Tvheadend - Feature #5741

Patch folder for user patches

2019-10-07 12:08 - saen acro

Status:	New	**Start date:**	2019-10-07
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Patches	**Estimated time:**	0.00 hour
Target version:			

Description

Idea is simple folder named "patches" in main build folder
when there is a "*.patch" or "*.diff" to be applied to source and after compilation finish to be reverted /not to interrupt feature PULL/ aprouvment of **Autobuild.sh --enable-patches**

positives is:
easy implementation of patches
easy testing

This is example of oscam builder.
https://github.com/gorgone/s3_releases

```
#!/bin/bash

get_patch(){ _get_patch; };
_get_patch(){
    cd $pdir;
    [ -f patch.url ] && source patch.url;
    clear;
    echo -e $C;
    ologo;
    echo -e $WH;
    [ -f $PATCHNAME ] && echo -e "$Y   old patch found\n  remove  $P$PATCHNAME" && rm -f
$PATCHNAME;
    echo -e $W"  load $PATCHNAME";
    wget -q -O$PATCHNAME $PATCHURL;
    [ -f $PATCHNAME ] && echo -e $G"   ok$W new $PATCHNAME loaded\n" || echo -e $R
"   patch not found\n"$W;
};
_apply_menupatch(){
    [ -f "$workdir/SVN-IS-PATCHED" ] && quicksvnrestore $_toolchainname 2>/dev/null;
        cd "$pdir";
            (if [ "$(ls -1 "$pdir"/*.patch 2>/dev/null | wc -l)" -gt "0" ];then
                cd "$pdir";
                unset patchlist;patchlist=`ls *.patch | sort -st '/' -k1,1`;
                patchlog="$(mktemp)";
                for e in ${patchlist[@]};do
                    _w="ok";
                    cd "$svndir";
                    echo "PATCH : apply $e";
                    patch -f -p0 < "$pdir/$e" >>"$patchlog" 2>/dev/null;
                    hunks=$(grep -c1 '^Hunk' "$patchlog");
                    fails=$(grep -c1 'hunks FAILED' "$patchlog");
                    if [ "$hunks" -gt "0" ];then
                        echo "PATCH : $hunks x HUNK for $e";
                        _w="hunk";
                    fi;
                    if [ "$fails" -gt "0" ];then
                        echo "PATCH : $fails x FAILS for $e";
                        _w="fail";
                    fi;
                done;
```

```
                case $_w in
                    ok)
                            echo "PATCH : done all ok";
                            touch "$workdir/SVN-IS-PATCHED";;
                    hunk)
                            echo "PATCH : done with warnings save $e.log";
                            cat $patchlog >>"$ldir/$e.log";
                            ln -sf "$ldir/$e.log" "$workdir/lastpatch.log";
                            touch "$workdir/SVN-IS-PATCHED";;
                    fail)
                            echo "PATCH : break build save $e.log";
                            cat $patchlog >>"$ldir/$e.log";
                            ln -sf "$ldir/$e.log" "$workdir/lastpatch.log";
                            touch "$workdir/SVN-IS-PATCHED";;
                esac;
                rm -rf "$patchlog";
            fi;)|"$gui" "$st_" "$bt_" "$title_" "$pb_" 12 62;sleep 2;
};
_apply_consolepatch(){
    if [ ! -f "$workdir/SVN-IS-PATCHED" ];then
        cd "$pdir";
        if [ "$(ls -1 "$pdir"/*.patch 2>/dev/null | wc -l)" -gt "0" ];then
            unset patchlist;patchlist=`ls *.patch | sort -st '/' -k1,1`;
            patchlog="$(mktemp)";
            for e in ${patchlist[@]};do
                _w=0;cd "$svndir";
                echo -e "$y_l |     PATCH : apply $e";
                patch -F 10 -f -p0 < "$pdir/$e" >>"$patchlog" 2>/dev/null;
                hunks=$(grep -c1 "^Hunk" "$patchlog");
                fails=$(grep -c1 "hunks FAILED" "$patchlog");
                if [ "$hunks" -gt "0" ];then
                    echo -e "$y_l |     PATCH :$w_l $hunks x HUNK for $e";
                    _w=1;
                fi;
                if [ "$fails" -gt "0" ];then
                    echo -e  "$y_l |     PATCH :$r_n FAIL (breaking Build) = $fails";
                    echo -en "$w_l |   RESTORE :$c_w LAST SVN BACKUP in ";
                    for (( i=6; i>0; i--));do
                        sleep 1 & echo -en "$i\b";
                        wait;
                    done;
                    svnrestore lastsvn;
                    exit;
                fi;
                cat $patchlog >"$ldir/$e.log";
                ln -sf "$ldir/$e.log" "$workdir/lastpatch.log";
            done;
            rm -rf "$patchlog";
            if [ "$_w" -gt "0" ];then echo -e "$y_l |     PATCH : done with warnings";
                touch "$workdir/SVN-IS-PATCHED";
            else
                echo -e "$y_l |     PATCH :$g_l done all ok$rs_";
                touch "$workdir/SVN-IS-PATCHED";
            fi;
        else
            echo -en "$y_l |     PATCH : no patch found\n";
        fi;
    fi;
};
```

History

#1 - 2019-10-07 20:50 - Joe User

There is no need when it all can be managed by git.

The example you gave is perfect, because with gorgone's s3, I do just the opposite of what you are requesting. Instead of using his patch mechanism, I have cloned the oscam-emu git into the oscam-svn folder and just use "continue with local SVN" and build as normal.

You can either use the tvheadend repository and use the "git stash"/"git pop" commands to manage your local changes.
[[https://git-scm.com/book/en/v1/Git-Tools-Stashing]]

Or you can fork into your own repository and set the official tvheadend repository as upstream. The you can pull upstream and merge into you locally modified branch.

I am far from being an expert on git, but can try to answer any questions you have.

#2 - 2019-10-07 21:21 - Joe User

Sorry, should be "git stash"/"git stash pop|apply".
More info here: https://git-scm.com/docs/git-stash

#3 - 2019-10-08 07:54 - saen acro

@ Joe User You totally not understand what is the idea of request.

#4 - 2019-10-08 08:05 - saen acro

Joe User wrote:

> Sorry, should be "git stash"/"git stash pop|apply".
> More info here: https://git-scm.com/docs/git-stash

If user have downloaded archive what git commands?
https://github.com/tvheadend/tvheadend/archive/master.zip

#5 - 2019-10-08 16:30 - Em Smith

I'm not sure exactly what you're trying to do, but you wouldn't download the zip file. Instead you would:

```
git clone https://github.com/tvheadend/tvheadend.git
```

This would create a "tvheadend" directory.

Then you apply whatever diffs you want. Just commit your patches (locally) with "git commit -a". When you want to update your repository, issue a "git pull" to pull any new changes and it will automatically merge any new features in with your already locally applied patch.

Or, if you don't want to commit locally, then just apply the patches and before updating just checkout the files and re-patch afterwards with a "git reset --hard HEAD && git pull"

#6 - 2019-10-08 19:49 - saen acro

Em Smith wrote:

> I'm not sure exactly what you're trying to do, but you wouldn't download the zip file. Instead you would:
> [...]
> This would create a "tvheadend" directory.
>
> Then you apply whatever diffs you want. Just commit your patches (locally) with "git commit -a". When you want to update your repository, issue a "git pull" to pull any new changes and it will automatically merge any new features in with your already locally applied patch.
>
> Or, if you don't want to commit locally, then just apply the patches and before updating just checkout the files and re-patch afterwards with a "git reset --hard HEAD && git pull"

Exactly this I do not want to do.
Just want to have **patches** folder
To put all moded stuff there.
To start compilation, patches te applyed, after compilation to be reverted
To git ignore this folder, and not to have any other conflicts with main git
100% offline.

#7 - 2019-10-08 20:03 - Luis Alves

I also don't see the point in this request...
You can easily do this with git assuming you're familiar with (mainly) branching, rebasing, cherry-picking.

#8 - 2019-10-08 21:10 - saen acro

I don't want to use git.

#9 - 2019-10-09 10:16 - Joe User

Don't be afraid - git is your friend! smile.png

The problem with patches is they become unusable after time as the source code changes. By using git, the "patch" is essentially updated after each incremental change to the source so the change has a better chance of surviving for longer. Of course direct changes to the patched code will cause failure for both methods, but I find it easier to resolve git merge problems than patch rejects.

That is one of the reasons I changed oscam-emu to be a git repo with a fully patched source instead of a patch and use it instead of gorgone's s3 patch system. (I use s3 mainly for the ease of doing cross compile builds...)

And in the long run, you have a much better chance of people updating their pull requests than fixing an old patch that was posted to the website for a quick, experimental test...

#10 - 2019-10-09 13:09 - saen acro

Install GIT on sh4 architecture or crosscompile it.
Solution is usable but not implemented.
Simple solution is better then git.

#11 - 2019-10-09 16:40 - Joe User

sh4? Hopefully you are not compiling on an sh4 box - that would take fooooorrrreeeeeeevvvvveeeeeerrrr! wink.png

Maybe you can describe better your entire build process.

I tested tvheadend only briefly on my sh4 box a few years ago. But, I may start using in in the near future.

#12 - 2019-10-09 18:15 - saen acro

Joe User wrote:

> sh4? Hopefully you are not compiling on an sh4 box - that would take fooooorrrreeeeeeevvvvveeeeeerrrr! wink.png
>
> Maybe you can describe better your entire build process.
>
> I tested tvheadend only briefly on my sh4 box a few years ago. But, I may start using in in the near future.

https://github.com/Audioniek/buildsystem

#13 - 2019-10-10 09:02 - Joe User

I used his git also when I built tvheadend a long time ago. I recently used his new bitbake build for enigma2 but haven't had much time to test it. I did not notice if he include tvheadend as a build option with bitbake, but there are recipes for it for other systems (librelec) that I built.

In "buildsystem/make/tvheadend.mk", you can modify the line:

```
REPO_TVHE="https://github.com/tvheadend/tvheadend.git"
```

and point to your forked tvheadend git. Then you can modify your own version as you like using git tools.

I did this when I was testing my extended_cw changes I made before Jaroslav made a real implementation. smile.png

As a side note, I run my sh4 box (alien2) from an nfsmount (using tftp boot and nfsroot) so I never have to flash my box. It is very convenient for testing. I have 10-15 images extracted on my nfs server and just need to change a soft link to point to the image I want to boot/run from. It also makes it much easier for modifying files since my nfs server is also the machine I use for building.

Also, I only tested tvheadend with FTA, and would be interested to know if use are able to use the hardware descramblers with tvheadend or you use software?

#14 - 2019-10-10 09:13 - saen acro

Joe User wrote:

> In "buildsystem/make/tvheadend.mk", you can modify the line:
> [...]

https://github.com/Audioniek/buildsystem/tree/master/Patches

Same scenario as this with I offer to be implemented.
How many branches need to have if want to make x86, x86_64, arm, mips. sh4