1001 bugs – or: the golden rules of bad programming

Christian Boltz
openSUSE beta tester, PostfixAdmin developer,
battle-hardened AppArmorer, …
and: BBfH

Never use any libraries or existing functions

Re-inventing the wheel is fun!

```
function myprint ($text) {
    $handle = fopen("/dev/stdout");
    fput($handle, $text);
}
```

Photo: http://www.flickr.com/photos/vrogy/514733529/

Handle special values in a special way

looks like you have some special code in yast for password "x", maybe I should use the even more secure new password "y" in the future ?! ;-)

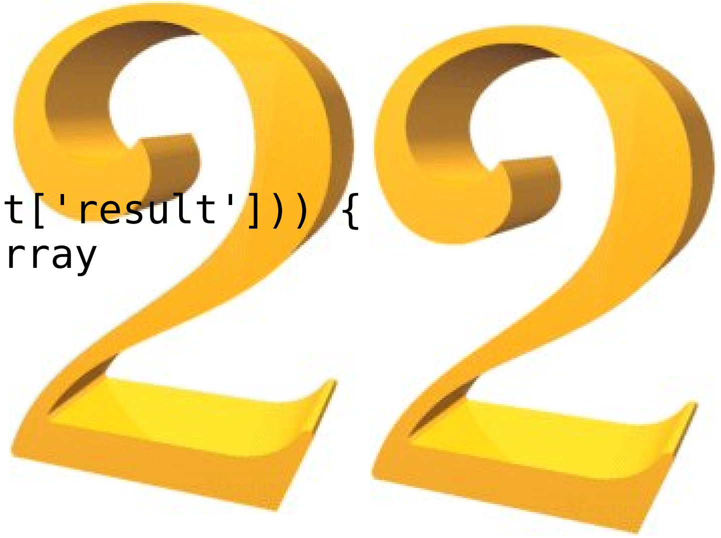
[Harald Koenig, bnc#148464]

Invent new ways to make your program slow

```
while ( $current < $list['alias_count'] ) {
    $query = "SELECT $table_alias.address FROM $table_alias
        [...] LIMIT $current, 1";
    $result = db_query ("$query");
    $row = db_array ($result['result']);
    $tmpstr = $row['address'];
    $idxlabel = $tmpstr[0] . $tmpstr[1]; // first two chars
    $current = $current + $page_size;
    $pagebrowser[]=$idxlabel;
}
```

Invent new ways to make your program slow

```
$initcount = "SET @row=-1";
$result = db_query($initcount);
# get labels for relevant rows (first and last of each page)
$page_size_zerobase = $page_size - 1;
$query = "
    SELECT * FROM (
        SELECT $idxfield AS label,
        @row := @row + 1 AS row $querypart
    ) idx WHERE MOD(idx.row, $page_size)
    IN (0,$page_size_zerobase) OR idx.row = $count_results
";
$result = db_query ($query);
if ($result['rows'] > 0) {
    while ($row = db_array ($result['result'])) {
        # store all labels in an array
    }
}
```



Users hate error messages

Conclusion: never print an error message.

Fail silently instead.

openSUSE developers seem to love this rule:

- no error message when RPM database is missing (bnc#148105)
- rcxdm doesn't start X in failsafe mode if xorg.conf.install was deleted (bnc#394316)

kweather not installed – and now?

> Hmm, what about looking out of the window?

Outside of the window is another window, the desktop background or the border of the monitor. How exactly would that help?

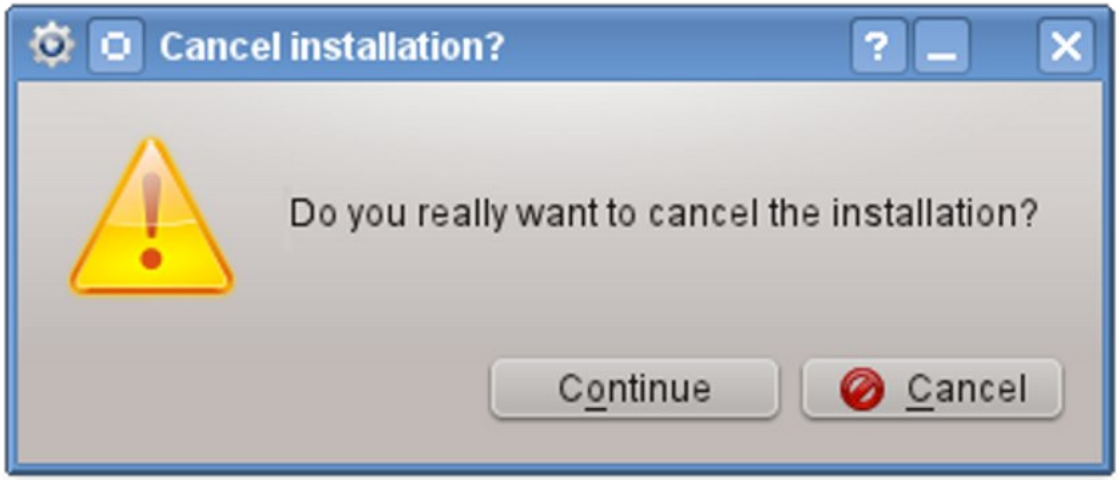
[full story: bnc#141107]

Never drop any privileges

... you might need them later again

- aa-notify broken on 11.4 because of missing permissions
- might apply to real world also

Make the UI easy to understand



Do you really want to cancel the installation?

[bnc#21867]

Make the UI easy to understand

vi commands are quite easy to remember.

Once you know what `dw` `db` `de` `d)` `d(` `d}` `d{` `dd` `d^` `d$` `d0` `dG` as well as `cw` and `yw` do, you'll also know what `cb` `ce` `c)` `c(` `c}` `c{` `cc` `c^` `c$` `c0` `cG` and `yb` `ye` `y)` `y(` `y}` `y{` `yy` `y^` `y$` `y0` `yG` do.

[Bernd Bordesser in suse-linux, translated]

Take great care in setting bad defaults

```
zypper ar obs://home:cboltz home:cboltz
```

always adds the Factory repo instead of the repo for the installed version.

But there is a config option to hardcode the distribution in zypper.conf (and to have fun after upgrading the distribution)

(/etc/SuSE-release anyone?)

[bnc#648892]
[wontfix, ENOTIME]

It's enough to expect the usual data

(cron.daily doesn't run because the system is on battery)

Yes Karl, your machine has a battery! But no ac adapter :-)

Unfortunately it is the battery in the BT Mouse and it won't be able to power your system for very long :-)

[Stefan Seyfried, bnc#221999]

It's enough to expect the usual data

```
#define IM_TEXT_LEN      32
char numstr[20];
char str[IM_TEXT_LEN];
sprintf(numstr, "%%2d%%%% %%.%ds",
    IM_TEXT_LEN-6);


sprintf(str, numstr,
    (int)(percent * 100),
    graph->pairs[i]->name);
```

[modlogan, bnc#517602]

It's enough to expect the usual data

```
#define IM_TEXT_LEN      32
char numstr[20];
char str[IM_TEXT_LEN];
sprintf(numstr, "%%2d%%%% %%.%ds",
    IM_TEXT_LEN-6);
# numstr = "%2d%% %.27s"
sprintf(str, numstr,
    (int)(percent * 100),
    graph->pairs[i]->name);
```

[modlogan, bnc#517602]

Never make your code reusable

... especially if it has more than 1000 lines

Reusing pieces of code is like picking off sentences from other people's stories and trying to make a magazine article. [Bob Frankston]

Besides that:

Nobody needs that much code a second time ;-)

Make ALL your code reuseable

(including each and every little script you write)

bad:

```
    echo "Hello World!";
```

Make ALL your code reuseable

good:

```
    Class HelloWorld {
        my $greeting = "Hello World!";
        function setGreet($newGreeting) {
            $greeting = $newGreeting;
        }
        function greet() {
            echo $greeting;
        }
    }
    $greeter = new HelloWorld;
    # default text is fine
    $greeter->greet;
```

Make ALL your code reuseable

This will save you lots of work in the future.

You can then just do:

```
    $greeter = new HelloWorld;
    $greeter->setGreet("Hello openSUSE!");
    $greeter->greet;
```

Make ALL your code reuseable

This will save you lots of work in the future.

You can then just do:

```
    $greeter = new HelloWorld;
    $greeter->setGreet("Hello openSUSE!");
    $greeter->greet;
```

That's easier than

```
    echo "Hello openSUSE!";
```

Don't use small if blocks. Use cp instead.

```
cp edit-mailbox.php admin/edit-mailbox.php
vi admin/edit-mailbox.php # remove permission checks
diff edit-mailbox.php admin/edit-mailbox.php | wc -l
   15
... 5 years later ...
diff edit-mailbox.php admin/edit-mailbox.php | wc -l
   250
```

Don't use small if blocks. Use cp instead.

Boring:

```
[Unit]
Description=Daemon to detect crashing apps
After=syslog.target

[Service]
ExecStart=/usr/sbin/abrtd
Type=forking

[Install]
WantedBy=multi-user.target
```

Don't use small if blocks. Use cp instead.

Also boring:

CPAN_service in the buildservice

That would make it too obvious that you are a lazybone as packager and just use a template-based specfile.

It's much more maintenance fun to checkin the cpanspec-generated specfiles.

Always trust your users
or: never check user input

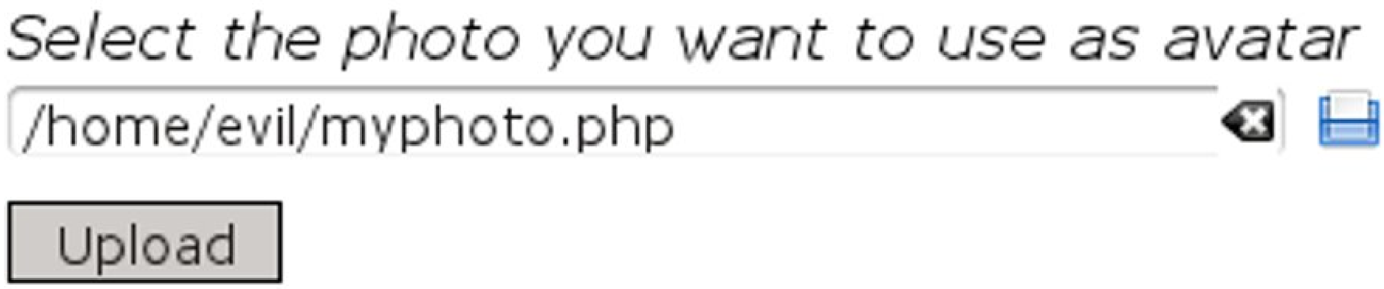
Even AppArmor followed this rule, so it can't be too wrong ;-)

```
echo 'AAA AAA' > /proc/$$/attr/current
Segmentation fault
```

```
[107353.169142] kernel BUG at kernel BUG at
/usr/src/packages/BUILD/kernel-desktop-
2.6.37.6/linux-2.6.37/security/apparmor/audit.c:183!
[107353.169159] invalid opcode: 0000 [#7] SMP
[...]
```

[https://bugs.launchpad.net/bugs/789409]

Always trust your users
or: never check user input



Select the photo you want to use as avatar

/home/evil/myphoto.php

Upload

http://server/wbb/images/avatars/myphoto.php
will give the admin some fun...

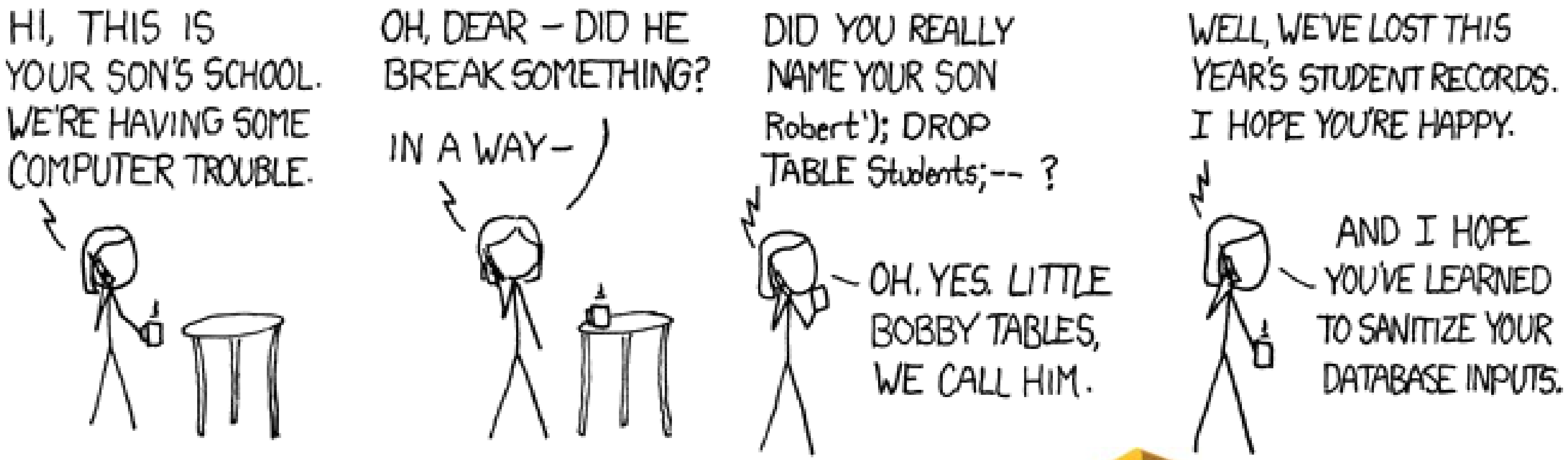
[Woltlab burning board 1.0.2]

Always trust your users
or: never check user input

The input you may expect will be completely unrelated to the input given by a disgruntled employee, a cracker with months of time on their hands, or a housecat walking across the keyboard.

[http://www.php.net/manual/en/security.general.php]

Always trust your users
or: never check user input


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

[http://xkcd.com/327/]

Always check for errors

Especially if you write a library.

It isn't trivial to write correct code:

```
    exit(-1);
    syslog(LOG_ERROR, “Can't exit.\n”);
```

[Lutz Donnerhacke in dclp, translated]

Never think about error handling

Just expect your code to work.

- error handling is only boring programming work, it's much more exciting to fix bugs after deploying the software on production-critical systems
- better invest your time in developing new features
- thinking about errors is the job of bugreporters anyway

Bugzilla speed with Coolo

> Status?

NEW

[Ihno Krumreich and Stephan Kulow, bnc#159223]

> which camera is this?

Marcus, this is my bug :)

[Marcus Meissner and Stephan Kulow, bnc#217731]

Never expect someone will use your software

Hardcoding your username is fine.

> mkdir: Can't create directory »»/home/ratti««
> Looks like you should use ~ instead of
> /home/ratti ;-)

Wait and see – 0.0.3 will even work if your name is not “ratti” ;-)

[Fontlinge development fun with Ratti, translated]

Never expect someone will use your software

Never remove any debugging code
(at least until someone forces you to do it)

mcelog should NOT email trenn@suse.de by default

[bnc#713562]

Never expect someone will use your software

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

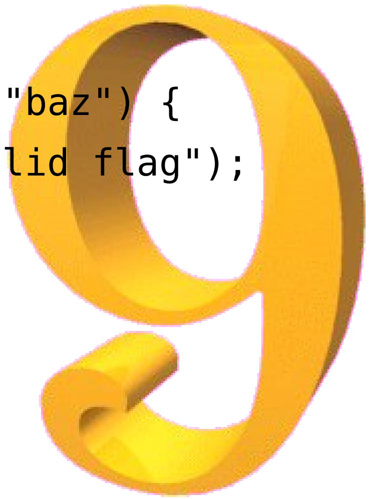
[Linus Torwalds, August 1991]

Nest your code as deep as possible

```
if ($name == “foo”) {
    if ($value == “bar”) {
        if ($number > 0) {
            if ($flags == "baz") {
                # 1000 lines of code
            } else {
                die ("invalid flag");
            }
        } else {
            die ("invalid number");
        }
    } else {
        die ("invalid value");
    }
} else {
    die ("invalid name");
}
```

Never do something like this:

```
if ($name != “foo”) {
    die("invalid name");
}
if ($value != "bar") {
    die ("invalid value");
}
if ($number <= 0) {
    die("invalid number");
}
if ($flags != "baz") {
    die ("invalid flag");
}
# 1000
# lines
# of
# code
```



Offer some brain training for your users

```
# zypper ar --help |grep refresh
-f, --refresh      Enable autorefresh of the repository.
# zypper mr --help |grep refresh
-r, --refresh      Enable auto-refresh of the repository.
```

[bnc#661410]

Ignore compiler and rpmlint warnings

- real problems cause errors, not warnings
- conclusion: warnings are not a problem

Never submit your patches upstream

Keeping the patches in your package is fun:

- you look like a professional if you can handle 50 patches in a package
- you save upstream some work on reviewing and integrating the patches
- you always have some fun when updating the package and your patches to the next version

Never write any documentation

- if some old documentation exists, never update it
- nobody reads documentation anyways
- comments in the code also count as documentation – avoid adding them whenever possible

Never write any documentation

- nobody reads documentation anyways

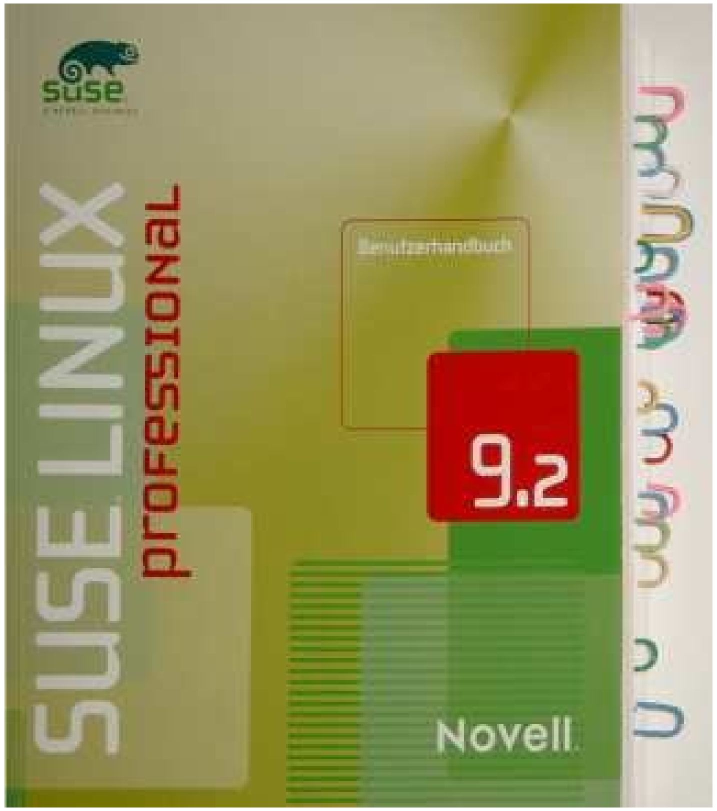
Really?

I've been doing this 10.1 test work just like a real user: In other words I never read any release notes or documenta-tion :-)

[tomhorsley(at)adelphia.net
in opensuse-factory]

Never write any documentation

- nobody reads documentation anyways



Really?

Beware of the paperclips!

[bnc#65000]

Never trust a bugreporter

> \> \> RESOLVED INVALID
> \> Henne, did you actually test this before closing
> \> the bug as invalid?

of course i did not test it. do you think i'm bored?

[bnc#420972]

Never trust a bugreporter

general rule: if Olaf reports a bug, it is a valid bug.
(Olaf Hering while reopening bnc#168595)

NEEDINFO fun

I am supposed to be the info provider,
so here is my answer:

42

By the way:

What is the question?

[Johannes Meixner, bnc#190173]

Never test small changes

switch2nvidia:

```
  * fixed disabling Composite extension;
     script replaced "Option" with "Optioff" :-(
```

[commit message by Stefan Dirsch]

```
KMS_IN_INITRD="noyes"
```

[sed result, bnc#619218]

```
-ao=pulse,alsa
+,alsa
```

[setup-pulseaudio fun, bnc#681113]

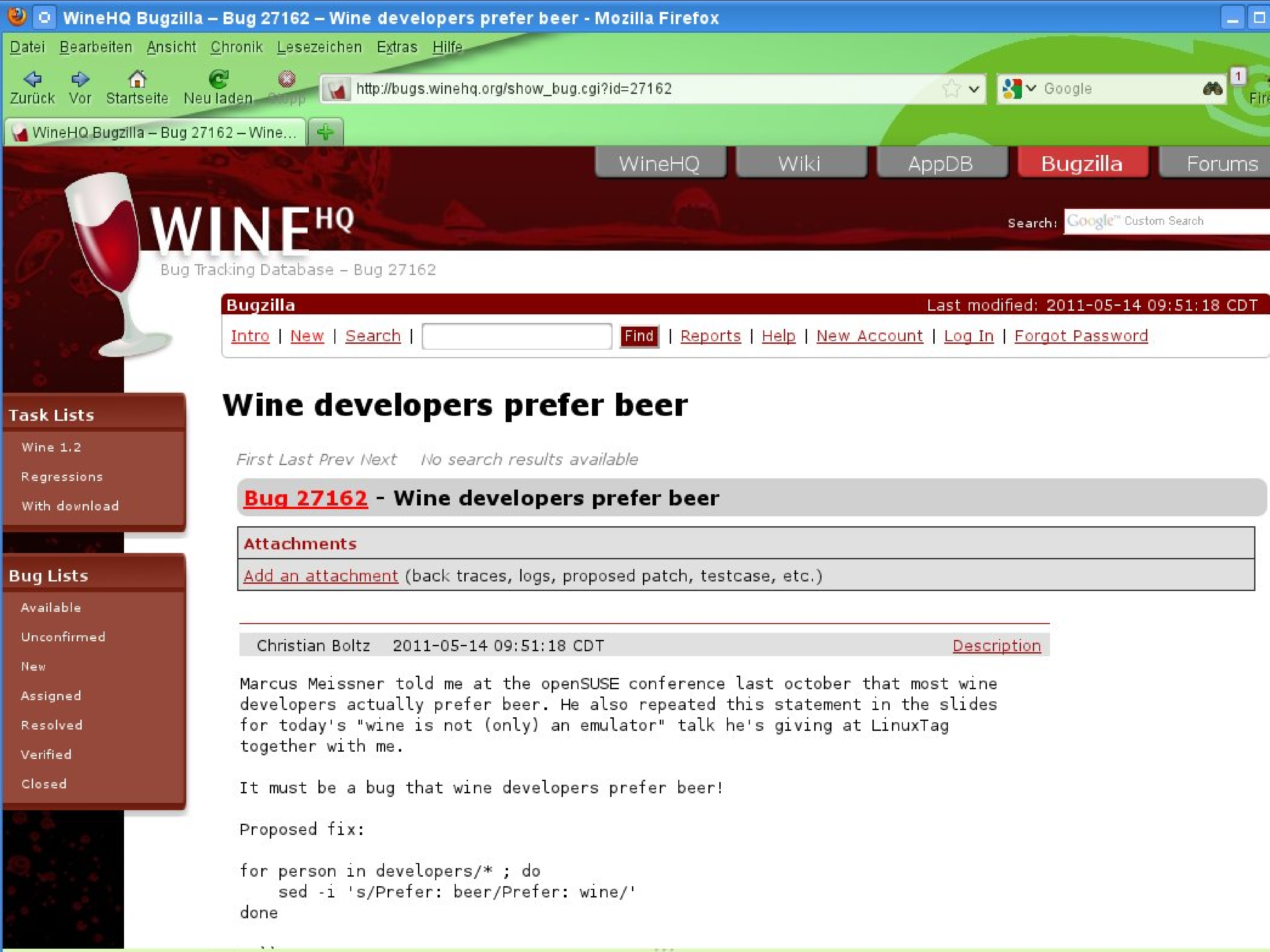
Quoting in shell scripts is overestimated

```
@@ -352,1 +352,1 @@
-   rm -rf /usr /lib/nvidia-current/xorg/xorg
+   rm -rf /usr/lib/nvidia-current/xorg/xorg
```

Probably the most-commented commit on github.

https://github.com/MrMEEE/bumblebee/commit/a047be

Hey, this makes you famous! ;-)


WineHQ Bugzilla – Bug 27162 – Wine developers prefer beer - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
Zurück Vor Startseite Neu laden Stopp
http://bugs.winehq.org/show_bug.cgi?id=27162
Google
WineHQ Bugzilla – Bug 27162 – Wine...
WineHQ Wiki AppDB Bugzilla Forums
WINE HQ
Search: Google Custom Search
Bug Tracking Database – Bug 27162
Bugzilla
Last modified: 2011-05-14 09:51:18 CDT
Intro | New | Search | Find | Reports | Help | New Account | Log In | Forgot Password
Task Lists
Wine 1.2
Regressions
With download
Bug Lists
Available
Unconfirmed
New
Assigned
Resolved
Verified
Closed

Wine developers prefer beer

First Last Prev Next   No search results available

Bug 27162 - Wine developers prefer beer

Attachments
Add an attachment (back traces, logs, proposed patch, testcase, etc.)

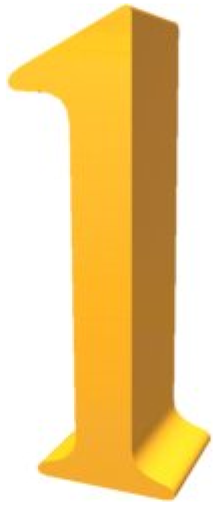
Christian Boltz 2011-05-14 09:51:18 CDT Description

```
Marcus Meissner told me at the openSUSE conference last october that most wine
developers actually prefer beer. He also repeated this statement in the slides
for today's "wine is not (only) an emulator" talk he's giving at LinuxTag
together with me.

It must be a bug that wine developers prefer beer!

Proposed fix:

for person in developers/* ; do
    sed -i 's/Prefer: beer/Prefer: wine/'
done
```

and the winner is…

Make your code easy to understand

```
   #!/usr/bin/perl
   eval eval '"'.

                    '#'.'!'.'/'.('['^'.')      .+(
               '['^'(').('['^')').'/'.('`'|'"').('`'|')')')
           .('`'|'.').'/'.('['^'+').('`'|'%').('['^')')'   ).
         ('`'|',').('!'^'+').('!'^'+').('['^'+').('[' ^+ ( ((
      ')')))).('`'|')').('`'|'.').('['^'/').('{'^'[') .''.  ((
    '\\')).'"'.('`'^')')."'".('`'|'-').('{'^'['). (((  ((  '`'
   ))))|'!').('{'^'[').('`'|"'").('`'|'%').(('`')|    "\%").(
  '`'|'+')   .('`'|'/').'!'.('{'^'[').':'.'-'.')'.'\\'.    (((
 '\\')            )).('`'|'.').'\\'.'"'.';'.('!'^'+').("\!"^
'+').   "\"";$:=    '.'^'~';$~='@'|('(');$^=
')'^   '['   ;($/)   ="\`"|         '.';$,=
'('^   '}'     ;$\=   "\`"|           "\!";
 ($:)  ="\)"^   '}'    ;$~=             '*'|
  '`';         ($^)                        =(
   '+')^'_' ;($/)=
     '&'|'@';#;#
```



© September 27, 2011 Christian Boltz

special rule for openSUSE:

special rule for openSUSE:

Always code as if the guy who
ends up maintaining your code
will be a violent psychopath
who knows where you live.
[John F. Woods]

Thanks!

- everybody who accidently ;-) contributed to my talk
- for the inspiration by

  http://www.karzauninkat.com/Goldhtml/
  “golden rules of bad HTML” (german)

  http://www.sapdesignguild.org/community/design/golden_rules.asp
  “golden rules for bad user interfaces”

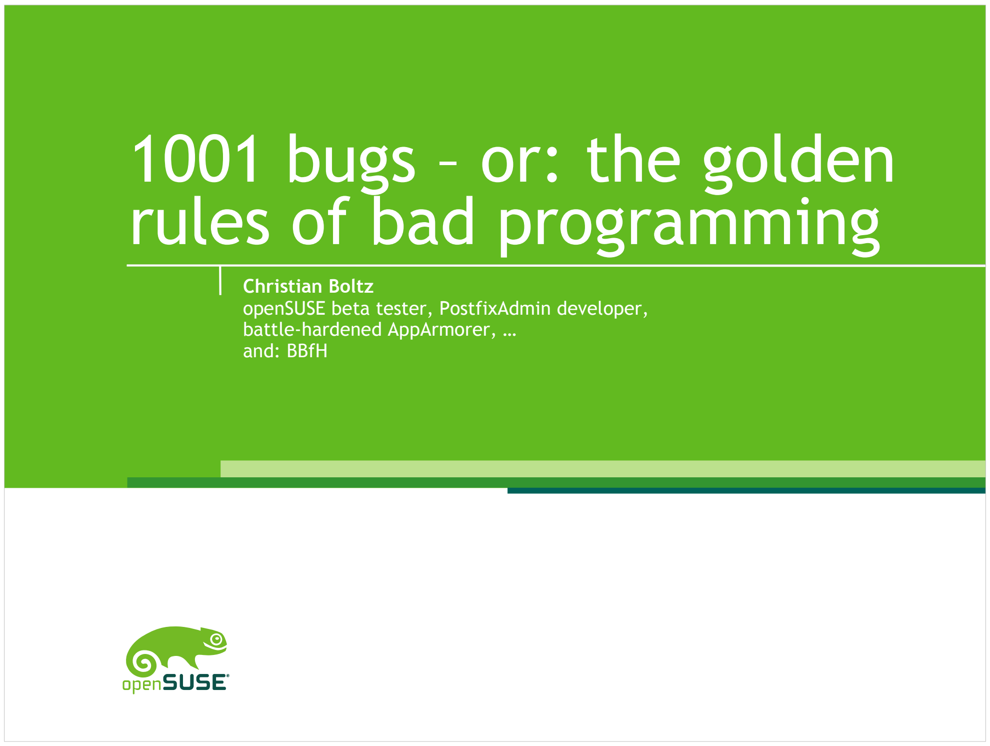
  http://blog.koehntopp.de/archives/2127-The-Importance-Of-FAIL.html

  http://blog.koehntopp.de/archives/2611-Was-bedeutet-eigentlich-Never-check-for-an-error-condition-you-dont-know-how-to-handle.html
  two great articles about FAIL by Kris Köhntopp
- perl Acme::EyeDrops for rendering rule #1
- for listening

Questions?

Opinions?

Flames?



1001 bugs – or: the golden rules of bad programming

Christian Boltz
openSUSE beta tester, PostfixAdmin developer,
battle-hardened AppArmorer, …
and: BBfH

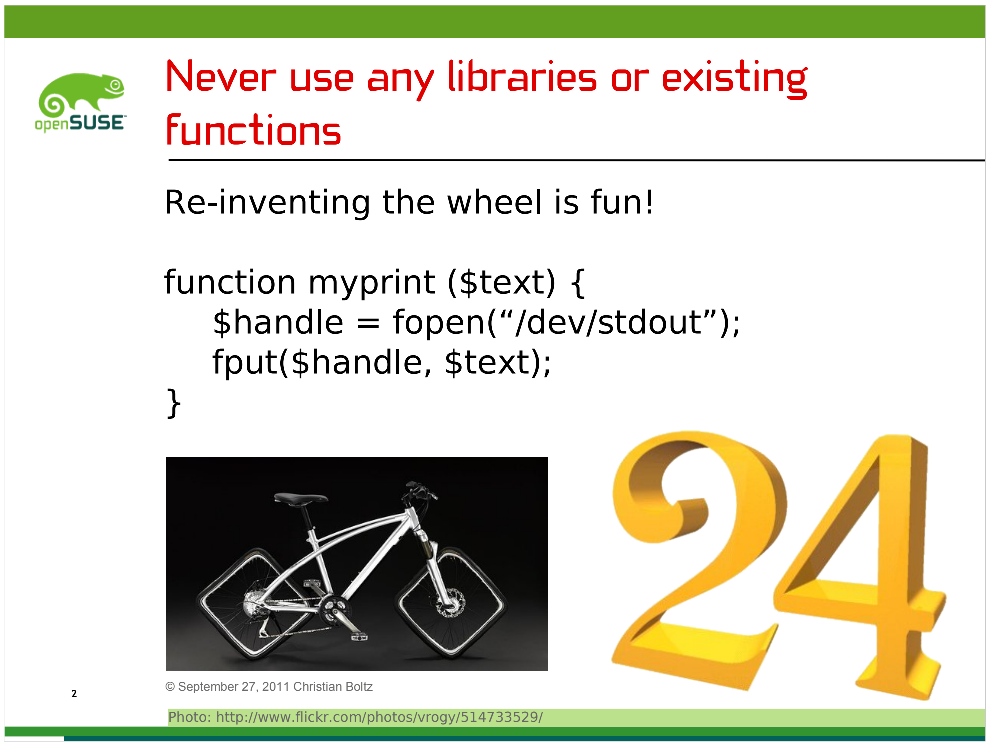
Battle-hardened AppArmorer:by Sascha Peilicke

BBfH: Bastard Bugreporter from Hell

You'll find lots of books telling you how to write good code. That's nice and maybe even useful, but boring ;-)

My talk will give you something more inspiring: the golden rules of bad programming.

BTW: I have no idea why rules have to be golden, but I won't break this tradition.



Never use any libraries or existing functions

Re-inventing the wheel is fun!

```
function myprint ($text) {
    $handle = fopen("/dev/stdout");
    fput($handle, $text);
}
```

© September 27, 2011 Christian Boltz

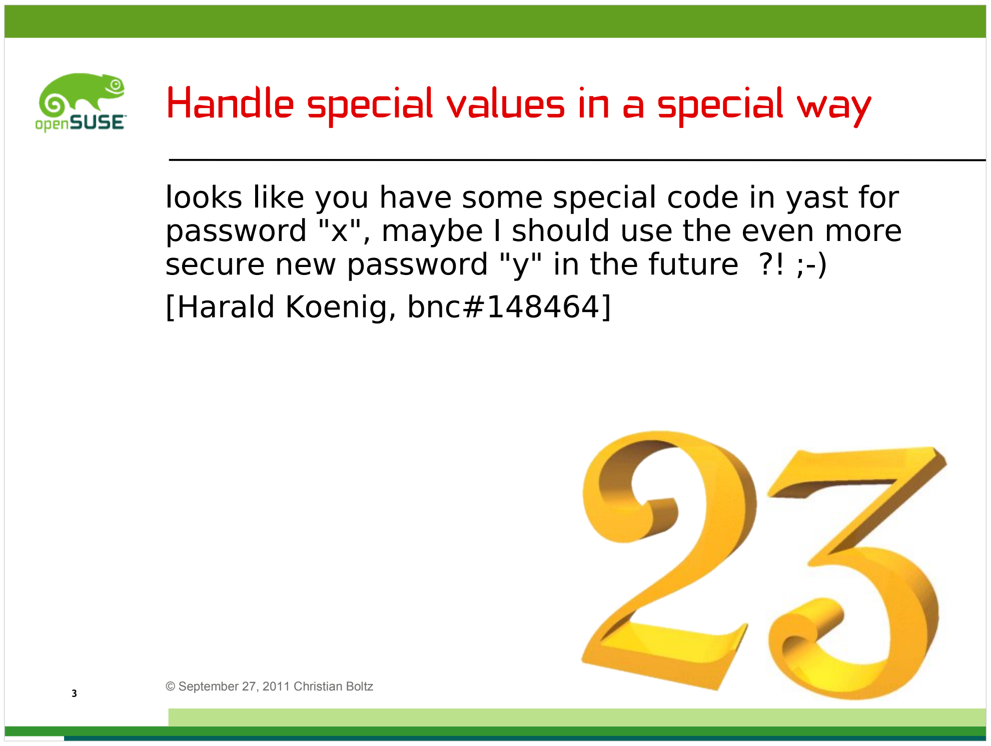
Photo: http://www.flickr.com/photos/vrogy/514733529/

Using existent libraries is evil. Think of dependencies and bigger installed size if you use lots of libraries – do you really want that?

I have to admit that the example is very extreme, but I also have an example from practise – parsing commandline options.

You might think "hey, I only need to handle two options" - no need for a library. That's also what happened in one of this year's GSoC projects. Sooner or later, you add more commandline switches, some of them accept options and so on.

Sooner or later you have to switch to getopts that already provides all the details you need.



Handle special values in a special way

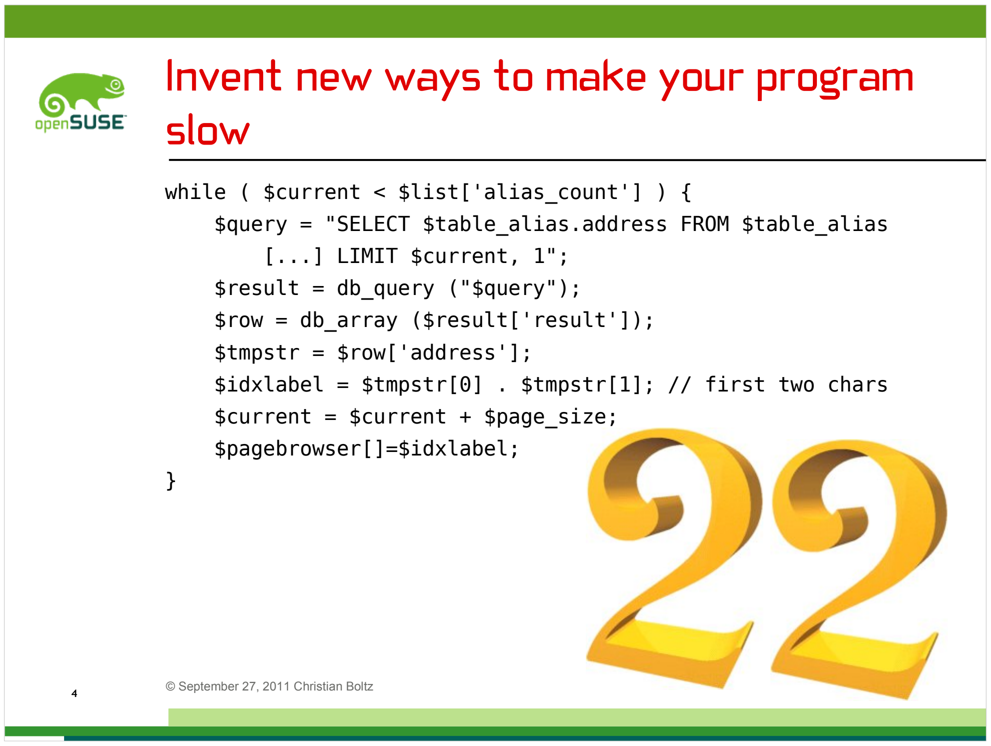
looks like you have some special code in yast for password "x", maybe I should use the even more secure new password "y" in the future ?! ;-)
[Harald Koenig, bnc#148464]

YaST had some special code that locked a newly created user if you give him password 'x'.

There are more examples where special values were handled in a special way.

Think of all the y2k bugs where the year zero was handled in a funny way.

And Linux will get a similar problem in 2038 when the unix time (seconds since 1970) doesn't fit in a 32bit variable anymore. I doubt someone will still use 32bit systems by then, but nevertheless the problem might be embedded in data structures, file formats and in embedded devices like machine control units.



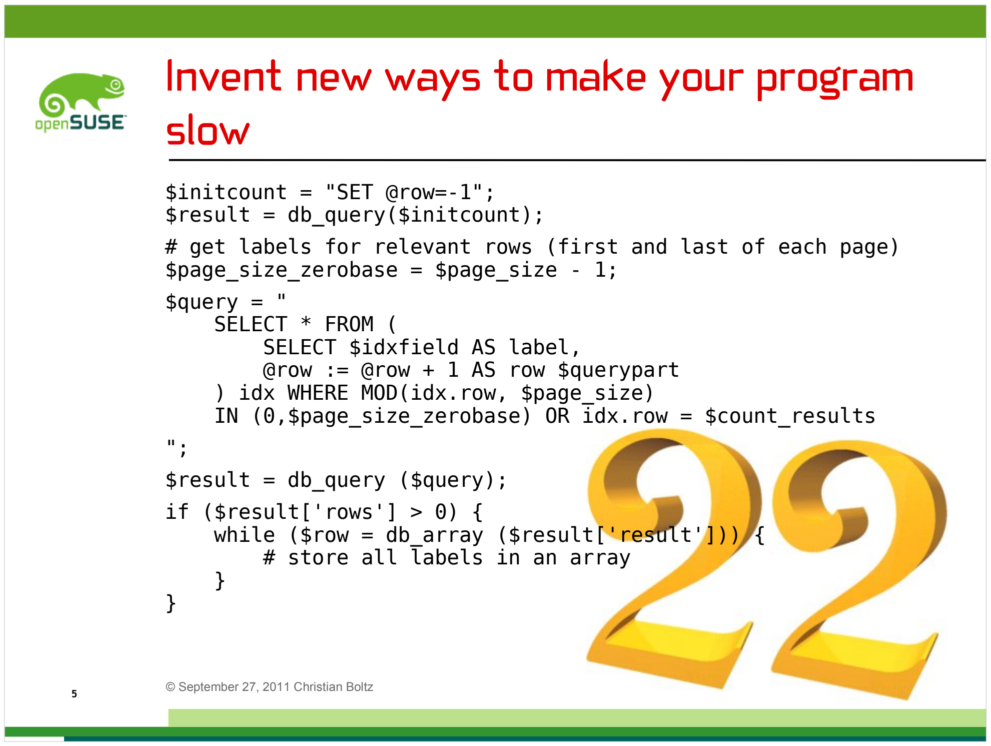
Invent new ways to make your program slow

```
while ( $current < $list['alias_count'] ) {
    $query = "SELECT $table_alias.address FROM $table_alias
        [...] LIMIT $current, 1";
    $result = db_query ("$query");
    $row = db_array ($result['result']);
    $tmpstr = $row['address'];
    $idxlabel = $tmpstr[0] . $tmpstr[1]; // first two chars
    $current = $current + $page_size;
    $pagebrowser[]=$idxlabel;
}
```

A very good way to make a program slow is to do SQL queries in a loop.

The above code is a shortened sniplet from PostfixAdmin 2.3 to generate the pagebrowser (you know – the “a-c, d-f, g-k” links) in the listing of mail addresses. The code on the slide only fetches the starting point of each page, the original code fetches start and end point, which doubles the number of queries.

Following this rule worked quite well – with 1000 pages of mail addresses, it took 10 minutes until the pagebrowser was generated. And, very strange, there were even some users that complained about the loading time of that page...



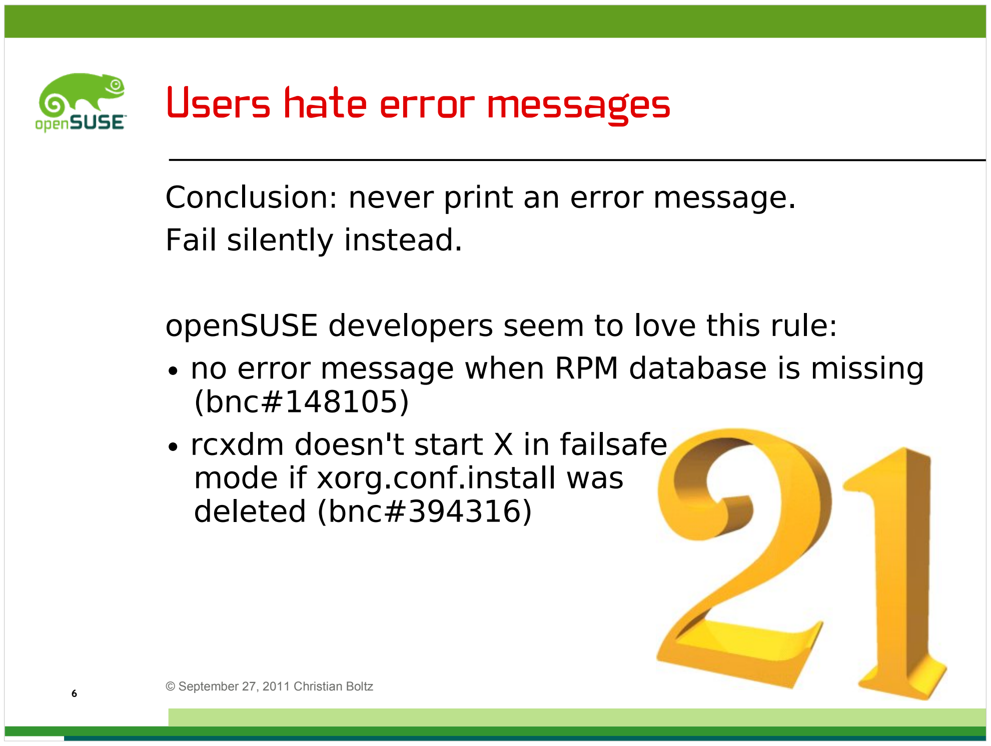
Invent new ways to make your program slow

```
$initcount = "SET @row=-1";
$result = db_query($initcount);
# get labels for relevant rows (first and last of each page)
$page_size_zerobase = $page_size - 1;
$query = "
    SELECT * FROM (
        SELECT $idxfield AS label,
        @row := @row + 1 AS row $querypart
    ) idx WHERE MOD(idx.row, $page_size)
    IN (0,$page_size_zerobase) OR idx.row = $count_results
";
$result = db_query ($query);
if ($result['rows'] > 0) {
    while ($row = db_array ($result['result'])) {
        # store all labels in an array
    }
}
```

Now here's what you should never do if you want to see your CPU busy: Let MySQL count over the rows and just get the relevant lines with one big query.

The downside is that this doesn't work with postgresql – if someone knows a similar working solution for postgresql, please tell me after the talk.

Oh, and while we are talking about databases: There's also the good old and simple way to make your program slow – don't add an index to your tables. This small detail can already make your queries 100 times slower.



Users hate error messages

Conclusion: never print an error message.
Fail silently instead.

openSUSE developers seem to love this rule:

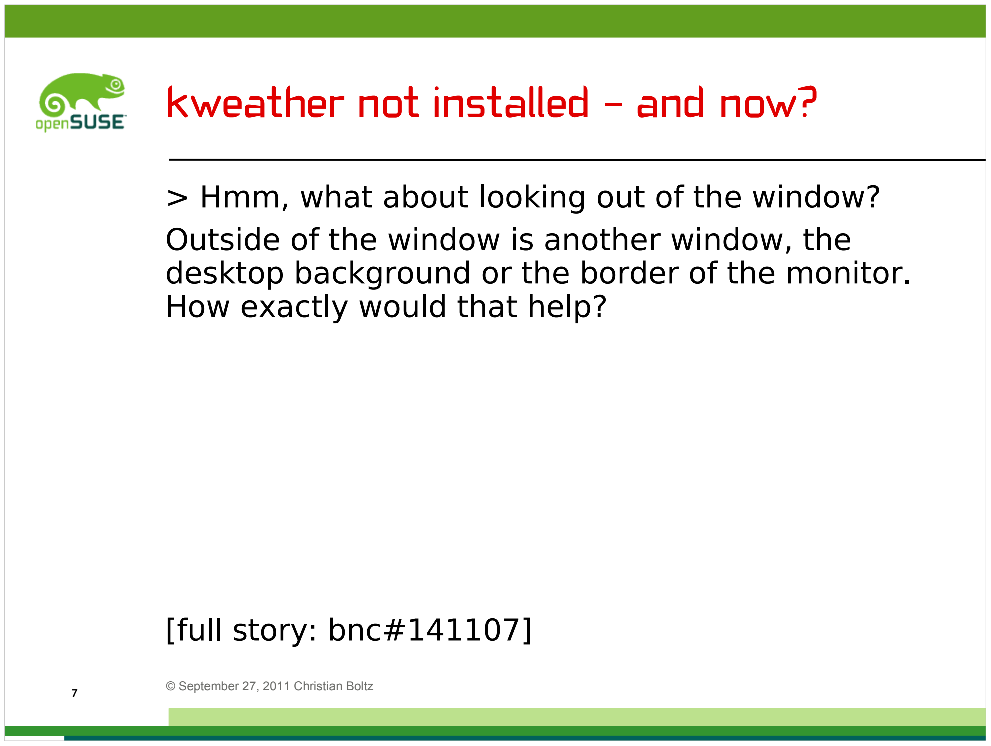
- no error message when RPM database is missing (bnc#148105)
- rcxdm doesn't start X in failsafe mode if xorg.conf.install was deleted (bnc#394316)

6

© September 27, 2011 Christian Boltz

That's the hardest type of bugreports – trying to explain a developer that you want to see an error message added. Usually they'll explain you why the thing you are doing can't work and that it's impossible to add working code for that case, which is usually a corner case.

About 5 reopens later, they finally understand that all you want is an additional error message, which is what was already stated in the initial bugreport.



kweather not installed – and now?

> Hmm, what about looking out of the window?

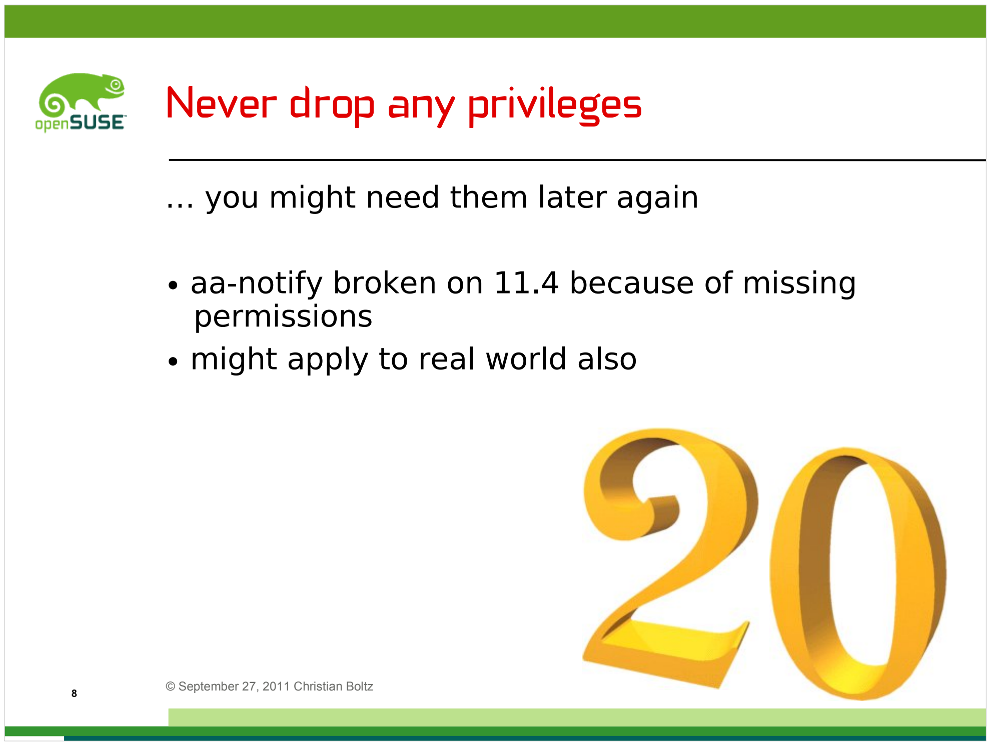
Outside of the window is another window, the desktop background or the border of the monitor. How exactly would that help?

[full story: bnc#141107]

Sometimes the developers in the SUSE office have really difficult problems to solve.

One day Marcus noticed that kweather was not installed by default, and I proposed to look out of the window.

Rasmus Plewe tried this, but not too successful.



Never drop any privileges

… you might need them later again

- aa-notify broken on 11.4 because of missing permissions
- might apply to real world also

8

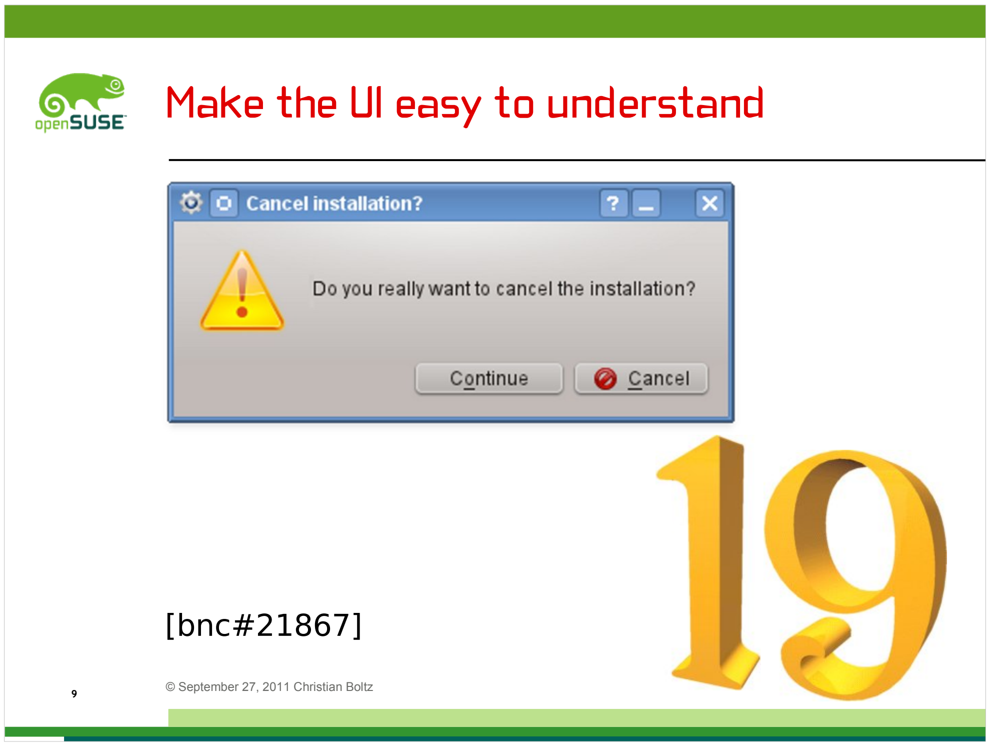
© September 27, 2011 Christian Boltz

aa-notify is part of AppArmor and can be used to display desktop notifications when a program violates the AppArmor policy.

By dropping permissions, it lost the ability to read /var/log/audit/

Funnily, it somehow followed the “never drop any privileges” rule and didn't drop the group permissions. This means it was able to access the /var/log/audit/ directory with its group permissions on Ubuntu, but not on openSUSE because of stricter permissions.

That's the funny thing – sometimes two bugs neutralize each other. At least until someone like me comes and has a too strict permission set to make the group permission bug working.



Make the UI easy to understand

Cancel installation?

Do you really want to cancel the installation?

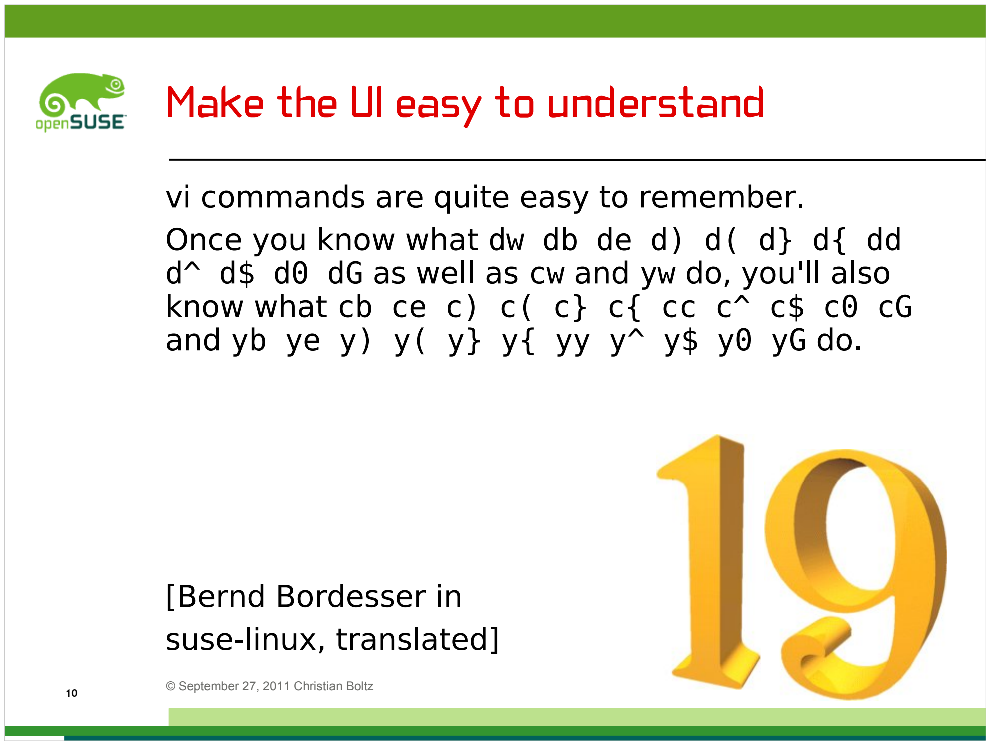
[bnc#21867]

Now guess what happens if you click continue.

The answer is: YaST continues to cancel the installation – or, translated for the confused listeners: continue cancels the installation.

If you click cancel, the installation will continue.

Buttons named “yes” and “no” might be a better idea in such cases.



Make the UI easy to understand

vi commands are quite easy to remember.

Once you know what `dw db de d) d( d} d{ dd d^ d$ d0 dG` as well as `cw` and `yw` do, you'll also know what `cb ce c) c( c} c{ cc c^ c$ c0 cG` and `yb ye y) y( y} y{ yy y^ y$ y0 yG` do.

[Bernd Bordesser in suse-linux, translated]

This example speaks for itsself – vim is really easy to use

Take great care in setting bad defaults

```
zypper ar obs://home:cboltz home:cboltz
```

always adds the Factory repo instead of the repo for the installed version.

But there is a config option to hardcode the distribution in zypper.conf (and to have fun after upgrading the distribution)

(/etc/SuSE-release anyone?)

[bnc#648892]
[wontfix, ENOTIME]

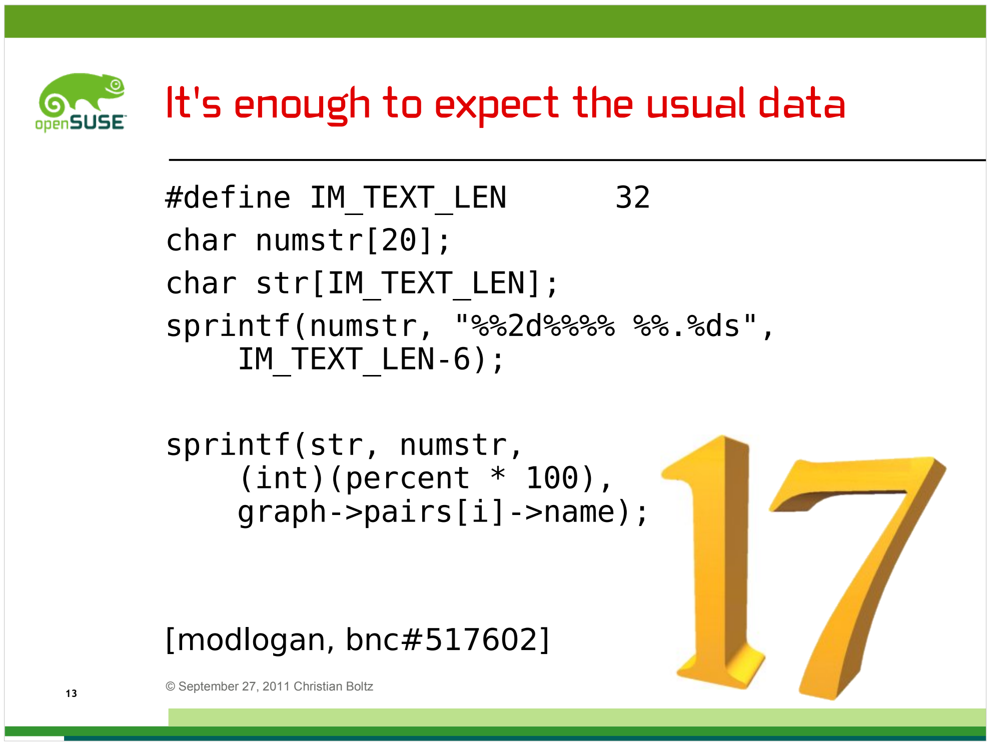
It's enough to expect the usual data

(cron.daily doesn't run because the system is on battery)

Yes Karl, your machine has a battery! But no ac adapter :-)

Unfortunately it is the battery in the BT Mouse and it won't be able to power your system for very long :-)

[Stefan Seyfried, bnc#221999]



It's enough to expect the usual data

```
#define IM_TEXT_LEN      32
char numstr[20];
char str[IM_TEXT_LEN];
sprintf(numstr, "%%2d%%%% %%.%ds",
    IM_TEXT_LEN-6);

sprintf(str, numstr,
    (int)(percent * 100),
    graph->pairs[i]->name);
```

[modlogan, bnc#517602]

Here's a nice format string example from modlogan – who can tell me what it does?

First it generates the formatstring numstr:

numstr: %2d → 2 digits from (int)(percent*100)

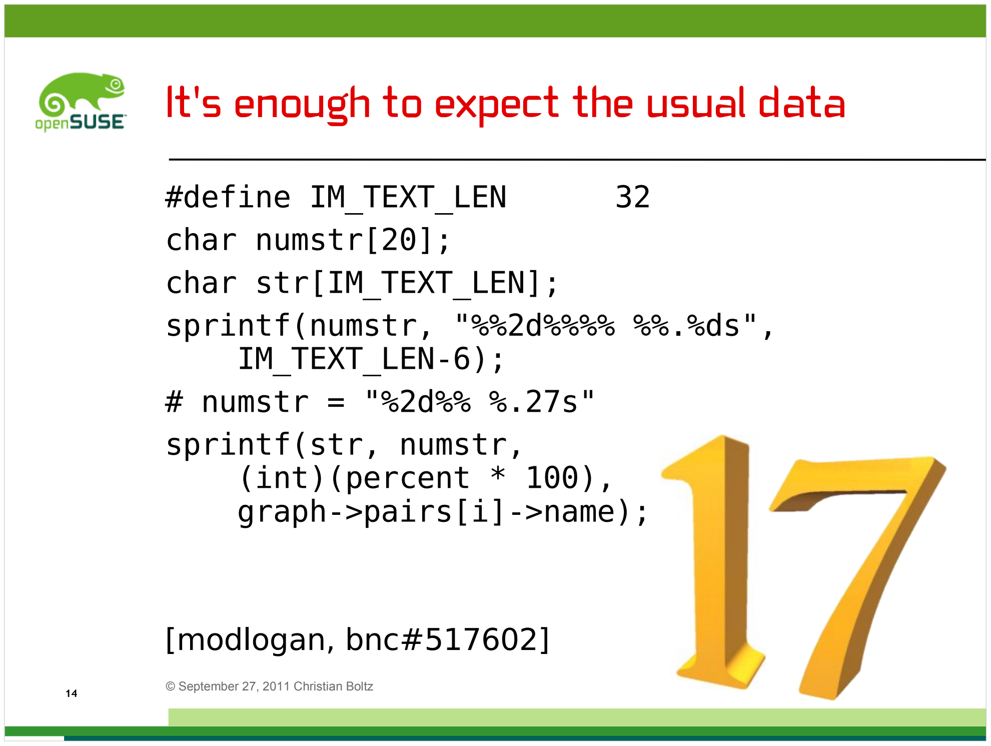
“%% “ → “%“ + space (2 byte)

%.27s → 27 chars string with dots as fillers

total: 31 bytes

And the even more interesting question: Why/how can it cause a buffer overflow?

Answer: with only one line in the log (in other words: only one client, only one browser type), you'll get 100%. That's 3 digits and makes the string one byte longer – 32 byte total, no space left for the null byte at the end



It's enough to expect the usual data

```
#define IM_TEXT_LEN      32
char numstr[20];
char str[IM_TEXT_LEN];
sprintf(numstr, "%%2d%%%% %%.%ds",
    IM_TEXT_LEN-6);
# numstr = "%2d%% %.27s"
sprintf(str, numstr,
    (int)(percent * 100),
    graph->pairs[i]->name);
```

[modlogan, bnc#517602]

Here's a nice format string example from modlogan – who can tell me what it does?

First it generates the formatstring numstr:

numstr: %2d → 2 digits from (int)(percent*100)

“%% “ → “%“ + space (2 byte)

%.27s → 27 chars string with dots as fillers

total: 31 bytes

And the even more interesting question: Why/how can it cause a buffer overflow?

Answer: with only one line in the log (in other words: only one client, only one browser type), you'll get 100%. That's 3 digits and makes the string one byte longer – 32 byte total, no space left for the null byte at the end

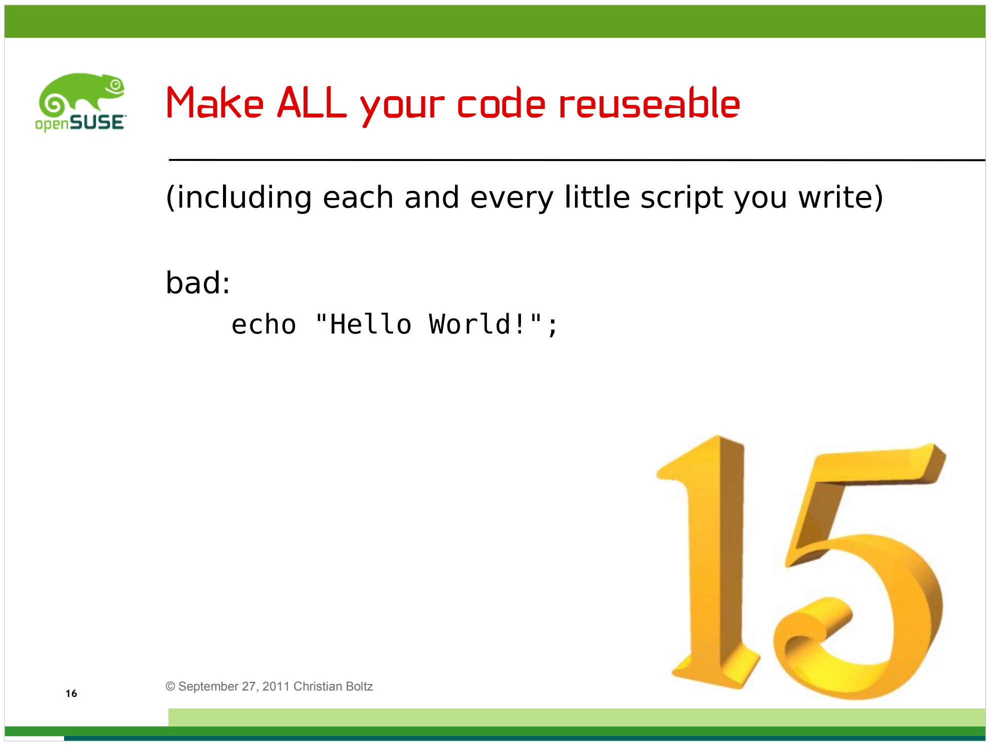
Never make your code reusable

… especially if it has more than 1000 lines

Reusing pieces of code is like picking off sentences from other people's stories and trying to make a magazine article. [Bob Frankston]

Besides that:

Nobody needs that much code a second time ;-)



Make ALL your code reuseable

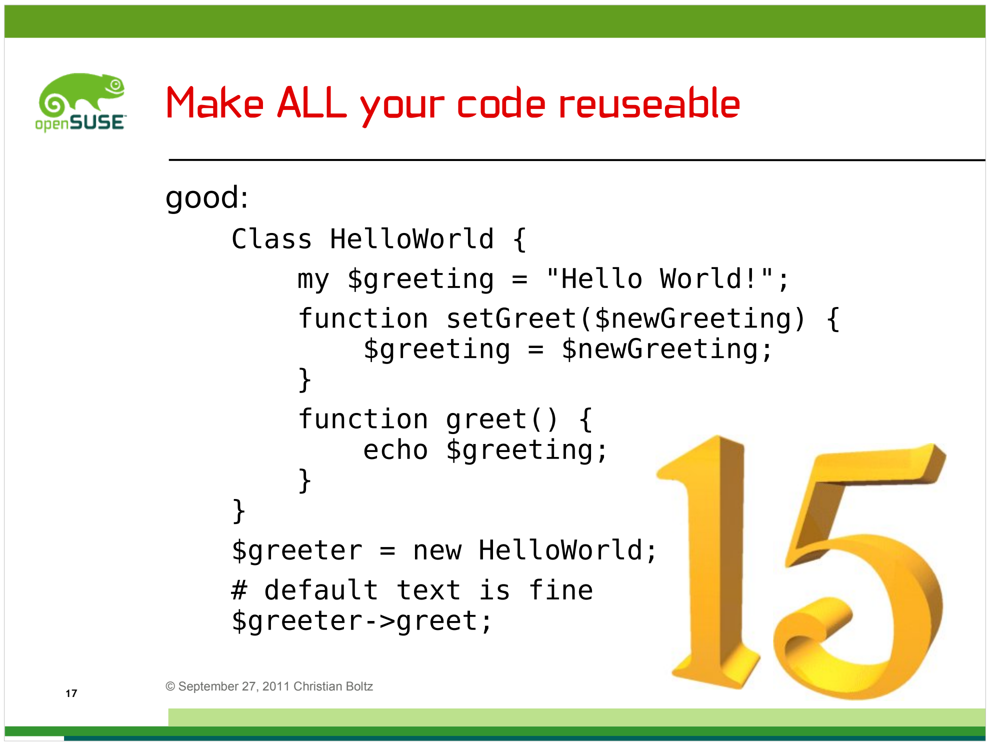
(including each and every little script you write)

bad:

```
    echo "Hello World!";
```

That's bad code because it is not reuseable

Let's make it better...



Make ALL your code reuseable

good:

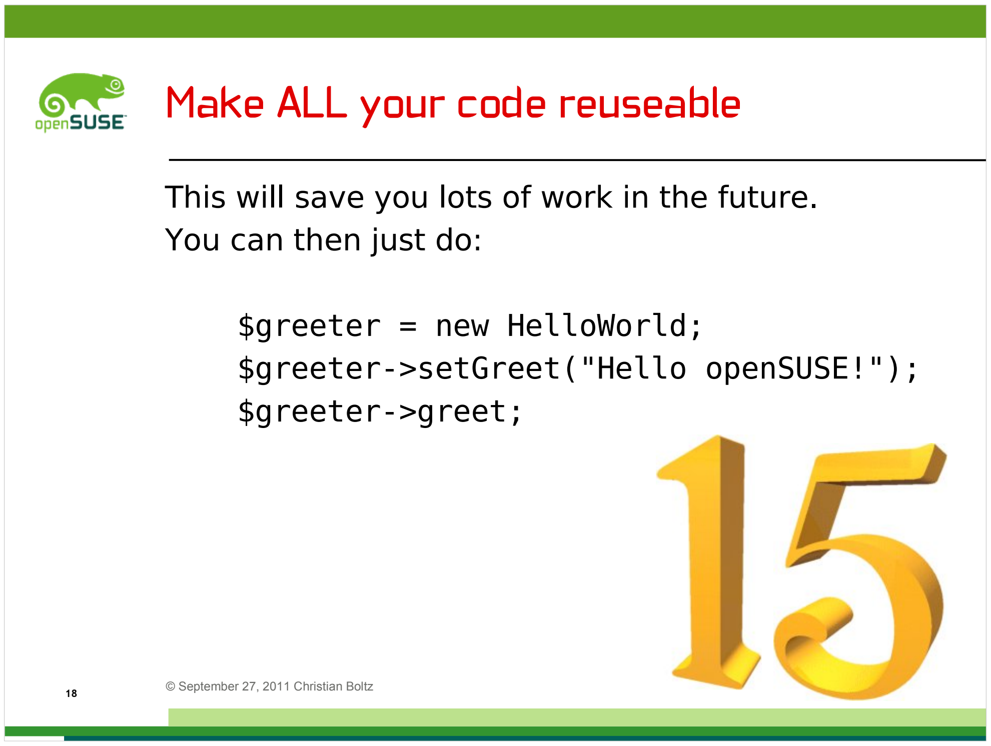
```
Class HelloWorld {
    my $greeting = "Hello World!";
    function setGreet($newGreeting) {
        $greeting = $newGreeting;
    }
    function greet() {
        echo $greeting;
    }
}
$greeter = new HelloWorld;
# default text is fine
$greeter->greet;
```

Here's the good code:

- the greeting text is in a variable – the code is flexible enough to work with different texts

- everything is encapsulated in a class – no internals are visible to the outside

- and, most important: it's reuseable



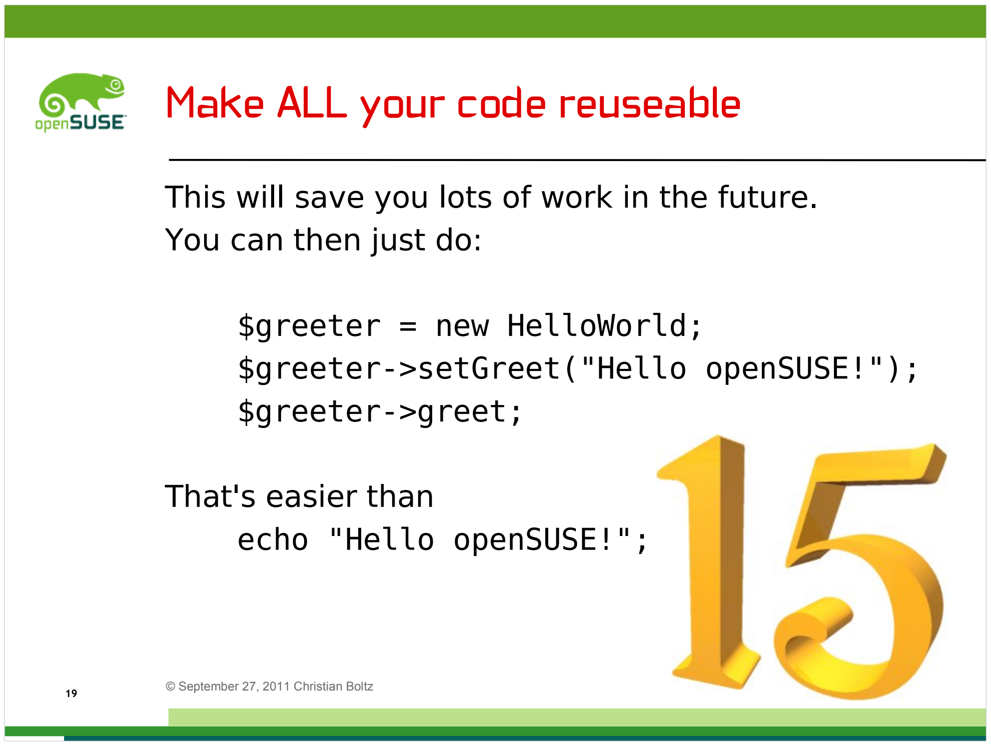
Make ALL your code reuseable

This will save you lots of work in the future.
You can then just do:

```
$greeter = new HelloWorld;
$greeter->setGreet("Hello openSUSE!");
$greeter->greet;
```

See how much work reuseable code can save you in the future. Just create an instance of the class, set the text you want and let it print its greeting.

And, most important...



Make ALL your code reuseable

This will save you lots of work in the future.
You can then just do:

```
    $greeter = new HelloWorld;
    $greeter->setGreet("Hello openSUSE!");
    $greeter->greet;
```

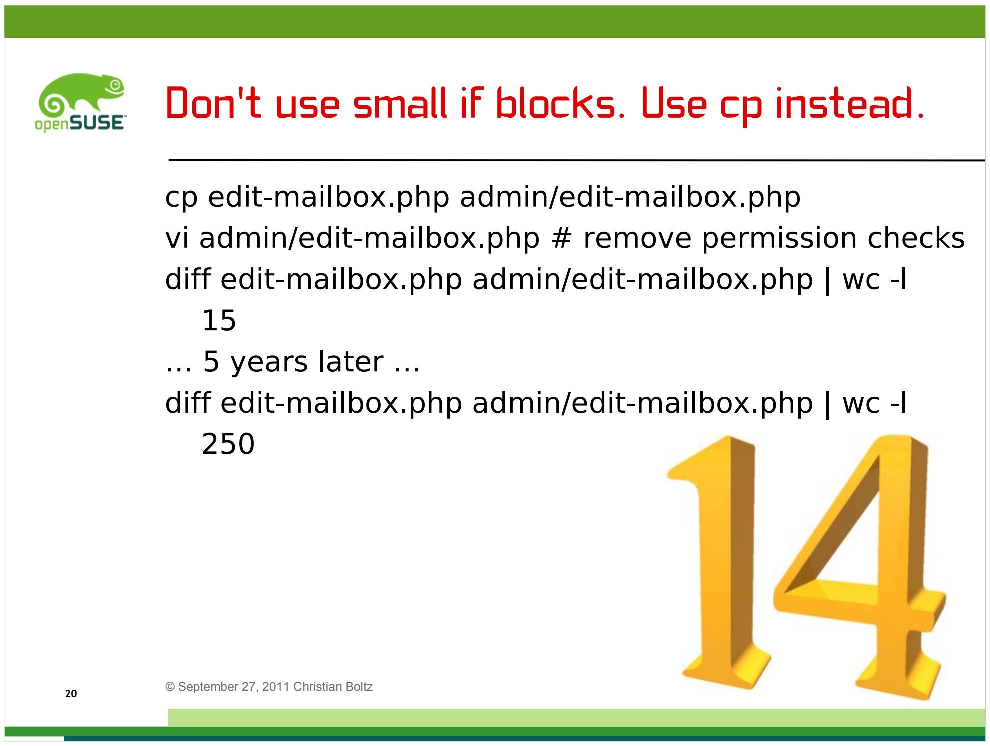
That's easier than

```
    echo "Hello openSUSE!";
```

… that's much easier than using code you can't re-use ;-)

In practise, the way to go is somewhere between this and the previous rule – big programs should have units of reuseable code whereever it makes sence. OTOH, it doesn't make sense to make everything reuseable, as the “hello world” example showed.

Usually a pragmatic approach is the best choise – do what you think is the best



Don't use small if blocks. Use cp instead.

```
cp edit-mailbox.php admin/edit-mailbox.php
vi admin/edit-mailbox.php # remove permission checks
diff edit-mailbox.php admin/edit-mailbox.php | wc -l
   15
… 5 years later …
diff edit-mailbox.php admin/edit-mailbox.php | wc -l
   250
```

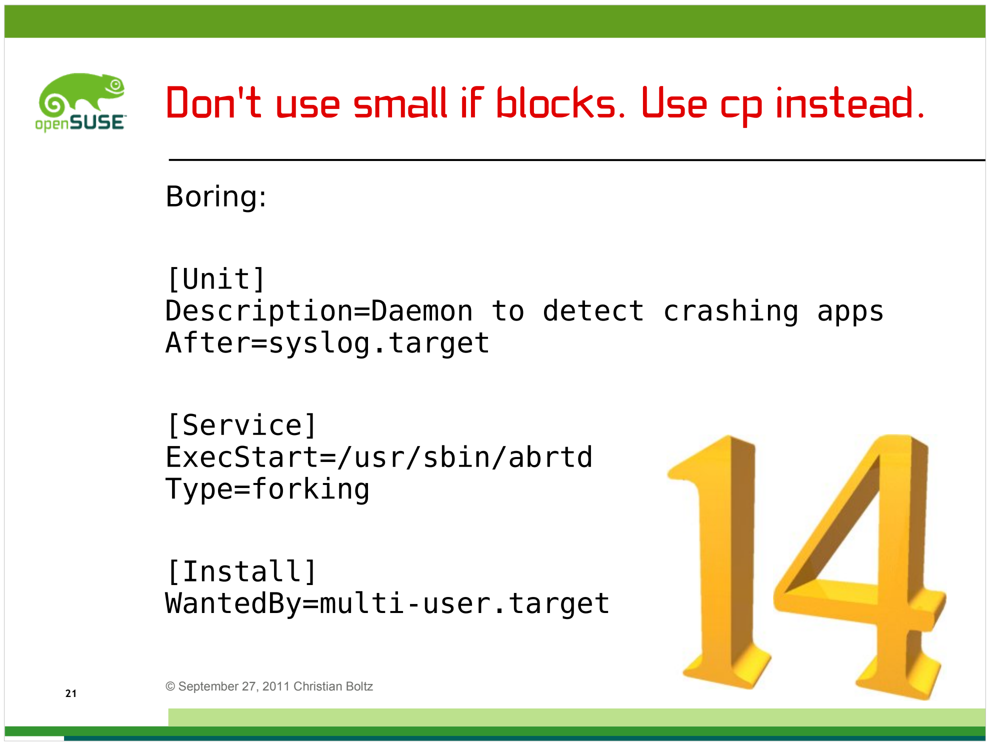
This is a real-world example from PostfixAdmin, however the numbers are just a not-so-wild guess.

Several years ago, edit-mailbox (for admins that have permissions only for some domains) and admin/edit-mailbox (for superadmins, think "root") had the same code, with the only exception that the superadmin code did not check for domain permissions.

Over the years, several changes and bugfixes were done – but only in one copy of the code.

The result was that several copies of nearly the same code existed, but each copy came with a different set of bugs.

That's what I found when I started working on PostfixAdmin in 2007. Since then I'm more or less doing code cleanup and remove duplicated code. But yes, we also introduced some new cool features.



Don't use small if blocks. Use cp instead.

Boring:

```
[Unit]
Description=Daemon to detect crashing apps
After=syslog.target

[Service]
ExecStart=/usr/sbin/abrtd
Type=forking

[Install]
WantedBy=multi-user.target
```

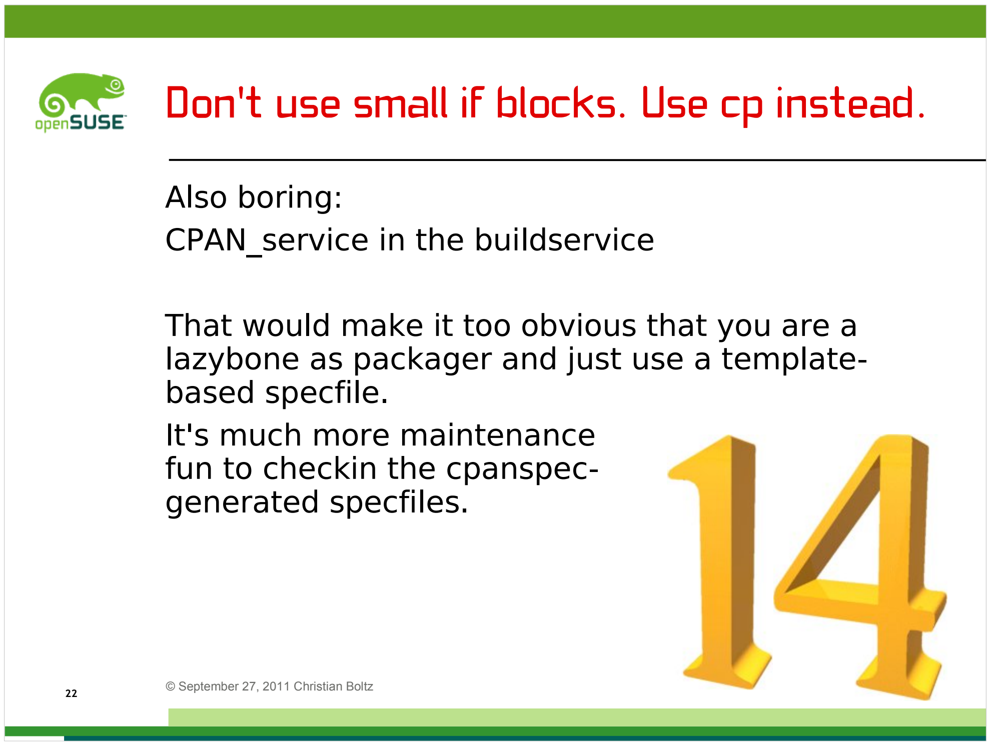
© September 27, 2011 Christian Boltz

21

A more openSUSE-related example are initscripts.

You all know how long the old-style initscripts are, and if you compare them, you'll find out that 80 or 90% of the code is the same in all initscripts.

Now compare that to the systemd unit files. In short: the systemd unit files are ways too short, boring and too easy to maintain.



Don't use small if blocks. Use cp instead.

Also boring:
CPAN_service in the buildservice

That would make it too obvious that you are a lazybone as packager and just use a template-based specfile.

It's much more maintenance fun to checkin the cpanspec-generated specfiles.

Even more openSUSE-related: Source services in the buildservice.

In general they are a good idea (or at least would be if they work)

For example the cpanspec service would make it much easier to package 90% of the perl packages.

However, that would make it too obvious that you are a lazybone as packager and just use template-based specfiles.

Checking in the cpanspec-generated specfile will give you much more maintenance fun. When you update the package, you have the choice of

a) edit the version number in the specfile and hope that everything else continues to work

b) run cpanspec to recreate the specfile and hope that it didn't have to use an additional parameter to cpanspec that you don't remember

My personal solution is to checkin a “run-cpanspec.sh” script in my packages.

And it's even funnier to discuss this with yaloki and darix who both hate source services ;-)

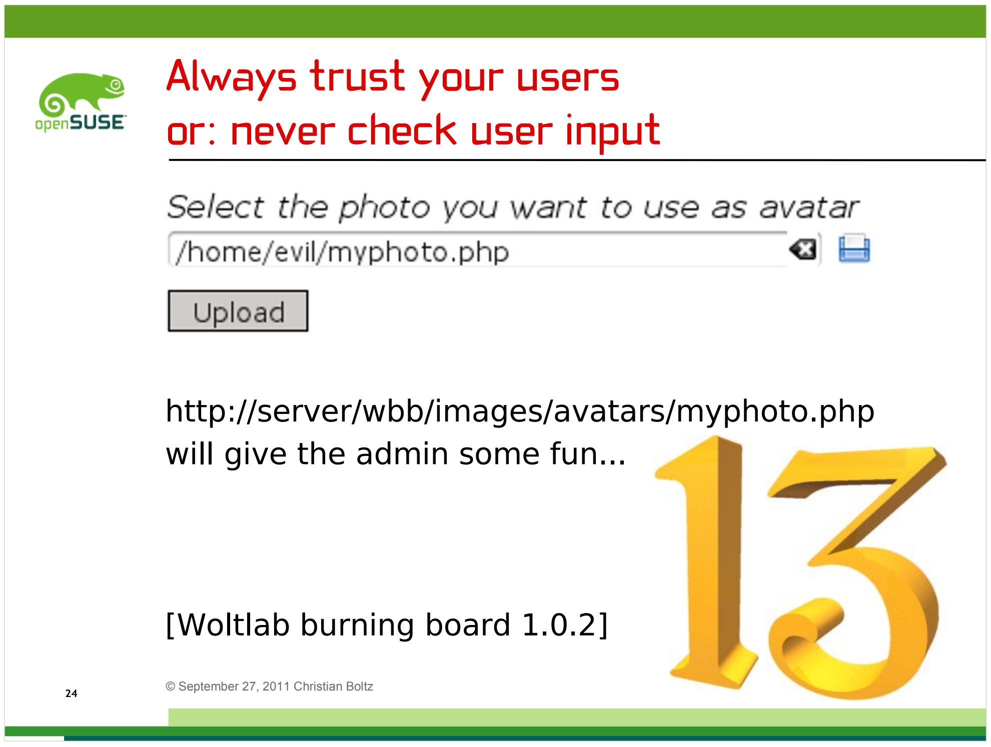
Always trust your users
or: never check user input

Even AppArmor followed this rule, so it can't be too wrong ;-)

```
echo 'AAA AAA' > /proc/$$/attr/current
Segmentation fault
```

```
[107353.169142] kernel BUG at kernel BUG at
/usr/src/packages/BUILD/kernel-desktop-
2.6.37.6/linux-2.6.37/security/apparmor/audit.c:183!
[107353.169159] invalid opcode: 0000 [#7] SMP
[…]
```

[https://bugs.launchpad.net/bugs/789409]



Always trust your users or: never check user input

Select the photo you want to use as avatar

/home/evil/myphoto.php

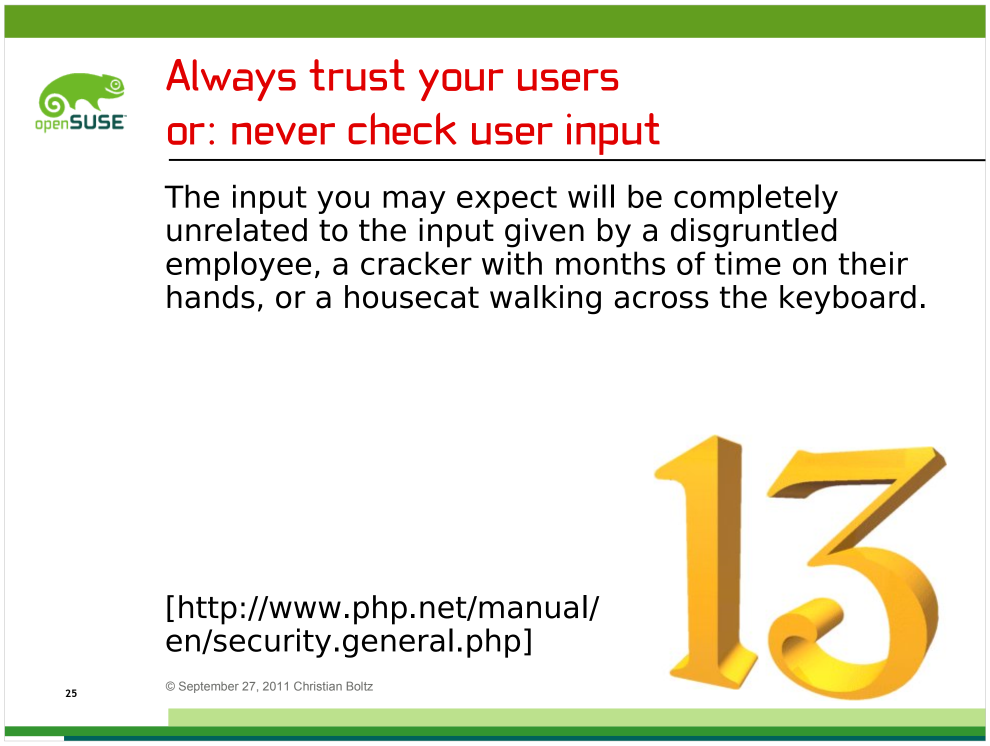
http://server/wbb/images/avatars/myphoto.php will give the admin some fun...

[Woltlab burning board 1.0.2]

Real-world case: a customer's forum software allowed to upload any file as avatar. Including PHP scripts...

BTW: The next version fixed the upload vulnerability, but allowed to download any file the webserver could read – including PHP files with database passwords and /etc/passwd

I'd say the PHP security team isn't too wrong. They say...



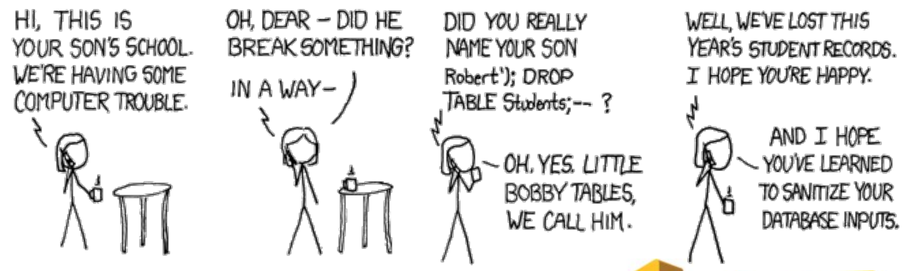
Always trust your users
or: never check user input

The input you may expect will be completely unrelated to the input given by a disgruntled employee, a cracker with months of time on their hands, or a housecat walking across the keyboard.

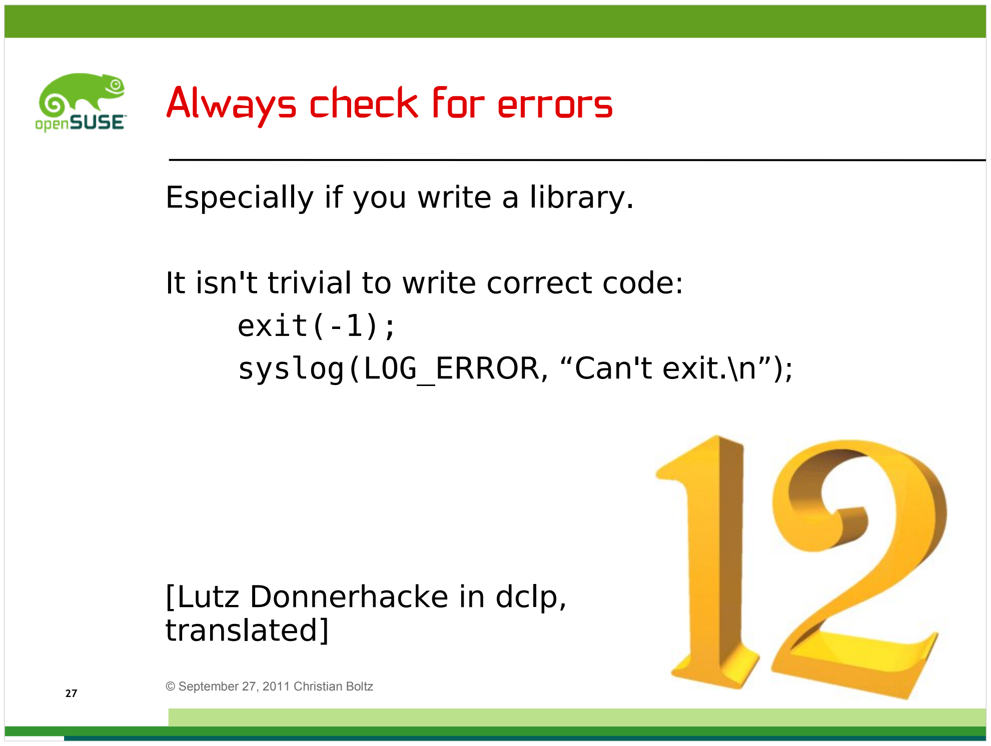
[http://www.php.net/manual/en/security.general.php]

… but this rule wouldn't be complete without little bobby tables...

Always trust your users
or: never check user input


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

[http://xkcd.com/327/]



Always check for errors

Especially if you write a library.

It isn't trivial to write correct code:

```
     exit(-1);
     syslog(LOG_ERROR, “Can't exit.\n”);
```

[Lutz Donnerhacke in dclp, translated]

This rule is especially valid if you write a library.

For example, there could be a failure in allocating memory. What do you do?

There might be cases where you don't know how an error should be handled. In that case, it's indeed better to forward the error to the calling application instead of handling it yourself in a way that breaks the application. Please forward all details of the error – telling the application “something didn't work” isn't really helpful.

Back to the memory allocation failure:

The library can't simply exit with an error message – that would also exit the calling program in an undefined state.

the only solution I'd accept in a library is that it does a quick online order for _free_ memory modules.

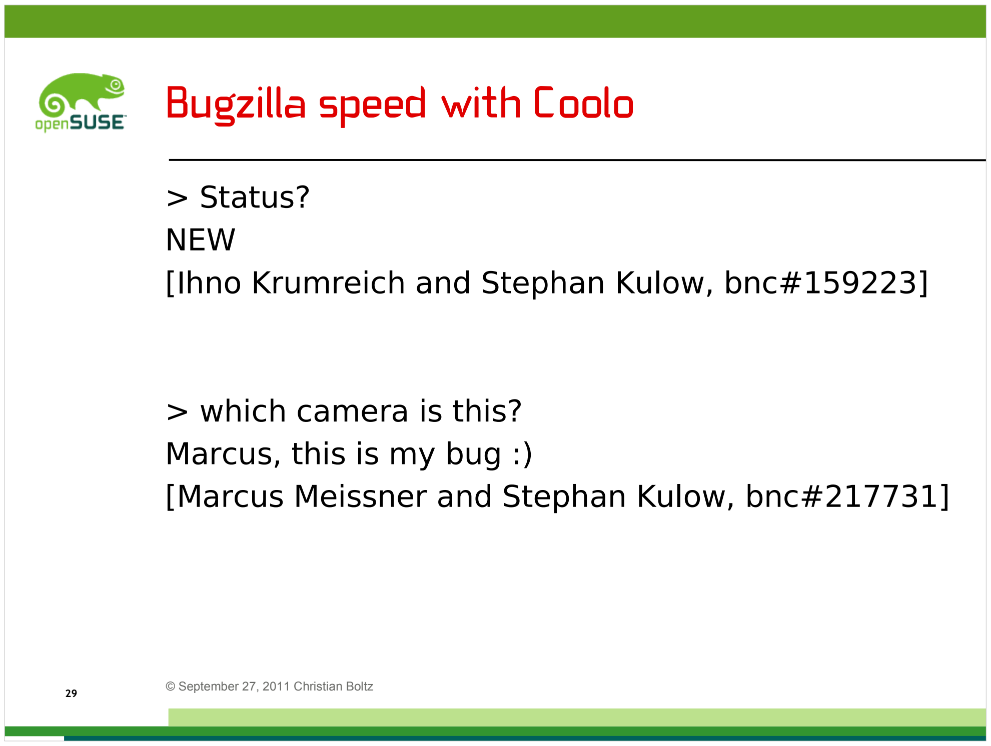
Or, more seriously, tell the calling application “memory allocation failure” and hope that it will handle this error in a sane way.

So the conclusion is...

Never think about error handling

Just expect your code to work.

- error handling is only boring programming work, it's much more exciting to fix bugs after deploying the software on production-critical systems
- better invest your time in developing new features
- thinking about errors is the job of bugreporters anyway



Bugzilla speed with Coolo

> Status?
NEW
[Ihno Krumreich and Stephan Kulow, bnc#159223]

> which camera is this?
Marcus, this is my bug :)
[Marcus Meissner and Stephan Kulow, bnc#217731]

Before we come to the top 10, here's some bugzilla speed comparison with Coolo.

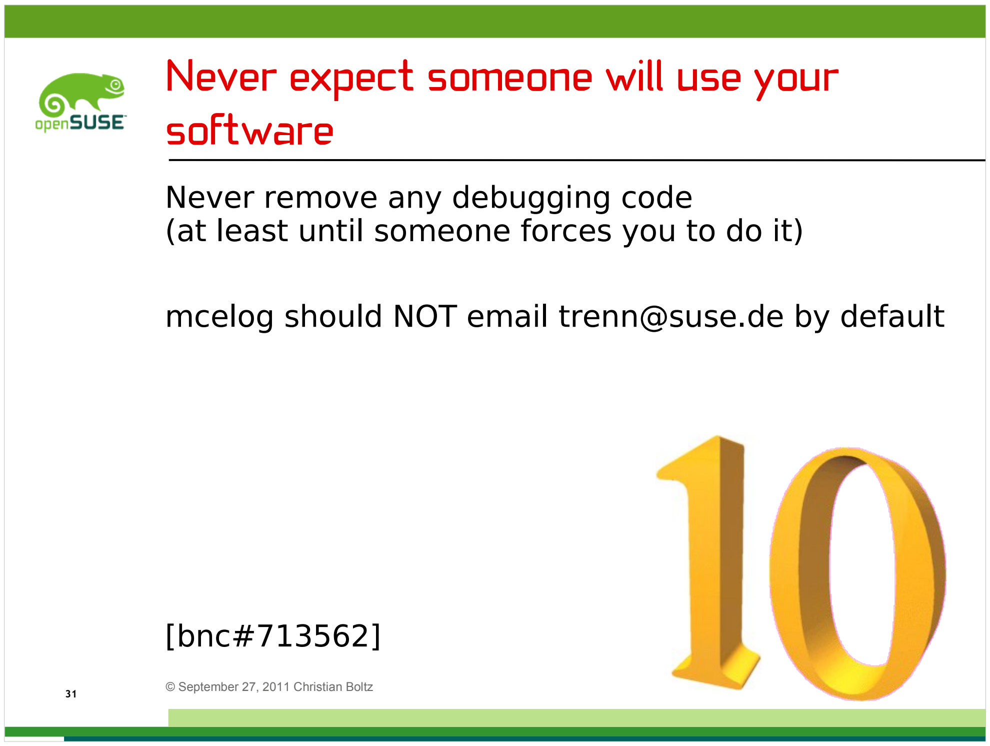
Never expect someone will use your software

Hardcoding your username is fine.

> mkdir: Can't create directory »»/home/ratti««
> Looks like you should use ~ instead of
> /home/ratti ;-)

Wait and see – 0.0.3 will even work if your name is not “ratti” ;-)

[Fontlinge development fun with Ratti, translated]



Never expect someone will use your software

Never remove any debugging code
(at least until someone forces you to do it)

mcelog should NOT email trenn@suse.de by default

[bnc#713562]

You probably have seen the online update to fix this bugreport some weeks ago ...

and finally the most prominent example of a person who never expected that his software would be used by anyone...

Never expect someone will use your software

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

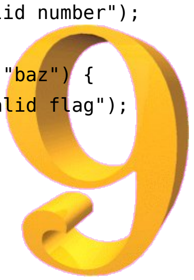
[Linus Torwalds, August 1991]

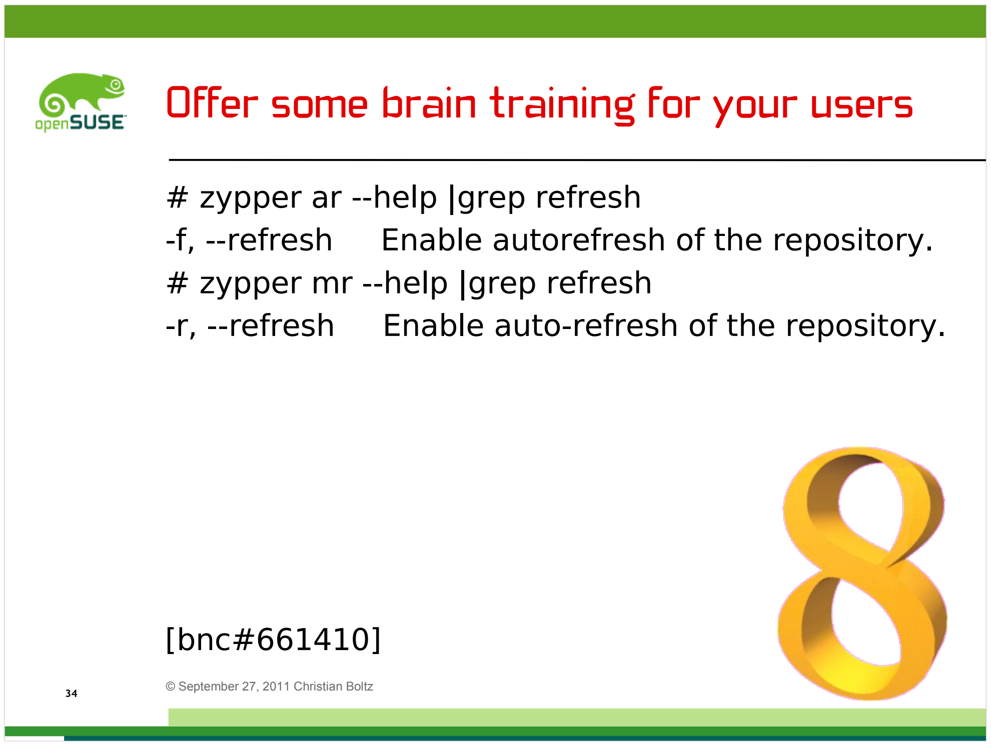
Nest your code as deep as possible

```
if ($name == “foo”) {
    if ($value == “bar”) {
        if ($number > 0) {
            if ($flags == "baz") {
                # 1000 lines of code
            } else {
                die ("invalid flag");
            }
        } else {
            die ("invalid number");
        }
    } else {
        die ("invalid value");
    }
} else {
    die ("invalid name");
}
```

Never do something like this:

```
if ($name != “foo”) {
    die("invalid name");
}
if ($value != "bar") {
    die ("invalid value");
}
if ($number <= 0) {
    die("invalid number");
}
if ($flags != "baz") {
    die ("invalid flag");
}
# 1000
# lines
# of
# code
```



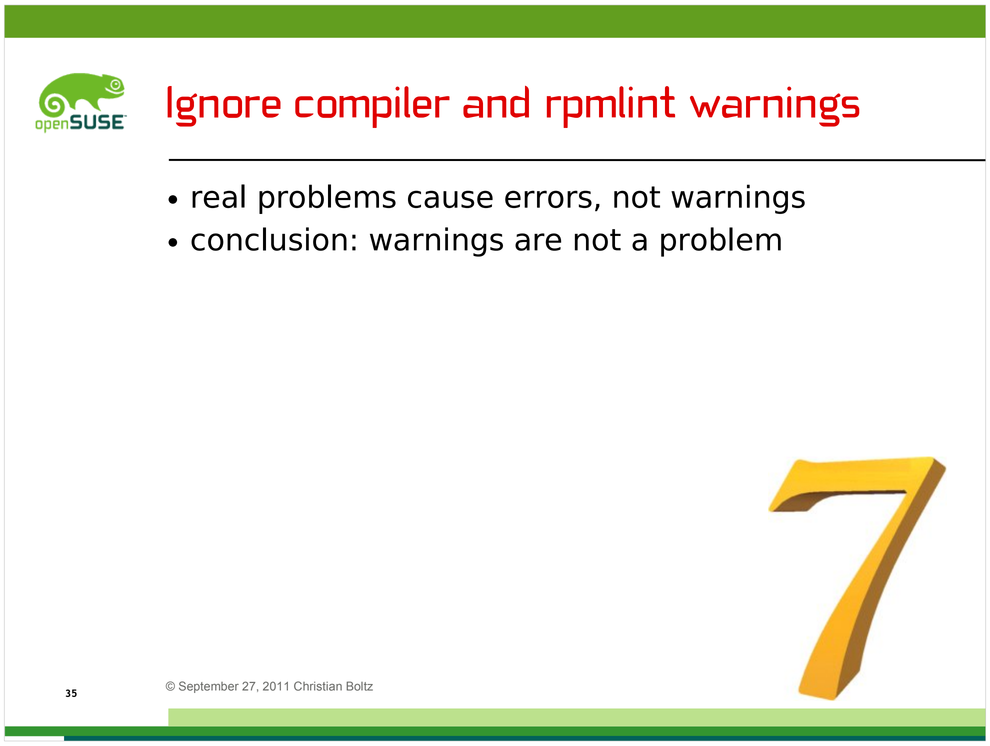


Offer some brain training for your users

```
# zypper ar --help |grep refresh
-f, --refresh     Enable autorefresh of the repository.
# zypper mr --help |grep refresh
-r, --refresh     Enable auto-refresh of the repository.
```

[bnc#661410]

it would be too boring if the same short option would work for both, right?



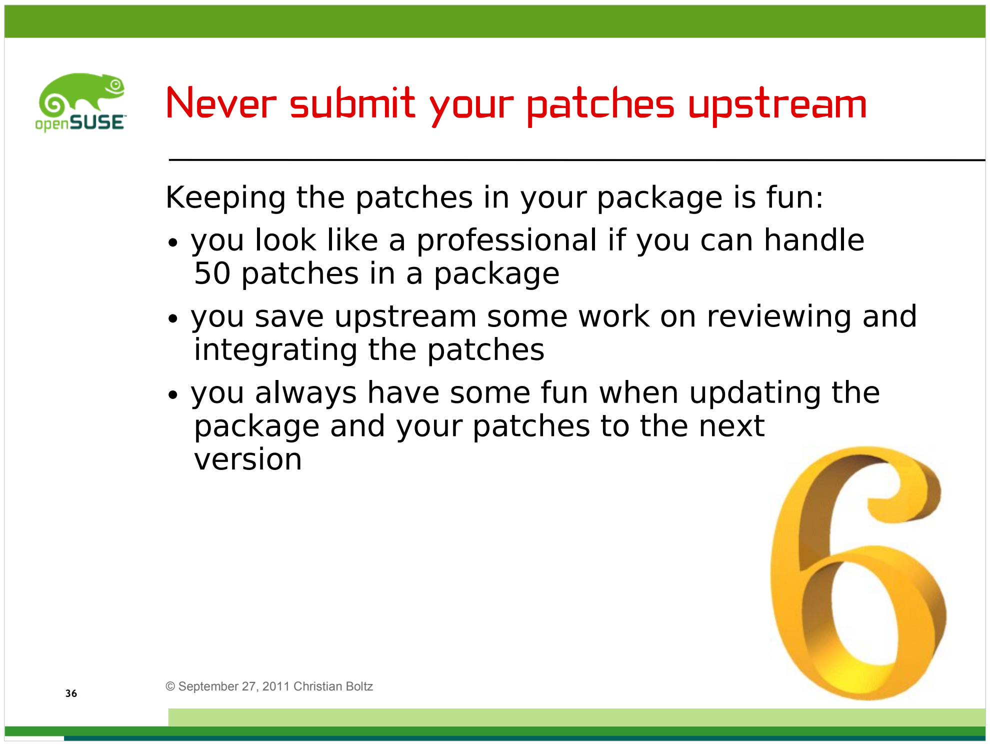
Ignore compiler and rpmlint warnings

- real problems cause errors, not warnings
- conclusion: warnings are not a problem

examples:

- uninitialized variables can't be real. The memory is always full, therefore the variable must have some content

- and it will never happen that you mistyped the variable name.



Never submit your patches upstream

Keeping the patches in your package is fun:

- you look like a professional if you can handle 50 patches in a package
- you save upstream some work on reviewing and integrating the patches
- you always have some fun when updating the package and your patches to the next version

This slide is dedicated to the AppArmor package, but probably applies to lots of packages.

I'd say the AppArmor package looks quote professional with 24 patches

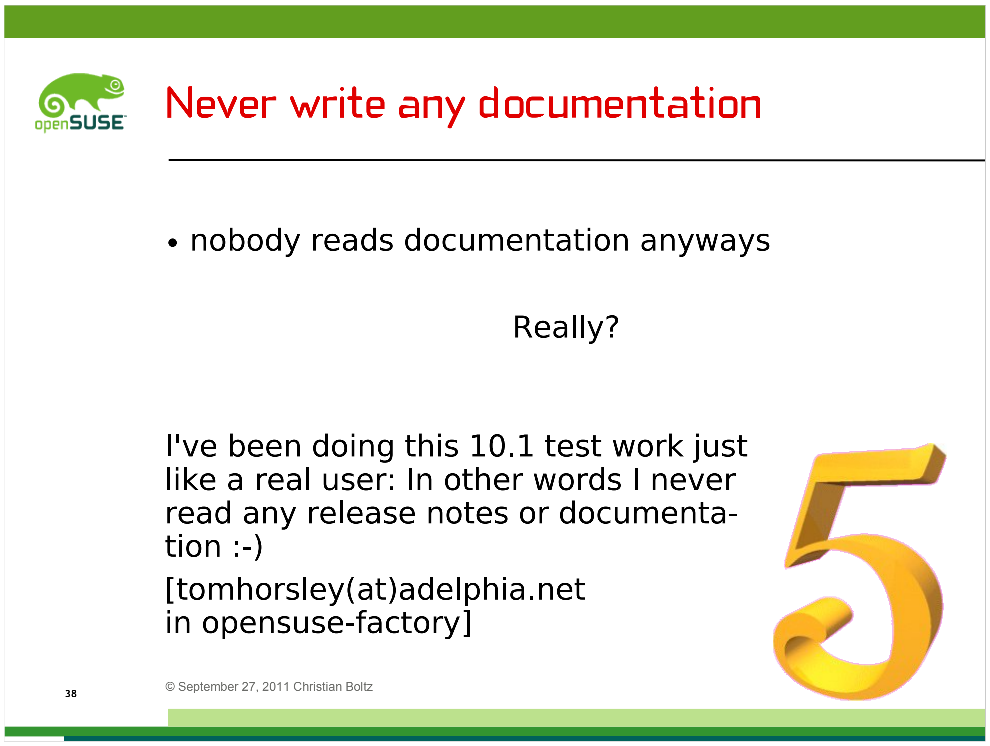
And I can confirm that upstreaming the patches causes some review work for upstream – I submitted most of the AppArmor patches upstream, and it took them some days until they turned up from my patch flood again.

The result is that as soon as someone updates the AppArmor package to 2.7 beta, less than 10 patches will be left. That someone might even be me, but unfortunately one of the patches is 370 kB big and doesn't have a real chance to be accepted upstream.

Final reason not to upstream patches: The openSUSE package should always be better than the packages in other distributions, and having some important patches others don't have indeed makes the package better.

Never write any documentation

- if some old documentation exists, never update it
- nobody reads documentation anyways
- comments in the code also count as documentation – avoid adding them whenever possible



Never write any documentation

- nobody reads documentation anyways

Really?

I've been doing this 10.1 test work just like a real user: In other words I never read any release notes or documentation :-)

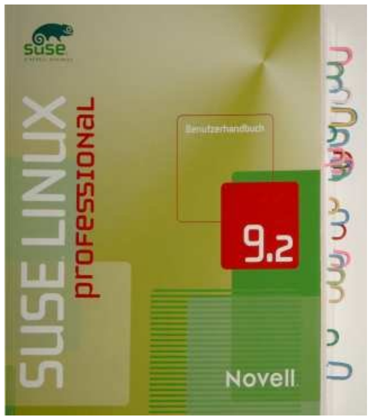
[tomhorsley(at)adelphia.net in opensuse-factory]

Well, most users don't read the documentation.

But it might happen that the BBfH strikes out again

Never write any documentation

- nobody reads documentation anyways



Really?
Beware of the paperclips!

[bnc#65000]

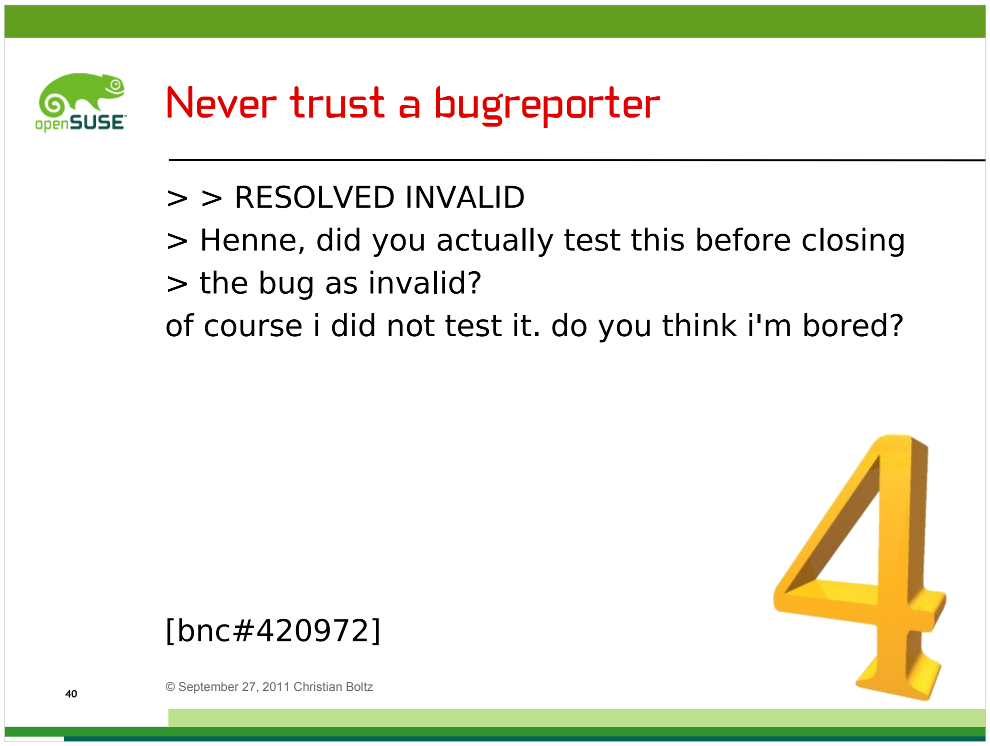
Some people might remember the printed manuals from the good old time. Here's one of them.

Each paperclip marks a bug in the “shell” chapter of the manual.

This results in a bugreport that is about 3 printed pages long.

First bugzilla comment:

@bugreporter: please do not ever touch a paperclip again ... :)



Never trust a bugreporter

> > RESOLVED INVALID
> Henne, did you actually test this before closing
> the bug as invalid?
of course i did not test it. do you think i'm bored?

[bnc#420972]

Bug reporters are evil, you remember?

They steal your time by reporting defects and expect them to be fixed. By YOU, of course.

And even worse, if you try to get rid of them by closing a bug as invalid, they reopen it...

In this case it was a regression in the courier-imap initscript.
Needless to say that the bugreport was valid...

Never trust a bugreporter

general rule: if Olaf reports a bug, it is a valid bug.
(Olaf Hering while reopening bnc#168595)

NEEDINFO fun

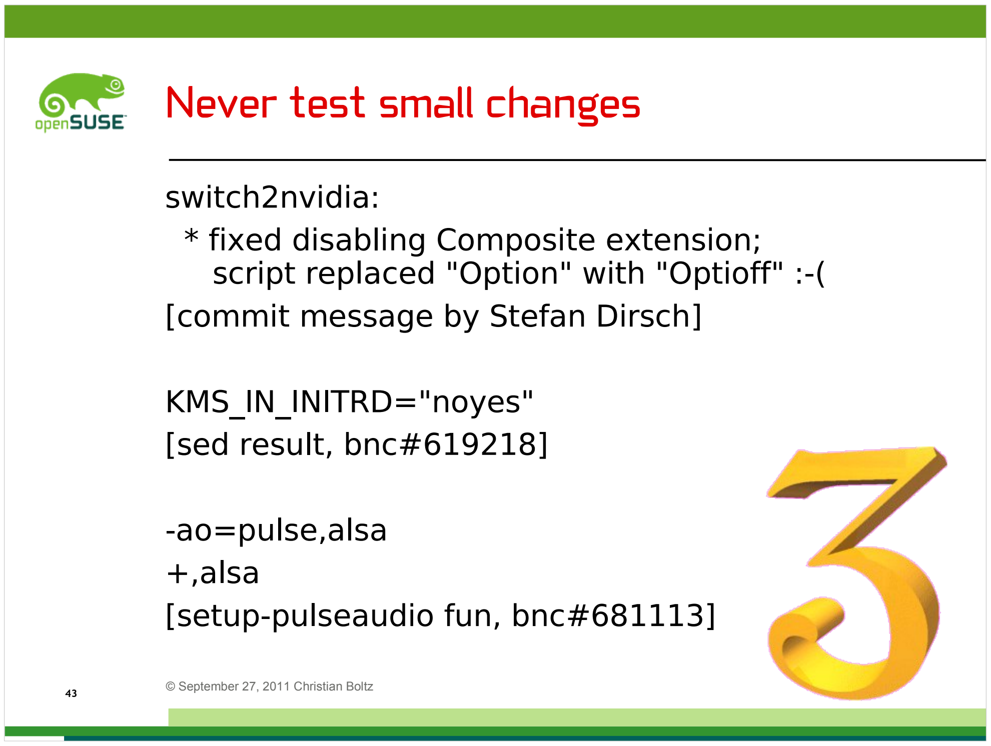
I am supposed to be the info provider,
so here is my answer:

42

By the way:

What is the question?

[Johannes Meixner, bnc#190173]



Never test small changes

switch2nvidia:
  * fixed disabling Composite extension;
    script replaced "Option" with "Optioff" :-(
[commit message by Stefan Dirsch]

KMS_IN_INITRD="noyes"
[sed result, bnc#619218]

-ao=pulse,alsa
+,alsa
[setup-pulseaudio fun, bnc#681113]

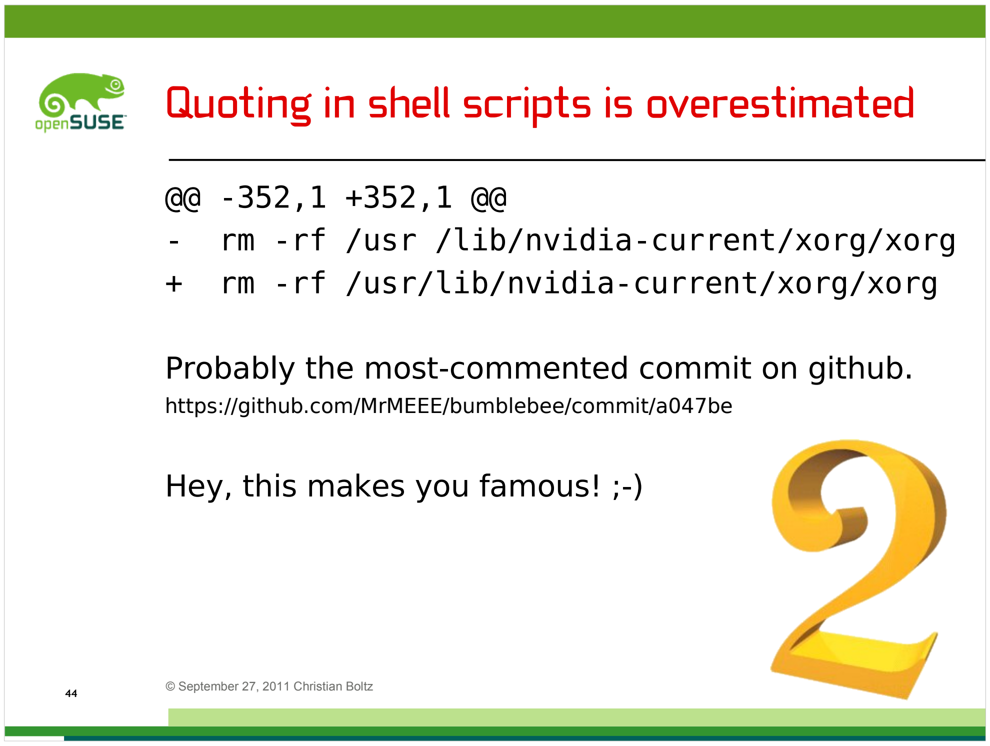
valid for changes and scripts up to 200 lines ;-)

You'll probably find lots of similar examples, but those are the most interesting ones I could find.

First the “optioff” - but it worked and successfully disabled the composite extension

In the second example, sed couldn't decide and finally put in “noyes” so that everyone is happy.

And finally setup-pulseaudio broke the mplayer config file. Additionally mplayer didn't report in which file the error was, which meant some fun until I had found out what happened.



Quoting in shell scripts is overestimated

```
@@ -352,1 +352,1 @@
-   rm -rf /usr /lib/nvidia-current/xorg/xorg
+   rm -rf /usr/lib/nvidia-current/xorg/xorg
```

Probably the most-commented commit on github.
https://github.com/MrMEEE/bumblebee/commit/a047be

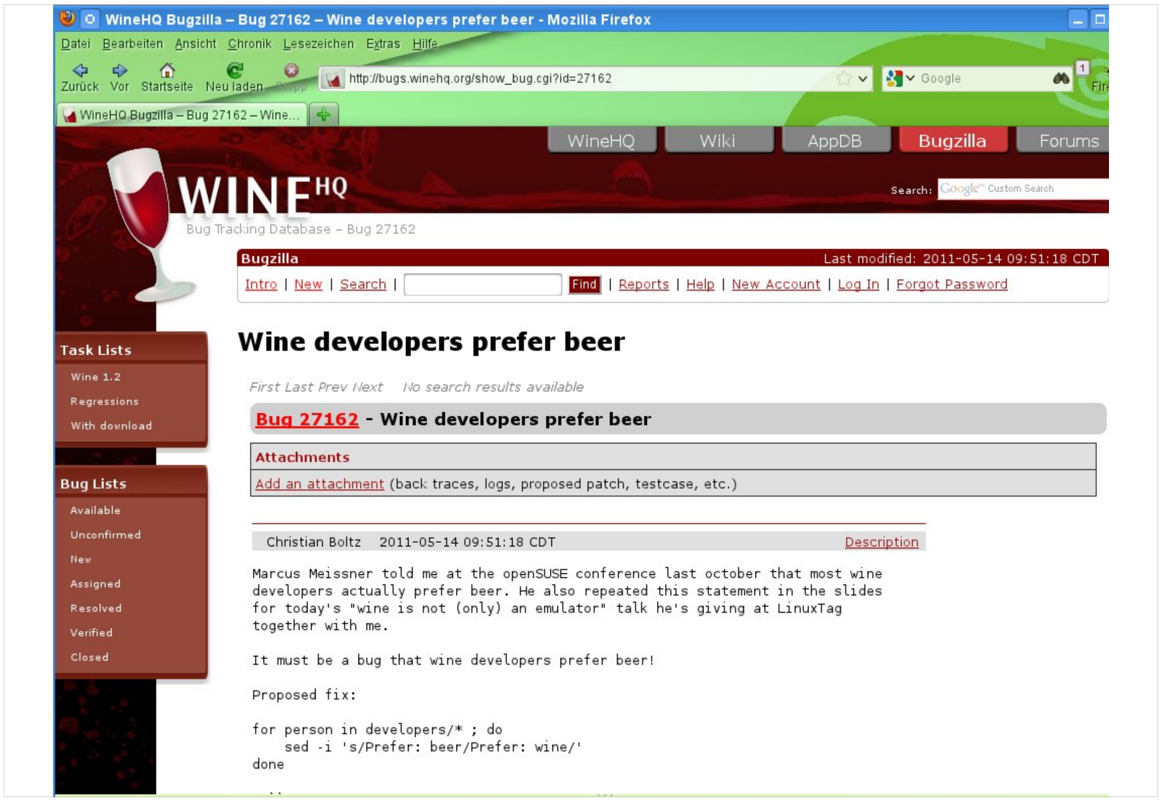
Hey, this makes you famous! ;-)

…

before we come to rule number one, I have to add something to my wine talk from last year.

You probably remember that Marcus said that most wine developers prefer beer.

Well, that must be a bug, and therefore I entered a bugreport at the wine bugtracker.


WineHQ Bugzilla – Bug 27162 – Wine developers prefer beer - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
Zurück Vor Startseite Neu laden
http://bugs.winehq.org/show_bug.cgi?id=27162
WineHQ Bugzilla – Bug 27162 – Wine...
WineHQ Wiki AppDB Bugzilla Forums
WINE HQ
Search:
Bug Tracking Database – Bug 27162
Bugzilla
Last modified: 2011-05-14 09:51:18 CDT
Intro | New | Search | Find | Reports | Help | New Account | Log In | Forgot Password
Task Lists
Wine 1.2
Regressions
With download
Bug Lists
Available
Unconfirmed
New
Assigned
Resolved
Verified
Closed

Wine developers prefer beer

First Last Prev Next No search results available

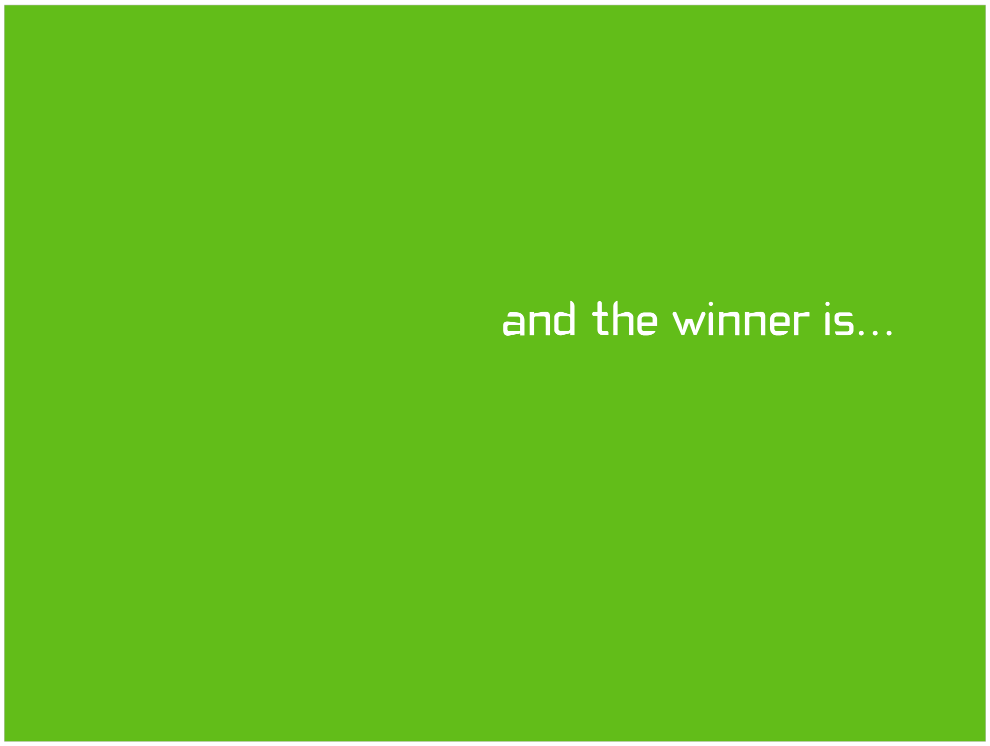
Bug 27162 - Wine developers prefer beer

Attachments

Add an attachment (back traces, logs, proposed patch, testcase, etc.)

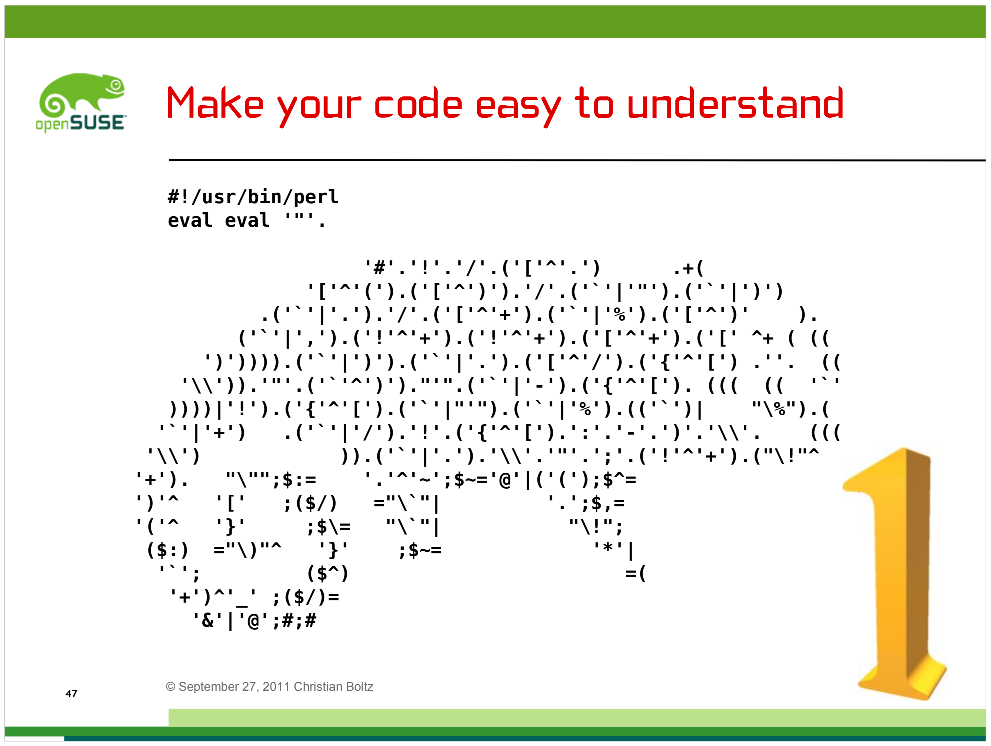
Christian Boltz 2011-05-14 09:51:18 CDT Description

```
Marcus Meissner told me at the openSUSE conference last october that most wine
developers actually prefer beer. He also repeated this statement in the slides
for today's "wine is not (only) an emulator" talk he's giving at LinuxTag
together with me.

It must be a bug that wine developers prefer beer!

Proposed fix:

for person in developers/* ; do
    sed -i 's/Prefer: beer/Prefer: wine/'
done
```


and the winner is...

Ladies and and gentlemen, here is number one of the golden rules of bad programming.

*** drum roll ***



Make your code easy to understand

```
#!/usr/bin/perl
eval eval '"'.

                    '#'.'!'.'/'.('['^'.')       .+(
               '['^'(').('['^')').'/'.('`'|'"').('`'|')')
           .('`'|'.').'/'.('['^'+').('`'|'%').('['^')'    ).
         ('`'|',').('!'^'+').('!'^'+').('['^'+').('[' ^+ ( ((
      ')')))).('`'|')').('`'|'.').('['^'/').('{'^'[') .''.  ((
    '\\')).'"'.('`'^')')."'".('`'|'-').('{'^'['). (((  ((  '`'
   ))))|'!').('{'^'[').('`'|'"').('`'|'%').(('`')|    "\%").(
  '`'|'+')   .('`'|'/').'!'.('{'^'[').':'.'-'.')'.'\\'.    (((
 '\\')            )).('`'|'.').'\\'.'"'.';'.('!'^'+').("\!"^
'+').   "\"";$:=    '.'^'~';$~='@'|('(');$^=
')'^   '['   ;($/)  ="\`"|         '.';$,=
'('^   '}'     ;$\=   "\`"|           "\!";
 ($:)  ="\)"^   '}'    ;$~=             '*'|
  '`';         ($^)                        =(
   '+')^'_' ;($/)=
     '&'|'@';#;#
```

I think this code speaks for itsself.

I'll pay a glass of wine for the first person who tells me what this little program does. Yes, it is valid perl code.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

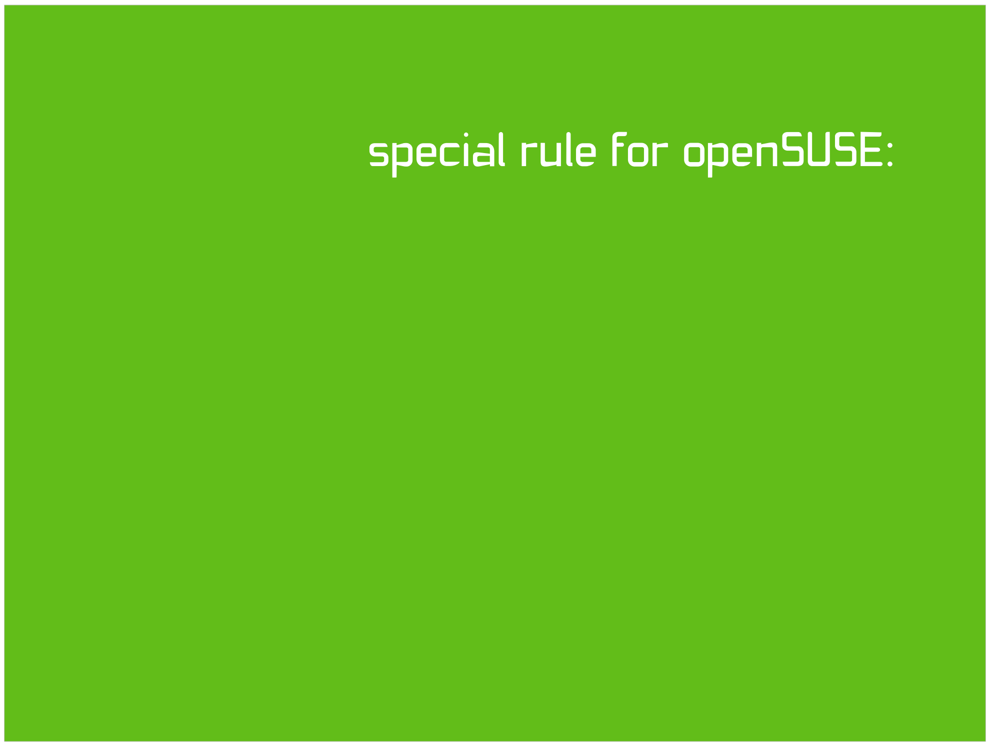
.

.

.

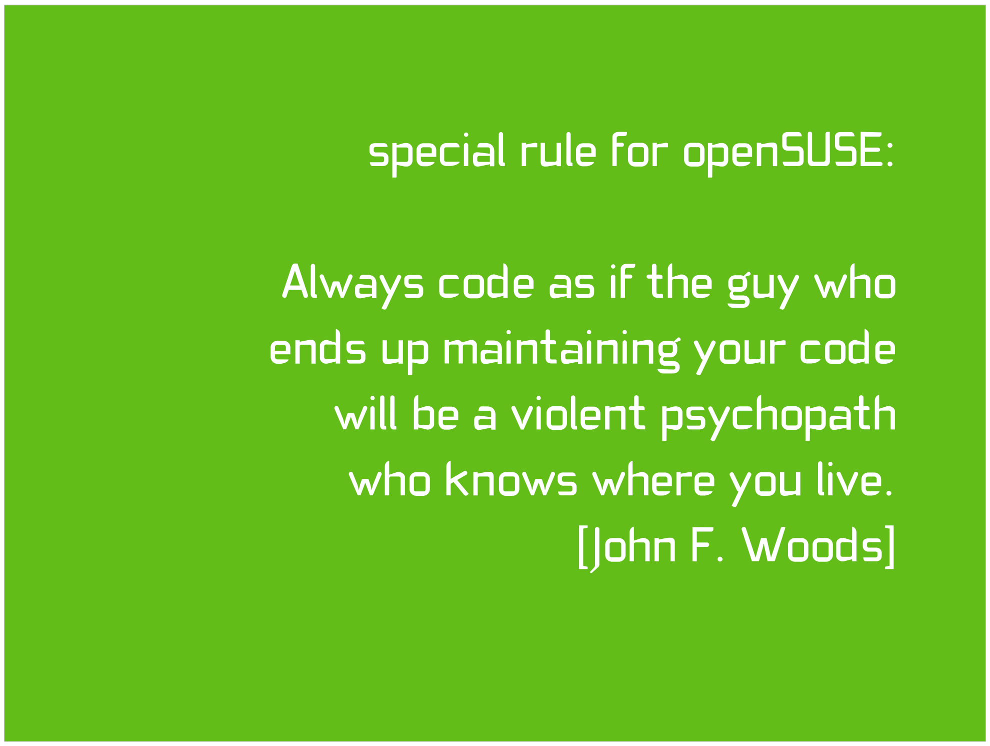
.

.



special rule for openSUSE:

There is one special rule for openSUSE, and it might be more important than the other rules I told you:



special rule for openSUSE:

Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. [John F. Woods]

There is one special rule for openSUSE, and it might be more important than the other rules I told you:

Thanks!

- everybody who accidently ;-) contributed to my talk
- for the inspiration by
  http://www.karzauninkat.com/Goldhtml/
  “golden rules of bad HTML” (german)
  http://www.sapdesignguild.org/community/design/golden_rules.asp
  “golden rules for bad user interfaces”
  http://blog.koehntopp.de/archives/2127-The-Importance-Of-FAIL.html
  http://blog.koehntopp.de/archives/2611-Was-bedeutet-eigentlich-Never-check-for-an-error-condition-you-dont-know-how-to-handle.html
  two great articles about FAIL by Kris Köhntopp
- perl Acme::EyeDrops for rendering rule #1
- for listening

Questions?

Opinions?

Flames?