What’s Up With DCHECKs

With Special Guest Peter

Intro

DCHECKs. You’ve seen them around and been asked to add them in code review, but what exactly are DCHECKS? Today we’re sitting down with Peter, who works on desktop and core UI, and is currently working on improving crash reports, which includes DCHECKs.

What is a DCHECK?

- Along with a CHECK, these enforce invariants that matter, asserting what you think is true actually is true
  - E.g. When a certain line is run, we evaluate the condition and see if it’s true. If it’s not, we crash
- Traditionally, DCHECKs only ran in debug builds, either locally or on trybots, so they incurred no overhead on a stable build
  - Think of the D in DCHECK as standing for ‘debug’ or ‘developer’
- D/CHECKs are good because like tests, they make sure future code changes don’t affect existing expectations, and are better than comments because they’ll let you know if the assertion they’re making no longer holds

static_assert

- D/CHECKs have similar goals to static_assert in C++, but differ in practice
  - static_assert is part of the C++ language and checks something that can be asserted at compile time. If a static_assert fails, your program won’t compile. This also means performance wise, they’re free!
  - D/CHECKs are evaluated at runtime, and are more similar to assert() in C++
- Is a language feature, not a library feature
- assert is part of the standard library and is a more basic form of a DCHECK. It doesn’t go through crash handling, and doesn’t give you a stack trace. We don’t use these in Chrome
- For example, if you have an array that is a constant size, this can be checked with a static_assert at compile time

History of D/CHECKs

- (As far as we know), CHECKs used to be more like logging. If they were hit, they would give you a file name and line number, log some stuff, then crash

    - This has size overhead, since you take the file name and line number for every instance, which adds to the binary size
    - As a result, historically, the guidance was to only use these when needed
- Now, CHECKs are just if conditions that crash. The file name and line number come from debugging symbols instead. You can add logging to the end of a CHECK, but they aren't necessarily included in the crash report
- CHECKs now have less debug info, but are much cheaper

When should I use a DCHECK vs a CHECK?

- Lean towards using a CHECK, unless you're in performance critical code
    - Examples of this would be a CHECK that gets evaluated a lot (i.e. every pixel for every frame in a video) or checking reachability in a graph which would jump all over memory
    - If you're not sure, it's probably not expensive
- EXPENSIVE_DCHECK explicitly states this DCHECK is expensive and as a result should remain a DCHECK
- There's no clear policy on this and is up to the owners, but hopefully there will be in the future

Security implications

- Security folks aren't fans of DCHECKs
- DCHECKs used to just be for developers for debug builds, which are super slow because of how huge Chrome is. That means if your code isn't well exercised under tests or in development, these checks are never actually run or enforced
- To combat this, DCHECKs are now on by default so whenever you build and run locally, they are in use and are able to provide much more coverage
- Now that CHECKs are much less expensive, they can be used instead of DCHECKs and the developer will actually know if it the CHECK they added crashed, which is good and what we want
    - This is important because a lot of DCHECKs are added in places where the program can get into a weird state if the DCHECK fails, but if the developer never finds out about the DCHECK crash, then they can't fix it
    - Crashing CHECKs also make it easier to debug because the program stops at that line. With a DCHECK, the program keeps running until maybe it crashes somewhere else in a weird state, which makes it much harder to debug

When your D/CHECK fails

- Code running **in the wild** means that it's running on users' machines, on one of the Chrome release channels (Canary, Dev, Beta, Stable)

- If your D/CHECK fails in the wild, a crash report is generated. This produces a stack trace that you can then use to debug what went wrong
- When a CHECK crashes in the wild, the user gets a sad tab
- Recently, on Windows, some users are getting non-crashing DCHECKs. This provides useful information on which assertions are failing and are helping to change the cultural impression of DCHECKs so people start to care about them they way they do about CHECKs and bring their perceptions closer
    - The exception here is expensive DCHECKs as mentioned before
- These checks are uploaded to the same crash reporting page as regular crashes, as DumpWithoutCrashing, and you will probably get a bug assigned to you if one of your DCHECKs fails often enough
- DumpWithoutCrashing runs on all release channels, while DCHECKs only will give crash reports on Canary

Debugging Crash Reports

- There are crash keys that can add logging to the crash dump when your CHECK or DumpWithoutCrashing is run, which can be useful for debugging
- You can also add additional checks to help isolate an issue
    - I.e. if x does not equal the expected value, 7, but you have an idea of why that might be, you can add checks to see if x == 6 or x == 8
- CHECK_GT(x,y) and CHECK_EQ(x,y) also exist, although the actual values of x and y are unfortunately lost in compilation on official builds

Callback to Pointers

- If you're CHECKing to see if something is non-null, that's a good indicator that the type should be SafeRef and raw_ref
- By using these types, the type does the checking for you, and reduces the number of invariants you then need to enforce
    - Plus anything else that uses the type gets the benefit of the built in enforcements as well
- You will also get more informative crash dumps as a result
- Always try to minimize the number of branches in your code!

Goal for DCHECKs in the future

- One possible future goal is for all DCHECKs to be CHECKs, unless they're notably expensive
- One way of working towards this is treating DCHECKs with the seriousness of checks, and the work on making DCHECKs run in the wild is working towards this goal

- Another part of the rollout is DumpWithoutCrashing, which generates a crash report like a failing CHECK, but without the crashing, as the name suggests
    - If all DCHECKs started crashing, that would make rolling out these changes very difficult and probably be a very poor user experience
- Current work is in place to make DCHECKs run on more platforms and channels

Join In

- For more, check out #base #cxx, and #halp on Slack