive ]; then
   ${1} ${2}
   return $?
fi
```

```
while true; do
   read -p "Run ${1} ${2} (Yes/no/continue)? " -n 1 runit
   echo
```

```
   case ${runit} in
      c | C)
         interactive=""
         ${i} ${2}
         ret=${?}
         break;
         ;;
```

```
      n | N)
         return 0
         ;;
```

```
      y | Y)
         ${i} ${2}
         ret=${?}
```

```
         break
         ;;
   esac
done
```

```
return $ret
```

}

Read any local settings/overrides [-r /etc/sysconfig/rc.site] && source /etc/sysconfig/rc.site

DISTRO=${DISTRO:-«Linux From Scratch»} DISTRO_CONTACT=${DISTRO_CONTACT:-«lfs-dev@lists.linuxfromscratch.org (Registration required)»} DISTRO_MINI=${DISTRO_MINI:-«LFS»} IPROMPT=${IPROMPT:-«no»}

These 3 signals will not cause our script to exit trap «» INT QUIT TSTP

[«${1}» != «»] && runlevel=${1}

if [«${runlevel}» == «»]; then

```
echo "Usage: ${0} <runlevel>" >&2
exit 1
```

fi

previous=${PREVLEVEL} [«${previous}» == «»] && previous=N

if [! -d /etc/rc.d/rc${runlevel}.d]; then

```
log_info_msg "/etc/rc.d/rc${runlevel}.d does not exist.\n"
exit 1
```

fi

if [«$runlevel» == «6» -o «$runlevel» == «0»]; then IPROMPT=«no»; fi

Note: In ${LOGLEVEL:-7}, it is ':' 'dash' '7', not minus 7 if [«$runlevel» == «S»]; then

```
[ -r /etc/sysconfig/console ] && source /etc/sysconfig/console
dmesg -n "${LOGLEVEL:-7}"
```

fi

if [«${IPROMPT}» == «yes» -a «${runlevel}» == «S»]; then

```
# The total length of the distro welcome string, without escape codes
wlen=${wlen:-$(echo "Welcome to ${DISTRO}" | wc -c )}
welcome_message=${welcome_message:-"Welcome to ${INFO}${DISTRO}${NORMAL}"}
```

```
# The total length of the interactive string, without escape codes
ilen=${ilen:-$(echo "Press 'I' to enter interactive startup" | wc -c )}
```

```
 i_message=${i_message:-"Press '${FAILURE}I${NORMAL}' to enter interactive
startup"}
```

```
 # dcol and icol are spaces before the message to center the message
 # on screen. itime is the amount of wait time for the user to press a key
 wcol=$(( ( ${COLUMNS} - ${wlen} ) / 2 ))
 icol=$(( ( ${COLUMNS} - ${ilen} ) / 2 ))
 itime=${itime:-"3"}
```

```
 echo -e "\n\n"
 echo -e "\\033[${wcol}G${welcome_message}"
 echo -e "\\033[${icol}G${i_message}${NORMAL}"
 echo ""
 read -t "${itime}" -n 1 interactive 2>&1 > /dev/null
```

fi

Make lower case [«${interactive}» == «I»] && interactive=«i» [«${interactive}» != «i»] && interactive=«»

Read the state file if it exists from runlevel S [-r /run/interactive] && source /run/interactive

Stop all services marked as K, except if marked as K in the previous # runlevel: it is the responsibility of the script to not try to kill # a non running service if [«${previous}» != «N»]; then

```
 for i in $(ls -v /etc/rc.d/rc${runlevel}.d/K* 2> /dev/null)
 do
    check_script_status
    if [ "${SCRIPT_STAT}" == "1" ]; then
       SCRIPT_STAT="0"
       continue
    fi
```

```
    suffix=${i#/etc/rc.d/rc${runlevel}.d/K[0-9][0-9]}
    [ -e /etc/rc.d/rc${previous}.d/K[0-9][0-9]$suffix ] && continue
```

```
    run ${i} stop
    error_value=${?}
```

```
    if [ "${error_value}" != "0" ]; then print_error_msg; fi
 done
```

fi

if [«${previous}» == «N»]; then export IN_BOOT=1; fi

if [«$runlevel» == «6» -a -n «${FASTBOOT}»]; then

```
 touch /fastboot
```

fi

Start all services marked as S in this runlevel, except if marked as # S in the previous runlevel # it is the responsibility of the script to not try to start an already running # service for i in $(ls -v /etc/rc.d/rc${runlevel}.d/S* 2> /dev/null) do

```
if [ "${previous}" != "N" ]; then
   suffix=${i#/etc/rc.d/rc${runlevel}.d/S[0-9][0-9]}
   [ -e /etc/rc.d/rc${previous}.d/S[0-9][0-9]$suffix ] && continue
fi
```

```
check_script_status
if [ "${SCRIPT_STAT}" == "1" ]; then
   SCRIPT_STAT="0"
   continue
fi
```

```
run ${i} start
```

```
error_value=${?}
```

```
if [ "${error_value}" != "0" ]; then print_error_msg; fi
```

done

Store interactive variable on switch from runlevel S and remove if not if [«${runlevel}» == «S» -a «${interactive}» == «i»]; then

```
 echo "interactive=\"i\"" > /run/interactive
```

else

```
 rm -f /run/interactive 2> /dev/null
```

fi

Copy the boot log on initial boot only if [«${previous}» == «N» -a «${runlevel}» != «S»]; then

```
cat $BOOTLOG >> /var/log/boot.log
```

```
# Mark the end of boot
echo "--------" >> /var/log/boot.log
```

```
# Remove the temporary file
rm -f $BOOTLOG 2> /dev/null
```

fi

End rc D.2. /lib/lsb/init-functions #!/bin/sh ## ########### # # Begin /lib/lsb/init-funtions # # Description : Run Level Control Functions #

Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # : DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 # # Notes : With code based on Matthias Benkmann's simpleinit-msb # http://winterdrache.de/linux/newboot/index.html # # The file should be located in /lib/lsb # ##

Environmental setup # Setup default values for environment umask 022 export PATH=«/bin:/usr/bin:/sbin:/usr/sbin»

Set color commands, used via echo # Please consult `man console_codes for more information # under the «ECMA-48 Set Graphics Rendition» section # # Warning: when switching from a 8bit to a 9bit font, # the linux console will reinterpret the bold (1;) to # the top 256 glyphs of the 9bit font. This does # not affect framebuffer consoles

NORMAL=«\\033[0;39m» # Standard console grey SUCCESS=«\\033[1;32m» # Success is green WARNING=«\\033[1;33m» # Warnings are yellow FAILURE=«\\033[1;31m» # Failures are red INFO=«\\033[1;36m» # Information is light cyan BRACKET=«\\033[1;34m» # Brackets are blue

Use a colored prefix BMPREFIX=« » SUCCESS_PREFIX=«${SUCCESS} * ${NORMAL} » FAILURE_PREFIX=«${FAILURE}*${NORMAL} » WARNING_PREFIX=«${WARNING} *** ${NORMAL} » SKIP_PREFIX=«${INFO} S ${NORMAL}»

SUCCESS_SUFFIX=«${BRACKET}[${SUCCESS} OK ${BRACKET}]${NORMAL}» FAILURE_SUFFIX=«${BRACKET}[${FAILURE} FAIL ${BRACKET}]${NORMAL}» WARNING_SUFFIX=«${BRACKET}[${WARNING} WARN ${BRACKET}]${NORMAL}» SKIP_SUFFIX=«${BRACKET}[${INFO} SKIP ${BRACKET}]${NORMAL}»

BOOTLOG=/run/bootlog KILLDELAY=3 SCRIPT_STAT=«0»

Set any user specified environment variables e.g. HEADLESS [-r /etc/sysconfig/rc.site] && . /etc/sysconfig/rc.site

If HEADLESS is set, use that. # If file descriptor 1 or 2 (stdout and stderr) is not open or # does not refer to a terminal, consider the script headless. [! -t 1 -o ! -t 2] && HEADLESS=${HEADLESS:-yes}

if [«x$HEADLESS» != «xyes»] then

```
## Screen Dimensions
# Find current screen size
if [ -z "${COLUMNS}" ]; then
  COLUMNS=$(stty size)
  COLUMNS=${COLUMNS##* }
fi
```

else

```
 COLUMNS=80
```

fi

When using remote connections, such as a serial port, stty size returns 0 if [«${COLUMNS}» = «0»]; then

```
COLUMNS=80
```

fi

Measurements for positioning result messages COL=$[1] WCOL=$[2]

Set Cursor Position Commands, used via echo SET_COL=«\\033[${COL}G» # at the $COL char SET_WCOL=«\\033[${WCOL}G» # at the $WCOL char CURS_UP=«\\033[1A\\033[0G» # Up one line, at the 0'th char CURS_ZERO=«\\033[0G»

start_daemon() # # Usage: start_daemon [-f] [-n nicelevel] [-p pidfile] pathname [args…] # # # # Purpose: This runs the specified program as a daemon # # # # Inputs: -f: (force) run the program even if it is already running. # # -n nicelevel: specify a nice level. See 'man nice(1)'. # # -p pidfile: use the specified file to determine PIDs. # # pathname: the complete path to the specified program # # args: additional arguments passed to the program (pathname) # # # # Return values (as defined by LSB exit codes): # # 0 - program is running or service is OK # # 1 - generic or unspecified error # # 2 - invalid or excessive argument(s) # # 5 - program is not installed # ## start_daemon() {

```
  local force=""
  local nice="0"
  local pidfile=""
  local pidlist=""
  local retval=""
```

```
  # Process arguments
  while true
  do
      case "${1}" in
```

1. f)

force=«1»

```
              shift 1
              ;;
```

1. n)

nice=«${2}»

```
              shift 2
              ;;
```

1. p)

pidfile=«${2}»

```
                shift 2
                ;;
```

1. *)

return 2

```
                ;;
```

-)

program=«${1}»

```
                break
                ;;
        esac
    done
```

```
    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi
```

```
    # Execute
    if [ -z "${force}" ]; then
        if [ -z "${pidfile}" ]; then
            # Determine the pid by discovery
            pidlist=`pidofproc "${1}"`
            retval="${?}"
        else
            # The PID file contains the needed PIDs
            # Note that by LSB requirement, the path must be given to
pidofproc,
            # however, it is not used by the current implementation or
standard.
            pidlist=`pidofproc -p "${pidfile}" "${1}"`
            retval="${?}"
        fi
```

```
        # Return a value ONLY
        # It is the init script's (or distribution's functions) responsibility
        # to log messages!
        case "${retval}" in
```

```
            0)
                # Program is already running correctly, this is a
                # successful start.
                return 0
                ;;
```

```
        1)
            # Program is not running, but an invalid pid file exists
            # remove the pid file and continue
            rm -f "${pidfile}"
            ;;
```

```
        3)
            # Program is not running and no pidfile exists
            # do nothing here, let start_deamon continue.
            ;;
```

-)

Others as returned by status values shall not be interpreted

```
            # and returned as an unspecified error.
            return 1
            ;;
    esac
fi
```

```
# Do the start!
nice -n "${nice}" "${@}"
```

}

killproc() # # Usage: killproc [-p pidfile] pathname [signal] # # # # Purpose: Send control signals to running processes # # # # Inputs: -p pidfile, uses the specified pidfile # # pathname, pathname to the specified program # # signal, send this signal to pathname # # # # Return values (as defined by LSB exit codes): # # 0 - program (pathname) has stopped/is already stopped or a # # running program has been sent specified signal and stopped # # successfully # # 1 - generic or unspecified error # # 2 - invalid or excessive argument(s) # # 5 - program is not installed # # 7 - program is not running and a signal was supplied # ## killproc() {

```
local pidfile
local program
local prefix
local progname
local signal="-TERM"
local fallback="-KILL"
local nosig
local pidlist
local retval
local pid
local delay="30"
local piddead
local dtime
```

```
  # Process arguments
  while true; do
      case "${1}" in
          -p)
              pidfile="${2}"
              shift 2
              ;;
```

-)

program=«${1}»

```
              if [ -n "${2}" ]; then
                  signal="${2}"
                  fallback=""
              else
                  nosig=1
              fi
```

```
              # Error on additional arguments
              if [ -n "${3}" ]; then
                  return 2
              else
                  break
              fi
              ;;
      esac
  done
```

```
  # Check for a valid program
  if [ ! -e "${program}" ]; then return 5; fi
```

```
  # Check for a valid signal
  check_signal "${signal}"
  if [ "${?}" -ne "0" ]; then return 2; fi
```

```
  # Get a list of pids
  if [ -z "${pidfile}" ]; then
      # determine the pid by discovery
      pidlist=`pidofproc "${1}"`
      retval="${?}"
  else
      # The PID file contains the needed PIDs
      # Note that by LSB requirement, the path must be given to pidofproc,
      # however, it is not used by the current implementation or standard.
      pidlist=`pidofproc -p "${pidfile}" "${1}"`
      retval="${?}"
  fi
```

```
  # Return a value ONLY
```

```
  # It is the init script's (or distribution's functions) responsibility
  # to log messages!
  case "${retval}" in
```

```
      0)
          # Program is running correctly
          # Do nothing here, let killproc continue.
          ;;
```

```
      1)
          # Program is not running, but an invalid pid file exists
          # Remove the pid file.
```

```
          progname=${program##*/}
```

```
          if [[ -e "/run/${progname}.pid" ]]; then
              pidfile="/run/${progname}.pid"
              rm -f "${pidfile}"
          fi
```

```
          # This is only a success if no signal was passed.
          if [ -n "${nosig}" ]; then
              return 0
          else
              return 7
          fi
          ;;
```

```
      3)
          # Program is not running and no pidfile exists
          # This is only a success if no signal was passed.
          if [ -n "${nosig}" ]; then
              return 0
          else
              return 7
          fi
          ;;
```

-)

Others as returned by status values shall not be interpreted

```
          # and returned as an unspecified error.
          return 1
          ;;
  esac
```

```
  # Perform different actions for exit signals and control signals
  check_sig_type "${signal}"
```

```
  if [ "${?}" -eq "0" ]; then # Signal is used to terminate the program
```

```
      # Account for empty pidlist (pid file still exists and no
      # signal was given)
      if [ "${pidlist}" != "" ]; then
```

```
          # Kill the list of pids
          for pid in ${pidlist}; do
```

```
              kill -0 "${pid}" 2> /dev/null
```

```
              if [ "${?}" -ne "0" ]; then
                  # Process is dead, continue to next and assume all is well
                  continue
              else
                  kill "${signal}" "${pid}" 2> /dev/null
```

```
                  # Wait up to ${delay}/10 seconds to for "${pid}" to
                  # terminate in 10ths of a second
```

```
                  while [ "${delay}" -ne "0" ]; do
                      kill -0 "${pid}" 2> /dev/null || piddead="1"
                      if [ "${piddead}" = "1" ]; then break; fi
                      sleep 0.1
                      delay="$(( ${delay} - 1 ))"
                  done
```

```
                  # If a fallback is set, and program is still running, then
                  # use the fallback
                  if [ -n "${fallback}" -a "${piddead}" != "1" ]; then
                      kill "${fallback}" "${pid}" 2> /dev/null
                      sleep 1
                      # Check again, and fail if still running
                      kill -0 "${pid}" 2> /dev/null && return 1
                  fi
              fi
          done
      fi
```

```
      # Check for and remove stale PID files.
      if [ -z "${pidfile}" ]; then
          # Find the basename of $program
          prefix=`echo "${program}" | sed 's/[^/]*$//'`
          progname=`echo "${program}" | sed "s@${prefix}@@"`
```

```
          if [ -e "/run/${progname}.pid" ]; then
              rm -f "/run/${progname}.pid" 2> /dev/null
          fi
      else
```

```
            if [ -e "${pidfile}" ]; then rm -f "${pidfile}" 2> /dev/null; fi
        fi
```

```
    # For signals that do not expect a program to exit, simply
    # let kill do its job, and evaluate kill's return for value
```

```
    else # check_sig_type - signal is not used to terminate program
        for pid in ${pidlist}; do
            kill "${signal}" "${pid}"
            if [ "${?}" -ne "0" ]; then return 1; fi
        done
    fi
```

}

pidofproc() # # Usage: pidofproc [-p pidfile] pathname # # # # Purpose: This function returns one or more pid(s) for a particular daemon # # # # Inputs: -p pidfile, use the specified pidfile instead of pidof # # pathname, path to the specified program # # # # Return values (as defined by LSB status codes): # # 0 - Success (PIDs to stdout) # # 1 - Program is dead, PID file still exists (remaining PIDs output) # # 3 - Program is not running (no output) # ## pidofproc() {

```
    local pidfile
    local program
    local prefix
    local progname
    local pidlist
    local lpids
    local exitstatus="0"
```

```
    # Process arguments
    while true; do
        case "${1}" in
```

1. p)

pidfile=«${2}»

```
                shift 2
                ;;
```

-)

program=«${1}»

```
                if [ -n "${2}" ]; then
                    # Too many arguments
                    # Since this is status, return unknown
                    return 4
```

```
            else
                break
            fi
            ;;
    esac
done
```

```
# If a PID file is not specified, try and find one.
if [ -z "${pidfile}" ]; then
    # Get the program's basename
    prefix=`echo "${program}" | sed 's/[^/]*$//'`
```

```
    if [ -z "${prefix}" ]; then
       progname="${program}"
    else
       progname=`echo "${program}" | sed "s@${prefix}@@"`
    fi
```

```
    # If a PID file exists with that name, assume that is it.
    if [ -e "/run/${progname}.pid" ]; then
        pidfile="/run/${progname}.pid"
    fi
fi
```

```
# If a PID file is set and exists, use it.
if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
    # Use the value in the first line of the pidfile
    pidlist=`/bin/head -n1 "${pidfile}"`
else
    # Use pidof
    pidlist=`pidof "${program}"`
fi
```

```
# Figure out if all listed PIDs are running.
for pid in ${pidlist}; do
    kill -0 ${pid} 2> /dev/null
```

```
    if [ "${?}" -eq "0" ]; then
        lpids="${lpids}${pid} "
    else
        exitstatus="1"
    fi
done
```

```
if [ -z "${lpids}" -a ! -f "${pidfile}" ]; then
    return 3
else
    echo "${lpids}"
    return "${exitstatus}"
```

```
  fi

}

################################################################################ # statusproc() # # Usage: statusproc [-p pidfile] pathname # # # # Purpose: This function prints the status of a particular daemon to stdout # # # # Inputs: -p pidfile, use the specified pidfile instead of pidof # # pathname, path to the specified program # # # # Return values: # # 0 - Status printed # # 1 - Input error. The daemon to check was not specified. # ################################################################################ statusproc() {

local pidfile
local pidlist

if [ "${#}" = "0" ]; then
   echo "Usage: statusproc [-p pidfle] {program}"
   exit 1
fi

# Process arguments
while true; do
    case "${1}" in
```

1. p)

pidfile=«${2}»

```
            shift 2
            ;;
```

-)

if [-n «${2}»]; then

```
                echo "Too many arguments"
                return 1
            else
                break
            fi
            ;;
    esac
done

if [ -n "${pidfile}" ]; then
   pidlist=`pidofproc -p "${pidfile}" $@`
else
   pidlist=`pidofproc $@`
fi

# Trim trailing blanks
```

```
pidlist=`echo "${pidlist}" | sed -r 's/ +$//'`
```

```
base="${1##*/}"
```

```
if [ -n "${pidlist}" ]; then
   /bin/echo -e "${INFO}${base} is running with Process" \
      "ID(s) ${pidlist}.${NORMAL}"
else
   if [ -n "${base}" -a -e "/run/${base}.pid" ]; then
      /bin/echo -e "${WARNING}${1} is not running but" \
         "/run/${base}.pid exists.${NORMAL}"
   else
      if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
         /bin/echo -e "${WARNING}${1} is not running" \
            "but ${pidfile} exists.${NORMAL}"
      else
         /bin/echo -e "${INFO}${1} is not running.${NORMAL}"
      fi
   fi
fi
```

```
}
```

```
########################################################################
##################### # timespec() # # # # Purpose: An internal utility function to format a timestamp # # a boot log file. Sets the STAMP variable. # # # # Return value: Not used #
########################################################################
##################### timespec() {
```

```
STAMP="$(echo `date +"%b %d %T %:z"` `hostname`) "
return 0
```

```
}
```

```
########################################################################
##################### # log_success_msg() # # Usage: log_success_msg [«message»] # # # # Purpose: Print a successful status message to the screen and # # a boot log file. # # # # Inputs: $@ - Message # # # # Return values: Not used #
########################################################################
##################### log_success_msg() {
```

```
 if [ "x$HEADLESS" != "xyes" ]
 then
   /bin/echo -n -e "${BMPREFIX}${@}"
   /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"
 else
   logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
   /bin/echo -e "${logmessage} OK"
 fi
 # Strip non-printable characters from log file
```

```
  logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
```

```
  timespec
  /bin/echo -e "${STAMP} ${logmessage} OK" >> ${BOOTLOG}
```

```
  return 0
```

```
}
```

log_success_msg2() {

```
  if [ "x$HEADLESS" != "xyes" ]
  then
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"
  else
    echo " OK"
  fi
```

```
  echo " OK" >> ${BOOTLOG}
```

```
  return 0
```

```
}
```

########## # log_failure_msg() # # Usage: log_failure_msg [«message»] # # # # Purpose: Print a failure status message to the screen and # # a boot log file. # # # # Inputs: $@ - Message # # # # Return values: Not used # ########## log_failure_msg() {

```
  if [ "x$HEADLESS" != "xyes" ]
  then
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"
  else
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -e "${logmessage} FAIL"
  fi
```

```
  # Strip non-printable characters from log file
```

```
  timespec
  logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
  /bin/echo -e "${STAMP} ${logmessage} FAIL" >> ${BOOTLOG}
```

```
  return 0
```

```
}
```

log_failure_msg2() {

```
  if [ "x$HEADLESS" != "xyes" ]
  then
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"
  else
    echo "FAIL"
  fi
```

```
  echo "FAIL" >> ${BOOTLOG}
```

```
  return 0
```

}

log_warning_msg() # # Usage: log_warning_msg [«message»] # # # # Purpose: Print a warning status message to the screen and # # a boot log file. # # # # Return values: Not used # ## log_warning_msg() {

```
  if [ "x$HEADLESS" != "xyes" ]
  then
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${WARNING_PREFIX}${SET_COL}${WARNING_SUFFIX}"
  else
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -e "${logmessage} WARN"
  fi
```

```
  # Strip non-printable characters from log file
  logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
  timespec
  /bin/echo -e "${STAMP} ${logmessage} WARN" >> ${BOOTLOG}
```

```
  return 0
```

}

log_skip_msg() {

```
  if [ "x$HEADLESS" != "xyes" ]
  then
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SKIP_PREFIX}${SET_COL}${SKIP_SUFFIX}"
  else
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo "SKIP"
```

```
  fi
```

```
  # Strip non-printable characters from log file
  logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
  /bin/echo "SKIP" >> ${BOOTLOG}
```

```
  return 0
```

```
}
```

```
########################################################################
#################### # log_info_msg() # # Usage: log_info_msg message # # # # Purpose: Print an information message to the screen and # # a boot log file. Does not print a trailing newline character. # # # # Return values: Not used #
########################################################################
#################### log_info_msg() {
```

```
  if [ "x$HEADLESS" != "xyes" ]
  then
    /bin/echo -n -e "${BMPREFIX}${@}"
  else
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -n -e "${logmessage}"
  fi
```

```
  # Strip non-printable characters from log file
  logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
  timespec
  /bin/echo -n -e "${STAMP} ${logmessage}" >> ${BOOTLOG}
```

```
  return 0
```

```
}
```

```
log_info_msg2() {
```

```
  if [ "x$HEADLESS" != "xyes" ]
  then
    /bin/echo -n -e "${@}"
  else
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -n -e "${logmessage}"
  fi
```

```
  # Strip non-printable characters from log file
  logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
  /bin/echo -n -e "${logmessage}" >> ${BOOTLOG}
```

```
  return 0
```

```
}

########################################################################
##################### # evaluate_retval() # # Usage: Evaluate a return value and print success or failure as appropriate # # # # Purpose: Convenience function to terminate an info message # # # # Return values: Not used #
########################################################################
##################### evaluate_retval() {

  local error_value="${?}"

  if [ ${error_value} = 0 ]; then
     log_success_msg2
  else
     log_failure_msg2
  fi

}

########################################################################
##################### # check_signal() # # Usage: check_signal [ -{signal} ] # # # # Purpose: Check for a valid signal. This is not defined by any LSB draft, # # however, it is required to check the signals to determine if the # # signals chosen are invalid arguments to the other functions. # # # # Inputs: Accepts a single string value in the form of -{signal} # # # # Return values: # # 0 - Success (signal is valid # # 1 - Signal is not valid #
########################################################################
##################### check_signal() {

   local valsig

   # Add error handling for invalid signals
   valsig=" -ALRM -HUP -INT -KILL -PIPE -POLL -PROF -TERM -USR1 -USR2"
   valsig="${valsig} -VTALRM -STKFLT -PWR -WINCH -CHLD -URG -TSTP -TTIN"
   valsig="${valsig} -TTOU -STOP -CONT -ABRT -FPE -ILL -QUIT -SEGV -TRAP"
   valsig="${valsig} -SYS -EMT -BUS -XCPU -XFSZ -0 -1 -2 -3 -4 -5 -6 -8 -9"
   valsig="${valsig} -11 -13 -14 -15 "

   echo "${valsig}" | grep -- " ${1} " > /dev/null

   if [ "${?}" -eq "0" ]; then
       return 0
   else
       return 1
   fi

}

########################################################################
##################### # check_sig_type() # # Usage: check_signal [ -{signal} | {signal} ] # # # # Purpose: Check if signal is a program termination signal or a control signal # # This is not
```

defined by any LSB draft, however, it is required to # # check the signals to determine if they are intended to end a # # program or simply to control it. # # # # Inputs: Accepts a single string value in the form or -{signal} or {signal} # # # # Return values: # # 0 - Signal is used for program termination # # 1 - Signal is used for program control # ## ########## check_sig_type() {

```
  local valsig
```

```
  # The list of termination signals (limited to generally used items)
  valsig=" -ALRM -INT -KILL -TERM -PWR -STOP -ABRT -QUIT -2 -3 -6 -9 -14 -15
"
```

```
  echo "${valsig}" | grep -- " ${1} " > /dev/null
```

```
  if [ "${?}" -eq "0" ]; then
      return 0
  else
      return 1
  fi
```

```
}
```

```
########################################################## ########## # wait_for_user() # # # # Purpose: Wait for the user to respond if not a headless system # # #
########################################################## ########## wait_for_user() {
```

```
 # Wait for the user by default
 [ "${HEADLESS=0}" = "0" ] && read ENTER
 return 0
```

```
}
```

```
########################################################## ########## # is_true() # # # # Purpose: Utility to test if a variable is true | yes | 1 # # #
########################################################## ########## is_true() {
```

```
 [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ] ||  [ "$1" = "y" ]
||
 [ "$1" = "t" ]
```

```
}
```

End /lib/lsb/init-functions D.3. /etc/rc.d/init.d/mountvirtfs #!/bin/sh ## ########## # Begin mountvirtfs # # Description : Ensure proc, sysfs, run, and dev are mounted # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # DJ Lucas - dj AT

linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # Xi Ruoyao - xry111@xry111.site # # Version : LFS 12.0 # ##

BEGIN INIT INFO # Provides: mountvirtfs # Required-Start: $first # Should-Start: # Required-Stop: # Should-Stop: # Default-Start: S # Default-Stop: # Short-Description: Mounts various special fs needed at start # Description: Mounts /sys and /proc virtual (kernel) filesystems. # Mounts /run (tmpfs) and /dev (devtmpfs). # This is done only if they are not already mounted. # with the kernel config proposed in the book, dev # should be automatically mounted by the kernel. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

case «${1}» in

```
start)
   # Make sure /run is available before logging any messages
   if ! mountpoint /run >/dev/null; then
      mount /run || failed=1
   fi
```

```
   mkdir -p   /run/lock
   chmod 1777 /run/lock
```

```
   log_info_msg "Mounting virtual file systems: ${INFO}/run"
```

```
   if ! mountpoint /proc >/dev/null; then
      log_info_msg2 " ${INFO}/proc"
      mount -o nosuid,noexec,nodev /proc || failed=1
   fi
```

```
   if ! mountpoint /sys >/dev/null; then
      log_info_msg2 " ${INFO}/sys"
      mount -o nosuid,noexec,nodev /sys || failed=1
   fi
```

```
   if ! mountpoint /dev >/dev/null; then
      log_info_msg2 " ${INFO}/dev"
      mount -o mode=0755,nosuid /dev  || failed=1
   fi
```

```
   mkdir -p /dev/shm
   log_info_msg2 " ${INFO}/dev/shm"
   mount -o nosuid,nodev /dev/shm || failed=1
```

```
   mkdir -p /sys/fs/cgroup
   log_info_msg2 " ${INFO}/sys/fs/cgroup"
   mount -o nosuid,noexec,nodev /sys/fs/cgroup || failed=1
```

```
    (exit ${failed})
    evaluate_retval
    if [ "${failed}" = 1 ]; then
       exit 1
    fi
```

```
    log_info_msg "Create symlinks in /dev targeting /proc:
${INFO}/dev/stdin"
    ln -sf /proc/self/fd/0 /dev/stdin  || failed=1
```

```
    log_info_msg2 " ${INFO}/dev/stdout"
    ln -sf /proc/self/fd/1 /dev/stdout || failed=1
```

```
    log_info_msg2 " ${INFO}/dev/stderr"
    ln -sf /proc/self/fd/2 /dev/stderr || failed=1
```

```
    log_info_msg2 " ${INFO}/dev/fd"
    ln -sfn /proc/self/fd  /dev/fd     || failed=1
```

```
    if [ -e /proc/kcore ]; then
       log_info_msg2 " ${INFO}/dev/core"
       ln -sf /proc/kcore  /dev/core   || failed=1
    fi
```

```
    (exit ${failed})
    evaluate_retval
    exit $failed
    ;;
```

-)

echo «Usage: ${0} {start}»

```
    exit 1
    ;;
```

esac

End mountvirtfs D.4. /etc/rc.d/init.d/modules #!/bin/sh ## ########## # Begin modules # # Description : Module auto-loading script # # Authors : Zack Winkles # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 # ##

BEGIN INIT INFO # Provides: modules # Required-Start: mountvirtfs # Should-Start: # Required-Stop: # Should-Stop: # Default-Start: S # Default-Stop: # Short-Description: Loads required modules. # Description: Loads modules listed in /etc/sysconfig/modules. # X-LFS-Provided-By: LFS ### END INIT INFO

```
# Assure that the kernel has module support. [ -e /proc/modules ] || exit 0

. /lib/lsb/init-functions

case «${1}» in

start)
   # Exit if there's no modules file or there are no
   # valid entries
   [ -r /etc/sysconfig/modules ]             || exit 0
   grep -E -qv '^($|#)' /etc/sysconfig/modules || exit 0

   log_info_msg "Loading modules:"

   # Only try to load modules if the user has actually given us
   # some modules to load.

   while read module args; do

      # Ignore comments and blank lines.
      case "$module" in
         ""|"#"*) continue ;;
      esac

      # Attempt to load the module, passing any arguments provided.
      modprobe ${module} ${args} >/dev/null

      # Print the module name if successful, otherwise take note.
      if [ $? -eq 0 ]; then
         log_info_msg2 " ${module}"
      else
         failedmod="${failedmod} ${module}"
      fi
   done < /etc/sysconfig/modules

   # Print a message about successfully loaded modules on the correct line.
   log_success_msg2

   # Print a failure message with a list of any modules that
   # may have failed to load.
   if [ -n "${failedmod}" ]; then
      log_failure_msg "Failed to load modules:${failedmod}"
      exit 1
   fi
   ;;
```

-)

```
echo «Usage: ${0} {start}»
```

```
    exit 1
    ;;
```

esac

exit 0

End modules D.5. /etc/rc.d/init.d/udev #!/bin/sh ### ########## # Begin udev # # Description : Udev cold-plugging script # # Authors : Zack Winkles, Alexander E. Patrakov # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # Xi Ruoyao - xry111@xry111.site # # Version : LFS 12.0 # ###

BEGIN INIT INFO # Provides: udev $time # Required-Start: localnet # Should-Start: modules # Required-Stop: # Should-Stop: # Default-Start: S # Default-Stop: # Short-Description: Populates /dev with device nodes. # Description: Mounts a tempfs on /dev and starts the udevd daemon. # Device nodes are created as defined by udev. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

case «${1}» in

```
start)
   log_info_msg "Populating /dev with device nodes... "
   if ! grep -q '[[:space:]]sysfs' /proc/mounts; then
      log_failure_msg2
      msg="FAILURE:\n\nUnable to create "
      msg="${msg}devices without a SysFS filesystem\n\n"
      msg="${msg}After you press Enter, this system "
      msg="${msg}will be halted and powered off.\n\n"
      log_info_msg "$msg"
      log_info_msg "Press Enter to continue..."
      wait_for_user
      /etc/rc.d/init.d/halt start
   fi
```

```
   # Start the udev daemon to continually watch for, and act on,
   # uevents
   SYSTEMD_LOG_TARGET=kmsg /sbin/udevd --daemon
```

```
   # Now traverse /sys in order to "coldplug" devices that have
   # already been discovered
   /bin/udevadm trigger --action=add    --type=subsystems
   /bin/udevadm trigger --action=add    --type=devices
   /bin/udevadm trigger --action=change --type=devices
```

```
   # Now wait for udevd to process the uevents we triggered
   if ! is_true "$OMIT_UDEV_SETTLE"; then
      /bin/udevadm settle
```

```
    fi

    # If any LVM based partitions are on the system, ensure they
    # are activated so they can be used.
    if [ -x /sbin/vgchange ]; then /sbin/vgchange -a y >/dev/null; fi

    log_success_msg2
    ;;
```

-)

echo «Usage ${0} {start}»

```
    exit 1
    ;;
```

esac

exit 0

End udev D.6. /etc/rc.d/init.d/swap #!/bin/sh ## ########## # Begin swap # # Description : Swap Control Script # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 # ##

BEGIN INIT INFO # Provides: swap # Required-Start: udev # Should-Start: modules # Required-Stop: localnet # Should-Stop: $local_fs # Default-Start: S # Default-Stop: 0 6 # Short-Description: Activates and deactivates swap partitions. # Description: Activates and deactivates swap partitions defined in # /etc/fstab. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

case «${1}» in

```
start)
    log_info_msg "Activating all swap files/partitions..."
    swapon -a
    evaluate_retval
    ;;

stop)
    log_info_msg "Deactivating all swap files/partitions..."
    swapoff -a
    evaluate_retval
    ;;

restart)
    ${0} stop
```

```
    sleep 1
    ${0} start
    ;;
```

```
 status)
    log_success_msg "Retrieving swap status."
    swapon -s
    ;;
```

-)

echo «Usage: ${0} {start|stop|restart|status}»

```
    exit 1
    ;;
```

esac

exit 0

End swap D.7. /etc/rc.d/init.d/setclock #!/bin/sh ## ########## # Begin setclock # # Description : Setting Linux Clock # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 # ##

BEGIN INIT INFO # Provides: # Required-Start: # Should-Start: modules # Required-Stop: # Should-Stop: $syslog # Default-Start: S # Default-Stop: # Short-Description: Stores and restores time from the hardware clock # Description: On boot, system time is obtained from hwclock. The # hardware clock can also be set on shutdown. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

[-r /etc/sysconfig/clock] && . /etc/sysconfig/clock

case «${UTC}» in

```
 yes|true|1)
    CLOCKPARAMS="${CLOCKPARAMS} --utc"
    ;;
```

```
 no|false|0)
    CLOCKPARAMS="${CLOCKPARAMS} --localtime"
    ;;
```

esac

case ${1} in

```
 start)
```

```
    hwclock --hctosys ${CLOCKPARAMS} >/dev/null
    ;;
```

```
stop)
    log_info_msg "Setting hardware clock..."
    hwclock --systohc ${CLOCKPARAMS} >/dev/null
    evaluate_retval
    ;;
```

-)

echo «Usage: ${0} {start|stop}»

```
    exit 1
    ;;
```

esac

exit 0 D.8. /etc/rc.d/init.d/checkfs #!/bin/sh ## # Begin checkfs # # Description : File System Check # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # A. Luebke - luebke@users.sourceforge.net # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 # # Based on checkfs script from LFS-3.1 and earlier. # # From man fsck # 0 - No errors # 1 - File system errors corrected # 2 - System should be rebooted # 4 - File system errors left uncorrected # 8 - Operational error # 16 - Usage or syntax error # 32 - Fsck canceled by user request # 128 - Shared library error # ## ############

BEGIN INIT INFO # Provides: checkfs # Required-Start: udev swap # Should-Start: # Required-Stop: # Should-Stop: # Default-Start: S # Default-Stop: # Short-Description: Checks local filesystems before mounting. # Description: Checks local filesystems before mounting. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

case «${1}» in

```
start)
    if [ -f /fastboot ]; then
       msg="/fastboot found, will omit "
       msg="${msg} file system checks as requested.\n"
       log_info_msg "${msg}"
       exit 0
    fi
```

```
    log_info_msg "Mounting root file system in read-only mode... "
    mount -n -o remount,ro / >/dev/null
```

```
    if [ ${?} != 0 ]; then
```

```
      log_failure_msg2
      msg="\n\nCannot check root "
      msg="${msg}filesystem because it could not be mounted "
      msg="${msg}in read-only mode.\n\n"
      msg="${msg}After you press Enter, this system will be "
      msg="${msg}halted and powered off.\n\n"
      log_failure_msg "${msg}"
```

```
      log_info_msg "Press Enter to continue..."
      wait_for_user
      /etc/rc.d/init.d/halt start
   else
      log_success_msg2
   fi
```

```
   if [ -f /forcefsck ]; then
      msg="/forcefsck found, forcing file"
      msg="${msg} system checks as requested."
      log_success_msg "$msg"
      options="-f"
   else
      options=""
   fi
```

```
   log_info_msg "Checking file systems..."
   # Note: -a option used to be -p; but this fails e.g. on fsck.minix
   if is_true "$VERBOSE_FSCK"; then
     fsck ${options} -a -A -C -T
   else
     fsck ${options} -a -A -C -T >/dev/null
   fi
```

```
   error_value=${?}
```

```
   if [ "${error_value}" = 0 ]; then
      log_success_msg2
   fi
```

```
   if [ "${error_value}" = 1 ]; then
      msg="\nWARNING:\n\nFile system errors "
      msg="${msg}were found and have been corrected.\n"
      msg="${msg}      You may want to double-check that "
      msg="${msg}everything was fixed properly."
      log_warning_msg "$msg"
   fi
```

```
   if [ "${error_value}" = 2 -o "${error_value}" = 3 ]; then
      msg="\nWARNING:\n\nFile system errors "
      msg="${msg}were found and have been "
      msg="${msg}corrected, but the nature of the "
```

```
       msg="${msg}errors require this system to be rebooted.\n\n"
       msg="${msg}After you press enter, "
       msg="${msg}this system will be rebooted\n\n"
       log_failure_msg "$msg"
```

```
       log_info_msg "Press Enter to continue..."
       wait_for_user
       reboot -f
    fi
```

```
    if [ "${error_value}" -gt 3 -a "${error_value}" -lt 16 ]; then
       msg="\nFAILURE:\n\nFile system errors "
       msg="${msg}were encountered that could not be "
       msg="${msg}fixed automatically.\nThis system "
       msg="${msg}cannot continue to boot and will "
       msg="${msg}therefore be halted until those "
       msg="${msg}errors are fixed manually by a "
       msg="${msg}System Administrator.\n\n"
       msg="${msg}After you press Enter, this system will be "
       msg="${msg}halted and powered off.\n\n"
       log_failure_msg "$msg"
```

```
       log_info_msg "Press Enter to continue..."
       wait_for_user
       /etc/rc.d/init.d/halt start
    fi
```

```
    if [ "${error_value}" -ge 16 ]; then
       msg="FAILURE:\n\nUnexpected failure "
       msg="${msg}running fsck.  Exited with error "
       msg="${msg} code: ${error_value}.\n"
       log_info_msg $msg
       exit ${error_value}
    fi
```

```
    exit 0
    ;;
 *)
    echo "Usage: ${0} {start}"
    exit 1
    ;;
```

esac

End checkfs D.9. /etc/rc.d/init.d/mountfs #!/bin/sh ## ########## # Begin mountfs # # Description : File System Mount Script # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 #

##########

BEGIN INIT INFO # Provides: $local_fs # Required-Start: udev checkfs # Should-Start: modules # Required-Stop: localnet # Should-Stop: # Default-Start: S # Default-Stop: 0 6 # Short-Description: Mounts/unmounts local filesystems defined in /etc/fstab. # Description: Remounts root filesystem read/write and mounts all # remaining local filesystems defined in /etc/fstab on # start. Remounts root filesystem read-only and unmounts # remaining filesystems on stop. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

case «${1}» in

```
start)
   log_info_msg "Remounting root file system in read-write mode..."
   mount --options remount,rw / >/dev/null
   evaluate_retval
```

```
   # Remove fsck-related file system watermarks.
   rm -f /fastboot /forcefsck
```

```
   # Make sure /dev/pts exists
   mkdir -p /dev/pts
```

```
   # This will mount all filesystems that do not have _netdev in
   # their option list.  _netdev denotes a network filesystem.
```

```
   log_info_msg "Mounting remaining file systems..."
   failed=0
   mount --all --test-opts no_netdev >/dev/null || failed=1
   evaluate_retval
   exit $failed
   ;;
```

```
stop)
   # Don't unmount virtual file systems like /run
   log_info_msg "Unmounting all other currently mounted file systems..."
   # Ensure any loop devices are removed
   losetup -D
   umount --all --detach-loop --read-only \
          --types notmpfs,nosysfs,nodevtmpfs,noproc,nodevpts >/dev/null
   evaluate_retval
```

```
   # Make sure / is mounted read only (umount bug)
   mount --options remount,ro /
```

```
   # Make all LVM volume groups unavailable, if appropriate
   # This fails if swap or / are on an LVM partition
   #if [ -x /sbin/vgchange ]; then /sbin/vgchange -an > /dev/null; fi
   if [ -r /etc/mdadm.conf ]; then
      log_info_msg "Mark arrays as clean..."
```

```
        mdadm --wait-clean --scan
        evaluate_retval
    fi
    ;;
```

-)

echo «Usage: ${0} {start|stop}»

```
    exit 1
    ;;
```

esac

End mountfs D.10. /etc/rc.d/init.d/udev_retry #!/bin/sh ## ########## # Begin udev_retry # # Description : Udev cold-plugging script (retry) # # Authors : Alexander E. Patrakov # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # Bryan Kadzban - # # Version : LFS 7.0 # ##

BEGIN INIT INFO # Provides: udev_retry # Required-Start: udev # Should-Start: $local_fs cleanfs # Required-Stop: # Should-Stop: # Default-Start: S # Default-Stop: # Short-Description: Replays failed uevents and creates additional devices. # Description: Replays any failed uevents that were skipped due to # slow hardware initialization, and creates those needed # device nodes # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

case «${1}» in

```
start)
    log_info_msg "Retrying failed uevents, if any..."
```

```
    rundir=/run/udev
    # From Debian: "copy the rules generated before / was mounted
    # read-write":
```

```
    for file in ${rundir}/tmp-rules--*; do
        dest=${file##*tmp-rules--}
        [ "$dest" = '*' ] && break
        cat $file >> /etc/udev/rules.d/$dest
        rm -f $file
    done
```

```
    # Re-trigger the uevents that may have failed,
    # in hope they will succeed now
    /bin/sed -e 's/#.*$//' /etc/sysconfig/udev_retry | /bin/grep -v '^$' | \
    while read line ; do
        for subsystem in $line ; do
```

```
         /bin/udevadm trigger --subsystem-match=$subsystem --action=add
      done
   done
```

```
   # Now wait for udevd to process the uevents we triggered
   if ! is_true "$OMIT_UDEV_RETRY_SETTLE"; then
      /bin/udevadm settle
   fi
```

```
   evaluate_retval
   ;;
```

-)

echo «Usage ${0} {start}»

```
   exit 1
   ;;
```

esac

exit 0

End udev_retry D.11. /etc/rc.d/init.d/cleanfs #!/bin/sh ## ########### # Begin cleanfs # # Description : Clean file system # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 # ##

BEGIN INIT INFO # Provides: cleanfs # Required-Start: $local_fs # Should-Start: # Required-Stop: # Should-Stop: # Default-Start: S # Default-Stop: # Short-Description: Cleans temporary directories early in the boot process. # Description: Cleans temporary directories /run, /var/lock, and # optionally, /tmp. cleanfs also creates /run/utmp # and any files defined in /etc/sysconfig/createfiles. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

Function to create files/directory on boot. create_files() {

```
# Input to file descriptor 9 and output to stdin (redirection)
exec 9>&0 < /etc/sysconfig/createfiles
```

```
while read name type perm usr grp dtype maj min junk
do
   # Ignore comments and blank lines.
   case "${name}" in
      ""|\#*) continue ;;
   esac
```

```
   # Ignore existing files.
```

```
    if [ ! -e "${name}" ]; then
       # Create stuff based on its type.
       case "${type}" in
          dir)
             mkdir "${name}"
             ;;
          file)
             :> "${name}"
             ;;
          dev)
             case "${dtype}" in
                char)
                   mknod "${name}" c ${maj} ${min}
                   ;;
                block)
                   mknod "${name}" b ${maj} ${min}
                   ;;
                pipe)
                   mknod "${name}" p
                   ;;
                *)
                   log_warning_msg "\nUnknown device type: ${dtype}"
                   ;;
             esac
             ;;
          *)
             log_warning_msg "\nUnknown type: ${type}"
             continue
             ;;
       esac
```

```
       # Set up the permissions, too.
       chown ${usr}:${grp} "${name}"
       chmod ${perm} "${name}"
    fi
 done
```

```
 # Close file descriptor 9 (end redirection)
 exec 0>&9 9>&-
 return 0
```

```
}
```

case «${1}» in

```
 start)
    log_info_msg "Cleaning file systems:"
```

```
    if [ "${SKIPTMPCLEAN}" = "" ]; then
       log_info_msg2 " /tmp"
```

```
       cd /tmp &&
       find . -xdev -mindepth 1 ! -name lost+found -delete || failed=1
    fi
```

```
    > /run/utmp
```

```
    if grep -q '^utmp:' /etc/group ; then
       chmod 664 /run/utmp
       chgrp utmp /run/utmp
    fi
```

```
    (exit ${failed})
    evaluate_retval
```

```
    if grep -E -qv '^(#|$)' /etc/sysconfig/createfiles 2>/dev/null; then
       log_info_msg "Creating files and directories... "
       create_files      # Always returns 0
       evaluate_retval
    fi
```

```
    exit $failed
    ;;
*)
    echo "Usage: ${0} {start}"
    exit 1
    ;;
```

esac

End cleanfs D.12. /etc/rc.d/init.d/console #!/bin/sh ## ########## # Begin console # # Description : Sets keymap and screen font # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # Alexander E. Patrakov # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 # ##

BEGIN INIT INFO # Provides: console # Required-Start: $local_fs # Should-Start: udev_retry # Required-Stop: # Should-Stop: # Default-Start: S # Default-Stop: # Short-Description: Sets up a localised console. # Description: Sets up fonts and language settings for the user's # local as defined by /etc/sysconfig/console. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

Native English speakers probably don't have /etc/sysconfig/console at all [-r /etc/sysconfig/console] && . /etc/sysconfig/console

failed=0

case «${1}» in

```
start)
   # See if we need to do anything
   if [ -z "${KEYMAP}"         ] && [ -z "${KEYMAP_CORRECTIONS}" ] &&
      [ -z "${FONT}"           ] && [ -z "${LEGACY_CHARSET}"     ] &&
      ! is_true "${UNICODE}"; then
      exit 0
   fi

   # There should be no bogus failures below this line!
   log_info_msg "Setting up Linux console..."

   # Figure out if a framebuffer console is used
   [ -d /sys/class/graphics/fbcon ] && use_fb=1 || use_fb=0

   # Figure out the command to set the console into the
   # desired mode
   is_true "${UNICODE}" &&
      MODE_COMMAND="echo -en '\033%G' && kbd_mode -u" ||
      MODE_COMMAND="echo -en '\033%@\033(K' && kbd_mode -a"

   # On framebuffer consoles, font has to be set for each vt in
   # UTF-8 mode. This doesn't hurt in non-UTF-8 mode also.

   ! is_true "${use_fb}" || [ -z "${FONT}" ] ||
      MODE_COMMAND="${MODE_COMMAND} && setfont ${FONT}"

   # Apply that command to all consoles mentioned in
   # /etc/inittab. Important: in the UTF-8 mode this should
   # happen before setfont, otherwise a kernel bug will
   # show up and the unicode map of the font will not be
   # used.

   for TTY in `grep '^[^#].*respawn:/sbin/agetty' /etc/inittab |
      grep -o '\btty[[:digit:]]*\b'`
   do
      openvt -f -w -c ${TTY#tty} -- \
         /bin/sh -c "${MODE_COMMAND}" || failed=1
   done

   # Set the font (if not already set above) and the keymap
   [ "${use_fb}" == "1" ] || [ -z "${FONT}" ] || setfont $FONT || failed=1

   [ -z "${KEYMAP}" ] ||
      loadkeys ${KEYMAP} >/dev/null 2>&1 ||
      failed=1

   [ -z "${KEYMAP_CORRECTIONS}" ] ||
      loadkeys ${KEYMAP_CORRECTIONS} >/dev/null 2>&1 ||
      failed=1
```

```
    # Convert the keymap from $LEGACY_CHARSET to UTF-8
    [ -z "$LEGACY_CHARSET" ] ||
       dumpkeys -c "$LEGACY_CHARSET" | loadkeys -u >/dev/null 2>&1 ||
       failed=1
```

```
    # If any of the commands above failed, the trap at the
    # top would set $failed to 1
    ( exit $failed )
    evaluate_retval
```

```
    exit $failed
    ;;
```

-)

echo «Usage: ${0} {start}»

```
    exit 1
    ;;
```

esac

End console D.13. /etc/rc.d/init.d/localnet #!/bin/sh ## # Begin localnet # # Description : Loopback device # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 #

BEGIN INIT INFO # Provides: localnet # Required-Start: mountvirtfs # Should-Start: modules # Required-Stop: # Should-Stop: # Default-Start: S # Default-Stop: 0 6 # Short-Description: Starts the local network. # Description: Sets the hostname of the machine and starts the # loopback interface. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions [-r /etc/sysconfig/network] && . /etc/sysconfig/network [-r /etc/hostname] && HOSTNAME=`cat /etc/hostname`

case «${1}» in

```
 start)
    log_info_msg "Bringing up the loopback interface..."
    ip addr add 127.0.0.1/8 label lo dev lo
    ip link set lo up
    evaluate_retval
```

```
    log_info_msg "Setting hostname to ${HOSTNAME}..."
    hostname ${HOSTNAME}
    evaluate_retval
    ;;
```

```
stop)
   log_info_msg "Bringing down the loopback interface..."
   ip link set lo down
   evaluate_retval
   ;;
```

```
restart)
   ${0} stop
   sleep 1
   ${0} start
   ;;
```

```
status)
   echo "Hostname is: $(hostname)"
   ip link show lo
   ;;
```

-)

echo «Usage: ${0} {start|stop|restart|status}»

```
   exit 1
   ;;
```

esac

exit 0

End localnet D.14. /etc/rc.d/init.d/sysctl #!/bin/sh ## ########## # Begin sysctl # # Description : File uses /etc/sysctl.conf to set kernel runtime # parameters # # Authors : Nathan Coulson (nathan AT linuxfromscratch D0T org) # Matthew Burgress (matthew AT linuxfromscratch D0T org) # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 # ##

BEGIN INIT INFO # Provides: sysctl # Required-Start: mountvirtfs # Should-Start: console # Required-Stop: # Should-Stop: # Default-Start: S # Default-Stop: # Short-Description: Makes changes to the proc filesystem # Description: Makes changes to the proc filesystem as defined in # /etc/sysctl.conf. See 'man sysctl(8)'. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

case «${1}» in

```
start)
   if [ -f "/etc/sysctl.conf" ]; then
      log_info_msg "Setting kernel runtime parameters..."
      sysctl -q -p
      evaluate_retval
```

```
    fi
    ;;
```

```
status)
    sysctl -a
    ;;
```

-)

echo «Usage: ${0} {start|status}»

```
    exit 1
    ;;
```

esac

exit 0

End sysctl D.15. /etc/rc.d/init.d/sysklogd #!/bin/sh ## ########## # Begin sysklogd # # Description : Sysklogd loader # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org LFS12.1 # Remove kernel log daemon. The functionality has been # merged with syslogd. # # Version : LFS 7.0 # ##

BEGIN INIT INFO # Provides: $syslog # Required-Start: $first localnet # Should-Start: # Required-Stop: $local_fs # Should-Stop: sendsignals # Default-Start: 2 3 4 5 # Default-Stop: 0 1 6 # Short-Description: Starts system log daemon. # Description: Starts system log daemon. # /etc/fstab. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

case «${1}» in

```
start)
    log_info_msg "Starting system log daemon..."
    parms=${SYSKLOGD_PARMS-'-m 0'}
    start_daemon /sbin/syslogd $parms
    evaluate_retval
    ;;
```

```
stop)
    log_info_msg "Stopping system log daemon..."
    killproc /sbin/syslogd
    evaluate_retval
    ;;
```

```
reload)
    log_info_msg "Reloading system log daemon config file..."
```

```
    pid=`pidofproc syslogd`
    kill -HUP "${pid}"
    evaluate_retval
    ;;
```

```
 restart)
    ${0} stop
    sleep 1
    ${0} start
    ;;
```

```
 status)
    statusproc /sbin/syslogd
    ;;
```

-)

echo «Usage: ${0} {start|stop|reload|restart|status}»

```
    exit 1
    ;;
```

esac

exit 0

End sysklogd D.16. /etc/rc.d/init.d/network #!/bin/sh ## ########## # Begin network # # Description : Network Control Script # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # Nathan Coulson - nathan AT linuxfromscratch D0T org # Kevin P. Fleming - kpfleming@linuxfromscratch.org # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 # ##

BEGIN INIT INFO # Provides: $network # Required-Start: $local_fs localnet swap # Should-Start: $syslog firewalld iptables nftables # Required-Stop: $local_fs localnet swap # Should-Stop: $syslog firewalld iptables nftables # Default-Start: 2 3 4 5 # Default-Stop: 0 1 6 # Short-Description: Starts and configures network interfaces. # Description: Starts and configures network interfaces. # X-LFS-Provided-By: LFS ### END INIT INFO

case «${1}» in

```
 start)
    # if the default route exists, network is already configured
    if ip route | grep -q "^default"; then return 0; fi
    # Start all network interfaces
    for file in /etc/sysconfig/ifconfig.*
    do
       interface=${file##*/ifconfig.}
```

```
      # Skip if $file is * (because nothing was found)
      if [ "${interface}" = "*" ]; then continue; fi
```

```
      /sbin/ifup ${interface}
   done
   ;;
```

```
stop)
   # Unmount any network mounted file systems
    umount --all --force --types nfs,cifs,nfs4
```

```
   # Reverse list
   net_files=""
   for file in  /etc/sysconfig/ifconfig.*
   do
      net_files="${file} ${net_files}"
   done
```

```
   # Stop all network interfaces
   for file in ${net_files}
   do
      interface=${file##*/ifconfig.}
```

```
      # Skip if $file is * (because nothing was found)
      if [ "${interface}" = "*" ]; then continue; fi
```

```
      # See if interface exists
      if [ ! -e /sys/class/net/$interface ]; then continue; fi
```

```
      # Is interface UP?
      ip link show $interface 2>/dev/null | grep -q "state UP"
      if [ $? -ne 0 ];  then continue; fi
```

```
      /sbin/ifdown ${interface}
   done
   ;;
```

```
restart)
   ${0} stop
   sleep 1
   ${0} start
   ;;
```

-)

echo «Usage: ${0} {start|stop|restart}»

```
   exit 1
   ;;
```

esac

exit 0

End network D.17. /etc/rc.d/init.d/sendsignals #!/bin/sh ## ########## # Begin sendsignals # # Description : Sendsignals Script # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 # ##

BEGIN INIT INFO # Provides: sendsignals # Required-Start: # Should-Start: # Required-Stop: $local_fs swap localnet # Should-Stop: # Default-Start: # Default-Stop: 0 6 # Short-Description: Attempts to kill remaining processes. # Description: Attempts to kill remaining processes. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

case «${1}» in

```
stop)
   omit=$(pidof mdmon)
   [ -n "$omit" ] && omit="-o $omit"
```

```
   log_info_msg "Sending all processes the TERM signal..."
   killall5 -15 $omit
   error_value=${?}
```

```
   sleep ${KILLDELAY}
```

```
   if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
      log_success_msg
   else
      log_failure_msg
   fi
```

```
   log_info_msg "Sending all processes the KILL signal..."
   killall5 -9 $omit
   error_value=${?}
```

```
   sleep ${KILLDELAY}
```

```
   if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
      log_success_msg
   else
      log_failure_msg
   fi
   ;;
```

-)

echo «Usage: ${0} {stop}»

```
    exit 1
    ;;
```

esac

exit 0

End sendsignals D.18. /etc/rc.d/init.d/reboot #!/bin/sh
Begin reboot # # Description : Reboot Scripts # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # DJ Lucas - dj AT linuxfromscratch D0T org # Updates : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # : Pierre Labastie - pierre AT linuxfromscratch D0T org # # Version : LFS 7.0 # # Notes : Update March 24th, 2022: change «stop» to «start». # Add the $last facility to Required-start
##

BEGIN INIT INFO # Provides: reboot # Required-Start: $last # Should-Start: # Required-Stop: # Should-Stop: # Default-Start: 6 # Default-Stop: # Short-Description: Reboots the system. # Description: Reboots the System. # X-LFS-Provided-By: LFS ### END INIT INFO

. /lib/lsb/init-functions

case «${1}» in

```
start)
   log_info_msg "Restarting system..."
   reboot -d -f -i
   ;;
```

-)

echo «Usage: ${0} {start}»

```
    exit 1
    ;;
```

esac

End reboot D.19. /etc/rc.d/init.d/halt #!/bin/sh
Begin halt # # Description : Halt Script # # Authors : Gerard Beekmans - gerard AT linuxfromscratch D0T org # DJ Lucas - dj AT linuxfromscratch D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # : Pierre Labastie - pierre AT linuxfromscratch D0T org # # Version : LFS 7.0 # # Notes : Update March 24th, 2022: change «stop» to «start». # Add the $last facility to Required-start
##

BEGIN INIT INFO # Provides: halt # Required-Start: $last # Should-Start: # Required-Stop: # Should-Stop: # Default-Start: 0 # Default-Stop: # Short-Description: Halts the system. # Description: Halts the System. # X-LFS-Provided-By: LFS ### END INIT INFO

case «${1}» in

```
start)
   halt -d -f -i -p
   ;;
```

-)

echo «Usage: {start}»

```
   exit 1
   ;;
```

esac

End halt D.20. /etc/rc.d/init.d/template #!/bin/sh ## ########## # Begin scriptname # # Description : # # Authors : # # Version : LFS x.x # # Notes : # ##

BEGIN INIT INFO # Provides: template # Required-Start: # Should-Start: # Required-Stop: # Should-Stop: # Default-Start: # Default-Stop: # Short-Description: # Description: # X-LFS-Provided-By: ### END INIT INFO

. /lib/lsb/init-functions

case «${1}» in

```
start)
   log_info_msg "Starting..."
 # if it is possible to use start_daemon
   start_daemon fully_qualified_path
 # if it is not possible to use start_daemon
 # (command to start the daemon is not simple enough)
   if ! pidofproc daemon_name_as_reported_by_ps >/dev/null; then
      command_to_start_the_service
   fi
   evaluate_retval
   ;;
```

```
stop)
   log_info_msg "Stopping..."
 # if it is possible to use killproc
   killproc fully_qualified_path
 # if it is not possible to use killproc
```

```
  # (the daemon shouldn't be stopped by killing it)
    if pidofproc daemon_name_as_reported_by_ps >/dev/null; then
       command_to_stop_the_service
    fi
    evaluate_retval
    ;;
```

```
 restart)
    ${0} stop
    sleep 1
    ${0} start
    ;;
```

-)

echo «Usage: ${0} {start|stop|restart}»

```
    exit 1
    ;;
```

esac

exit 0

End scriptname D.21. /etc/sysconfig/modules
Begin /etc/sysconfig/modules # # Description : Module auto-loading configuration # # Authors : # # Version : 00.00 # # Notes : The syntax of this file is as follows: # <module> [<arg1> <arg2> ...] # # Each module should be on its own line, and any options that you want # passed to the module should follow it. The line deliminator is either # a space or a tab.
##########

End /etc/sysconfig/modules D.22. /etc/sysconfig/createfiles
Begin /etc/sysconfig/createfiles # # Description : Createfiles script config file # # Authors : # # Version : 00.00 # # Notes : The syntax of this file is as follows: # if type is equal to «file» or «dir» # <filename> <type> <permissions> <user> <group> # if type is equal to «dev» # <filename> <type> <permissions> <user> <group> <devtype> # <major> <minor> # # <filename> is the name of the file which is to be created # <type> is either file, dir, or dev. # file creates a new file # dir creates a new directory # dev creates a new device # <devtype> is either block, char or pipe # block creates a block device # char creates a character device # pipe creates a pipe, this will ignore the <major> and # <minor> fields # <major> and <minor> are the major and minor numbers used for # the device.
##########

End /etc/sysconfig/createfiles D.23. /etc/sysconfig/udev-retry
Begin /etc/sysconfig/udev_retry # # Description : udev_retry script configuration # # Authors : # # Version : 00.00 # # Notes : Each subsystem that may need to be re-triggered after mountfs # runs should be listed in this file. Probable subsystems to be # listed here are rtc (due to

/var/lib/hwclock/adjtime) and sound # (due to both /var/lib/alsa/asound.state and /usr/sbin/alsactl). # Entries are whitespace-separated.
##

rtc

End /etc/sysconfig/udev_retry D.24. /sbin/ifup #!/bin/sh
Begin /sbin/ifup # # Description : Interface Up # # Authors : Nathan Coulson - nathan AT linuxfromscratch D0T org # Kevin P. Fleming - kpfleming@linuxfromscratch.org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # DJ Lucas - dj AT linuxfromscratch D0T org # # Version : LFS 7.7 # # Notes : The IFCONFIG variable is passed to the SERVICE script # in the /lib/services directory, to indicate what file the # service should source to get interface specifications.
##

up() {

```
log_info_msg "Bringing up the ${1} interface..."
```

```
if ip link show $1 > /dev/null 2>&1; then
   link_status=`ip link show $1`
```

```
   if [ -n "${link_status}" ]; then
      if ! echo "${link_status}" | grep -q UP; then
         ip link set $1 up
      fi
   fi
```

```
else
   log_failure_msg "Interface ${IFACE} doesn't exist."
   exit 1
fi
```

```
evaluate_retval
```

}

RELEASE=«7.7»

USAGE=«Usage: $0 [-hV] [–help] [–version] interface» VERSTR=«LFS ifup, version ${RELEASE}»

while [$# -gt 0]; do

```
 case "$1" in
    --help | -h)     help="y"; break ;;
```

1. -version | -V) echo «${VERSTR}»; exit 0 ;;

1. *) echo «ifup: ${1}: invalid option» >&2

echo «${USAGE}» >& 2

```
                        exit 2 ;;
```

-) break ;;

esac done

if [-n «$help»]; then

```
 echo "${VERSTR}"
 echo "${USAGE}"
 echo
 cat << HERE_EOF
```

ifup is used to bring up a network interface. The interface parameter, e.g. eth0 or eth0:2, must match the trailing part of the interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

HERE_EOF

```
 exit 0
```

fi

file=/etc/sysconfig/ifconfig.${1}

Skip backup files [«${file}» = «${file%»«~»«}»] || exit 0

. /lib/lsb/init-functions

if [! -r «${file}»]; then

```
 log_failure_msg "Unable to bring up ${1} interface! ${file} is missing or
cannot be accessed."
 exit 1
```

fi

. $file

if [«$IFACE» = «»]; then

```
 log_failure_msg "Unable to bring up ${1} interface! ${file} does not define
an interface [IFACE]."
 exit 1
```

fi

Do not process this service if started by boot, and ONBOOT # is not set to yes if [«${IN_BOOT}» = «1» -a «${ONBOOT}» != «yes»]; then

```
  exit 0
```

fi

Bring up the interface if [«$VIRTINT» != «yes»]; then

```
  up ${IFACE}
```

fi

for S in ${SERVICE}; do

```
if [ ! -x "/lib/services/${S}" ]; then
  MSG="\nUnable to process ${file}.  Either "
  MSG="${MSG}the SERVICE '${S} was not present "
  MSG="${MSG}or cannot be executed."
  log_failure_msg "$MSG"
  exit 1
fi
```

done

#if [«${SERVICE}» = «wpa»]; then log_success_msg; fi

Create/configure the interface for S in ${SERVICE}; do

```
IFCONFIG=${file} /lib/services/${S} ${IFACE} up
```

done

Set link up virtual interfaces if [«${VIRTINT}» == «yes»]; then

```
  up ${IFACE}
```

fi

Bring up any additional interface components for I in $INTERFACE_COMPONENTS; do up $I; done

Set MTU if requested. Check if MTU has a «good» value. if test -n «${MTU}»; then

```
 if [[ ${MTU} =~ ^[0-9]+$ ]] && [[ $MTU -ge 68 ]] ; then
    for I in $IFACE $INTERFACE_COMPONENTS; do
       ip link set dev $I mtu $MTU;
    done
 else
    log_info_msg2 "Invalid MTU $MTU"
 fi
```

fi

Set the route default gateway if requested if [-n «${GATEWAY}»]; then

```
 if ip route | grep -q default; then
    log_warning_msg "Gateway already setup; skipping."
 else
    log_info_msg "Adding default gateway ${GATEWAY} to the ${IFACE}
interface..."
    ip route add default via ${GATEWAY} dev ${IFACE}
    evaluate_retval
 fi
```

fi

End /sbin/ifup D.25. /sbin/ifdown #!/bin/bash
Begin /sbin/ifdown # # Description : Interface Down # # Authors : Nathan Coulson - nathan AT linuxfromscratch D0T org # Kevin P. Fleming - kpfleming@linuxfromscratch.org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 # # Notes : the IFCONFIG variable is passed to the scripts found # in the /lib/services directory, to indicate what file the # service should source to get interface specifications.
##

RELEASE=«7.0»

USAGE=«Usage: $0 [-hV] [–help] [–version] interface» VERSTR=«LFS ifdown, version ${RELEASE}»

while [$# -gt 0]; do

```
 case "$1" in
    --help | -h)     help="y"; break ;;
```

1. -version | -V) echo «${VERSTR}»; exit 0 ;;

1. *) echo «ifup: ${1}: invalid option» >&2

echo «${USAGE}» >& 2

```
                      exit 2 ;;
```

-) break ;;

esac done

if [-n «$help»]; then

```
 echo "${VERSTR}"
 echo "${USAGE}"
 echo
 cat << HERE_EOF
```

ifdown is used to bring down a network interface. The interface parameter, e.g. eth0 or eth0:2, must

match the trailing part of the interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

HERE_EOF

```
 exit 0
```

fi

file=/etc/sysconfig/ifconfig.${1}

Skip backup files [«${file}» = «${file%»«~»«}»] || exit 0

. /lib/lsb/init-functions

if [! -r «${file}»]; then

```
 log_warning_msg "${file} is missing or cannot be accessed."
 exit 1
```

fi

. ${file}

if [«$IFACE» = «»]; then

```
 log_failure_msg "${file} does not define an interface [IFACE]."
 exit 1
```

fi

We only need to first service to bring down the interface S=`echo ${SERVICE} | cut -f1 -d« «`

if ip link show ${IFACE} > /dev/null 2>&1; then

```
 if [ -n "${S}" -a -x "/lib/services/${S}" ]; then
   IFCONFIG=${file} /lib/services/${S} ${IFACE} down
 else
   MSG="Unable to process ${file}.  Either "
   MSG="${MSG}the SERVICE variable was not set "
   MSG="${MSG}or the specified service cannot be executed."
   log_failure_msg "$MSG"
   exit 1
fi
```

else

```
 log_warning_msg "Interface ${1} doesn't exist."
```

fi

Leave the interface up if there are additional interfaces in the device link_status=`ip link show

${IFACE} 2>/dev/null`

if [-n «${link_status}»]; then

```
 if [ "$(echo "${link_status}" | grep UP)" != "" ]; then
    if [ "$(ip addr show ${IFACE} | grep 'inet ')" == ""  ]; then
       log_info_msg "Bringing down the ${IFACE} interface..."
       ip link set ${IFACE} down
       evaluate_retval
    fi
 fi
```

fi

End /sbin/ifdown D.26. /lib/services/ipv4-static #!/bin/sh
##
########## # Begin /lib/services/ipv4-static # # Description : IPV4 Static Boot Script # # Authors : Nathan Coulson - nathan AT linuxfromscratch D0T org # Kevin P. Fleming - kpfleming@linuxfromscratch.org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 #
##
##########

. /lib/lsb/init-functions . ${IFCONFIG}

if [-z «${IP}»]; then

```
 log_failure_msg "\nIP variable missing from ${IFCONFIG}, cannot continue."
 exit 1
```

fi

if [-z «${PREFIX}» -a -z «${PEER}»]; then

```
 log_warning_msg "\nPREFIX variable missing from ${IFCONFIG}, assuming 24."
 PREFIX=24
 args="${args} ${IP}/${PREFIX}"
```

elif [-n «${PREFIX}» -a -n «${PEER}»]; then

```
 log_failure_msg "\nPREFIX and PEER both specified in ${IFCONFIG}, cannot
continue."
 exit 1
```

elif [-n «${PREFIX}»]; then

```
 args="${args} ${IP}/${PREFIX}"
```

elif [-n «${PEER}»]; then

```
 args="${args} ${IP} peer ${PEER}"
```

fi

if [-n «${LABEL}»]; then

```
 args="${args} label ${LABEL}"
```

fi

if [-n «${BROADCAST}»]; then

```
 args="${args} broadcast ${BROADCAST}"
```

fi

case «${2}» in

```
 up)
    if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" = "" ]; then
       log_info_msg "Adding IPv4 address ${IP} to the ${1} interface..."
       ip addr add ${args} dev ${1}
       evaluate_retval
    else
       log_warning_msg "Cannot add IPv4 address ${IP} to ${1}.  Already
present."
    fi
 ;;
```

```
 down)
    if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" != "" ]; then
       log_info_msg "Removing IPv4 address ${IP} from the ${1} interface..."
       ip addr del ${args} dev ${1}
       evaluate_retval
    fi
```

```
    if [ -n "${GATEWAY}" ]; then
       # Only remove the gateway if there are no remaining ipv4 addresses
       if [ "$(ip addr show ${1} 2>/dev/null | grep 'inet ')" != "" ]; then
          log_info_msg "Removing default gateway..."
          ip route del default
          evaluate_retval
       fi
    fi
 ;;
```

-)

echo «Usage: ${0} [interface] {up|down}»

```
    exit 1
 ;;
```

esac

End /lib/services/ipv4-static D.27. /lib/services/ipv4-static-route #!/bin/sh
##
########## # Begin /lib/services/ipv4-static-route # # Description : IPV4 Static Route Script #
Authors : Kevin P. Fleming - kpfleming@linuxfromscratch.org # DJ Lucas - dj AT linuxfromscratch
D0T org # Update : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org # # Version : LFS 7.0 #
##
##########

. /lib/lsb/init-functions . ${IFCONFIG}

case «${TYPE}» in

```
 ("" | "network")
    need_ip=1
    need_gateway=1
 ;;
```

```
 ("default")
    need_gateway=1
    args="${args} default"
    desc="default"
 ;;
```

```
 ("host")
    need_ip=1
 ;;
```

```
 ("unreachable")
    need_ip=1
    args="${args} unreachable"
    desc="unreachable "
 ;;
```

```
 (*)
    log_failure_msg "Unknown route type (${TYPE}) in ${IFCONFIG}, cannot
continue."
    exit 1
 ;;
```

esac

if [-n «${GATEWAY}»]; then

```
 MSG="The GATEWAY variable cannot be set in ${IFCONFIG} for static
routes.\n"
 log_failure_msg "$MSG Use STATIC_GATEWAY only, cannot continue"
 exit 1
```

fi

if [-n «${need_ip}»]; then

```
 if [ -z "${IP}" ]; then
    log_failure_msg "IP variable missing from ${IFCONFIG}, cannot continue."
    exit 1
 fi
```

```
 if [ -z "${PREFIX}" ]; then
    log_failure_msg "PREFIX variable missing from ${IFCONFIG}, cannot
continue."
    exit 1
 fi
```

```
 args="${args} ${IP}/${PREFIX}"
 desc="${desc}${IP}/${PREFIX}"
```

fi

if [-n «${need_gateway}»]; then

```
 if [ -z "${STATIC_GATEWAY}" ]; then
    log_failure_msg "STATIC_GATEWAY variable missing from ${IFCONFIG},
cannot continue."
    exit 1
 fi
 args="${args} via ${STATIC_GATEWAY}"
```

fi

if [-n «${SOURCE}»]; then

```
      args="${args} src ${SOURCE}"
```

fi

case «${2}» in

```
 up)
    log_info_msg "Adding '${desc}' route to the ${1} interface..."
    ip route add ${args} dev ${1}
    evaluate_retval
 ;;
```

```
 down)
    log_info_msg "Removing '${desc}' route from the ${1} interface..."
    ip route del ${args} dev ${1}
    evaluate_retval
 ;;
```

-)

echo «Usage: ${0} [interface] {up|down}»

```
    exit 1
;;
```

esac

End /lib/services/ipv4-static-route

1)

${COLUMNS} - 8

2)

${COL} - 2