```
#!/bin/sh

# script for udhcpc
# Copyright (c) 2008 Natanael Copa <natanael.copa@gmail.com>

UDHCPC="/etc/udhcpc"
UDHCPC_CONF="$UDHCPC/udhcpc.conf"

RESOLV_CONF="/etc/resolv.conf"
[ -f $UDHCPC_CONF ] && . $UDHCPC_CONF

export broadcast
export dns
export domain
export interface
export ip
export mask
export metric
export staticroutes
export router
export subnet

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

run_scripts() {
	local dir=$1
	if [ -d $dir ]; then
		for i in $dir/*; do
			[ -f $i ] && $i
		done
	fi
}

deconfig() {
	ip -4 addr flush dev $interface
}

is_wifi() {
	test -e /sys/class/net/$interface/phy80211
}

if_index() {
	if [ -e  /sys/class/net/$interface/ifindex ]; then
		cat /sys/class/net/$interface/ifindex
	else
		ip -4 link show dev $interface | head -n1 | cut -d: -f1
	fi
}

calc_metric() {
	local base=
	if is_wifi; then
		base=300
	else
		base=200
	fi
	echo $(( $base + $(if_index) ))
}

route_add() {
	local to=$1 gw=$2 num=$3
	# special case for /32 subnets:
	# /32 instructs kernel to always use routing for all outgoing packets
	# (they can never be sent to local subnet - there is no local subnet
```

```
for /32).
      # Used in datacenters, avoids the need for private ip-addresses between
two hops.
      if [ "$subnet" = "255.255.255.255" ]; then
            ip -4 route add $gw dev $interface
      fi
      ip -4 route add $to via $gw dev $interface \
            metric $(( $num + ${IF_METRIC:-$(calc_metric)} ))
}

routes() {
      [ -z "$router" ] && [ -z "$staticroutes" ] && return
      for i in $NO_GATEWAY; do
            [ "$i" = "$interface" ] && return
      done
      while ip -4 route del default via dev $interface 2>/dev/null; do
            :
      done
      local num=0
      # RFC3442:
      # If the DHCP server returns both a Classless Static Routes option
      # and a Router option, the DHCP client MUST ignore the Router option.
      if [ -n "$staticroutes" ]; then
            # static routes format: dest1/mask gw1 ... destn/mask gwn
            set -- $staticroutes
            while [ -n "$1" ] && [ -n "$2" ]; do
                  local dest="$1" gw="$2"
                  if [ "$gw" != "0.0.0.0" ]; then
                        route_add $dest $gw $num && num=$(( $num + 1))
                  fi
                  shift 2
            done
      else
            local gw=
            for gw in $router; do
                  route_add 0.0.0.0/0 $gw $num && num=$(( $num + 1 ))
            done
      fi
}

resolvconf() {
      local i
      [ -n "$IF_PEER_DNS" ] && [ "$IF_PEER_DNS" != "yes" ] && return
      if [ "$RESOLV_CONF" = "no" ] || [ "$RESOLV_CONF" = "NO" ] \
                  || [ -z "$RESOLV_CONF" ] || [ -z "$dns" ]; then
            return
      fi
      for i in $NO_DNS; do
            [ "$i" = "$interface" ] && return
      done
      echo -n > "$RESOLV_CONF.$$"
      if [ -n "$search" ]; then
            echo "search $search" >> "$RESOLV_CONF.$$"
      elif [ -n "$domain" ]; then
            echo "search $domain" >> "$RESOLV_CONF.$$"
      fi
      for i in $dns; do
            echo "nameserver $i" >> "$RESOLV_CONF.$$"
      done
      chmod a+r "$RESOLV_CONF.$$"
      mv "$RESOLV_CONF.$$" "$RESOLV_CONF"
}

bound() {
```

```
	ip -4 addr add $ip/$mask ${broadcast:+broadcast $broadcast} dev $interface
	ip -4 link set dev $interface up
	routes
	resolvconf
}

renew() {
	if ! ip -4 addr show dev $interface | grep $ip/$mask; then
		ip -4 addr flush dev $interface
		ip -4 addr add $ip/$mask ${broadcast:+broadcast $broadcast} dev
$interface
	fi

	local i
	for i in $router; do
		if ! ip -4 route show | grep ^default | grep $i; then
			routes
			break
		fi
	done

	if ! grep "^search $domain"; then
		resolvconf
		return
	fi
	for i in $dns; do
		if ! grep "^nameserver $i"; then
			resolvconf
			return
		fi
	done
}

case "$1" in
	deconfig|renew|bound)
		run_scripts $UDHCPC/pre-$1
		$1
		run_scripts $UDHCPC/post-$1
		;;
	leasefail)
		echo "udhcpc failed to get a DHCP lease" >&2
		;;
	nak)
		echo "udhcpc received DHCP NAK" >&2
		;;
	*)
		echo "Error: this script should be called from udhcpc" >&2
		exit 1
		;;
esac
exit 0
```