Slackware ARM local mirror repository - SAREPO

It is acknowledged that there are more complete and distinguished Slackware repository solutions readily available. Such as, alienBOB's gen_repos_files.sh and mirror-slackware-current.sh scripts, and Dave Spencer's slackrepo. These are renowned, highly automated, tools for managing Slackware package repositories and mirrors.

Preface

The script featured here, 'local-slackwarearm-repo.sh' [nickname: 'SAREPO'] does not attempt to supplant or undermine the above (or other) contemporary efforts, by *re-inventing the wheel*, so to speak. It's just a much less sophisticated method of achieving the same results, and very much **Slackware ARM specific**.

When I was looking into creating a local Slackware ARM mirror, most of the currently available scripts and software to perform this task was just way too advanced and extensive for my needs with regards to Slackware "*repodemia*". Something much more basic and bespoke was preferable, and that's the main reason why this script was created.

Question & Answer

Q. What is it and what is its purpose?
A. It's bash script called "local-slackwarearm-repo.sh" which downloads Slackware ARM package files from a remote repository (or mirror site) onto your local system and maintains the hierarchial directory/file structure. Thus creating a local mirror repository for whatever purpose users may have for it.

Q. What's the point of this "local-slackwarearm-repo.sh" script when there are much *better* alternatives available and why should I use it in preference to my existing method(s)?
A. Firstly, if your existing methods work well for you then abide by the common-sense dictum, "if it's not broken, don't fix it." Secondly, This script can be set to maintain a local mirror repository for any number of available Slackware ARM ports and versions (and also 'devtools') and was created primarily for its simplicity and for the fun/experience factor. Nobody is saying it's *better* than any existing method(s). It's there and if you have a purpose for it then feel free to use it. 🙂

Q. What does this script do that other alternatives don't?
A. It does nothing more, but is certainly much **less** capable, than gen_repos_files.sh and slackrepo, for example. Plus it's designed to be Slackware ARM specific.

Notes

- The local-slackwarearm-repo.sh script can be used on any Slackware installed system, BUT it is

purposely intended for Slackware ARM installed devices (e.g. Raspberry Pi, Orange Pi, Banana Pi, RockPro64, etc.) - where storage space may be limited and/or only certain Slackware ARM ports-versions are required.

- This script uses arrays with defined elements to download [via rsync] from a remote repository to your local system.
- In order to maintain synchronicity, with a main Slackware ARM repository or mirror, this script can be called in a crontab so that it may be executed periodically to keep a local mirror repository *automagically* up-to-date. It can, of course, also be executed manually on the CLI at the user's discretion.
- This script will create a '/home/$(whoami)/slackwarearm' directory, by default, in which to store any mirror repository data. It will also create a '/home/<username>/bin' directory to store a .database file which contains a list of all the local repository content, and a .log file which records a very basic account of all update (in)activity.
- User/directory names, filenames, and PATHs, can be configured to suit your personal requirements or setups. The defaults are quite generic.

Everyone should be aware that SD cards can fail at the best, and worst, of times and usually without any warning. In an attempt to maintain a high level of reliability I chose to install Slackware ARM on a Raspberry Pi 2 using a Kingston Canvas Go! Plus 64GB UHS-1 (U3) which I have found to be fast, cheap, and unwaveringly dependable over the past few years. When hosting a local Slackware ARM package repository contemplate your choices regarding SD cards (where applicable) because constantly reading and writing data wears them out quickly. If you have any SSDs laying around and/or doing not much, it might be prudent to consider using one of those for repository storage space instead of a SD card.

Requirements

- A computer system running Slackware - bearing in mind that this 'local-slackwarearm-repo.sh' script is designed to deal only with Slackware ARM files.
- Enough free storage space on which to keep the mirror repository files - prior calculations are advised if your intended storage space capacity is sparse.
- ONLY edit the values specified in the “## local-slackwarearm-repo.sh SETTINGS ##” section of the script code - unless you know what you're doing - or else things might break!

local-slackwarearm-repo.sh script code

I prefer to put scripts like this one into my 'home/<username>/bin' directory. You don't have to do the same but, initially (with default settings) this directory will be created by running this script, if it doesn't already exist, to store 'local-slackwarearm-

repo.database' and 'local-slackwarearm-repo.log' files - which are also created at the same time.

- Download the 'local-slackwarearm-repo.sh' file by clicking the link at the top of the code below, or copy and paste the code therein if you prefer.
- Make the file executable by using 'chmod +x local-slackwarearm-repo.sh' or 'chmod 755 local-slackwarearm-repo.sh' command.
- Refer to the commented sections of the script code for usage. These are in the top sections of the script code below.

*** Don't edit anything commented as "!!! _DO_NOT_ Change !!!" or below the "## END OF local-slackwarearm-repo.sh SETTINGS ##" comment unless you know the results to expect from making such modifications!**

local-slackwarearm-repo.sh

```
#!/bin/bash

# Create Slackware ARM local repository mirror utility script.
#
# local-slackwarearm-repo.sh - SAREPO [v2.0.3] -  13 Mar 2021
#
# Copyright (c) 2021 Exaga - SARPi Project - sarpi.penthux.net
#
# Version - 08 Mar 2021 [v0.1a] - progenitor
#         - 10 Mar 2021 [v1]    - associative array mechanics
#         - 12 Mar 2021 [v2]    - indexed array mechanics
#
# ######
#
# This script creates a local Slackware ARM mirror repository of any
# version(s) [i.e. ARM, Aarch64, 15.0, current] which are defined in
# the settings. Only change the settings which suit your own personal
# preferences, unless you really know what you're doing!
#
# This script will create a /home/$(whoami)/slackwarearm directory, by
# default, which to store any repository data. It will also create a
# /home/<username>/bin directory to store a database and logfile which
# contains a list of all the local repository files and used to verify
# (diff) with a remote repository to check if there's any updates. This
# script can also be added to crontab to run periodically.
#
# Put this script anywhere you choose and run like this:
#
# ~$ chmod +x local-slackwarearm-repo.sh
# ~$ ./local-slackwarearm-repo.sh
#
```

```
# It's also possible to run the apache server software on the system
and
# create a symlink to the local Slackware ARM repository so that it can
# be accessed from the browser and/or used as a local mirror for
whatever
# use you may find for it. After setting up and starting the httpd
daemon
# just create a symlink to the repository directory. For example:
#
# ~# ln -sf /home/$(whoami)/slackwarearm /var/www/htdocs/slackwarearm
#
# Then it should be accessible in your browser and can be used as a URL
# during Slackware ARM 'setup' when selecting source media.
#
# ######


### Slackware brand name !!! Explicitly _DO_NOT_ Change !!!
SLACKNAM=(slackware);

# Slackware ARM Project element !!! Explicitly _DO_NOT_ Change !!!
ARMPROJECT=(arm)

# Slackware ARM [array-based] remote repo dir !!! _DO_NOT_ Change !!!
REMOTE_SAREPO_DIR="::${SLACKNAM[@]}${ARMPROJECT[@]}"

######################################################################
##               local-slackwarearm-repo.sh SETTINGS                ##
######################################################################

# -- Edit the settings in this section to suit your requirements -- ##
#
# PRGNAM vars
PRGNAM="$(basename $BASH_SOURCE .sh)"
PETNAM="SAREPO"

# User directory vars
USERDIR="/home/$(whoami)"
SOURCEDIR="${USERDIR}/slackwarearm"
USERBINDIR="${USERDIR}/bin"
LOG_FILE="${USERBINDIR}/${PRGNAM}.log"

# Choose the Slackware ARM version(s) you wish to mirror and enter
# any between the brackets, seperated by a space. Omit any versions
# which you do not want to download. NOTE: It _MUST_ already exist
# on the remote server before you can download it. Obviously!
#
# Slackware ARMVERS elements [ 14.2 | 15.0 | current | devtools ]
ARMVERS=(14.2 current devtools)
```

```
# Same as the above but this is for Slackware AARCH64 versions when
# it is released.
#
# Slackware A64VERS elements [ 15.0 | current ]
A64VERS=(current)

# Specify the URL of ONE remote Slackware repository or mirror site
# WITHOUT any leading "http://" or "ftp://" and WITHOUT a trailing
# forward slash "/".
# e.g. "ftp.arm.slackware.com" or "ftp.halifax.rwth-aachen.de"
# or "mirror.slackbuilds.org" or "slackware.uk"
#
# Remote Slackware ARM repository [ !!! NO ftp:// or trailing "/" !!! ]
SAREPO_URL="slackware.uk"

# Set BANDWIDTH_LIMIT to cap download speed of remote repository data,
# or set a value of "0" [zero] for no limit [Kilobits per second].
#
# BANDWIDTH_LIMIT [Kbps]
BANDWIDTH_LIMIT="0"

##########################################################################
##            END OF local-slackwarearm-repo.sh SETTINGS            ##
##########################################################################
# Halt build process on error
#set -ue
IFS="$(printf '\n\t')"

# Local .database and .lock files
LOCAL_SAREPO_DB="${USERBINDIR}/${PRGNAM}.database"
TMP_DATA_DB="${USERBINDIR}/.${PRGNAM}.TMP"
LOCK_FILE="${TMP_DATA_DB}.lock"

# Delete LOCK_FILE TMP_DATA_DB on error trap EXIT
trap "{ echo 'EXIT ${PIPESTATUS[@]}'; rm -rf ${LOCK_FILE}
${TMP_DATA_DB}; exit; }" INT TERM EXIT

# LOG_FILE function
function log {
  echo "$(date +"%F %T") [$$] $1"
  echo "$(date +"%F %T") [$$] $1" >> "${LOG_FILE}"

}

# Recreate SOURCEDIR USERBINDIR
echo "Starting ${PETNAM} update [PID $$] ..."
log "${PETNAM} : initiating local repository audit"
mkdir -p "${SOURCEDIR}" "${USERBINDIR}"
rm -f "${TMP_DATA_DB}"
touch "${LOCAL_SAREPO_DB}"
```

```
# Create LOCKFILE - exit if it already exists
if [ -f "${LOCK_FILE}" ]; then
  for PID in $(pidof -x "${PRGNAM}.sh"); do
    if [ "${PID}" != "$$" ]; then
        echo "[!] ERROR : ${PRGNAM} : Process is already running with
PID ${PID} ..."
        log "[!] ERROR : ${PRGNAM} : Process is already running with
PID ${PID} ..."
        exit 1
    fi
  done
else
  rm -f "${LOCK_FILE}"
  touch "${LOCK_FILE}"
fi

# Slackware ARM PORT-VERSION array element mechanics function
array_element_mechanics () {
  log "${PETNAM} : validating local repository data ..."
  # Slackware ARM elements
  if [[ "${#ARMVERS[@]}" ]]; then
REMOTE_SAREPO_ARM_PATH="${SAREPO_URL}${REMOTE_SAREPO_DIR}"/"${SLACKNAM[
0]}${ARMPROJECT[0]}"
    for elementarm in "${!ARMVERS[@]}"; do
      log "> [${SAREPO_URL}] ${SLACKNAM[0]}${ARMPROJECT[0]}-
${ARMVERS[$elementarm]}"
      rsync -av --no-motd --contimeout=30 --timeout=60 --delete --
itemize-changes --human-readable \
        --log-file="${LOG_FILE}" --log-file-format="%o %n %'''b" --
bwlimit="${BANDWIDTH_LIMIT}" \
    "${REMOTE_SAREPO_ARM_PATH}-${ARMVERS[$elementarm]}" "${SOURCEDIR}"
|| \
        exit "${PIPESTATUS[@]}"
    done
  fi
  # Slackware AARCH64 elements
  if [[ "${A64VERS[@]}" ]]; then
REMOTE_SAREPO_A64_PATH="${SAREPO_URL}${REMOTE_SAREPO_DIR}"/"${SLACKNAM[
0]}${ARMPROJECT[0]+aarch64}"
    for elementa64 in "${!A64VERS[@]}"; do
      log "> [${SAREPO_URL}] ${SLACKNAM[0]}${ARMPROJECT[0]+aarch64}-
${A64VERS[$elementa64]}"
      rsync -av --no-motd --contimeout=30 --timeout=60 --delete --
itemize-changes --human-readable \
        --log-file="${LOG_FILE}" --log-file-format="%o %n %'''b" --
bwlimit="${BANDWIDTH_LIMIT}" \
    "${REMOTE_SAREPO_A64_PATH}-${A64VERS[$elementa64]}" "${SOURCEDIR}"
|| \
```

```
        exit "${PIPESTATUS[@]}"
    done
  fi
  # Process database file
  build_database

}

# Process LOCAL_SAREPO_DB file
build_database() {
  cd "${SOURCEDIR}"
  echo "${PETNAM} : verifying ${PRGNAM}.database ..."
  find . -type f ! -name "index.html" -print0 | xargs -0 ls -la --time-
style=full >> "${TMP_DATA_DB}"
  cmp -s "${TMP_DATA_DB}" "${LOCAL_SAREPO_DB}" && CMPSTATUS=0 ||
CMPSTATUS=1
  if [[ $CMPSTATUS -eq 0 ]]; then
    log "${PETNAM} : Local repository database is up-to-date ..."
  else
    rm -f "${LOCAL_SAREPO_DB}" && mv "${TMP_DATA_DB}"
"${LOCAL_SAREPO_DB}" && log "${PETNAM} : [!] Local repository database
updated!"
  fi
  log "${PETNAM} : local repository audit complete"
  log "+-----------------------------------------------+" && echo >>
${LOG_FILE}
  echo "${PETNAM} update complete"
  # Done
  exit

}

# run rsync
array_element_mechanics

#EOF<*>
```

Running the script manually

To run the script at any time via the CLI just do the same as you would with any other executable bash script...

```
myuser@slackware:~/bin$ ./local-slackwarearm-repo.sh
```

Or, if you want to know how long the process takes, run the command with the 'time' prefix like this...

```
myuser@slackware:~/bin$ time ./local-slackwarearm-repo.sh
Verifying local repository files ...
```

```
2021-03-08 19:32:32 : local-slackwarearm-repo.sh : Local repository up-to-
date ...

real    8m13.797s
user    4m35.653s
sys     3m36.734s
myuser@slackware:~$
```

You can press the 'CTRL+z' keys, once the “Verifying local repository files ...” message appears, to put the process into the background (and use the 'bg' command) while you carry on with on other things. To bring it to the foreground again use the 'fg' command. Or you can start a new 'screen -S <name>' session and execute it in there (etc.) so you're not waiting around, twiddling your thumbs, while this script process finishes.

Automating with cron

Adding a crontab is almost as easy, but you'll need to know the PATH to the 'local-slackwarearm-repo.sh' script and how to set it to execute at your preferred time interval(s). This is made easy by visiting the crontab guru website and looking at a few examples, and/or learning about the spatial layout and structure (syntax) of crontab if you need to.

Type the 'crontab -e' command on the CLI...

```
myuser@slackware:~/bin$ crontab -e
```

Now, say you want cron to execute this script every day at 8:00AM - if you didn't previously change any of the default script settings, you would enter this into the crontab file:

```
0 8 * * * /home/<username>/bin/local-slackwarearm-repo.sh
```

The crontab works in 24HR format so if you wanted to change 8AM to 8PM [i.e. 20:00 in 24HR format] you would enter this into the crontab file:

```
0 20 * * * /home/<username>/bin/local-slackwarearm-repo.sh
```

If you wanted to have crontab execute the script every 8 hours on the hour you would enter this into the crontab file:

```
0 */8 * * * /home/<username>/bin/local-slackwarearm-repo.sh
```

You will find that crontab covers every eventuality for whenever you want the system to *automagically* execute a command or run a shell script. It can be configured with any interval(s), for differing requirements, and is good for more than just running your

'local-slackwarearm-repo.sh' script periodically.

Local mirror repository log file

The first time the script runs it will create a plain-text 'local-slackwarearm-repo.log' file and record every instance from then on. Each time the script runs this log file will be updated with the result(s). Here's an example of a typical log file entry from a successful update...

```
2021-03-13 15:59:55 [1645] SAREPO : initiating local repository audit
2021-03-13 15:46:45 [17698] > [slackware.uk] slackwarearm-14.2
2021/03/13 15:46:48 [17713] receiving file list
2021/03/13 15:46:49 [17715] sent 2.18K bytes  received 311.62K bytes  69.73K
bytes/sec
2021/03/13 15:46:49 [17715] total size is 3.54G  speedup is 11,286.64
2021-03-13 15:46:49 [17698] > [slackware.uk] slackwarearm-current
2021/03/13 15:46:51 [17718] receiving file list
2021/03/13 15:46:52 [17719] sent 2.25K bytes  received 380.06K bytes
109.23K bytes/sec
2021/03/13 15:46:52 [17719] total size is 3.44G  speedup is 8,988.90
2021-03-13 15:46:52 [17698] > [slackware.uk] slackwarearm-devtools
2021/03/13 15:46:55 [17722] receiving file list
2021/03/13 15:46:55 [17722] slackwarearm-devtools/
2021/03/13 15:46:55 [17722] slackwarearm-devtools/buildcluster/
2021/03/13 15:46:55 [17722] slackwarearm-devtools/buildcluster/distcc/
2021/03/13 15:46:55 [17724] recv slackwarearm-devtools/ 0
2021/03/13 15:46:55 [17724] recv slackwarearm-devtools/buildcluster/ 0
2021/03/13 15:46:55 [17724] recv slackwarearm-
devtools/buildcluster/distcc-3.1-i486-1.txz 261.91K
2021/03/13 15:46:55 [17724] recv slackwarearm-
devtools/buildcluster/distcc-3.1-x86_64-1.txz 269.98K
2021/03/13 15:46:55 [17724] recv slackwarearm-devtools/buildcluster/fstab
112
2021/03/13 15:46:55 [17724] recv slackwarearm-devtools/buildcluster/rc.local
972
2021/03/13 15:46:55 [17724] recv slackwarearm-devtools/buildcluster/sp 119
2021/03/13 15:46:55 [17724] recv slackwarearm-devtools/buildcluster/distcc/
0
2021/03/13 15:46:55 [17724] recv slackwarearm-
devtools/buildcluster/distcc/distcc-3.1.tar.bz2 575.12K
2021/03/13 15:46:55 [17724] recv slackwarearm-
devtools/buildcluster/distcc/distcc.SlackBuild 2.91K
2021/03/13 15:46:55 [17724] recv slackwarearm-
devtools/buildcluster/distcc/slack-desc 980
2021/03/13 15:46:55 [17724] sent 226 bytes  received 1.14M bytes  326.82K
bytes/sec
2021/03/13 15:46:55 [17724] total size is 1.54G  speedup is 1,344.88
2021-03-13 15:46:55 [17698] > [slackware.uk] slackwareaarch64-current
2021/03/13 15:46:58 [17727] receiving file list
```

```
2021/03/13 15:46:58 [17728] sent 72 bytes  received 10.77K bytes  3.10K
bytes/sec
2021/03/13 15:46:58 [17728] total size is 887.48K  speedup is 81.86
2021-03-13 15:55:20 [17698] SAREPO : [!] Local repository database updated!
2021-03-13 15:55:20 [17698] SAREPO : local repository audit complete
2021-03-13 15:55:20 [17698] +----------------------------------------------
+
```

When no new files are found within the local repsitory there's still a log entry to cover that eventuality...

```
2021-03-13 16:12:57 [1645] Local repository database is up-to-date ...
```

Using logrotate to manage log files

With every Slackware [ARM] installation there's tool included called 'logrotate' which is designed to ease the management of many log files generated by the system. For example, if there's any Slackware system that runs a httpd daemon then 'logrotate' will make administration of the log files much easier. So, 'logrotate' can perform an automatic change-over in system log files depending on their filenames, sizes, timestamps, locations (etc.), and many other functions (hint: type '**man logrotate**' from the CLI to find out more). What's more, 'logrotate' can also be configured to manage the 'local-slackwarearm-repo.sh' log file within a normal '/home/<username>/bin' directory or elsewhere. Say if there's a lot of writing to the 'local-slackwarearm-repo.log' and filesize is becoming somewhat too large to read easily, or you're managing the log(s) manually and want to automate the process (etc.), then 'logrotate' will save you a whole lot of time and effort. **To make this work successfully you'll need 'root' access.**

Incidentally, for those who don't have any understanding of the terminology involved, "*rotate*" in this case is just another way of saying, "*copy a currently generated log file to another filename and/or location (while compressing it [optional]) and create a new log file in its place*".

So, by default settings, the 'local-slackwarearm-repo.log' file is located in '/home/<username>/bin/local-slackwarearm-repo.log' and that's what 'logrotate' will be configured to process here. First you need to make a few decisions about where to store the rotated log files, how often to rotate the log(s) and such. The easiest thing to do is to create a new '/home<username/logs' directory for this purpose. We'll do this as a normal user like this...

```
user@slackware:~$ mkdir -p ~/logs
```

Now we'll 'su - root' user in order to create a new configuration for 'logrotate' to process the 'local-slackwarearm-repo.log' file. This configuration file will be placed in the '/etc/logrotate.d/' directory, where all other 'logrotate' configurations are located. The name of our configuration file is 'sarepo'...

```
user@slackware:~$ su - root
root@slackware:~# nano -w /etc/logrotate.d/sarepo
```

While tailored to meet our own specific requirements, here's what the contents of the

'/etc/logrotate.d/sarepo' file looks like...

```
# logrotate /home/<username>/bin/local-slackwarearm-repo.log config
/home/<username>/bin/local-slackwarearm-repo.log {
    su <username> users
    olddir /home/<username>/logs
    missingok
    notifempty
    dateext
    dateformat _%Y%m%d-%H%M%S
    minsize 100k
    rotate 12
    weekly
    compress
    compresscmd /bin/xz
    compressoptions -9
    compressext .xz
    create 644 <username> users
}
```

The function of these configuration settings, along with many more which can be specified, are covered in the 'man logrotate' page, which should be consulted in order to get the best from 'logrotate' and whatever suits your personal requirements.

Some things to note include:
'**<username>**' = the username who owns the 'local-slackwarearm-repo.log' file.
'**su <username> users**' = rotate log files under '<username>' 'group' permissions instead of 'root'.
'**olddir**' = the location [PATH] of the directory where any rotated logs will reside.
'**dateext**' = suffix the date to the resulting rotated filename [default is: date +'%Y-%m-%d'].
'**dateformat**' = here we have specified a custom 'dateext' which also includes the time.
'**rotate 12**' = only keep the past 12 most recent logs - the rest will be deleted [weekly x 12 = 84 days].
'**weekly**' = rotate the logs on a weekly basis (this can be set either daily or monthly too).
'**compress**', '**compresscmd**' = compress the file, using the 'xz' command.
'**compressoptions**', '**compressext**' = compress the file using max '-9' compression, giving it an '.xz' extension.
'**create 644 <username> users**' = create a NEW log file in place of the rotated file under '<username>' 'group' permissions.

Things to do or avoid:
DO read the '**man logrotate**' page to gain some understanding of the different settings and parameters - https://linux.die.net/man/8/logrotate
DON'T set an '**olddir**' PATH (where applicable) if the '**<username>**' does not have access or else 'logrotate' will fail-error.

Save and exit the file once you are happy with the configuration settings within.

Now run 'logrotate' as 'root' user on this specific configuration file and do it like this...

```
root@slackware:~# logrotate -vf /etc/logrotate.d/sarepo
```

After running this 'logrotate' command you will see some output. Ignoring any errors it throws out - unless they are a show-stopper and the log file was not processed. So now we can check on the status of the 'local-slackwarearm-repo.log' file and also the rotated archive in our recently created '/home/<username>/logs' directory...

```
root@slackware:~# ls -lah /home/<username>/bin/local-slackwarearm-repo.log
-rw-r--r-- 1 <username> users 0 Mar 14 15:36 /home/<username>/bin/local-
slackwarearm-repo.log
root@slackware:~# ls -lah /home/<username>/logs/
drwxr-xr-x 2 <username> users 4.0K Mar 14 14:52 ./
drwx--x--x 8 <username> users 4.0K Mar 14 14:52 ../
-rw-r--r-- 1 <username> users  876 Mar 14 15:36 local-slackwarearm-
repo.log_20210314-153618.xz
```

There it is! The '/home/<username>/bin/local-slackwarearm-repo.log' file is 0 bytes size (which means it has been replaced) and the rotated '/home/<username>/logs/local-slackwarearm-repo.log_20210314-152618.xz' file has a timestamp in the filename and .xz extension. Looks like it worked perfectly, as expected.

Hope this shows how useful 'logrotate' really is. 😎

Automated daily logrotate

What's else will happen as a result of running the 'logrotate' command? Well, in the '/etc/cron.daily' directory there is a file...

```
root@slackware:~# ls -lah /etc/cron.daily/logrotate
-rwxr-xr-x 1 root root 129 Jan 18 12:56 /etc/cron.daily/logrotate*
```

The presence of this file means that (once per day) 'logrotate' will be executed by the system and, because we created a '/etc/logrotate.d/sarepo' configuration file, the 'local-slackwarearm-repo.log' will be processed along with all the existing system logs. Due to our configuration settings 'logrotate' will not create new backups daily, but weekly, and only if the 'local-slackwarearm-repo.log' file size is 100Kb or larger. Incidentally, if our log file was becoming large very quickly then we would modify that weekly setting to a daily one. Or ultimately, we could also create a crontab (as 'root' user) that runs the 'logrotate -f /etc/logrotate.d/sarepo' command individually and periodically to suit our requirements.

Local repository .database file

A plain-text 'local-slackwarearm-repo.database' file is initially created when the 'local-slackwarearm-repo.sh' script is first executed, and then maintained as an up-to-date and accurate record of all existent files within the local mirror repository. This database file is verified each time the script is

executed, and regenerated when new files appear, or when current files are supplanted by updates. Until any variation occurs within the local mirror repository content an incumbent 'local-slackwarearm-repo.database' file remains unchanged.

Setting up Apache web server - httpd

As stated previously, the Apache web-server software included with Slackware ARM can easily be configured and started in order to view the mirror repository contents in your web browser. For example, if there's an ARM device running Slackware that's connected to the local network and being used to host a Slackware ARM package repository then viewing the files and directories it contains in a browser (as seen on ftp.arm.slackware.com) is perhaps something that users would be interested in doing. This can be very useful for many reasons, convenient, and just great fun to get working. Plus, with Slackware, it's not difficult at all.

First find out the IP address of the ARM device that the local mirror repository resides on. You need to do this as 'root' user and not a normal user. Use the 'hostname -I' (that's a capital “i”) command like this...

```
root@slackware:~# hostname -I
192.168.1.17
root@slackware:~#
```

That will tell you the IP address of the system. Make a note of it. Your own IP will most likely be different to the example shown here.

Next, use your favourite text editor to open the '/etc/httpd/httpd.conf' file. In this example 'nano' is used to perform this task but 'pico', 'vi', 'vim', 'elvis' (etc.) will work just as well.

```
root@slackware:~# nano -w /etc/httpd/httpd.conf
```

Scroll down the file until you find 'ServerAdmin you@example.com' and edit the email address to your own preference - this is not a precondition and is entirely optional, but a very good practice none-the-less.

Now just below that in the next section you will see this...

```
# If your host doesn't have a registered DNS name, enter its IP address
here.
#
#ServerName www.example.com:80
```

Edit the ServerName as instructed and don't forget to remove the #comment [i.e. get rid of the '#' in front of '#ServerName']. Most users will not have a registered DNS name to use here, so the IP address gained from the 'hostname -I' command should be specified instead. In this example the ServerName will be changed to this...

```
# If your host doesn't have a registered DNS name, enter its IP address
here.
#
```

```
ServerName 192.168.1.17:80
```

Save and exit the file once that's been done. Configuring the Apache web-server further (e.g. for php, htaccess, etc.) is not within the scope of this exercise but, for those who are interested, there are dozens of tutorials and guides around that can further assist you in doing that.

Now create a symlink from the '/var/www/htdocs' directory to the local mirror repository directory in your normal user '/home/<username>/slackwarearm' directory. If the default PATHs in the 'local-slackwarearm-repo.sh' script haven't been modified, this PATH will be *as seen below* with "<username>" being the name of your user. In order to make this work, the essential symlink is created like this...

```
root@slackware:~# ln -sf /home/<username>/slackwarearm
/var/www/htdocs/slackwarearm
```

The penultimate task is to start the httpd daemon. So, make sure the 'rc.httpd' initialization file has been made executable and then run the command to start it...

```
root@slackware:~# chmod +x /etc/rc.d/rc.httpd
root@slackware:~# /etc/rc.d/rc.httpd start
```

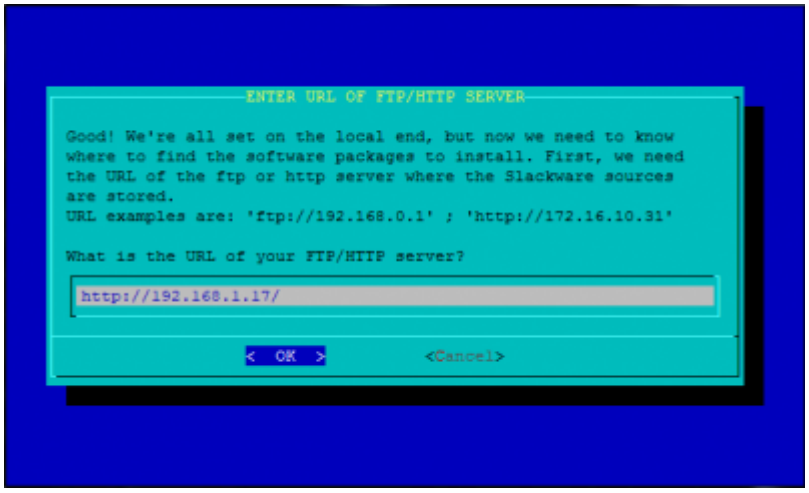
The last and final small step now is to open the browser on another computer system and enter the URL using the IP address gained from the ARM device. So, in this example it's:
http://192.168.1.17/slackwarearm

To access the same content on the ARM device itself you would use this URL:
http://localhost/slackwarearm

Installing Slackware ARM using local network mirror repository source media

Using a local network repository as source media for installing Slackware ARM couldn't be much easier or more simple.

During the Slackware ARM installation 'setup' process, when the "**SELECT SOURCE MEDIA**" titled dialog appears, just choose "**[5] Install from FTP/HTTP server**" from the menu. Then enter the IP address of the system which hosts the local mirror repository (followed by a forward slash "/") in the "What is the URL of your FTP/HTTP server?" input field/box. As shown in the example screenshot below...



As long as the IP address is correct, the Slackware ARM installer will take the source media files from the local mirror repository. This method is generally much faster than downloading over the Internet (NB: not in all circumstances), and saves using the bandwidth of the main repositories and mirrors.

In the "**SELECT SOURCE DIRECTORY**" dialog which follows, the PATH is are no different to if/when using a remote server for Slackware ARM source media. That is, unless the default settings in the 'local-slackwarearm-repo.sh' script have been changed, or stored in a different location - then you would use the PATH to the directory which contains the Slackware ARM source media files.

Using slackpkg on a local network repository

It's possible to configure the 'slackpkg' tool to use a mirror repository on any local network to update a Slackware ARM system. This is easily achieved by editing the '/etc/slackpkg/mirrors' file and entering the IP address of the system which hosts the local mirror repository in the URL - i.e. exactly the same way as you would with a remote repository. Obviously, make sure that you replace the IP address of the example below with your own...

```
#------------------------------------------------------------------
# Slackware ARM current: 32bit armv7, hardware floating point ABI.
#------------------------------------------------------------------
http://192.168.1.17/slackwarearm/slackwarearm-current/
#http://slackware.uk/slackwarearm/slackwarearm-current/
#http://ftp.halifax.rwth-aachen.de/slackwarearm/slackwarearm-current/
#http://ftp.slackware.pl/pub/slackwarearm/slackwarearm-current/
```

Now whenever 'slackpkg update' is run it will refer to the specified URL and get any new packages from the local network repository...

```
Downloading
http://192.168.1.17/slackwarearm/slackwarearm-current/testing/PACKAGES.TXT..
.
--2021-03-10 17:29:40--
http://192.168.1.17/slackwarearm/slackwarearm-current/testing/PACKAGES.TXT
```

```
Connecting to 192.168.1.17:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 224 [text/plain]
Saving to: '/tmp/slackpkg.RfY09f/testing-PACKAGES.TXT'
```

As long as the local mirror repository is updated regularly (e.g. by utilising the 'local-slackwarearm-repo.sh' script) this can/will be a permanently viable solution. If, like me, you are forever (re)installing and testing Slackware ARM then this kind of setup is very convenient and saves on bandwidth. It's a win-win situation all round!

Neat huh? Hope this local mirror repository script and related information has been fun and helpful. Have fun with it! 😎

Help

Any requests for help/assitance, questions, suggestions, good or bad or indifferent feedback, can be addressed on the Slackware ARM LQ forum.

Thank YOU very much for your interest in a Slackware ARM local mirror respository and this SlackDoc page.

Sources

Documentation which assisted in this SlackDoc project:

Trap { $LOCKFILE } EXIT - https://tldp.org/LDP/Bash-Beginners-Guide/html/sect_12_02.html

Thanks to MrJackson for suggesting the idea that gave me the idea which lead to creating this 'local-slackwarearm-repo.sh' script.
Special thanks to Tadgy (https://slackware.uk) whose help and advice greatly facilitated success with httpd.conf and autoindex.conf settings.
Thanks also to Aal (https://fatdog.eu) for the logging function idea/code used in this 'local-slackwarearm-repo.sh' script.

* Originally written by Exaga - 2021-03-08 18:02:17 [GMT]

howtos, slackware, arm, local, mirror, respository, cron, logrotate, httpd, slackpkg, author exaga