Chris Anderson: This is such a strange thing. Your software, Linux, is in millions of computers, it probably powers much of the Internet. And I think that there are, like, a billion and a half active Android devices out there. Your software is in every single one of them. It's kind of amazing. You must have some amazing software headquarters driving all this. That's what I thought -- and I was shocked when I saw a picture of it. I mean, this is -- this is the Linux world headquarters.
(Laughter)
(Applause)
Linus Torvalds: It really doesn't look like much. And I have to say, the most interesting part in this picture, that people mostly react to, is the walking desk. It is the most interesting part in my office and I'm not actually using it anymore. And I think the two things are related.
The way I work is ... I want to not have external stimulation. You can kind of see, on the walls are this light green. I'm told that at mental institutions they use that on the walls.
(Laughter)
It's like a calming color, it's not something that really stimulates you.
What you can't see is the computer here, you only see the screen, but the main thing I worry about in my computer is -- it doesn't have to be big and powerful, although I like that -- it really has to be completely silent. I know people who work for Google and they have their own small data center at home, and I don't do that. My office is the most boring office you'll ever see. And I sit there alone in the quiet. If the cat comes up, it sits in my lap. And I want to hear the cat purring, not the sound of the fans in the computer.
CA: So this is astonishing, because working this way, you're able to run this vast technology empire -- it is an empire -- so that's an amazing testament to the power of open source.
Tell us how you got to understand open source and how it lead to the development of Linux.
LT: I mean, I still work alone. Really -- I work alone in my house, often in my bathrobe. When a photographer shows up, I dress up, so I have clothes on.
(Laughter)
And that's how I've always worked. I mean, this was how I started Linux, too. I did not start Linux as a collaborative project. I started it as one in a series of many projects I had done at the time for myself, partly because I needed the end result, but even more because I just enjoyed programming. So it was about the end of the journey, which, 25 years later, we still have not reached. But it was really about the fact that I was looking for a project on my own and there was no open source, really, on my radar at all.
And what happened is ... the project grows and becomes something you want to show off to people. Really, this is more of a, "Wow, look at what I did!" And trust me -- it was not that great back then. I made it publicly available, and it wasn't even open source at that point. At that point it was source that was open, but there was no intention behind using the kind of open-source methodology that we think of today to improve it. It was more like, "Look, I've been

working on this for half a year, I'd love to have comments."
And other people approached me. At the University of Helsinki, I had a friend who was one of the open source -- it was called mainly "free software" back then -- and he actually introduced me to the notion that, hey, you can use these open-source licenses that had been around. And I thought about it for a while. I was actually worried about the whole commercial interests coming in. I mean, that's one of the worries I think most people who start out have, is that they worry about somebody taking advantage of their work, right? And I decided, "What the hell?" And --
CA: And then at some point, someone contributed some code that you thought, "Wow, that really is interesting, I would not have thought of that. This could actually improve this."
LT: It didn't even start by people contributing code, it was more that people started contributing ideas. And just the fact that somebody else takes a look at your project -- and I'm sure it's true of other things, too, but it's definitely true in code -- is that somebody else takes an interest in your code, looks at it enough to actually give you feedback and give you ideas. That was a huge thing for me.
I was 21 at the time, so I was young, but I had already programmed for half my life, basically. And every project before that had been completely personal and it was a revelation when people just started commenting, started giving feedback on your code. And even before they started giving code back, that was, I think, one of the big moments where I said, "I love other people!" Don't get me wrong -- I'm actually not a people person.
(Laughter)
I don't really love other people --
(Laughter)
But I love computers, I love interacting with other people on email, because it kind of gives you that buffer. But I do love other people who comment and get involved in my project. And it made it so much more.
CA: So was there a moment when you saw what was being built and it suddenly started taking off, and you thought, "Wait a sec, this actually could be something huge, not just a personal project that I'm getting nice feedback on, but a kind of explosive development in the whole technology world"?
LT: Not really. I mean, the big point for me, really, was not when it was becoming huge, it was when it was becoming little. The big point for me was not being alone and having 10, maybe 100 people being involved -- that was a big point. Then everything else was very gradual. Going from 100 people to a million people is not a big deal -- to me. Well, I mean, maybe it is if you're --
(Laughter)
If you want to sell your result then it's a huge deal -- don't get me wrong. But if you're interested in the technology and you're interested in the project, the big part was getting the community. Then the community grew gradually. And there's actually not a single point where I went like, "Wow, that just took off!" because it -- I mean -- it took a long time, relatively.
CA: So all the technologists that I talk to really credit you with massively changing their work. And it's not just Linux, it's this

thing called Git, which is this management system for software development. Tell us briefly about that and your role in that.
LT: So one of the issues we had, and this took a while to start to appear, is when you ... When you grow from having 10 people or 100 people working on a project to having 10,000 people, which -- I mean, right now we're in the situation where just on the kernel, we have 1,000 people involved in every single release and that's every two months, roughly two or three months. Some of those people don't do a lot. There's a lot of people who make small, small changes. But to maintain this, the scale changes how you have to maintain it. And we went through a lot of pain. And there are whole projects that do only source-code maintenance. CVS is the one that used to be the most commonly used, and I hated CVS with a passion and refused to touch it and tried something else that was radical and interesting and everybody else hated.
CA: (Laughs)
LT: And we were in this bad spot, where we had thousands of people who wanted to participate, but in many ways, I was the kind of break point, where I could not scale to the point where I could work with thousands of people.
So Git is my second big project, which was only created for me to maintain my first big project. And this is literally how I work. I don't code for -- well, I do code for fun -- but I want to code for something meaningful so every single project I've ever done has been something I needed and --
CA: So really, both Linux and Git kind of arose almost as an unintended consequence of your desire not to have to work with too many people.
LT: Absolutely. Yes.
(Laughter)
CA: That's amazing. LT: Yeah.
(Applause)
And yet, you're the man who's transformed technology not just once but twice, and we have to try and understand why it is. You've given us some clues, but ... Here's a picture of you as a kid, with a Rubik's Cube. You mentioned that you've been programming since you were like 10 or 11, half your life.
Were you this sort of computer genius, you know, übernerd, were you the star at school who could do everything? What were you like as a kid?
LT: Yeah, I think I was the prototypical nerd. I mean, I was ... I was not a people person back then. That's my younger brother. I was clearly more interested in the Rubik's Cube than my younger brother.
(Laughter)
My younger sister, who's not in the picture, when we had family meetings -- and it's not a huge family, but I have, like, a couple of cousins -- she would prep me beforehand. Like, before I stepped into the room she would say, "OK. That's so-and-so ..." Because I was not -- I was a geek. I was into computers, I was into math, I was into physics. I was good at that. I don't think I was particularly exceptional. Apparently, my sister said that my biggest exceptional quality was that I would not let go.
CA: OK, so let's go there, because that's interesting. You would not let go. So that's not about being a geek and being smart, that's

about being ... stubborn?
LT: That's about being stubborn. That's about, like, just starting something and not saying, "OK, I'm done, let's do something else -- Look: shiny!"
And I notice that in many other parts in my life, too. I lived in Silicon Valley for seven years. And I worked for the same company, in Silicon Valley, for the whole time. That is unheard of. That's not how Silicon Valley works. The whole point of Silicon Valley is that people jump between jobs to kind of mix up the pot. And that's not the kind of person I am.
CA: But during the actual development of Linux itself, that stubbornness sometimes brought you in conflict with other people. Talk about that a bit. Was that essential to sort of maintain the quality of what was being built? How would you describe what happened?
LT: I don't know if it's essential. Going back to the "I'm not a people person," -- sometimes I'm also ... shall we say, "myopic" when it comes to other people's feelings, and that sometimes makes you say things that hurt other people. And I'm not proud of that.
(Applause)
But, at the same time, it's -- I get people who tell me that I should be nice. And then when I try to explain to them that maybe you're nice, maybe you should be more aggressive, they see that as me being not nice.
(Laughter)
What I'm trying to say is we are different. I'm not a people person; it's not something I'm particularly proud of, but it's part of me. And one of the things I really like about open source is it really allows different people to work together. We don't have to like each other -- and sometimes we really don't like each other. Really -- I mean, there are very, very heated arguments. But you can, actually, you can find things that -- you don't even agree to disagree, it's just that you're interested in really different things.
And coming back to the point where I said earlier that I was afraid of commercial people taking advantage of your work, it turned out, and very quickly turned out, that those commercial people were lovely, lovely people. And they did all the things that I was not at all interested in doing, and they had completely different goals. And they used open source in ways that I just did not want to go. But because it was open source they could do it, and it actually works really beautifully together.
And I actually think it works the same way. You need to have the people-people, the communicators, the warm and friendly people who like --
(Laughter)
really want to hug you and get you into the community. But that's not everybody. And that's not me. I care about the technology. There are people who care about the UI. I can't do UI to save my life. I mean, if I was stranded on an island and the only way to get off that island was the make a pretty UI, I'd die there.
(Laughter)
So there's different kinds of people, and I'm not making excuses, I'm trying to explain.
CA: Now, when we talked last week, you talked about some other trait

that you have, which I found really interesting. It's this idea called taste.
And I've just got a couple of images here. I think this is an example of not particularly good taste in code, and this one is better taste, which one can immediately see. What is the difference between these two?
LT: So this is -- How many people here actually have coded?
CA: Oh my goodness.
LT: So I guarantee you, everybody who raised their hand, they have done what's called a singly-linked list. And it's taught -- This, the first not very good taste approach, is basically how it's taught to be done when you start out coding. And you don't have to understand the code.
The most interesting part to me is the last if statement. Because what happens in a singly-linked list -- this is trying to remove an existing entry from a list -- and there's a difference between if it's the first entry or whether it's an entry in the middle. Because if it's the first entry, you have to change the pointer to the first entry. If it's in the middle, you have to change the pointer of a previous entry. So they're two completely different cases.
CA: And that's better.
LT: And this is better. It does not have the if statement. And it doesn't really matter -- I don't want you understand why it doesn't have the if statement, but I want you to understand that sometimes you can see a problem in a different way and rewrite it so that a special case goes away and becomes the normal case. And that's good code. But this is simple code. This is CS 101. This is not important -- although, details are important.
To me, the sign of people I really want to work with is that they have good taste, which is how ... I sent you this stupid example that is not relevant because it's too small. Good taste is much bigger than this. Good taste is about really seeing the big patterns and kind of instinctively knowing what's the right way to do things.
CA: OK, so we're putting the pieces together here now. You have taste, in a way that's meaningful to software people. You're --
(Laughter)
LT: I think it was meaningful to some people here.
CA: You're a very smart computer coder, and you're hellish stubborn. But there must be something else. I mean, you've changed the future. You must have the ability of these grand visions of the future. You're a visionary, right?
LT: I've actually felt slightly uncomfortable at TED for the last two days, because there's a lot of vision going on, right? And I am not a visionary. I do not have a five-year plan. I'm an engineer. And I think it's really -- I mean -- I'm perfectly happy with all the people who are walking around and just staring at the clouds and looking at the stars and saying, "I want to go there." But I'm looking at the ground, and I want to fix the pothole that's right in front of me before I fall in. This is the kind of person I am.
(Cheers)
(Applause)
CA: So you spoke to me last week about these two guys. Who are they and how do you relate to them?
LT: Well, so this is kind of cliché in technology, the whole Tesla

versus Edison, where Tesla is seen as the visionary scientist and crazy idea man. And people love Tesla. I mean, there are people who name their companies after him.
(Laughter)
The other person there is Edison, who is actually often vilified for being kind of pedestrian and is -- I mean, his most famous quote is, "Genius is one percent inspiration and 99 percent perspiration." And I'm in the Edison camp, even if people don't always like him. Because if you actually compare the two, Tesla has kind of this mind grab these days, but who actually changed the world? Edison may not have been a nice person, he did a lot of things -- he was maybe not so intellectual, not so visionary. But I think I'm more of an Edison than a Tesla.
CA: So our theme at TED this week is dreams -- big, bold, audacious dreams. You're really the antidote to that.
LT: I'm trying to dial it down a bit, yes.
CA: That's good.
(Laughter) We embrace you, we embrace you.
Companies like Google and many others have made, arguably, like, billions of dollars out of your software. Does that piss you off?
LT: No. No, it doesn't piss me off for several reasons. And one of them is, I'm doing fine. I'm really doing fine.
But the other reason is -- I mean, without doing the whole open source and really letting go thing, Linux would never have been what it is. And it's brought experiences I don't really enjoy, public talking, but at the same time, this is an experience. Trust me. So there's a lot of things going on that make me a very happy man and thinking I did the right choices.
CA: Is the open source idea -- this is, I think we'll end here -- is the open source idea fully realized now in the world, or is there more that it could go, are there more things that it could do?
LT: So, I'm of two minds there. I think one reason open source works so well in code is that at the end of the day, code tends to be somewhat black and white. There's often a fairly good way to decide, this is done correctly and this is not done well. Code either works or it doesn't, which means that there's less room for arguments. And we have arguments despite this, right? In many other areas -- I mean, people have talked about open politics and things like that -- and it's really hard sometimes to say that, yes, you can apply the same principles in some other areas just because the black and white turns into not just gray, but different colors.
So, obviously open source in science is making a comeback. Science was there first. But then science ended up being pretty closed, with very expensive journals and some of that going on. And open source is making a comeback in science, with things like arXiv and open journals. Wikipedia changed the world, too. So there are other examples, I'm sure there are more to come.
CA: But you're not a visionary, and so it's not up to you to name them.
LT: No.
(Laughter)
It's up to you guys to make them, right?
CA: Exactly.
Linus Torvalds, thank you for Linux, thank you for the Internet,

thank you for all those Android phones.
Thank you for coming here to TED and revealing so much of yourself.
LT: Thank you.
(Applause)