final

Abdul Rahim

24 July 2024

some useful scripts to install:

stable_rc_checkout.sh

```
1        #!/bin/bash
2        ## SPDX-License-Identifier: GPL-2.0
3        # Copyright(c) Shuah Khan <
             skhan@linuxfoundation.org>
4       #
5        # LICENSE: GPLv2
6        # Example usage: stable_rc_checkout.sh <
             stable-rc e.g 5.2>
7        mkdir -p stable_rc
8        cd stable_rc
9        git clone git://git.kernel.org/pub/scm/
             linux/kernel/git/stable/linux-stable-
             rc.git
```

linux-$1.y

```
1       cd linux-$1.y
2       #cp /boot/<currentconfig> .config #
            update script
3       make -j2 all
4       rc=$?; if [[ $rc !=0 ]]; then exit $rc;
            fi
5       su -c "make modules_install install"
6       echo Ready for reboot test of Linux-$1
```

Download and applying stable release patches

Alternately, you can download and apply the patch. The following is my workflow for getting the repository ready, applying the patch, compiling, and installing. Run the `stable_checkout.sh` script once to set up your stable repository. After that, run `pre_compile_setup.sh` to get the patch file and apply whenever a stable release patch is released. I apply patches and use the same repository to be able to detect regressions. I save dmesg for the current rc to compare with the next rc. Please feel free to make changes to suit your needs. Also, make sure to pass in the correct release information from the stable release emails as arguments to this script.

stable_checkout.sh

```
#!/bin/bash
## SPDX-License-Identifier: GPL-2.0
# Copyright(c) Shuah Khan <skhan@linuxfoundation.org>
#
# License: GPLv2
# Example usage: stable_checkout.sh <stable-release-version e.g 5.2>
mkdir -p stable
cd stable
git clone git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git linux_$1_stable
cd linux_$1_stable
git checkout linux-$1.y
#cp /boot/ .config # update script
```

pre_compile_setup.sh

```
1  #!/bin/bash
2  ## SPDX-License-Identifier: GPL-2.0
3  # Copyright(c) Shuah Khan <
       skhan@linuxfoundation.org>
4  #
5  # License: GPLv2
6  # Example usage: pre_compile_setup.sh 5.2.11 1
       5
7  # Arg 1 is the stable release version which is
       typically 5.2.x
8  # Arg2 is the 1 for rc1 or 2 for rc2
9  # Arg3 is 4.x or 5.x used to call wget to get
       the patch file
```

```
1  echo Testing patch-$1-rc$2
2  wget https://www.kernel.org/pub/linux/kernel/
       v$3.x/stable-review/patch-$1-rc$2.gz
3  git reset --hard
4  make clean
5  git pull
6  gunzip patch-$1-rc$2.gz
7  git apply --index patch-$1-rc$2
8  echo "Patch-$1-rc$2 applied"
9  head Makefile
10 make -j2 all
11 rc=$?; if [[ $rc != 0 ]]; then exit $rc; fi
12 su -c "make modules_install install"
13 echo Ready for reboot test of Linux-$1-$2
```

dmesg_checks.shi

```
1  # !/bin/bash
2  #
3  #SPDX-License-Identifier: GPL-2.0
4  # Copyright(c) Shuah Khan <
       skhan@linuxfoundation.org>
5  #
6  # License: GPLv2
7
8        if [ "$1" == "" ]; then
9           echo "$0 " <old name -r>
10          exit -1
11 fi
```

```
release=`uname -r`
echo "Start dmesg regression check for
    $release" > dmesg_checks_results

echo "----------------------------" >>
    dmesg_checks_results

dmesg -t -l emerg > $release.dmesg_emerg
echo "dmesg emergency regressions" >>
    dmesg_checks_results
echo "----------------------------" >>
    dmesg_checks_results
diff $1.dmesg_emerg $release.dmesg_emerg >>
    dmesg_checks_results
echo "----------------------------" >>
    dmesg_checks_results
```

```
1  dmesg -t -l crit > $release.dmesg_crit
2  echo "dmesg critical regressions" >>
       dmesg_checks_results
3  echo "----------------------------" >>
       dmesg_checks_results
4  diff $1.dmesg_crit $release.dmesg_crit >>
       dmesg_checks_results
5  echo "----------------------------" >>
       dmesg_checks_results
6
7  dmesg -t -l alert > $release.dmesg_alert
8  echo "dmesg alert regressions" >>
       dmesg_checks_results
9  echo "----------------------------" >>
       dmesg_checks_results
10 diff $1.dmesg_alert $release.dmesg_alert >>
       dmesg_checks_results
11 echo "----------------------------" >>
       dmesg_checks_results
```

```
dmesg -t -l err > $release.dmesg_err
echo "dmesg err regressions" >>
    dmesg_checks_results
echo "----------------------------" >>
    dmesg_checks_results
diff $1.dmesg_err $release.dmesg_err >>
    dmesg_checks_results
echo "----------------------------" >>
    dmesg_checks_results

dmesg -t -l warn > $release.dmesg_warn
echo "dmesg warn regressions" >>
    dmesg_checks_results
echo "----------------------------" >>
    dmesg_checks_results
diff $1.dmesg_warn $release.dmesg_warn >>
    dmesg_checks_results
echo "----------------------------" >>
    dmesg_checks_results
```

```
1  dmesg -t > $release.dmesg
2  echo "dmesg regressions" >>
       dmesg_checks_results
3  echo "----------------------------" >>
       dmesg_checks_results
4  diff $1.dmesg $release.dmesg >>
       dmesg_checks_results
5  echo "----------------------------" >>
       dmesg_checks_results
```

```
dmesg -t > $release.dmesg_kern
echo "dmesg_kern regressions" >>
    dmesg_checks_results
echo "----------------------------" >>
    dmesg_checks_results
diff $1.dmesg_kern $release.dmesg_kern >>
    dmesg_checks_results
echo "----------------------------" >>
    dmesg_checks_results

echo "----------------------------" >>
    dmesg_checks_results

echo "End dmesg regression check for $release"
     >> dmesg_checks_results
```