Linux kernel management style

This is a short document describing the preferred (or made up, depending on who you ask) management style for the linux kernel. It's meant to mirror the CodingStyle document to some degree, and mainly written to avoid answering (*) the same (or similar) questions over and over again.

Management style is very personal and much harder to quantify than simple coding style rules, so this document may or may not have anything to do with reality. It started as a lark, but that doesn't mean that it might not actually be true. You'll have to decide for yourself.

Btw, when talking about "kernel manager", it's all about the technical lead persons, not the people who do traditional management inside companies. If you sign purchase orders or you have any clue about the budget of your group, you're almost certainly not a kernel manager. These suggestions may or may not apply to you.

First off, I'd suggest buying "Seven Habits of Highly Successful People", and NOT read it. Burn it, it's a great symbolic gesture.

(*) This document does so not so much by answering the question, but by making it painfully obvious to the questioner that we don't have a clue to what the answer is.

Anyway, here goes:

Chapter 1: Decisions

Everybody thinks managers make decisions, and that decision-making is important. The bigger and more painful the decision, the bigger the manager must be to make it. That's very deep and obvious, but it's not actually true.

The name of the game is to _avoid_ having to make a decision. In particular, if somebody tells you "choose (a) or (b), we really need you to decide on this", you're in trouble as a manager. The people you manage had better know the details better than you, so if they come to you for a technical decision, you're screwed. You're clearly not competent to make that decision for them.

(Corollary:if the people you manage don't know the details better than you, you're also screwed, although for a totally different reason. Namely that you are in the wrong job, and that _they_ should be managing your brilliance instead).

So the name of the game is to _avoid_ decisions, at least the big and painful ones. Making small and non-consequential decisions is fine, and makes you look like you know what you're doing, so what a kernel manager needs to do is to turn the big and painful ones into small things where nobody really cares.

It helps to realize that the key difference between a big decision and a small one is whether you can fix your decision afterwards. Any decision can be made small by just always making sure that if you were wrong (and you _will_ be wrong), you can always undo the damage later by backtracking. Suddenly, you get to be doubly managerial for making

two inconsequential decisions - the wrong one _and_ the right one.

And people will even see that as true leadership (*cough* bullshit *cough*).

Thus the key to avoiding big decisions becomes to just avoiding to do things that can't be undone. Don't get ushered into a corner from which you cannot escape. A cornered rat may be dangerous - a cornered manager is just pitiful.

It turns out that since nobody would be stupid enough to ever really let a kernel manager have huge fiscal responsibility _anyway_, it's usually fairly easy to backtrack. Since you're not going to be able to waste huge amounts of money that you might not be able to repay, the only thing you can backtrack on is a technical decision, and there back-tracking is very easy: just tell everybody that you were an incompetent nincompoop, say you're sorry, and undo all the worthless work you had people work on for the last year. Suddenly the decision you made a year ago wasn't a big decision after all, since it could be easily undone.

It turns out that some people have trouble with this approach, for two reasons:

- admitting you were an idiot is harder than it looks. We all like to maintain appearances, and coming out in public to say that you were wrong is sometimes very hard indeed.
- having somebody tell you that what you worked on for the last year wasn't worthwhile after all can be hard on the poor lowly engineers too, and while the actual _work_ was easy enough to undo by just deleting it, you may have irrevocably lost the trust of that engineer. And remember: "irrevocable" was what we tried to avoid in the first place, and your decision ended up being a big one after all.

Happily, both of these reasons can be mitigated effectively by just admitting up-front that you don't have a friggin' clue, and telling people ahead of the fact that your decision is purely preliminary, and might be the wrong thing. You should always reserve the right to change your mind, and make people very _aware_ of that. And it's much easier to admit that you are stupid when you haven't _yet_ done the really stupid thing.

Then, when it really does turn out to be stupid, people just roll their eyes and say "Oops, he did it again".

This preemptive admission of incompetence might also make the people who actually do the work also think twice about whether it's worth doing or not. After all, if _they_ aren't certain whether it's a good idea, you sure as hell shouldn't encourage them by promising them that what they work on will be included. Make them at least think twice before they embark on a big endeavor.

Remember: they'd better know more about the details than you do, and they usually already think they have the answer to everything. The best thing you can do as a manager is not to instill confidence, but rather a healthy dose of critical thinking on what they do.

Btw, another way to avoid a decision is to plaintively just whine "can't

we just do both?" and look pitiful. Trust me, it works. If it's not clear which approach is better, they'll eventually figure it out. The answer may end up being that both teams get so frustrated by the situation that they just give up.

That may sound like a failure, but it's usually a sign that there was something wrong with both projects, and the reason the people involved couldn't decide was that they were both wrong. You end up coming up smelling like roses, and you avoided yet another decision that you could have screwed up on.

Chapter 2: People

Most people are idiots, and being a manager means you'll have to deal with it, and perhaps more importantly, that _they_ have to deal with _you_.

It turns out that while it's easy to undo technical mistakes, it's not as easy to undo personality disorders. You just have to live with theirs - and yours.

However, in order to prepare yourself as a kernel manager, it's best to remember not to burn any bridges, bomb any innocent villagers, or alienate too many kernel developers. It turns out that alienating people is fairly easy, and un-alienating them is hard. Thus "alienating" immediately falls under the heading of "not reversible", and becomes a no-no according to Chapter 1.

There's just a few simple rules here:
 (1) don't call people d*ckheads (at least not in public)
 (2) learn how to apologize when you forgot rule (1)

The problem with #1 is that it's very easy to do, since you can say "you're a d*ckhead" in millions of different ways (*), sometimes without even realizing it, and almost always with a white-hot conviction that you are right.

And the more convinced you are that you are right (and let's face it, you can call just about _anybody_ a d*ckhead, and you often _will_ be right), the harder it ends up being to apologize afterwards.

To solve this problem, you really only have two options:
 - get really good at apologies
 - spread the "love" out so evenly that nobody really ends up feeling like they get unfairly targeted. Make it inventive enough, and they might even be amused.

The option of being unfailingly polite really doesn't exist. Nobody will trust somebody who is so clearly hiding his true character.

(*) Paul Simon sang "Fifty Ways to Lose Your Lover", because quite frankly, "A Million Ways to Tell a Developer He Is a D*ckhead" doesn't scan nearly as well. But I'm sure he thought about it.

Chapter 3: People II - the Good Kind

While it turns out that most people are idiots, the corollary to that is sadly that you are one too, and that while we can all bask in the secure knowledge that we're better than the average person (let's face it, nobody ever believes that they're average or below-average), we should also admit that we're not the sharpest knife around, and there will be other people that are less of an idiot that you are.

Some people react badly to smart people. Others take advantage of them.

Make sure that you, as a kernel maintainer, are in the second group. Suck up to them, because they are the people who will make your job easier. In particular, they'll be able to make your decisions for you, which is what the game is all about.

So when you find somebody smarter than you are, just coast along. Your management responsibilities largely become ones of saying "Sounds like a good idea - go wild", or "That sounds good, but what about xxx?". The second version in particular is a great way to either learn something new about "xxx" or seem _extra_ managerial by pointing out something the smarter person hadn't thought about. In either case, you win.

One thing to look out for is to realize that greatness in one area does not necessarily translate to other areas. So you might prod people in specific directions, but let's face it, they might be good at what they do, and suck at everything else. The good news is that people tend to naturally gravitate back to what they are good at, so it's not like you are doing something irreversible when you _do_ prod them in some direction, just don't push too hard.

Chapter 4: Placing blame

Things will go wrong, and people want somebody to blame. Tag, you're it.

It's not actually that hard to accept the blame, especially if people kind of realize that it wasn't _all_ your fault. Which brings us to the best way of taking the blame: do it for another guy. You'll feel good for taking the fall, he'll feel good about not getting blamed, and the guy who lost his whole 36GB porn-collection because of your incompetence will grudgingly admit that you at least didn't try to weasel out of it.

Then make the developer who really screwed up (if you can find him) know _in_private_ that he screwed up. Not just so he can avoid it in the future, but so that he knows he owes you one. And, perhaps even more importantly, he's also likely the person who can fix it. Because, let's face it, it sure ain't you.

Taking the blame is also why you get to be manager in the first place. It's part of what makes people trust you, and allow you the potential glory, because you're the one who gets to say "I screwed up". And if you've followed the previous rules, you'll be pretty good at saying that by now.

Chapter 5: Things to avoid

There's one thing people hate even more than being called "d*ckhead", and that is being called a "d*ckhead" in a sanctimonious voice. The

first you can apologize for, the second one you won't really get the chance. They likely will no longer be listening even if you otherwise do a good job.

We all think we're better than anybody else, which means that when somebody else puts on airs, it _really_ rubs us the wrong way. You may be morally and intellectually superior to everybody around you, but don't try to make it too obvious unless you really _intend_ to irritate somebody (*).

Similarly, don't be too polite or subtle about things. Politeness easily ends up going overboard and hiding the problem, and as they say, "On the internet, nobody can hear you being subtle". Use a big blunt object to hammer the point in, because you can't really depend on people getting your point otherwise.

Some humor can help pad both the bluntness and the moralizing. Going overboard to the point of being ridiculous can drive a point home without making it painful to the recipient, who just thinks you're being silly. It can thus help get through the personal mental block we all have about criticism.

(*) Hint: internet newsgroups that are not directly related to your work are great ways to take out your frustrations at other people. Write insulting posts with a sneer just to get into a good flame every once in a while, and you'll feel cleansed. Just don't crap too close to home.

Chapter 6: Why me?

Since your main responsibility seems to be to take the blame for other peoples mistakes, and make it painfully obvious to everybody else that you're incompetent, the obvious question becomes one of why do it in the first place?

First off, while you may or may not get screaming teenage girls (or boys, let's not be judgmental or sexist here) knocking on your dressing room door, you _will_ get an immense feeling of personal accomplishment for being "in charge". Never mind the fact that you're really leading by trying to keep up with everybody else and running after them as fast as you can. Everybody will still think you're the person in charge.

It's a great job if you can hack it.