Model Engineering College
Talk by Richard M. Stallman (INDIA 2001)
http://www.sslug.dk/~chlor/ogg/rms/trans_1.html
http://www.gnu.org/philosophy/stallman-mec-india.html
The_Danger_of_Software_Patents

Prof. Jyothi John, Head of Computer Engineering Department introduces Stallman:

It's my privilege and duty to welcome the most distinguished guest ever we had in this college.

Mr. Richard Mathew Stallman launched the development of the GNU operating system in 1984, the goal being to create a completely free Unix-like operating system. The organisation that was founded in 1985 to further this purpose is the Free Software Foundation.

Stallman is a visionary of computing in our times, and is the genius behind programs such as emacs, gcc, the GNU debugger and more. Most importantly, he's the author of the GNU general public licence, the licence under which more than half of all free software is distributed and developed. The combination of GNU with Linux, the kernel, called the GNU/Linux operating system, now has an estimated twenty million users worldwide.

Stallman's concept of free software talks about freedom, rather than about price. His ideas g a long way into ensuring development of software for the welfare of society, collectively developed by programmers who do not "lock up" their work, but rather release it for others to study, modify and redistribute.

Stallman received the Grace Hopper award from the Association for computing machinery for 1991, in 1990 he was awarded McArhtur Foundation Fellowship - other recipients of this prestigious award include Nom Chomsky and Tim-Berners Lee. In 1996, an honorary doctorate of Technology from the Royal Institute, Sweden was awarded to him. In 1998, he received the Electronic Frontier Foundation's Pioneer award, along with Linus Torvalds. In 1999 he received the Yuri Rubinski Memorial award.

Today, Stallman will be talking about the danger of software patents. In fact this is one of the most important aspect of the freedom of programming because the aspect of software patents may make all programmers potential lawbreakers because unknowingly they may be violating some of the patents registered by some other company.

Stallman's speech:

After that introduction, I am sure many of you want to know about Free Software. But unfortunately that's not what I am supposed to speak about. In fact, this topic, software patents, is not very closely related to the issue of Free Software. Software patents are a danger that affect all programmers and all computer users. I found out about them of course in working on Free Software because they are a danger to my project as well as to every other software project in the world.

There is a very unfortunate phrase that you may have heard. It is the phrase "intellectual property". Now, there are two things wrong with this phrase. One - it prejudges the most important policy question about how to treat some kind of ideas or practices or work. So, whatever it assumes

that they are going to be treated as some kind of property. Now, this is a public policy decision and you should be able to consider various alternatives to choose the best one. Which means you shouldn't name the whole field, name the question with with a term that prejudges what kind of answer you use.

But second and and even more fundamental, that term is actually a catalogue for totally different areas of law including copyrights, patents, trademarks, trade secrets and various other things as well. Now these areas of a law in fact have almost nothing in common. What the laws say is totally different from one to the next. Their origins are completely independent and the public policy issues that they raise are completely different. So the only intelligent way to think about them is to pick one of them and think about it; think about them separately.

So the intelligent way to talk about them is never to generalise about them but to talk about a specific one, you know, talk about copyrights or talk about patents or talk about trademarks, but never lump them all together as intellectual property because that is a recipe for simplistic conclusions. It is almost impossible to think intelligently about "intellectual property" and so I refuse to do that. I just tell people why the term is a mistake and then if you ask me for my opinion on copyrights or my opinion on patents it will take me an hour to tell you it. But they are two different opinions and my opinion on trademarks is something completely different as well.

So the most important thing for you to start with is never mix copyrights and patents as topics. They have nothing to do for each other. Let me tell you some of the basic differences between copyrights and patents: a copyright deals with a particular work, usually a written work and it has to do with the details of that work. Ideas are completely excluded from it. Patents, by contrast - well, patent covers an idea. It's that simple and any idea that you can describe - that's what a patent might restrict you from doing.

Now, copyrights have to do with copying; if you wrote something that was word for word the same as some famous novel and you could prove that you did this while you were locked up in a room and you have never seen that novel, this would not be copyright violation because it's not copying. But a patent is an absolute monopoly on using a particular idea. Even if you could show that you thought of it on your own that would be considered totally irrelevant. It doesn't help you.

Now, copyrights exist automatically. Whenever anything is written, it is copyrighted. Patents are issued through an expensive application process. There is an expensive fee and even more expense in paying lawyers, which of course tends to be good for big companies and the patent office says that it only issues patents for things that are unobvious. However, practically speaking, in many patent offices the criterion is unobvious to somebody with an IQ of fifty. And they have all sorts of excuses to ignore the fact that whenever any programmer looks at it his first statement is, "this is absurd, it's obvious." They say, "well, this is hindsight". So they have an excuse to completely ignore the judgement of everybody who really is a programmer.

Now, copyrights last an extremely long time. In the US today it's possible for copyrights to last for a 150 years, which is absurd. Patents don't last that long; they merely last for a long time - 20 years which in the field of software, as you can imagine is a long time. Now there are many other differences as well. In fact every detail is different. So the worst thing you should ever do is to learn something about copyrights and suppose that the same is true of patents. No, more likely it is not true of patents. If it's true of copyrights, it's not true of patents. That would be a better guideline if you had to guess.

Now most of the time when people describe how the patent system works they are people with a vested interest in the system. And so they describe the patent system from the point of view of

somebody who wants to get a patent and then point it at programmers and say, "hand me your money". This is natural, you know, when they sell lottery tickets they talk about people who win, not people who lose. Of course most of the people lose, but they don't want you to think about them and so they talk about the ones who win. It's the same with patents.

The patent system is a very expensive lottery for its participants. But of course the people who run the system want you to think about the small chance you might win. So to redress this imbalance, I am going to explain what the patent system looks like from the point of view of somebody who might be a victim of a patent. That is, somebody who wants to develop software. So, suppose that you want to develop a program and you were in a country that has software patents. How do you have to deal with the patent system.

Well, the first thing is you have to find out about the patents that might potentially affect your area. This is impossible, because patents that are in the pipeline being considered by the patent office, are secret. Well, in some countries they are published after 18 months but that still gives plenty of time for it to be secret. So you might develop a program this year, which is perfectly legal and safe this year. And then next year, a patent could be issued and all of a sudden you could be sued. It happens, or your users could get sued.

For instance, in 1984 the compress program was developed and since it was Free Software. It was distributed by many companies along with UNIX systems. Well, in 1985, a US patent was issued on the LZW compression algorithm used by compress. And after a few years UNISYS began squeezing money out of various companies.

Well, since we in the GNU project needed a data compression program and since we could not use compress we began looking for some other compression program. We found that somebody came forward and said, "I have been working on this algorithm for a year and now I have decided I am going to contribute it to you. Here is the code". We were a week away from releasing this program when I just happened to see a copy of the New York Times , which doesn't happen very often, and it just happened to have the weekly patents column and I noted it and so I read it. It said that somebody had got a patent for inventing a new method, a better method of data compression.

Well, that was not in fact true. When I saw this I thought we'd better get copy of this patent and see if it's a problem and it turned out to cover exactly the same algorithm that we were about to release. So this program was killed one week before it was released. And in fact that person, that patent holder had not invented a better method because in fact it wasn't new, but that doesn't matter. He had a monopoly.

So eventually we found another compression algorithm which is used in the program that's known as GISA. But this illustrates the danger that you face: even if you had unlimited resources, you couldn't find out about all the patents that might endanger your project. But you can find out about the issued patents because they are published by the patent office. So in principle you could read them all, and see what they restrict you from doing. Practically speaking, though, once there are software patents there are so many of them that you can't keep up with them.

In the US there are over a hundred thousand of them; may be two hundred thousand of them. This is just an estimate. I know that 10 years ago they were issuing 10,000 a year and I believe it has accelarated since then. So it's too much for you to keep track of them unless that's your full-time job. Now you can try to search for those that are relevant to what you are doing, and this works some of the time. If you search for keywords or follow links you'll find some patents that are relevant to what you are doing. You won't find them all.

Now, a few years ago somebody had a US patent - may be it's expired by now - on natural order recalculation in spreadsheets. Now, what does this mean ? It means the original spreadsheets did the recalculation always from top to bottom. Which meant, if a cell depends on a lower cell, then it wouldn't get recalculated the first time. You have to do another recalculation to get that one. Clearly it's better to do the recalculation in the order, you know, if A depends on B then do B first and then do A. This way a single recalculation will make everything consistent.

Well, that's what the patent covered. Now if you search for the term spreadsheet you would not have found that patent because that term did not appear in it. The phrase "natural order recalculation" didn't appear either. This algorithm, and it was indeed the algorithm they covered basically every imaginable way of coding this algorithm. The algorithm is called topological sorting and that term did not appear in the patent either. It presented itself as a patent on a technique for compilation. So reasonable searching would not have found this patent but it would have been a reason to sue you.

In fact you can't tell what a software patent covers even roughly, except by studying it carefully. This is different from patents in other areas because in other areas there is some physical thing happening and the details of that physical thing usually give you a sort of anger so that you can tell whether it relates or not. But in software there is no such thing and so it's easy for two totally different ways of saying something to cover, in fact, the same computation and it takes careful study to see that they cover the same one. Because of this even the patent office can't keep track. So there is not one, but two patents covering LZW data compression. The first one was issued in 1985 and the second, I think, in 1989.

But that one had been applied for even earlier. One of this patent belongs to Unisys and the other belongs to IBM. Now, this kind of mistake is not in fact that rare. It's not the only one.You see patent examiners don't have a lot of time to spend on one patent. In the US they have an average of 17 hours per patent. Now that's not enough to carefully study all the other patents in the area to see if they are really the same thing. So they are going to make this kind of mistake over and over. So you won't find all the patents that might threaten you but you might find some of them.

So then what do you do? You have to try to figure out precisely what these patents prohibit. That is very hard because patents are written in tortuous legal language which is very hard for an engineer to understand. You are going to have to work with a lawyer to do it.

In the 1980's the Australian Government commissioned a study of the patent system. The patent system in general, not software patents. This study concluded that Australia would be better off abolishing the patent system because it did very little good for society and cause a lot of trouble. The only reason they didn't recommend is that international pressure. So one of the things they cited was that patents which was supposed to disclose information so that they would no longer be secret or in fact useless, for that purpose, engineers never looked at patents to try and learn anything because it's too hard to read them. In fact they quoted that an engineer saying "I can't recognize my own inventions in patents".

Few years ago an engineer in US named Paul Heckel was sueing Apple. He had a couple of software patents in the late 80's for a software package and then when he saw hypercards and looked at inside - this is nothing like my program. He didn't think any more of it. But later on his lawyer explained to him that if you read this patent carefully hypercards fell into the prohibitted area. So he sued Apple feeling this is an opportunity to get some money. Well once when I give a speech like this, he was in the audience, and he said "oh no that's not true. I just wasn't aware of the scope of my protection" and I said "yeah, that's what I said".

So you are going to have to spend a lot of time working with a lawyer and explaining to the lawyer what projects you are working on, so that the lawyer can explain you what are the patents employed. This is going to be expensive when you're done the lawyer will tell you something like this : "If you do something in this area you are almost sure to loose a lawsuit. If you do something in this area you are in a substantial danger and if you really want to be safe you better stay out of this area and of course there is a substantial element of chance in the outcome of any lawsuit." So now that you have a predictable terrain for doing business what are you going to do .

Well, you have three options to consider - you can try to avoid the patent, you can try to licence the patent, or you can try to challenge its validity in court. Any one these three is sometimes a viable alternative and sometimes not.

First let's consider avoiding the patent. Well in some cases that's is easy. You know Unisys was frightening people using the patent on LZW compression which is have to find another data compression algorithm or we can avoid that patent. Well, that was somewhat difficult because there are many other patents covering lots of other data compression algorithms. But eventually we found one that was not in the area that those others' patents cover, eventually we did. So that program was implemented. It actually gave better compression results and so we now have GZIP and a lot of people use GZIP. So in that one case it was, there was considerable work and we were able to do it to avoid that patent.

But in the 80's Compuserv defined an image format called GIF and used LCW compression algorithm in defining it. Well, of course once the uproar about the patent become known, people defined another image format using a different compression algorithm. They used GZIP algorithm and that format is called the PNG format which I suppose to mean that PNG is Not GIF.

But there was a problem: lots of people have already started using GIF format and there were many programs that could display GIF format and produce GIF format but they couldn't display PNG format. So the result was people felt it too hard to switch. You see when you are dealing with a data compression program used by somebody who says "I want to compress some data or you can give them a data compression program". If he can get sued for using this one and you give him another one he will switch.

But what you want to do is make images that can be displayed by Netscape, then he can't switch unless Netscape handles the other format, and it didn't . It took years, I think, before Netscape is started to handle PNG format. So people essentially said "I can't switch, I just have't" and so the result was the society had invested so much in this one format but the inertia was too great for a switch even now there was another superior format available.

So even when a patent is rather narrow, avoiding it can be very hard. Postscript specifications includes LZW compression which we and our implementation of a postscript cannot implement. We supported another kind of compression in some sense that's not correct even though it does the useful job. So even a narrow patent is always feasible to avoid.

Now, some times a feature get patented. In that case, you can avoid the patents by taking out that feature. In the late 80's the users of the word processor ZIRITE got a downgrade in mail. That word processor had a feature where you could define a short word or sequence as an abbreviation. Whenever you type in that short sequence and then a space it would turn into a longer expansion. You could define this anywhere you write. Then somebody patented this and ZIRITE decided to deal with the patent by removing the feature. They contacted me because in fact I had put a feature like that into the original Emacs editor back in the 70's - many years before that patent. So there was a chance that I could provide evidence would enable them to fight the patent.

Well, this at least showed me that I had at least one patentable idea in my life. I know of it because someone else patented it. Now, of course you can respond to these patented features by taking the features out. Once your program starts being missing several features that users want, it might be useless as a program.

Now you may have heard of Adobe Photoshop. We have a program called the GIMP which is more powerful and general than Photoshop. But there is a one important feature that it doesn't have which is pantom colour matching. Which is very important for people who want actually print the images on paper and get reliable results. This feature is omitted because it is patented. And as a result is that the program for one substantial class of users is crippled. If you look at programs today you see that they are often provide many features and what the users demand is features. If any important feature is missing, well, it is easy to leave it out. But the results may be very bad .

Of course, sometimes a patent is so broad that is impossible to avoid it. Public key encryption is essential for computer users to have privacy. The whole field was patented. That patent expired just four years ago, so they could dream now Free Software in the US for public key encryption. Until then many programs free and non-free were wiped out by the Patent Office. And in fact that the whole area of computing was held back for more than a decade despite strong interest. So, that is the possibility of avoiding the patents.

Another possibility that is sometimes available is to licence the patent. Now the patent holder is not required to offer you licence that's his. The patent holder can say "I am not licensing this, you are just out of business. Period".

In the League for Programming Freedom we heard in the early 90's from somebody whose family business was making casino games, computerised of course, and he had then threatened by somebody who have a patent on a very broad category of computerised casino games. The patent covered a network where there is more than one machine and each machine support more than one type and they can display more than one game in progress at time. Now one thing you should realise is the patent office thinks that is really brilliant. If you see that other people implemented doing one thing and you decided to support two or more. You know if they made a system that plays one game and if you make it able to play more than one game that's an invention. If you can display one game and you decided to setup so that display two games at once that's an invention .

If he get with one computer and you do it with network having notebook computers thats an invention for them. They think that these steps are really brilliant. Of course we in Computer Science know this is just a rule that we can generalise anything from one to more than one. So most obvious principle there is. So every time you write a subroutine that's what you're doing. So this is one of the systematic reasons why the patent system produces and then oppose patents that we would all say are ridiculously obvious. You can presume just because it's ridiculously obvious that they wouldn't be upheld by a court. They may be legally valid despite the fact that are stupid. So he was faced with this patent and the patent holder was not even offering him the chance to get a licence. "Shutdown!" - that's what the patent holder said, and that's what he eventually did. He couldn't afford to fight it.

However many patent holders will offer you chance of a licence. But it cost you dearly. The owners of the natural order recalculation patent would be demanding five percent of the gross sales of every spreadsheet and that I was told was the cheap pre-lawsuit price. If you insisted on fighting over the matter they would be charging more. Now you could, I suppose, sign a licence like that for one patent, you could do it for two, you could do it for three. For what I think twenty different patents for your program and each patent holder wants five percent of the gross sales. What I have

said twenty - one of them disagrees, then you are pretty badly screwed.

But actually business people tell you that two or three such patents would be such a big burden that they would make the company fail. In practice, even if in theory, it might have the chance. So a licence for a patent is not necessarily a feasible thing to do and for us, the Free Software developers were in the worst position because we can't even count the copies and most licences domain in the field for copying so that's absolutely impossible for us to use one of these licences. You know, that if a licence charged one millionth part of a rupee for each copy, we would be unable to comply because we can't count the copies. The total amount of money that I might have in my pocket, but I can't count it so I can't pay it. So we suffers some special burden occasionally.

But there are one kind of organisations for which licensing patents works very well and that is the large multinational corporations and the reason is that they own many patents themselves and they use them towards cross licensing. What does that mean? Well, essentially the only defence against patents is deterrence, you have to have patents of your own. Then you hopes that somebody points patented you, you will be able point patent back and say "don't sue me, because I'll sue you".

However, deterrence doesn't work as well for patents as it does with nuclear weapons and the reason is that each patent is pointed in a fixed direction. It prohibits certain specified activities. So the result is that most of the companies that are trying to get some patents to defend themselves with. They have more chance of making a success. They might get a few patents that might get a patent that points there and they might get a patent that points there. And then if somebody over here threatens this company what are they going to do? They have a patent pointing over there, so they have no defence.

Meanwhile, sooner or later, somebody else who wander over there and the executive of the company will think "gee, we're not as profitable as I would like, why don't I like to squeeze some money out of them." So they say first to getting this patents for defensive purposes. But they often change their minds later when a victim wants to buy. And this by the way the reason why the fallacy in the myth that the patent system "protect" the "small inventor".

Let me tell the message in the myth of the starving genius. If somebody who's been working in isolation for years and starving and he has a brilliant new idea for how to do something or other. And so, now, he's starting a company and he is afraid some big companies like IBM will compete with him and so he gets a patent and this patent will "protect him". Well, of course, this is not the way of things work out in fields. People won't make this kind of progress in isolation is where you working with other people and talking with other people and developing software usually. And so the whole scenario doesn't make sense and besides, if there's such a good computer scientist there is no need for him to starve. He could have got a job at any time if he wanted.

But let's suppose this happened, and suppose he has this patent, and he says "IBM, you can't compete with me because I have got this patent ". But here is what IBM says: "well, gee, let's look at your product, hmm, I have this patent, this patent and this patent and this patent and this patent that your patent is violating. So how about a quick cross-licence?". And the starving genius says "hmm, I haven't had enough food in my belly to fight these things, so I better give in" and so they sign a cross-licence and now guess what .

IBM can compete with him - he wasn't protected at all. Now IBM can do this because they have a lot of patents. They have patents pointing here, here, here, everywhere. So that anybody from almost anywhere that attacks IBM is facing a stand-off. A small company can't do it but a big company can.

So IBM brought an artcle it was in Stink magazine, I believe issue number five of 1990, that's IBM's own magazine - an article about IBM's patent portfolio. IBM said that they have two kinds of benefits form it's 9000 active U.S patents. One benefit was collecting royalties from licences. But the other benefit, the bigger benefit, was access to things patented by others. From mission to not to be attacked by others but with their patents through cross licensing. And the article said that the second benefit was an order of magnitude greater than the first.

In other words, the benefit of IBM is to make it things freely, not being sued, was ten times the benefit of collecting money from all their patents. Now the patent system is lot like a lottery, in that the what happens with any given patents is largely around them and most of them don't get bring benefits of their owners. But IBM is so big that these things average out over the scale of IBM. So you can say IBM is measuring what the average is like. What we see is, and this is a little bit sudden, the benefit of IBM of being able to make use of ideas patented by others is equal to the harm that the patent system would have done to IBM if there were no cross licensing.

If the IBM were really prohibited from using all those ideas that were patented by others, so what I said is the harm that the patent system would do is ten times the benefit, on the average. Now, for IBM though, this harm doesn't happen. Because IBM does have 9000 patents and thus forces licences and cross-licences and avoids the problem. But if you were small, then you can't avoid the problem that way and you will really be facing ten time as much trouble as benefit. Anyway, this is why the big multinational corporations are in favour of software patents and they are lobbying governments around the world to adopt software patents and saying naive things like "this is a new kind of monopoly for software developers that has to be good for them, right ?"

Well, today, after you have heard my speech I hope you understand why that isn't true. You have to look carefully at how patents affect software developers to see whether they are good or bad, and explaining that is my overall purpose.

So, that is the possibility of licensing a patent. The third possible option is to go to court and challenge the validity of the patent. Now the outcome of this case will depend largely on technicalities, which means essentially on randomness.

You know, the dice were rolled a few years ago and you can investigate and find out what the dice came up saying and then you will find out you have got a chance. So its mainly historical accident that determines whether the patent is valid. The historical accident of whether precisely which things people happen to publish and when. So, sometimes, there is a possibility of invalidating. So even if a patent is ridiculously trivial sometimes there is a good chance of invalidating and sometimes there is not.

You can't expect the courts to recognise that it is trivial, because their standards are generally much lower than we would think are sensible. In fact, in the United States, this is been a persistent tendency. I saw a supreme court decision from something like 1954 which had a long list of patents that were invalidated by the Supreme Court starting in the 1800's. And they were utterly ridiculous like making a shape of door-knob out of rubber when previously they've been made out of wood. And this decision rebuked the patent system for going far far away from the proper standards, and they just keep on doing it.

So you can expect sensible results from that, but there are situations where when you look at the past record, you see there is a chance to invalidate a certain patent. It is worth the try, at least to investigate. But the actual court case is likely to be extremely expensive.

A few years ago, one defendant lost and had to pay 13 million dollars of which most went to

the lawyers on the two sides. I think only 5 million dollars was actually taken away by the patent holder and so then there were 8 million to the lawyers.

Now, these are your possible options. At this point, of course, you have to write the program. And there the problem is, that you face this situation not just once but over and over and over because programs today are complicated. Look at a word processor, you'll see a lot of features. Many different things each of which could be patented by somebody, or a combination of two of them could be patented by somebody. British Telecom has a patent in US on the combination of following hypertext links and letting the users dial up through a phone line. Now these are two basically separate things, but the combination of the two is patented.

So, that means if there are a 100 things in your program there are potentially some five thousand copairs of two that might be patented by somebody already and there is no law against patenting a combination of three of them either. So that's just the features, you know, there are going to be many techniques that you use in writing the programs, many algorithms they could be patented too. So there are lots and lots of things that could be patented and the result is that developing a program becomes like crossing a field of land mines. Sure, each step probably will not step on a patent, chances are it will be safe. But crossing a whole field becomes dangerous.

The best way for a nonprogrammer to understand what this is like is to compare the writing of these large programs with another area in which people write something very large symphonies. Imagine if the governments of Europe in the 1700's had wanted to promote progress in symphonic music by adopting a system of music patents. So that any idea that could be described in words could be patented if it seem to be new and original. So you'd be able to patent, say your three note melodic motive which is too short to be but it would have been patentable and may they could have patented a certain chord progression and may be patented using a certain combination of instruments playing at the same time or any other idea that somebody could describe.

Well, by 1800 there would have been thousands of these music idea patents. And then imagine that you are Beethoven and you want to write a symphony. To write a whole symphony, you are going to have to do lots of different things and at any point you could be using an idea that somebody else has patented. Of course, if you do that, he'll say "oh! you are just a thief, why can't you write something original". Well Beethoven had more than his share of new musical ideas. But he used a lot of existing musical ideas. He had to, because that is the only way to make it recognisable. If you don't do that, people won't listen at all. Paebulus thought he was going to totally reinvent the language of music and he tried and nobody listens to it because it doesn't use all the ideas that they were familiar with. So you have to use the old ideas that other people have thought of.

Nobody is such a genius that he can reinvent the entire field of software and do useful things without learning anything from anybody else. So in effect, those people were patent holders and their lawyers, they were accusing us of being cheaters because we don't totally reinvent the field from scratch. We have to build on previous work to make progress and that is exactly what the patent system is prohibits us from doing. We have to provide features that the users are accustomed to and can recognise where they'll find our software just too difficult to use no matter how good it is.

Now, people sometimes ask me why is software different from other fields. Sometimes, of course they ask this in a rather nasty fashion, they say the other fields can deal with patents why should software be an exception ? Now that's a nasty way of putting it because its making the assumption that it is wrong to want to escape from a problem. I could imagine I am saying when other people could get cancer, why shouldn't you ? Clearly, if it is a problem, enabling any field to

escape is good. But it is a good and serious question "are these fields in the same issue" ? The patents affect all these fields the same way ? Is the right policy for the software the same as the right policy for automobile engines or pharmaceuticals or chemical processes, you know, this is a serious question which is worth looking at. When you look at it, what you see is that the relationship between patents and products varies between the fields.

At one extreme you have pharmaceuticals were typically a whole chemical formula is patented. So if you come up with a new drug then its not patented by somebody else. At the other extreme is software were when you write a new program, you are combining dozens or hundreds of ideas and we can't expect them all to be new. Even in an innovative program which has the few new ideas has to use lots and lots of old ideas too. And in between you find the other fields. Even in other fields, you can get patent deadlocks.

When the United States entered the World War I, nobody in the US could make a modern airplane. And the reason was that modern airplanes use several different techniques that were patented by different companies and the owners hated each other. So nobody could get a license to use all these patents. Well, the US Government decided that this was an unacceptable state of affairs and essentially, paid those patent holders a lump sum and said we have nationalised these patents and now everybody go make airplanes for us. But the amount to which this happens, the frequency and the seriousness of it varies according to how many different ideas go in one product. It varies according to how many points of patent vulnerability there are in one product. And in that question, software is at the extreme.

Its not unusual of for a few people working for a couple of years to write a program that could have a million parts in it, different parts which is maybe, say 300,000 lines of code. To design a physical system that has a million different parts, that's a major project, that's very rare. Now you find many times people make a physical object with a million parts, but its typically many copies of the same subunit and that's much easier to design - that's not a million parts in the design. Now, so, why is this ?

The reason is that in other fields people have to deal with the propensity of matter. When you are designing circuits or cars or chemicals, you have to face the fact that these physical substances will do what they do, not what they are supposed to do. We in software don't have that problem and that makes it tremendously easier. We are designing a collection of idealised mathematical parts which have definitions. They do exactly what they are defined to do.

And so there are many problems we don't have. For instance, if we put an if statement inside a while statement, we don't have to worry about whether the if statement can get enough power to run at the speed its going to run. We don't have to worry about whether it would run at a speed that would generates radio frequency interference in it and induces wrong values in some other parts of the data. We don't have to worry about whether it would loop at a speed that causes resonance and eventually the if statement will vibrate against the while statement and one of them will crack. We don't have to worry that chemicals in the environment will get into the boundary between the if statement and the while statement and corrode them and cause a bad connection.

We don't have to worry other chemicals would get on them and cause short-circuit. We don't have to worry about whether the heat can be dissipated from the if statement through the surrounding while statement. We don't have to worry about whether the while statement would cause so much of voltage drop and the if statement so that the if statement won't function correctly. When you look at the value of a variable you don't have to worry about whether you've referenced that variable so many times that you exceed the fan out limit. You don't have to worry about how much capacitance there is in a certain variable and how much time it will take to store the value in

it.

All these things are defined a way and the system is defined to function in a certain way and it always does. The physical computer might not function, but that's not the programs fault. So because of all these problems we don't have to deal with, our field is tremendously easier. If you assume that the intelligence of programmers is the same as the intelligence of mechanical engineers, electrical engineers and chemical engineers and so on, what's going to happen. Those of us with the easiest field, fundamentally, are going to push it further. We make bigger and bigger the things and eventually it becomes hard again.

And that's why we can develop much bigger systems than people in the other fields. They just have these hard problems to deal with all the time.

In the other fields, it may be necessary to develop an idea, you may have the idea but then you may have to try out lots of different ways to get it work at all. In software now, like that, you have the idea and what you go and do is write a program which uses this idea and then the users may like it or not. And if they don't like it, probably you can just fix some details and get it to work. There is another problem that we don't have to worry about - manufacturing of copies.

Well, when we put the if statement inside the while statement, we don't have to worry about how the if statement is going to be inserted into the while statement as a copy is being built. We don't have to worry either about making sure we have access to remove and replace the if statement if it should burn out. So all we have to do is take copy in its all purpose copy anything facility. People making physical recruitment and physical products they can do that these things has to be built piece by piece each time. The result is that for them, the cost of designing a system of certain complexity may be (gesturing) this much and the factory may take this much to set up. So they have to deal with this much with the patent system, its a level of overhead they can live with.

For us, designing it may cost (gesturing) this much and manufacturing it may cost this much, so this much overhead from the patent system is crushing. Another way to look at it is that because we can, a few of us can, make a much bigger system, there are many more points of vulnerability where somebody might have patented something already.

We have to walk a long distance through the mine field where they they only have to walk a few feet through the minefield. So its much more of a dangerous system for us. Now, you have to realise that the extensible purpose of the patent system is to promote progress. This is something that is often forgotten because the companies that benefit from patents like to distract you from it. They like to give you the idea that patents exist because they deserves special treatment. But this is not what the patent system says.

Patent system says, the goal is to promote progress for the society. By encouraging certain behaviour like publishing new ideas and after certain - originally that was fairly short - time, everyone could use them. Of course there is a certain price that the society pays as well and we have to ask the question which is bigger - the benefit or the price. Well, in other fields, I am not sure. I am not an expert on other fields of engineering, I've never done them and I don't know whether having patents is good for progress in those fields.

I have been in software since before software patents existed and I know that software patents do a lot of harm and essentially no good. In the old days, ideas came along, either people in a university had an idea or somebody had an idea what he was working on developing a software. And either way, these ideas got published and then everyone could use them. Now why did the software publishers publish these ideas? Because they knew that the big job was writing the

program.

They knew that publishing the ideas would get them credit from the community and meanwhile anybody else who wanted to compete with them would still have to write a program, which is the big job. So they typically kept the details of the program secret, of course some of us think that it is wrong, but that is a different issue they kept the details of the program secret and they publish the ideas and meanwhile the software development because software development is going on that provided the field with a steady stream of ideas so ideas were not the limiting factor.

The Limiting factor was the job of writing programs that would work and that people would like using. So, in effect, applying the patent system to software focuses on facilitating a thing which is not the limiting factor while causing trouble for the thing which is the limiting factor. You see the software patents encourage somebody to have an idea but at the same time they encourage people to restrict its use, so in fact we are actually worse off now in terms of having ideas we could use, because in the past people have the ideas and publish them and we could use them and now they have the ideas and patented them and we can't use them for twenty years.

In the mean time, the real limiting factor - which is developing the programs - this is hampered by software patents because of other dangers I explained to you in the first half of this talk. So the result is that while the system is supposed to be promoting progress in software actually it is so screwed up its just obstructing progress.

Today we have some economic research showing Mathematically how this can happen. You can find it in www.researchoninnovation.org and I am not completely sure of the name of the paper, but its one that shows that in a field where incremental innovation is typical. Having a patent system can result in slower progress. In other words the system produces counter intuitive results that are the opposite of what it was intended to do. This backs up the intuitive conclusion of every programmer who sees that software patents are absurd.

So, what can a country do to avoid this problem ? Well, there are two approaches, one is to address the problem at the issue of granting patents, and the other is to approach it at the point where the patents are being enforced. Doing this at the stage of granting patents is not quite as easy as you might think, now I have been talking about software patents but strictly speaking you can't classify patents into hardware patents and software patents because one patent might cover both hardware and software so in fact my definition of a software patent is a patent that can restrict software development.

And if you look at many software patents you can often find that the system may describe has a large part of the computer itself as a part of the description of what's going on, that's a great way of making the whole thing seem complicated when it is really trivial. So its the way they can get the patent office to decide it is unobvious. But there is a different criterion that can be used, a slightly different place to draw the line that still does a reasonable job and that is between processes that transforms matter in a specific way and processes where the result is just calculation and display of information or a combination of data processing and display steps where others are put it as mental steps being carried out by equipment.

There are various ways of formulating this, but more or less equivalent. Now this is not exactly the same as prohibiting software patents, because in some cases computers are used as a part of specific physical equipment to make it do a specific thing. And software patents might be allowed if they are part of a specific physical activity. But that's not really a disaster, after all once people are involved in a specific physical activity or a specific physical product, they are bringing into their home business all those complexities of dealing with matter. So its more like those other fields of

engineering, may be its okay to have patents on that narrow kind of software.

As long as we can keep the core areas of the software, the purely software activities safe from patents we have solved the bulk of the problem. So that is a feasible approach and that's what people are working towards in Europe. However that is not going to be any use in the United States because United States already has tens of thousands and probably hundreds of thousands of software patents. Any change in the criterion for issuing patents does not help at all with the patents that already exists.

So what I propose to the United States is to change the criteria for applying patents to say that purely software systems running on general purpose computing hardware are immune from patents.They by definition cannot infringe a patent and this way patents can still be granted exactly the way they are now and they can still in the formal sense cover both hardware implementations and software implementations as they do now. But software would be safe.

That's the solution I propose to the US, but it could be used in other countries as well. Now, one of the tremendous dangers facing most countries today is the World Trade Organisation, which sets up a system of corporate regulated trade - not free trade as its proponents like to call it, but corporate regulated trade. It replaces regulation of trade by governments that are somewhat democratic and might listen to the interest of their citizens with regulation of trade by businesses which don't pretend to listen to the citizens. So it is fundamentally antidemocratic and ought to be abolished.

But it is crucial to note that the part of the GATT agreement which deals with patents does not require software patents. Many experts who have studied this in Europe makes this claim and the reason is that they interpret technical affect as there is a specific physical consequences for physical system going on. And the software vector doesn't do that, doesn't have to be, in the domains that the patents can cover. So at least you don't have to worry about the Word Trade Organisation causing problems here despite the tremendous problems they cause in other areas of life.

Preventing India from software patents here will be up to you - to the citizens of India. I am a foreigner, I have no influence except when I can convince other people through the logic of what I say. There is a chance that you can do this. When US started to have software patents the public policy question was not considered at all. Nobody even asked whether it was a good idea to have software patents. The Supreme Court made a decision which was then twisted around by an appeals court and ever since then there was software patents.

But when Europe started to consider officially authorising software patents a few years ago, public opposition started to rise and became so strong that the politicians and the parties began paying attention to it. And started saying that they were against it. In fact two attempts to authorise software patents have been blocked already in Europe. The French Minister of Industry says that the software patents would be a disaster and under no circumstances should they be allowed in France. Over the German political parties have taken a stand against the software patents.

The battle is not yet over, you know, we have not conclusively blocked software patents in Europe because the multinational companies and their servant, the United States government, is lobbying very hard and they have ignorance on their side it is so easy for somebody with a naive near-liberal view to be persuaded that new kind of monopoly has to be good.

You have to look at the details of how software patents affect software development to see that they cause problem. You have to study that economic research in its Mathematics in order to see why you shouldn't assume that patents always promote progress. So it is easy for IBM to centre

someone and say you should really adopt software patents, they are great for programming and look US is ahead in when US has software patents if you have software patents too you might catch up. You can get more dominant and - when US was ahead in computers before it had software patents, it can't be because of software patents. It is important to understand that each country has its own patent systems and its own patent laws and what you do in a certain country under the jurisdiction of that country's patent law. So the result is that if the US has software patents the US becomes a sort of battleground where anybody using computer might get sued.

If India avoid software patents then India is not a battleground, and computer users in India do not face this danger of getting sued. So it turns out that each country will issue patents to foreigners just as to its own citizens. So in fact in a place which has this idea of software patents foreigners can own those patents there are lots of non-US companies that own US software patents so they all welcome to get involved in the fighting in the US. Of course is we Americans to suffer become the victims of this. Meanwhile in India if there are no software patents that means Indian companies and foreign companies are prevented from coming to India and attacking people with software patents.

So, yes it is important that each countries has its own patents for that makes big difference but you can't understand what difference it makes. Having software patents in a certain country is not an advantage for developers in that country. It is a problem for anybody distributing and using software in that country. Now, if you in India are developing a program for use in US you may face the problem or your client may face the problem of US software patents. At least probably you can get sued here.

The client who commissioned the program and tried to use it might get sued in the US and indeed you will have to deal with the problem, the US is problem when you try doing business in the US. But at least you will be safe here you know at least it is a big difference between your client got sued because your client told you to make a product and that product is patented versus you get sued for making that product.

If there are software patents in India you will get sued. Wheas in the current situation at least you can say to the client "well, we did our job you told us to make this and we made it and so, I am sorry this happened to you but this is not our fault." If there are software patents in India you get sued yourself and there is nothing you can say about that.

So the ultimate conclusion is that software patents tie all software developers, all computer users and essentially all businesses in new kind of bureaucracy which serves no beneficial social purpose. So that is a bad policy and it should be avoided. Businesses don't like bureaucracy. If businesses knew that they were threatened with a new kind of bureaucracy they would have opposed software patents very strongly. But most of them aren't aware of this.

In the US software patents have led directly to business method patents. What does this mean? A business method is basically haw you make decisions about what to do in the business. And in the past these decisions were made by humans but now sometimes they are made by computers and that means they are carried out by software and that means decision policies can be patented. So software patents implied business method patents and business procedure patents. So the result is that any business could find itself, you know, once they decide "we're going to automate the way we carry out our procedures" but now they can get sued with software patents. So if the businesses only knew they would be organising through things like the chamber of commerce to demand opposition to software patents.

But most of them don't know and therefore it is going to be your job to inform them. Make

sure that they understand the danger that they are facing. And then India may be able with the help from other countries like France and Germany to reject software patents. It is important for the people in the Indian Government to make contact with officials in European countries so that this battle against software patents doesn't have to be fought at one country at a time, countries can work together to adopt an intelligent policy. May be there should be a no software patents treaty that various countries can sign and promise each other aid when they are threatened by economic pressure from the United States as part of its economic imperialism.

Because United States likes to do that, you know, one of the provisions in the GATT agreement is that the countries have the right to make compulsory licenses for making medicines to address a public health crisis. And the South African Government proposed to do this for medicine against AIDS. South Africa has a very bad problem with AIDS the figures I heard was that a quarter of the adult population is infected. And of course, most of them can't afford to buy these medicines at the prices charged by the US companies.

So the South African Government was going to issue compulsory license which even under GATT is allowed to do, but US Government threatened economic sanctions. Vice President Gore was directly involved with this and then he'd be facing the presidential election he realised that this was going to look bad and so he dropped out of the effort. But this kind of the thing is what the US Government does all the time with regard to patents and copyrights. They only mind if people get patented to death. So it is important for countries to work together against this.

For more information about the problems of software patents see www.programming-freedom.org and www.ffii.org. And there is also a petition to sign, www.knowepatents.org.

Please talk with all executives of businesses - any kind of businesses - about this issue. Make sure that they understand the extend of the problem they face and they think of going to business organisations to have them lobby against software patents.

Now I'll answer questions.

Oh, by the way any journos over here, I would recommend writing articles about software patents separately, from articles about free software. If you cover them in one article together, people may get the idea that software patents are only bad for the free software developers and they are okay for other software developers.

This is not true, if you think back of what I have said, hardly any of it relates to the question of whether the programs are free or not, the dangers of the same for all software developers. So please don't take the risk, the people will get confused, write separate articles.

Question: Sir, you said that companies like IBM are harmed 10 times as much as they benefit?

No. What I said is the harm that would had happened to them is 10 times the benefit, but this harm is purely theoretical, it doesn't occur. You see, they avoid it through crosslicensing. So in fact, the harm does not happen.

But it is only neutralised, they don't really benefit?

Well, they do, you see, because, the bare aspect they avoid through cross licensing and meanwhile they do collect money from some other licensees. So they are benefiting in total. There

is the small benefit which happens and the big potential harm which does not happen. So you have zero plus something for the benefit.

But for that something will oppose this movement against patents?

Right, IBM favours software patents. I had trouble one, I couldn't hear all the words in your sentence. I don't know whether there was a 'not' in it. I couldn't tell, there are two diametrically opposite meanings for what you have said, so what you can do is make sure that the situation is clear. IBM favours software patents, IBM thinks it stands to gain a lot from software patents. So what it stands to gain is that the IBM and the other very big companies would basically control software development, because it will be very hard to do independent software development.

To develop nontrivial programs you're going to have to infringe patents of IBMs. Now if you are big and often lucky enough, you might have some patents of your own and make IBM crosslicense with you. Otherwise you are completely at their mercy and you have to hope that they just let you pay the money.

Is someone else asking?

Sir, what was the reason for the development of the software patent?

Well, in the US, there was no reason, somebody tried to get a patent, that was a software patent, and, I think, the patent office said no, so he took it to court and eventually went to the supreme court and they, they didn't judge it as a public policy question, they judged it in terms of what did the law say.

So was it not the realisation that ..

Sorry, I can't ... could you try to pronounce your consonants more clearly, I'm having trouble understanding the words.

So was it not the realisation that copyright is notoriously weak for protecting software?

Copyright is not only what?

Notoriously weak...

Well, I think the whole sentence is not sensible, I don't understand the term protecting software, and I don't agree with you.

Most programmers don't agree with you.

So when you are saying that you are not favouring protection of software and you yourself is giving General Public Licence, where do you get that power to issue General Public Licence?

OK, you are asking questions about copyright and Free Software which is not the topic now, I will accept questions about that later on, but I gave the speech about software patents and i want to answer questions about software patents.

Sir I have a question about software patents, the thing is that how can one protect where there is a functional element ...

Protect what?

Functional element ...

What is gonna happen to them?

Sir, how can we get a protection when there is a...

Protection from what ? Somebody gonna come with a gun ?

No Sir ...

Basically the protection you need is the protection against being sued for the program you wrote, programmers need protection from software patents.

No it is not the programmers themselves sir, they are companies who have invested in something ?

And you what the company get sued because in your large program there are five different things that somebody, that five different people already patented. Now its clear to see the myth that you are operating on, its the naive idea that when you develop the program you will have the patent. Well, the idea, that very statement contains a mistake because there is no such thing as 'will patent' when you develop a program with many different things in it, there are many things each of which might be patented by somebody else already, and you find out about them one by one when they come to you saying either "pay us a lot of money or else shut down" and when you duel with five of them you never know when number six is going to come along. It's much safer to be in the software field if you know you are not going to get sued as long as you wrote the program yourself.

That's the way was before software patents - if you wrote the program yourself there is nothing to sue you about. Today you can write the program yourself, it may even be an useful and innovative program but because you didn't reinvent the whole field, you use some ideas that were already known, other people sue you. Of course, those people who wanna go around suing you, they are going to pretend that this extortion is protection for them, protection from what, protection from having competitors, I guess, they don't believe in competition, they want monopolies.

Well, to hell with them. It's not good for the public that they should get what they want, this is the question of public policy. We have to decide what is good for the citizens generally.

[applause]

Not have somebody saying "I wanna have a monopoly because I think I am so important, so I should have one, so protect me from anybody else being allowed to develop software."

You are suggesting that we should avoid making a battleground for patency, don't we still have to see the problem that there are a lot of American products being sold here and...

Well...

.. and we are still going to be mistaken..?

No! No, you misunderstood, US developers may be in trouble because of the patent system and what effect will that have, it means that their certain products that wouldn't be coming from the US and therefore they wouldn't be sold in the US or here. You see, if the developer is in the US and there is a US software patent, that software developer is going to get sued there, whether or not he tries with anybody in India he is going to get sued. But the fact that he is distributing the program in India is not going to cause him an additional problem because that is under the jurisdiction of India, that's the one thing he will not get sued for. So, basically, what it means is whatever exists can be distributed in India, safely ,and the developers who are lucky enough to be in India would be safe from this kind of gang warfare, and those who are unlucky enough to be in the US will not be safe.

Sir, are you basically against the very concept of intellectual property rights ?

As I said in the beginning, it is foolish even to think about that topic, that topic is an overgeneralisation. It lumps together totally different things like copyrights and patents and so any opinion about co-intellectual property is a foolish one. I don't have an opinion about intellectual property, I have opinions about copyrights and I have completely different opinions about patents and even in the area of patents, know, I have different opinions in different fields. Even that area is a big area. And then their trademarks which are also called intellectual property,
I think trademarks are basically a good idea.

The US has taken trademarks all little too far. But, basically it is reasonable to have labels that you can rely on. So you shouldn't try to have an opinion about intellectual property. If you are thinking about intellectual property, you are thinking at a simplistic level. And any conclusions you reach will be simplistic. So do as I do, you know, pick one topic at a time and focus on it and find out the details about that one area, then you can think intelligently about that area and later on you can think intelligently about the other areas too.

So there is an argument that if particular intellectual property right is not protected ...

I'm sorry,what you are saying makes no sense at all and is at a foolish general level....

Let me complete sir, if that particular intellectual property right is not protected, it may impede the investment, and this impedement ...

This generalistic thinking is so simplistic, it's totally stupid. It makes no sense at all. There is no principle of intellectual property. Copyrights and patents and trademarks originated completely separately, they have nothing in common, except later somebody else made up this term "intellectual property" to call them all while.

Sir, will you extend this concept to the physical property?

No, I'm sorry, none of these things has anything to do with the physical property rights, they are totally different. What do you say extend "this concept" ? Which is this "concept" ? The idea that the term "intellectual property" is a generalisation that leads you into simplistic thinking. Should we apply that to physical property? No, they are totally different. They have nothing in common.

So the basis under which this intellectual property is protected is "protect the labour", "intellectual labour"?

No! No, you are totally wrong, you are totally wrong. The purpose of you have been brainwashed, you have been listening to the propaganda of the companies that want to have these monopolies. If you ask what legal scholars say is the basis of these systems. They say that their attempts, for copyrights and for patents, their attempts to manipulate the behaviour of people to get benefit for the public. Trademarks are a different issue, I think the issues for trademark are completely different. So you are making an overgeneralisation also.

So why can't we extend the very same principle ...

But in any case, your principle is wrong and if you take a look at that economic research on www.researchoninnovation.org you will see that you are making naive statements, naive blind statements that are simply not true. You got the silly idea that creating a monopoly over some aspect of life always invariably makes that aspect of life thrive. Well this is dumb. Occasionally it might work, and occasionally it causes a lot of troubles.

Don't you think that the same kind of monopoly is created in favour of a party when he owns a physical property?

I'm sorry, I can't hear you.

Sir, don't you think that the same kind of monopoly rights are created if a particular physical property is allowed to be owned by a person, just like an intellectual property ?

Physical property can only be in one place at a time, you know, only one person can sit in a chair at a time in the normal way. You know these are totally different issues, you know trying to generalise to the utmost is a foolish thing to do, you are doing with complicated words that have many, many, many complicated details and you are asking us to ignore all these details and you are doing with words that have complicated effects in various fields and you are asking us to ignore the details of their effects. Don't bother judging this program by its results. I think that if we were talking about a public policy issue, we got to look at the actual results of the policy, not some myths to what results a certain ideology would predict, I'm telling you the real results, I'm telling you what I have seen and what other programmers have seen.

Sir,what about the LZW patent? Is it ..

What about the 'what'?

LZW patent ?

The LZW patent?

Yeah. Is it still in effect?

Yes, it is. Well,there are two LZW patents as I explained to you and they are both still in effect

Sir, so its for 20 years?

Yeah, it's not 20 years yet.

Sir, can we reduce the scope of the problem by reducing the period of the patent ?

Definitely, you could. If there were software patents, but they only lasted for, say, 5 years or three years, that would mostly solve the problem. Yes it's a pain to have to wait 3 or 5 years, but it's much, much less of a pain. But, but there is a difficulty there. The GATT agreements say that patents must last 20 years. So the only way you could have something like a software pattern which lasted for 3 or 5 years is as follows.

First, make it clear that ordinary patents do not apply and second, if you wish you could create this different system of five-year software idea monopolies. Well, it's not clear that there is any particular benefit in this five-year software monopolies but it would be much better in the current situation. So if you found the government prepared to make this deal, well, I would say, we should take it. But, but we have to realise, though, that the first step is to abolish software patents strictly speaking, and that has to be part of this deal.

So and patent has also now become victim of ...

I'm sorry , I couldn't hear you, could you speak louder?

Sir, patent has now become a way of making money by businesses rather than promoting inventions?

Yes, a lot of them use it that way

So, sir, can we reduce this problem further by assigning the patent to the actual inventor rather than a business ... ?

Not really. What you find is that, that aspect of the relationship between the employee and the business is something that gets negotiated and the business has more cloud, so they are always going in the end of ranging to have the employee and the patent of the company. The other thing is that it doesn't make a big difference who owns the patent. The point is that you are prohibited from developing a program using that idea and it may make some difference precisely who has the power to sue you. But what you really want is not to be sued on. So why look for a half measure like this ? Or its much better just to say that software shouldn't have patents ?

Okay, if you gonna pass a note, you better read it out loud. Any other questions?

People who are being to Malaysia say that, if we buy a PC there, the amount of money we would pay for all the standard software is about a tenth of what we should pay in this country. In Malaysia they are little more relaxed about patents and copyrights ?

Well, are you not sure what you are talking about, you seem to mixing together copyrights and patents, I'm not sure what you are talking about has anything to do with the issue of software patents.

Precisely what I want to know is about... this is something to do with patents..?

Probably not.

Different countries depending on how much, whether they are part of WTO or not part of WTO ...

No, no.

... I think matter ...

You see, I don't know for certain because I don't know what's going on their either, I'd never been there but I suspect that it's a matter of copyrights and has nothing to do with patents, because if you are talking about the same programs, remember, software patents are primarily a restriction on software developers. So it's the same program that was developed, say, in the US, the patent problems they have are independent of, you know, the patent problems they have are biggest in the US, not in either India or Malaysia, so, that probably has to do with copyrights not patents and that's a totally different issue, we mustn't lump these issues together.

Sir earlier you've told that ...

I'm sorry I can't hear you.

Earlier in your speech you've told that software that should be brought under the purvey of patents.. is what you defined that as what can be run on a general purpose machine.

I'm afraid I can't ... can anyone understand what he's saying ? I cannot understand your words. If you make an attempt to enunciate more clearly I may be able to understand.

You had spoken earlier that software that should be patented is, you defined that as, software that can be run on a general purpose machine ...

I'm sorry I didn't say that software should be patented, so I just cant make out these words. Maybe if you tell that to someone else, the other person could say it I could understand.

Software patents, like whatever you call software patents, like those are what can be run on a general purpose machine. So if some algorithm or some piece of software is capable of being executed on a general purpose machine, it should not be patented.

Yes. Now I can hear you, yes. One of the things I proposed was that patent should not apply to software for general purpose machines or the use of it on those general purpose machines. So that if you develop that program or if you are using that program you couldn't be sued.

We've an increasing number of software now being run on a general purpose machine.

Well, then that would be covered still by software patents so there wouldn't be a total a solution, but at least there would be a partial solution.

So if the defining line is general purpose machines, don't you see there's a possibility that people could find loopholes in it, like, to find or workarounds for ...

I'm sorry do I see a possibility for people would do what ?

... of finding loopholes or workarounds of converting what you would call software patents into get it actually patented.

I'm sorry I do not understand. Loopholes to do .. I'm sorry. What people would do, what software developers would do in that situation is use general purpose machines more.

Some algorithm can be run on a general purpose machine - what I'd say that that algorithm I'm using it for some embedded device and go ahead and patent it.

Why you could try it, you misunderstood. The point is that, you misunderstood what the solution is. the solution is that if I'm using in developing the software on general purpose machines then nobody can sue me for patent infringement. So yes, somebody could get a patent and may be he could sue others who are doing specialised things which involve particular hardware. But they couldn't sue me.

Excuse me sir, may I ask a question.

Yes.

Sir, you spoke of general purpose machines. In the sense, how would you define these machines, because these days you have a lot of custom made handheld devices etc. Now some way ...

No, hand held computers are general purpose when they are not designed to carry out a specific computation or a specific physical process. They're general purpose computers. They have general purpose computer chips in them.

Then the idea would be contestable in a court of law as together it's a general purpose ...

I guess, what has to be, yeah. The precise details of drawing these lines, one ends up having to leave to judges.

thank you sir.

Germany and France, the only countries who has said no to patents in Europe ...

Well, I don't know the full situation. Those are the just the ones I know of. The last time there was a vote there were going to be a majority of no votes, and so they dropped the issue. And I don't remember the other countries.

There's no European community decision on this ...

Not yet. In fact, the European Commission itself is divided. One of the agencies - the one which unfortunately is the lead agency on this issue - has been won over by multinationals and is in favour of software patents and then the agency that's trusted to encourage software development is against them, and so they're trying to work against it. So somebody who want to get in touch with the official incharge of the agency that is opposed to software patents, I can put them in touch.

Is there any country that said 'no' to software patents ?

Well, there are countries which don't have them, but it's not clear that there's any country which has affirmed this recently.

Sir, could you please elaborate on the benefits the software development community got in European countries from this policy ?

Well, the benefit is you don't have to be afraid someone will sue you because of some of the ideas were a combination of the ideas that you used in a program you wrote. Basically software patents mean that if you write a program, somebody else might sue you and say you're not allowed to write that program. The benefits of not having software patents is you're from that.

Now in India you've probably taken for granted that you're safe from that. But that will only long as long as they're no software patents in India.

Are there any threats to India not acceding to software regime ?

Well there's no software regime. The GATT agreement doesn't require software patents. There's no treaty requiring software patents.

Most people, if they had a chance to get a patent and make a lot of money out of it, they wouldn't pass it up ...

Well, many people if they had a chance to get a gun and make a lot of money of it wouldn't pass it up.

The point is, therefore, let me try not to hand them that opportunity. You know, since we don't have a government agency handing out guns to people on the steet we should not have a government agency handing out software patents to people on the street either.

Being an advocate of this non-patency, have you ever faced any ...

I'm having trouble hearing you. Please try to make an effort to pronounce every sound clearly that I might understand.

... You being an advocate of this non-patency, have you faced any problems with this multinationals or something ?

have I faced any problems ..

... so far in your life ?

I'm sorry. What did he say?

Have you faced any problems with multi-nationals in your life ?

Well, there are many. In the community where I develop software, there are many examples of

programs that had their features taken out, programs that didn't have the features put in the first place, programs that were not even written for many years, because of this. There are many examples of jobs we can't do, because we're not allowed to do them.

Now we collected examples of this and were looking for people to write them up. You've to look at each example and investigate it fully and write down a clear description of what happened and what the harm was and so on. We had had trouble finding people to do this. We're looking for more. So someone who is really good at writing clear English might want to volunteer for this.

I think he asked whether you had any threat to you by any multinational companies ...

Well they never threatened my life !

Yeah that's the question !

But they do threaten our work. Now they threaten to sue us.

Volunteer: There's a question from a gentleman at the back: If the multinational companies that produce hardware, like Intel, coming to a contract coming to a contract with big software companies to restrict Free Software by changing the microprocessor pattents, how will you overcome such a hazard ?

I see very little danger of that. Intel recently developed a new computer architecture,and far from trying to stop us from supporting it, they hired people to implement it.

So it looks like we should now we move to Free Software questions. I'd like to remind people that until this last answer, I was not speaking for the Free Software movement. I was speaking of something of vital interest to every programmer which is to be free to write programs and not get sued for having written them as long as you wrote them yourself. And that's the freedom that you've taken for granted until now and that's the freedom you will loose if you have software patents.

Now however we're moving to the topic of Free Software, which is what I spent most of my time working on, and the individual, actual software development project that I've lead which is developing the GNU operating system, which is a Free Software, Unix-like operating system used by some twenty million people estimated today. So I am going to answer questions on Free Software and GNU.

In the absence of a concrete revenue model for Free Software, will this also go bust like the dotcoms ?

I can't predict the future but I want to remind you that the dotcoms were businesses. And Free Software is not primarily a business. There are some Free Software businesses. Whether they will succeed or ultimately fail, I don't know. But those businesses do contribute to our community, but they are not what our community is all about. What our community is all about is having the freedom to redistribute and study and change software. A lot of Free Software is developed by volunteers and the amount is increasing and, no matter what happens with the companies that's not going away.

I understand companies like IBM are also investing considerably in making their systems and software compatible with free source code like Linux ...

You mean GNU ?

All right ...

Yes, they call it Linux. Actually the system is mainly GNU and Linux is one of the pieces.

[From audience] The kernel is hardly eighteen percent.

Well, really, that much? What I saw is three percent.

[From audience] You can see through a needle. Very insignificant.

But, I also understand that they've invested around a billion dollars in doing so. Now my question is ...

Well that's not true.

My question is, for a service that has no revenue model, will this be sustainable in the future and if I change my business onto ...

I'm sorry I can't predict the future. No one can.

How can I ...

Some god men can predict the future. I'm not. I'm a rationalist.

I can't tell you what's going to happen. What I can tell you is that what IBM claims to have put a billion dollars into the GNU plus Linux operating system, that's no entirely true. You've to look carefully on what they're spending this money on, and you'll find they are spending this money on various different things, some contribute and some don't.

For instance, they are funding some work on developing the GNU/Linux system. That's good, that contributes. They do develop some other Free Software packages that they've contributed to the community. That's a real contribution.

They are also developing many non-free programs to make them run with the GNU/Linux system and that is not a contribution. And they are publicising the system, well, it's not a primary contribution, but it does help. You know, having more user is not our primary goal. But it's nice if more people would try our software, so that does help, but then they're mistakenly calling this Linux which is not quite right, and they're lobbying for software patents in Europe, which is bad, so, yeah IBM is doing many different things, some are good and some are bad, and if you want to have a full view, it's important to look at the individual actions. Do not try to add it up because that just means you're missing the important aspects of the situation.

Are there any more questions ?

This question is not about patent or copyright or anything like that. But this is one example what you said about - if statement and while statement - that you said something about the differences in the field of computer science and differences other sciences, that is other engineering sciences. You said that if I change something in the if loop that's if statement, there won't be any effect, that you said ...

No I didn't say that.

You said that! You said that there isn't any heating effect. I remember that ...

I'm sorry, I know what I said. I said something that's partly similar to that ...

I'll tell the exact statement: you said there won't any heating effect.

Any whating effect ?

Heating effect. Heating ...

Oh yes we don't have to worry about how much heat is ...

Yeah, yeah, exactly. Then what is it that cascading effect ? If I change the structure of the loop, there will be an effect.

Oh sure. The program will behave differently when you change it, but I'm not saying that writing every program is easy that we never make mistakes. I listed a lot of specific kinds of problems, that would plague a mechanical or electrical engineer at every little detail, even if each detail gets to be very hard for them. Worse for us, the problems are because we do so much, we're doing it so fast, we don't think carefully about each one thing. So we make mistakes.

So you admit that there's an effect.

Of course. I never said otherwise, I'm sorry if you thought so. Sure if you change a program it's going to do different things.

Sir, can you comment on the commercial distributions ?

Well, you asked me to comment on the commercial distributions of GNU/Linux systems ? Well, I think that's fine. That's one of the freedoms Free Software gives you - the freedom to use it in business, the freedom to distribute it as part of the business, the freedom to sell copies in exchange for money. these are all legitimate.

Now, one thing I'm unhappy about is what the companies do this is that they add some non-free software to it.

That's the installation program ?

Yeah, any non-free software. Because the goal was you should be able to get a completely free operating system. Well, if I have a thing in a store which says I'm the GNU/Linux system, the public says Linux, but inside it there are some non-free programs, now you're not getting something that's entirely free anymore. It doesn't entirely respect your freedom. So the real goal for which we wrote the system is being lost.

So that's a major problem our community faces now. The tendency to put free software together with non-free software and make these non-free overall systems. And then, you know, it might seem that the software is a success because many people are using it. But if you look at our real goal, our real goal is not popularity. Our real goal is to spread a community of freedom and we're not succeeding in doing that if the people are using non-free software systems.

Unfortunately, I couldn't give both speeches. I can give a speech about software patents, or I can give a speech about free software. They're very different and each one of them is a long speech. So unfortunately what it means is that I can't explain about Free Software and the GNU project here. Am I giving anther speech in Kochi? Am I giving the Free Software speech in Kochi?

No.

Oh well. I gave that speech in Trivandrum.

So I'll answer five more questions and then I have to call it quits because it gets to be quite draining to answer so many.

Excuse me sir, question from me again. Sir, this is a personal question. Me, as such, I love programming. I spend a lot of time in front of my system. And I was listening to some of your earlier speeches where you said that back in the 70's, the community of programmers had a sense of goodwill among them. They used to share code, they used to develop on it.

Well, a specific community of programmers which I belonged to. This was not all programmers. This was one specific community. Continue.

Yes sir. In that context, I feel particularly, me as such, I feel very hurt when I see the so called interaction among programmers today. Because many of us are very good programmers but we look at each other in different colours depending upon the tools we use - "hey, he's a windows guy", "hey, he's a GNU/Linux guy", "hey, he's into Solaris systems", "he's a network programmer". And unfortunately most of this prejudice come from a lot of misinterpretation out of things like this. None of these guys promote Free Software as such, and it hurts me as a programmer and many of my colleagues, and I work in an environment ...

Could you speak a bit more slowly, I am hearing most of it, but there was one point that I miss, so you speak slowly.

Yeah, and that here that we work with in an environment you are judged according to the tools you use rather than the quality of work.

To me that, well, in one sense there is a situation where in a limited way that is rational. If there is a tool which is normally used for doing fairly easy jobs and there are lot of people who now had to do it than I would imagine now, I wouldn't want, I might not pay as much to them as somebody who does very hard jobs with a different tool that's used for hard jobs. But it's true if you're talking about hard jobs, it makes no sense that you prejudice about what tools people are using. Good programmers can use any tools.

That was not the focus here. The focus was that here it's a question of goodwill. Goodwill amongst programmers these days seems to be melted down into the, you know little boxes of this system and that system and that hurts.

I agree we should encourage people to study more different things and we should never be prejudiced against people because of some detail, you know the fact that this person likes Prolog and this person likes C, why should they hate each other ...

It's not even that distinct. It's like this person works on GNU/Linux and this person works on Windows, which are the major operating systems today in India at least.

Well, in that case, though it's not just a prejudice. You see Windows is a system, a social system, that keeps people helpless and devoid. Whereas, GNU/Linux is an alternative that is created specifically to liberate people and to encourage them to collaborate. So to sum it then, it's not like where you born in this country or that country, no this is like your choice of politics. And it does make to criticise people for their choices about important issues.

So, I would say, a person who's using Windows, well, either he's actively supporting this power structure or at least may be he's trapped in it and doesn't have the courage to get out. In that case you can forgive him, I guess, and encourage him. You know there are different situations in any place where people're different. Some people are making more or less effort to try to improve things. I believe in judging people as individuals, not as lumping them together by their groups.

But this is, in this one case it is, somewhat alike political choice with political consequences for society and that's exactly what makes sense to criticise people.

Sorry to continue again on this, but I'm a little persistent about this. It's ...

This is your last chance.

Yes sir, thank you. Generally when statements like these are made people who are not so much, you know, in connection with these things tend to assume that cooperative communities and sharing of source code and sharing of ideas and things like that don't exist in other environments but they do, and that's very unfortunate that they think so.

I'm sorry ... "that don't exist in other environments", I don't know which other environments that you're talking about I don't understand.

Other programming environments, other operating systems.

Well may be there are some users developing some Free Software that runs on Windows, in fact I'm sure ...

Note: At this point, there was a short blackout, and both the recording and the transcript is incomplete here.

Well, may be there, are there any questions? Could you speak louder? I can't hear you at all.

Sir may I ask you a question ?

Okay you can, sure.

In Free Software System we will be distributing the source code also together with the software. So

a person is entitled to change whatever he can in the source code. So don't you think there will be too many software versions of a particular software and this will in turn cause problems for a layman to find out which will suit him the most.

Practical experience is that this is not a problem and occasionally it happens. But not very often. Now, you see that the reason is that the users want interoperability and with Free Software the users are ultimately in control and what they want they tend to get. The Free Software developers realise that they had better - if they are going to make incompatible changes they are likely to make users unhappy and
their versions are not going to be used. So they generally draw the obvious conclusion and pay a lot of attention to interoperability.

What I feel is that like I'll be just loading a software into my computer and the next morning I'll find a better version then again I'll have to change it. The next morning again something has been done to the source code and that's a better version, so don't you ...

In general you are not going be finding a better version everyday and the reason is that typically for any given program there is usually only one version that is widely used. May be there will be two, once in a while there will be three - when there is no good maintainer that might happen. So you are just not going to keep finding out about more versions that are good everyday; there aren't so many. There won't be that many popular versions. There is one situation where you can get a new version everyday. That is, when there is one team doing a lot of work on development then every day you can get the latest version. That you can do. But there is only one version at any given time.

Sir, don't you think we will have to implement an organisation which will take into consideration all these updations and it will just provide a single software which will have all the updations right.

I'm sorry, I didn't hear that. Shouldn't we have an organisation that would do something with all these versions ... but i don't know what.

Like, say I have developed a version of ...

Did anyone else hear what she said? Could anyone else tell me what she said ?

The thing is that ...

It's a very valuable skill to learn to speak slowly and clearly. If you ever want to give a speech which as part of your career you will, it's very helpful to learn to enunciate clearly and slowly .

Thank you sir. Sir the thing is that don't you feel that we require an organisation which will just perform a number of updations together and make available a software which will club all the updations until that date?

You are saying, take various different applications and put them together?

Yes Sir.

I will tell you a lot of organisations are doing that; in fact every one of the GNU/Linux distributions is exactly that. Debian does that, Red Hat does that ... We to some extent do that also for the GNU packages. We work on making sure they work together.

Excuse me Sir, we have talked lot against patents. In US conditions have you ever been forced to put forward any applications for patents?

No. But no one can force me to make a patent application ...

Also do you own any patents?

I do not own any patents. Now, I have considered the possibility of applying for patents to use them as part of a mutual strategic defence alliance.

Do you mean to say that if I have twenty patents with me, I donate it to the FSF and you maintain it for me?

Well, not the FSF, it would be a separate specialised organisation that would exist specifically so that we would all contribute are patents and that organisations would use all of these patents to shelter anyone who wishes shelter. So anyone can join the organisation, even somebody who has no patents. And that person gets shelter of this organisation. But then we all do try to get patents so as to make the organisation stronger so it can protect us all better. That is the idea, but so far no one has been able to get this started. It's not an easy thing to do, and part of the reason is that applying for the patent is very expensive. And a lot of work as well.

So this will be the last question.

Why can't the Free Software foundation start its own distribution?

Oh well, the reason is that Debian is almost what we want, and it seems better to be friends with Debian and try to convince them to change it a little rather than to say, "Well, we are not going to use it; we are going to make our own thing". And also it seems likely to be more successful too because, after all there are a lot of people working on Debian already. Why try to make an alternative to that large community. Much better to work with them and convince them to support our goals better. If it works, of course, and we have our ways to go in that.

So that was the last question, I can't stay all day answering questions, I'm sorry. So at this point I am going to have to call a halt and get going and go have lunch. So Thank you for listening.

[Applause].