<html><head></head><body><pre style="word-wrap: break-word; white-space: pre-wrap;">
The perfect patch.
akpm@osdl.org
Updated 4 March 2006

The latest version of this document may be found at
http://www.zip.com.au/~akpm/linux/patches/stuff/tpp.txt

Please also see http://linux.yyz.us/patch-format.html

1: Delivery
===========

a) Patches are delivered via email only. Downloading them from internet
   servers is a pain.

b) One patch per email, with the changelog in the body of the email.

2: Subject:
===========

a) The email's Subject: should concisely describe the patch which that
   email contains. The Subject: should not be a filename. Use different
   Subject:s for each patch in a patch series.

   Bear in mind that the Subject: of your email becomes a globally-unique
   identifier for that patch. It propagates all the way into the kernel
   revision control system. The Subject: may later be used in developer
   discussions which refer to the patch. People will want to google for the
   patch's Subject: to read discussion regarding that patch.

b) It is nice if the Subject: includes mention of the subsystem which it
   affects. See example below.

c) Example Subject:

        [patch 2/5] ext2: improve scalability of bitmap searching

d) Note that various people's patch receiving scripts will strip away
   Subject: text which is inside brackets []. So you should place information
   which has no long-term relevance to the patch inside brackets. This
   includes the word "patch" and any sequence numbering. The subsystem
   identifier ("ext2:") should hence be outside brackets.

3: Attribution
==============

If someone else wrote the patch, they should be credited (and blamed) for it.
To communicate this, add a line:

From: John Doe <jdoe@wherever.com>

as the very first line of the email. Downstream tools will pick this up and
jdoe will get the git "Author" line.

4: Changelog
============

a) Bear in mind that the Subject: and changelog which you provide will
   propagate all the way into the permanent kernel record. Other developers
   will want to read and understand your patch and changelog years in the
   future.

   So the changelog should describe the patch fully:

   - why the kernel needed patching

   - the overall design approach in the patch

   - implementation details

- testing results

b) Don't bother mentioning what version of the kernel the patch applies to ("applies to 2.6.8-rc1"). This is not interesting information - once the patch is in the kernel.org kernel, of _course_ it applied, and it'll probably be merged into a later kernel than the one which you wrote it for.

c) Don't include text such as "this patch does ..." or "here is patch which ...". There's no point in telling us that it's a patch - once it's in the kernel SCM we _know_ it's a patch!

d) Do not refer to earlier patches when changelogging a new version of a patch. It's not very useful to have a final changelog which says "OK, this fixes the things you mentioned yesterday". Each iteration of the patch should contain a standalone changelog. This implies that you need a patch management system which maintains changelogs. See below.

e) Add a Signed-off-by: line, as per section 11 of the Documentation/SubmittingPatches file in the kernel tree.

Signed-off-by: implies that you had some part in the developent of the patch, or that you handled it and passed it on to another developer for merging.

If your role was purely review and/or testing, please use Acked-by: instead.

Sometimes we also use Cc: in a patch to keep particular individuals abreast of the patch's status and to give them an opportunity to review it.

f) Most people's patch receiving scripts will treat a ^--- string as the separator between the changelog and the patch itself. You can use this to ensure that any diffstat information is discarded when the patch is applied:

```
Another few #if/#ifdef cleanups, this time for the PPC architecture.

Signed-off-by: &lt;valdis.kletnieks@vt.edu&gt;
Signed-off-by: Andrew Morton &lt;akpm@osdl.org&gt;
---

 25-akpm/arch/ppc/kernel/process.c                       |    2 +-
 25-akpm/arch/ppc/platforms/85xx/mpc85xx_cds_common.c    |    2 +-
 25-akpm/arch/ppc/syslib/ppc85xx_setup.c                 |    4 ++--
 3 files changed, 4 insertions(+), 4 deletions(-)

--- 25/arch/ppc/kernel/process.c
+++ 25/arch/ppc/kernel/process.c
@@ -667,7 +667,7 @@ void show_stack(struct task_struct *tsk,
```

g) Non-changelog text:

Sometimes you want to include text in the email which isn't designed to go into the changelog in the git repository. Things like "this is already in Fred's tree" or "this is an updated version of last Friday's patch" or whatever.

You should place such text below the "^---" separator so that it is auto-removed when being placed into the revision control system.

5: The diff

a) Patches should be in `patch -p1' form:

```
--- a/kernel/sched.c
+++ b/kernel/sched.c
```

b) Make sure that your patches apply to the latest version of the kernel tree. Either straight from Linus's git tree or from ftp://ftp.kernel.org/pub/linux/kernel/v2.6/snapshots/

c) When raising patches for -mm kernels it's generally best to base them on
   the latest Linus tree. I'll work out any rejects/incompatibilities. There
   are of course exceptions to this.

d) Ensure that your patch does not add new trailing whitespace. The below
   script will fix up your patch by stripping off such whitespace.

```
#!/bin/sh

strip1()
{
        TMP=$(mktemp /tmp/XXXXXX)
        cp "$1" "$TMP"
        sed -e '/^+/s/[         ]*$//' &lt;"$TMP" &gt;"$1"
        rm "$TMP"
}

for i in "$@"
do
        strip1 "$i"
done
```

6: Patch series

a) When sending a series of patches, number them in the Subject:s thusly:

```
[patch 1/10] ext2: block allocation: frob the globnozzle
[patch 2/10] ext2: block allocation: wash the pizza
etc.
```

b) Some people like to introduce a patch series with an introductory email
   which doesn't actually carry a patch, such as:

```
[patch 0/10] ext2: block allocation changes
```

   Please don't do this. There is no facility in the git tree to carry
   changelog-only changesets such as this (or at least, we don't do that) so
   the information in the introductory email will be lost.

   So I end up copying and pasting your nice introduction into the
   changelog for the first patch, so it gets into git. I'll follow it with
   the text

```
    This patch:
```

   and then I'll include the changelog for the first patch of the series.

   It would be preferred if the patch originators were to do this.

c) Try very hard to ensure that the kernel builds and runs correctly at
   every step of the patch series. This requirement exists because of
   `git-bisect'. If someone is doing a bisection search for a kernel bug and
   they land upon your won't-compile point partway through the exercise, they
   will be unhappy.

d) If your patch series includes non-runtime-affecting things such as
   cleanups, whitespace fixes, file renames, moving functions around, etc. then
   this work should be done in the initial patches in the series. The
   functional changes should come later in the series.

   This is mainly so that reversion of problematic changes becomes simpler.

7: Overall

a) Avoid MIME and attachements if possible. Make sure that your email
   client does not wordwrap your patch. Make sure that your email client does
   not replace tabs with spaces.

   Mail yourself a decent-sized patch and check that it still applies.

b) If all this still hopelessly fails then MIME attachments are an acceptable
   reluctant fallback. You must ensure that the attached patches have

text/plain mimetypes.

The patch management scripts at http://www.zip.com.au/~akpm/linux/patches/
implement all of the above.

The patch management tools at https://savannah.nongnu.org/projects/quilt/ also
implement all of the above.

Many example patches may be found under
ftp://ftp.kernel.org/pub/linux/kernel/people/akpm/patches/2.6/
</pre><embed id="xunlei_com_thunder_helper_plugin_d462f475-c18e-46be-bd10-327458d045bd"
type="application/thunder_download_plugin" height="0" width="0"></body></html>