I have heavily modified the original install.sh so that it is more useful and flexible. This script is not 100% complete but it is functional and will simplify the initial process of compiling from source. Because it is still in development, I would recommend that you first follow the steps here Compiling DD-WRT Sources to lay out the necessary directory structure and required files. You don't have to use install.sh, instead you can use this script to compile the distribution you want. Before copying my script please read and understand its purpose and limitations.

<<WARNING>> This is still under development. Some features do not work as expected. <<WARNING>>

What this script will allow you to do:

- Compile a single distribution, eg. builddwrt mini
- Compile a combination of distributions, eg. builddwrt micro mini
- Create your build environment from scratch, eg. builddwrt -nd
- Download and update the source using subversion, eg. builddwrt -s

When you use the -d option, the script will automatically generate the correct filename based on the current date, then it will use wget to grab the file. If this attempt fails, the script will automatically try to download the previous day's snapshot. It will attempt to grab only the first snapshot it finds from the last 7 days. If it cannot find a snapshot by this method then you are asked to supply the date of the snapshot you want to grab.

You can name this script whatever you like and the help menu will reflect this so it appears personalized. It can also be placed most anywhere in your local path, but it's best placed directly in your home directory (~). Don't forget to issue chmod x to make this script executable.

You can combine some of the options, for example, builddwrt -ndp2 mini normal voip. This will create the directory structure for your build environment, download the latest snapshot, pause the output for 2 seconds between stages (in case you're watching), then build the image files for mini, normal and voip distributions. If you specify "all", this will expand to "micro mini normal voip openvpn".

What this script does NOT do:

This script only changes the interface for compiling your source, that is, you can consider this as install.sh on steroids. I did not make any attempts to change the actual scripts which perform the compilation, for example, install_mini.sh. Assuming we have the correct directory structure (as per the wiki), if install.sh does not properly compile your source then neither will my script.

Unless I have made a terrible mistake, if the original install.sh works for you then this script will do the same. You just have better control over the initial process.

The usual disclaimer applies here. I wrote this for my own edification and enjoyment and make no guarantees of any kind. I'm sharing this in case someone in this forum has the same interest and curiosity as myself, and may find some uses for it. I'm sure this script is rather trivial to the more experienced and technical individuals visiting this forum, but it works for me and I like it.

Copy the script below and save as builddwrt or something else you prefer.

```
#!/bin/sh

#####################################################
# Custom install.sh script for DD-WRT v23 SP1
# version 0.1b, Apr 20 2006 --jayC
```

```
#
# Based on the original install.sh script and ideas
# from http://forum.bsr-clan.de/ftopic7326.html
#
# License:  GPL2
#####################################################

###################################
## Declaration and Initialization ##
###################################

hr="-----------------------------------------------------------------"
debug=0
## Use hidden option "-G" to enable debug mode for this script.
## No actions will be committed while in debug mode.  This is only
## useful if you intend to modify this script and run some tests.

## Display usage information
usage() {
cat <<MENU
$hr

Compile DD-WRT v23 SP1 with your choice of distribution(s).

Syntax: `basename $0` -[ndfshC] [-p<seconds>] distribution

Distributions:  micro, mini, normal, openvpn, voip, all

Options:
-n                      Prepare a new build environment
-d                      Download latest source from snapshot
-f                      Force overwrite of previous files
-s                      Use subversion to grab latest source
-p <seconds>            Add pause to screen output
-h                      Display this information
-C                      Check build environment

Examples of Proper Use:
                        `basename $0` mini
                        `basename $0` -p 5 micro
                        `basename $0` normal voip
                        `basename $0` -df all
                        `basename $0` -ndp3 openvpn

To troubleshoot your build you can log all output by doing one of
the following examples:

`basename $0` voip | tee build.log    #View and log output at same time
`basename $0` voip > build.log        #No output, just create log file

You can find your log file in the same directory as your script.
$hr
MENU
exit
}

## This is our debug message
debug_script() {
cat <<DEBUG
$hr
Debug Mode (-G) Enabled
No actions will be committed.
```

```
Number of arguments passed to `basename $0`     = $arg
Parameters passed                               = $params
Distribution(s) selected                        = $@
Number of seconds to pause screen output        = $delay
$hr
DEBUG
}

## Initialize variables
arg=$#                          ## number of arguments
params="$@"                     ## parameter list
dist=""
opt="no"
new="no"
dl="no"
force="no"
svn="no"
delay=0
proceed="no"
build_all="micro mini normal openvpn voip"
env="no"
startdir=`pwd`
ddroot="/home/dd-wrt"
snaproot="$ddroot/snapshot"
buildroot="$snaproot/DD-WRT"
site="http://www.dd-wrt.com/dd-wrtv2/downloads/sourcecode"
chainfile="toolchains.x86.debian.sp1.tar.bz2"
toolchain_url="$site/$chainfile"
chainroot="$ddroot/toolchains"
tc3_full="3.4.6-uclibc-0.9.28"
tc3_short="3.4.6"
tc4_full="4.1.0-uclibc-0.9.28"
tc4_short="4.1.0"

## Looks like we need some help
need_help() {
  echo "Please use \"`basename $0 `-h\" to get help."
  exit
}

## Check for proper build environment
check_env() {
  if [ -d $buildroot ]; then
    env="yes"
  fi
}

## Check if any arguments passed
if [ $arg -eq 0 ]; then
  echo -e "\nNo arguments passed.  See below for usage."
  usage
fi

## Parse the options first, exit if unknown
## Using getopts is more efficient than shift alone
while getopts ndfsp:hG option
do
  case $option in
    n) new="yes"
      opt="yes" ;;
    d) dl="yes"
```

```
      opt="yes" ;;
    f) force="yes"
      opt="yes" ;;
    s) svn="yes"
      opt="yes" ;;
    p) delay=$OPTARG
      case $delay in
        0|1|2|3|4|5) opt="yes" ;;
        *) echo -e "\nSyntax error.  -p value must be between 0 - 5."
          need_help ;;
      esac ;;
    h) usage ;;
    G) debug=1
      opt="yes" ;;
    C) ;; ## Check default build environment
    \?) echo -e "\nSyntax error.  Unknown option in \"$params\"."
        need_help ;;
  esac
done

## Process the distributions next by dropping the options
if [ "$opt" = "yes" ]; then
  shift `expr $OPTIND - 1`
fi

## Optional debugging output
if [ $debug -eq 1 ]; then
  debug_script
fi

## Now validate the distributions
## Update these two variables with our new values
params="$@"
arg="$#"

## Exit immediately if distribution is unknown
if [ $arg -eq 1 ]; then
  if [ "$params" = "all" ]; then
    dist="$build_all"
    proceed="yes"
  else
    case $params in
      micro|mini|normal|voip|openvpn)
        dist="$params"
        proceed="yes" ;;
      *) echo -e "\nSyntax error.  Unknown distribution \"$params\"."
        need_help ;;
    esac
  fi
else  ## multiple distributions
  for each in $params; do
    if [ "$each" = "all" ]; then
      echo -e "\nSyntax error in \"$params\".  Cannot use \"all\" this way."
      need_help
    else
      case $each in
        micro|mini|normal|voip|openvpn)
          dist=$dist" $each"
          proceed="yes" ;;
      *) echo -e "\nSyntax error.  Unknown distribution in \"$params\"."
        need_help ;;
      esac
```

```
    fi
  done
fi

## Construct url for previous snapshot
## I haven't fully tested this routine, so it may
## not handle transitioning between months.  In that
## case you will have to enter the date manually for
## the snapshot you want to download.
back_one_day() {
  days=`expr $days + 1`
  monthday=`expr $(date +%d) - $days`
  snapday="$(date +%m)$monthday-$(date +%Y)"
  snapshot="$snapday-snapshot.tar.bz2"
  url=$site"/$snapshot"
}

## Extract snapshot into current location
extract_snapshot() {
  echo -e "\nExtracting files..."
  tar -xjvf $snapshot
}

## Download latest snapshot based on current date,
## and work backwards until we find the most current
## file.  Check only within the last seven days.  You
## can change the number of attempts, if you want to.
download_snapshot() {
  attempts=6
  success="no"
  days=0
  snapday=$(date +%m%d-%Y)
  snapshot="$snapday-snapshot.tar.bz2"
  url="$site/$snapshot"
  cd $snaproot
  while [ "$success" = "no" ] && [ $days -le $attempts ]
  do
    echo -e "\nAttempting to download $url"
    if [ $debug -eq 0 ]; then
      if [ ! -f $snapshot ]; then
        wget $url
      fi
    else
      echo -e "\nGrabbed $url"
    fi
    if [ -f $snapshot ]; then
      echo -e "\nDownload successful."
      success="yes"
    else
      echo -e "\nDownload failed."
      back_one_day
    fi
  done

  ## Can't find a snapshot within the last 7 days
  ## so we need to specify it manually.  There's no sanity
  ## checks here so be careful what you enter.
  if [ $days -gt $attempts ] && [ "$success" = "no" ]; then
    echo -e "\nPlease enter snapshot date (eg. 0420):"
    read snapdate
    if [ "$snapdate" = "" ]; then
      echo -e "Nothing entered.  Exiting."
```

```
      exit
    else
      snapshot="$snapdate$(date +-%Y-)snapshot.tar.bz2"
      url="$site/$snapshot"
      echo -e "\nAttempting to download $url"
      if [ $debug -eq 0 ]; then
        wget $url
      else
        echo -e "\nGrabbed $url"
      fi
      if [ $debug -eq 0 ]; then
        if [ -f $snapshot ]; then
          echo -e "\nDownload successful."
          success="yes"
        else
          echo -e "\nDownload failed.  Exiting."
          exit
        fi
      fi
    fi
  fi
}

## Download and extract toolchains
download_toolchain() {
if [ $debug -eq 0 ]; then
  wget $toolchain_url
  if [ -f $chainfile ]; then
    tar -xjvf $chainfile
    if [ -d $chainroot ]; then
      mv "$chainroot/$tc3_full" "$chainroot/$tc3_short"
      mv "$chainroot/$tc4_full" "$chainroot/$tc4_short"
    else
      echo -e "\nProblems creating $chainroot.  Exiting."
      exit
    fi
  fi
else
  echo -e "\nDebug Output:"
  echo -e "Grabbed \n$toolchain_url"
  echo "tar -xjvf $chainfile"
  echo "mv $chainroot/$tc3_full $chainroot/$tc3_short"
  echo "mv $chainroot/$tc4_full $chainroot/$tc4_short"
fi
}

## Setup our new build environment
setup_build_env() {
    if [ $debug -eq 1 ]; then
      mkdir -p $snaproot-debug
      snapdir=$snaproot-debug
    else
      mkdir -p $snaproot
      snapdir=$snaproot
    fi
    if [ -d $snapdir ] && [ "$dl" = "yes" ]; then
      download_snapshot
      if [ "$success" = "yes" ]; then
        if [ ! -d $buildroot ]; then
          extract_snapshot
        fi
        if [ -d $buildroot ]; then
```

```
          env="yes"
          cd $buildroot/opt
          rm libgcc_s.so
          ln -sf libgcc/libgcc_s.so.1 libgcc_s.so
        else
          echo -e "\nProblems creating $buildroot.  Exiting."
          exit
        fi
      fi
    fi
    cd $ddroot
    imageroot=$ddroot/image
    if [ $debug -eq 0 ]; then
      if [ ! -f $chainfile ]; then
        download_toolchain
      fi
      if [ ! -d $imageroot ]; then
        mkdir $imageroot
        ln -sf $imageroot /GruppenLW
      fi
    else
      echo -e "\nDebug Output:"
      echo "cd $ddroot"
      download_toolchain
      echo "mkdir $imageroot"
      echo "ln -sf $imageroot /GruppenLW"
    fi
}

## Download source using subversion
## to be completed
download_svn() {
  if [ `which svn` ]; then
    echo "Found svn"
  fi
}

## Execute options specified
process_options() {
  if [ "$new" = "yes" ] && [ "$env" = "no" ]; then
    setup_build_env
  fi
}

## Compile the kernel using the 3.4.6 toolchain
## Default version is v23
ver=23
compile_kernel() {
  export MYPATH=$PATH
  export PATH=/home/dd-wrt/toolchains/3.4.6/bin:$MYPATH
  echo "#define BUILD_DATE \"$(date +%D)\"" > build.h

  export SRCBASE=$(cd "../src" && pwd -P)
  echo $SRCBASE

  cd ../src
  cd linux/linux.v$ver
  make oldconfig
  make clean
  make dep
  make
  make modules
```

```
}

## Switch to 4.1.0 toolchain for compiling userspace
prep_userspace() {
  cd ../../
  make clean
  cd ../opt
  export PATH=/home/dd-wrt/toolchains/4.1.0/bin:$MYPATH
}

## Perform clean-up before installing
prep_install() {
  cd ../src/router
  rm -dfr mipsel-uclibc/install
  make httpd-clean
  make rc-clean
  make shared-clean
  cd ../../opt
}

## Mark start of build, in seconds
mark_start() {
  start=`date +%s`
}

## Mark end of build and compute total time taken
mark_stop() {
  stop=`date +%s`
  total=`expr $stop - $start`

  ## If ruby is installed then compute fancy total
  ## otherwise just echo total seconds
  if [ `which ruby` ]; then
    ruby <<CODE
      total = ($total - 3*$delay)
      puts "\n$build build completed in:  #{(total/60).to_s.strip} minute(s), #{(total%60).to_s.s
CODE
  else
    echo -e "\n$build build completed in:  $total seconds."
  fi
}

##########
## Main ##
##########

if [ "$opt" = "yes" ]; then
  process_options
fi

check_env

if [ "$env" = "yes" ]; then
  cd "$buildroot/opt"
else
cat <<NEW

Serious error:
$buildroot does not exist!

Please use -nd options to create the proper directories.
You must be root (or have root privileges) for this step.
```

```
Example:  `basename $0` -ndp2 mini

This will create $buildroot, download and extract
the latest snapshot, add a 2 seconds pause to the output,
and compile the mini image files.

See help:  `basename $0` -h

NEW
  exit
fi

## Okay to proceed?
if [ "$proceed" = "no" ]; then
  echo -e "\nNothing to compile.  Exiting."
  exit
fi

if [ "$params" = "all" ]; then
  distributions=$build_all
else
  distributions=$params
fi

echo -e "\nThe following distribution(s) will be compiled:  $distributions\n"

echo "Started at $(date +%D' - '%T)"

for build in $dist; do
  mark_start
  echo -e "\nWorking on $build distribution..."
  echo -e "\nStarting stage1 (compile kernel)..."
  sleep $delay
  if [ $debug -eq 0 ]; then
    compile_kernel
  fi
  echo -e "\nStarting stage2 (prep userspace)..."
  sleep $delay
  if [ $debug -eq 0 ]; then
    prep_userspace
  fi
  echo -e "\nFinal stage, creating $build image files..."
  sleep $delay
  if [ $debug -eq 0 ]; then
    prep_install
    ./install_$build.sh
  fi
  mark_stop
  sleep $delay
done

echo -e "\nCompleted at $(date +%D' - '%T)"
```