The "Multi"-verse

Description: Steve and Leo continue with their "fundamentals of computing" series this week, building upon all previous installments, to explain the details of multi-threading, multi-tasking, multi-processing, multi-core ... the "multi"-verse of modern computing.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-247.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-247-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 247 for May 6, 2010: The Multi-verse.

It's time for Security Now!, the show that covers everything you need to know about security, privacy, and how computers work. Joining us right now, the man behind the show, the myth, the legend, Mr. Steve - you're not a myth.

Steve Gibson: Wait a minute. The myth?

Leo: The man, the myth, the legend, Steve Gibson of GRC.com. Steve, every once in a while I should just remind people that you've been doing this for longer than, you know, almost anybody. You started, I think, was it out of college you were doing synthesizers?

Steve: No, actually before then. It was high school.

Leo: High school?

Steve: Yeah, I had my first programming job at age 14, when I was in high school.

Leo: Wow. And you, for the Apple II, you did the Gibson Light Pen, which was very famous at the time.

Steve: Yup, that was a prior company we did. I designed the hardware, electronics for it.

And in fact one of my favorite moments was I was in Boston at the Boston Applefest, which was the big, one of the big annual Apple shows, and Woz came up, and I was demonstrating the light pen with an Apple II up on the little - our little booth stage. We had a 20x10 booth. And he sort of stood there looking up at the screen. And there were a couple things that turned out really sort of spectacular about the pen. The Apple had the ability to show different regions of memory. You could do screen switching, which was often used for animation. But it occurred to me that you could use the light pen to trigger the screen switch. And so it was possible to literally, like, lift the screen up and see what was underneath it, which I used in a CAD program that I had written. And Woz just sort of shook his head, and he said, "It never ceases to amaze me what you are able to make my machine do, Steve."

Leo: Oh, isn't that neat.

Steve: So that was...

Leo: High praise.

Steve: Yeah, that was neat.

Leo: High praise.

Steve: So I just love computers.

Leo: And then after that a little thing called SpinRite. You know, it's funny, we were talking - where's my - oh, here. I want to do, on next Gadget Warehouse Friday with Dick DeBartolo I'm going to do a Zip drive. And so somebody sent me some Zip disks. Look at this. And it reminded me that I met you, the first time we worked together was - I was talking to Paul Thurrott about this, and he said, "Oh, you remember that click, and they would die?" I said, "Yes, the click of death. And guess who told me about the click of death?"

Steve: Yup.

Leo: Steve Gibson. In fact, you wrote a program called TIP - Trouble In Paradise.

Steve: Well, what happened - yes. What happened was SpinRite 5 was the first version which would run on a much wider array of drives, more than just strict, you know, motherboard-mounted hard drives. And people began writing in, saying hey, SpinRite just solved our click death problem. And I said, your what? And so that sort of got me into the whole Iomega stuff. And it turned out that what was happening with the Iomega drives was really different from what was going on with hard drives. And in fact, I decided running SpinRite really wasn't what you wanted to do. So, I mean, because it was - what happening on the Iomega drives was an actual drive failure. It was the drive no longer writing correctly.

So running SpinRite on a drive which is fundamentally broken can do more damage than good in the case of these Zip drives. So I created Trouble In Paradise, TIP, in order to give people a free diagnostic utility to tell them what the condition of their Zip drives was. And then only after you got - TIP gave you a clean bill of health, then you could use SpinRite to make things better. So, yeah, that was when you and I - I came up to San Francisco and hung out in the basement-looking Tech TV set and...

Leo: That was. It was Kate, I think. It was before Patrick.

Steve: Yup, Kate Botello was there.

Leo: It was in the first year, so that would have been '99 or '98.

Steve: And in fact my other favorite fun story is that she was the one who discovered ShieldsUP!.

Leo: That's right.

Steve: I have the video of it still where, like, she had her own little segment of the show, that she would show you something. And so you were just sort of there, and you showed up, and she said, "Oh, and I found this really amazing site that checks your Internet security. It's called ShieldsUP!." And so she goes to GRC.com. And you said - she said, "Yes, Steve Gibson did it." And you were sort of like, you know, distracted or - because you were, like...

Leo: I'm probably getting ready for the next segment, yeah.

Steve: Exactly. And you said, "Wait, who?"

Leo: [Laughing] Well, I read you religiously in InfoWorld. You did - was it InfoWorld?

Steve: Mm-hmm.

Leo: You did the best column, which I really - and I sorely miss. It was one of the - because it's, you know, most of the time people writing in tech magazines, the late lamented tech magazines, were journalists first, technologists second. And you were - it was vice versa. So you had the - they were great columns because they were so smart and well-informed, just like this show. And I just loved it. And I was very sad, I know you retired, you gave it up.

Steve: Well, eight years of a weekly column. And...

Leo: Was it that long? That's a long time.

Steve: Eight years, yeah.

Leo: What was the name of the column?

Steve: It was called Tech Talk.

Leo: That's right, yeah.

Steve: I originally named it Behind the Screens, which I liked. But CompuServe had a trademark on that for one of their own columns because there was a CompuServe magazine at the time. So they said to InfoWorld, we'd like you to change the name. So InfoWorld said, okay, Gibson, come up with something else. I said, okay, Tech Talk. Not very inspired, but still. And, yeah, in fact I was really - there was the gossip columnist, Robert X. Cringely was, before I came along, the most popular item that they had in the magazine. And after a couple years we were neck and neck. Sometimes they would do reader surveys. And sometimes I would come out first; sometimes Cringely would. And people often said that they turned to my column first and then went to the front page because they wanted to see what was going on. So it was great. It really improved my writing dramatically, too, the eight years of that discipline.

Leo: Sure, sure.

Steve: Made a big difference.

Leo: Just doing it, yeah. And you're a very good writer. It was very, it was always very clear and concise and very insightful. But anyway, I bring that up just to say this guy is not a newbie. He's been around. GRC.com is the website. You can still get SpinRite. It still is the, some versions later, the hard drive maintenance utility. But for the last 246 episodes, now 247, we have been talking about security and privacy on a podcast that really has ended up being, I think for a lot of people, kind of a must-listen-to introduction to technology, to how things work, how crypto works, all of this stuff. And we've been doing a series kind of off and on that I just love on the fundamentals of computing. We're going to continue that series today.

Steve: Yup. I called this one "The Multi-Verse." And I've been promising it for a number of weeks. But we've had some other things which have come up in the meantime that sort of preempted this. We're going to talk about multi-threading, multi-tasking, multi-processing, multi-core, and also hyper-threading doesn't quite fit in with the multi, but it's related. And it exactly builds on everything, the foundation that we've laid in the prior episodes about the fundamentals of computing technology. We left off a couple weeks ago talking about hardware interrupts and how hardware interrupts were able to be used to great advantage to manage the time spent in a computer. And with everything that we've covered so far, we can now take the next step and talk about basically how

computers are able to do so much, seemingly all at the same time: the multi-verse.

Leo: Mm-hmm. I love the name because of course that's a play on "Snow Crash," the Neal Stephenson novel; right?

Steve: And also it's the term that the Serenity uses. They talk about the multi-verse.

Leo: The multi-verse? Yeah, in "Snow Crash" it's the virtual reality world that you can go into and live in and really is - it's not, we're not talking Second Life here. I mean, it really is real because it's total virtual reality. I don't know what it is in "Serenity." But I always - I keep - I hope that they have a multi-verse before I pass on. Maybe somebody listening to this show...

Steve: I don't think we're even going to have aliens by the time you and I are gone.

Leo: Bummer. We might. You never know. That could happen at any time. You saw what Stephen Hawking said.

Steve: No.

Leo: Oh, he was - it's a show he's doing, a BBC show or something he's doing. He said, you know, we really oughtn't to talk to them. If we find aliens, don't say anything because it's probable that they've been in the ship, they've taken a very, very long time to get here. And when they get here, what are they going to need? Resources. We're just in the way. So he says don't talk to aliens.

Steve: What a cheery thought.

Leo: Yeah. So maybe let's hope those aliens don't show up. All right, Steve Gibson. Do you want to do, before we get into the multi-verse, do you want to do some security news?

Steve: Oh, yes. We have our - pretty much our regular lineup of people.

Leo: Usual, yeah.

Steve: Unfortunately. There is a, for those five people who are using Opera, I wanted to let them know what was regarded as a highly critical remote code execution vulnerability - the current version of Opera, as of the date of this podcast, is 10.53. Anyone using anything previous really should update soon. It's not that there's likely to be a great scattering of exploits against this because, as we know, Opera has a relatively small market share. They're stronger over in the portable side, I think, than they are on the PC side.

But there's a - it's been discovered that in versions prior to 10.53, there's an exploit which is pretty serious that allows JavaScript running on a web page to use the very common document.write function to continuously write until it overwrites improperly initialized memory, at which point it's able to run whatever code the bad guys want. So it's basically the idea would be you go to a website with JavaScript enabled, which is of course the default. And if you were using Opera, and the site was specifically targeted to this exploit, you could get something malicious installed on your machine. Which unfortunately is happening to people more and more. The reason...

Leo: Is this related to, now, wasn't there a bug, a zero-day exploit in Opera? Is this the same one, or is this a - when it first, the new version came out?

Steve: Hmm.

Leo: This is another one.

Steve: I think this is another one. This is different than the zero-day we had not long ago. And the reason it's a concern for Opera users, if, for example, your corporation had standardized on Opera, and it was known that all the employees, for example, in a company were Opera users, and somebody were targeting your company, then they could send a page to you which is going to take any of the company's employees to a malicious page and execute this. It's not - it's unlikely with the market share that Opera has that there will just be lots of pages out there that are using this exploit because they would expect probably no one to ever happen upon that site. But for targeted attacks, and what we're seeing increasingly in the threat model on the Internet is attack targeting, keeping really up to date is becoming increasingly important.

And once again we're at - just wanted to remind our listeners because the problem with the executable content in PDF files with Adobe Reader is continuing to escalate while Adobe sits around and does nothing about it. They are - they're still deciding whether they want to do anything about it. Meanwhile there are...

Leo: They're too busy yelling at Steve Jobs to fix their product.

Steve: Yeah, and trying to get him in trouble with the State Department, or I guess they're - clearly they're behind this ridiculous antitrust investigation which is going to be launched into Apple's decision not to allow the third-party development tools to be used. What's happened is there's now a Windows rootkit worm variant known by two names, as Auraax (A-u-r-a-a-x) and also Emold, which is using this now pretty well-known ability for PDFs to execute content. We've talked about it several weeks ago.

I had wanted to remind all of our listeners, if you had intended to disable this feature in Acrobat Reader and Adobe Reader, but forgot to, it would be a good time now not to forget. To do so is simple. You just open any PDF file. That'll get the reader going. And then under the Edit menu, down at the bottom is Preferences. That'll bring up the preferences dialogue. Then you'll see an alphabetical list of stuff over in the left-hand column, sort of categories. Down at the very bottom, near the bottom, is Trust Manager. Select that, and then there's a checkbox, the first one over on the right-hand panel,

which reads "Allow opening of non-PDF file attachments," that you want to turn off. If you turn that off, then you're safe.

This is being used, basically there's been a huge escalation in the use of this exploit because Adobe continues to do nothing. And they're saying, well, we may address it in our next quarterly update. Remember Adobe has adopted the four-times-a-year update policy, even though they haven't been following it because they've had so many problems with security lately. So I just wanted to remind users once again, if they had intended to get around to turning that off, now would be a real good time to do it. I understand people may be away from their computers while they're listening to the podcast and might want to do this, but forget to do so. So there's a little nudge because this is looking like it's catching more and more people.

We talked about what has been called in the security community Microsoft's "placebo patch" last week, which was released initially on a second Tuesday of the month, that is, of April, and has now been re-released. So it was something that affected only Windows 2000 running the Media Services service. So not affecting lots of people. But if you did notice that, if you're running Windows 2000, and Microsoft was saying, hey, we have a little update for you, that's what that was. They have rereleased it and fixed it this time. So it's no longer the placebo patch. It actually does patch the problem after they initially misfired on that.

And just in a little bit of security news, I thought it was sort of sad to read that the U.S. Treasury Department's websites are installing malware on their...

Leo: Oh, geez. Just in time for your taxes.

Steve: Your tax dollars hard at work installing malware in visitors' computers because they're hosted by Network Solutions.

Leo: What? Why is - first of all, why are they being hosted by Network Solutions would be one question.

Steve: I know. I know. Unfortunately...

Leo: Doesn't the government have its own servers?

Steve: Apparently these sites at the U.S. Treasury Department fall within a level of security clearance...

Leo: Right, because we use them.

Steve: Exactly. Because it's just for the public.

Leo: It's just for the public, so it's okay.

Steve: Yes, exactly. Which doesn't - which allows them to be outsourced. So they're hosted on Network Solutions' systems. And you'll remember from last week that Network Solutions unfortunately was the focus of an attack which got into and infected a whole raft of their sites. So it turns out the U.S. Treasury Department is among them. And so what happened is the websites had an iFrame added which caused web browsers that visited the U.S. Treasury Department sites to go get malicious content from another site, whose registration is the same as where the malware was coming from when the previous round of Network Solutions sites were hacked. Thus we believe it's the same bad guys who have now managed to infect the web pages of the U.S. Treasury Department.

Leo: Oh, interesting.

Steve: So that's not good.

Leo: No.

Steve: No.

Leo: No. I would say not good.

Steve: Yeah.

Leo: Not a good thing.

Steve: In errata...

Leo: You're not vulnerable, just to update this, just to reassure people, you're not vulnerable unless you haven't patched Windows; right?

Steve: Mmm, yes. It must be that it's...

Leo: It's probably using an existing exploit.

Steve: I didn't run across what it was that they were doing. But as is typically the case, I'm sure this is not a zero-day exploit. Yes, so it's not something unknown to the world. And as long as you're completely patched, you're fine. This is taking, as you say - good, I'm glad you brought that up, Leo - is taking advantage of almost certainly known exploits of various sorts. And typically the content you get will try five or six or seven different things, hoping to catch you out and find one that your machine hasn't patched.

Leo: That's pretty typical, right, yeah, yeah.

Steve: So are you sitting down?

Leo: I'm sitting down. I'm on my ball. Why?

Steve: Okay.

Leo: I'm nervous now. I know what it is, though. [Cackling evilly]

Steve: I have somewhat reluctantly moved into the 21st Century, Leo.

Leo: You know, you're going to disappoint a lot of people with this.

Steve: Yeah, I probably am. I probably am. I have two Twitter accounts.

Leo: Now, why, just out of curiosity, did you do this?

Steve: Well, okay. I first created an account with the not-very-inspired but hopefully memorable name of GibsonResearch. So if you just, you know, Twitter.com/GibsonResearch will, I guess, take you to my page.

Leo: It does. I'm looking at it right now. Now, you've got 57 followers. That's good.

Steve: Okay.

Leo: Do you plan to follow anybody? Or are you just going to...

Steve: I'm not sure. I don't think I'll follow anyone there.

Leo: Okay.

Steve: My intention is, as I understand it, this will allow me to easily post notices of what's going on with GRC.

Leo: I think that's a good idea. We use that for TWiT. We have a TWiT account, and that's exactly what we do.

Steve: Right. And I've been talking about CryptoLink and getting myself ready to get serious about starting it. So this is part of that. I thought it would be a good way for me to easily just sort of trickle out progress reports on what I'm working on, what's going on, new versions are available, what I'm doing. In general, what Gibson Research is doing.

Now, I then decided that maybe I ought to have a different account for, I don't know, the reason most people use Twitter, I guess, like, oh, look, I found some lint in my navel. Now, I didn't want to clutter up the GibsonResearch account with that. So the second account, my personal Twitter account, is AgileSynapse.

Leo: All right.

Steve: A-g-i-l-e-S-y-n-a-p-s-e. So I've created both of those, one for people who don't care about lint navel, or navel lint, rather. You can just follow Gibson Research, and I will keep that one relative to what GRC and I am doing. Or if you do wonder about the cappuccinos at Starbucks and lattes and, I guess, I mean, I'm very hesitant about this. I don't know why anyone would care. But that seems to be what people do these days. So AgileSynapse is where I will post sort of what's going on in my life. And at GibsonResearch Twitter account, what's going on with Gibson Research as it bears directly on sort of the professional side.

Leo: So, good, that makes sense. I'm going to - I just followed them both, Steve.

Steve: I don't know what that means, but I'm glad, Leo.

Leo: You'd better start learning the lingo, Steve. C'mon, now, dude. You're going to be part of the modern world, you've got to learn the lingo. So here's the deal. Basically the GibsonResearch account is a broadcast account, which we have TWiT and TWiTLive. I don't use it to follow stuff. I just - people follow it to find out what's on TWiTLive, like we tweet when the show starts, stuff like that, or that kind of thing. The TWiT account we use to update them on new stuff happening in TWiT and so forth. And then I have a LeoLaporte account. And that one I use both to talk about stuff, but also to read. And I think this is something you've got to consider, is you follow people who are saying interesting things. Doesn't have to be a thousand people. Probably shouldn't even be more than a hundred people. But that way, when you go to Twitter you'll get some stuff other than - and it's kind of considered, I don't know, polite to follow some people. You don't have to. You don't have to.

Steve: Okay. So but stuff's not being pushed on me; right? Like so I go to - I deliberately go to somewhere, and then I see...

Leo: You go to Twitter.com with your account, AgileSynapse, which is your personal account.

Steve: Right.

Leo: And then whoever you've followed will be down the page.

Steve: Oh, okay.

Leo: So that way, if you follow me or some other interesting people, who are saying interesting things, and I think there are probably quite a few people on Twitter that you'd be interested in, then, you know, programmers or tech pundits, I mean, people that are your friends.

Steve: So, and somehow the way this works is time is suspended during the time that you're doing this? So you get, like, extra time in the day? It doesn't count against your 24 hours?

Leo: No. That's the old Italian proverb, nobody ages when eating.

Steve: Oh.

Leo: And that's true. But not twittering. You age considerably while twittering. No, you don't have to. You don't have to go there. But even if you just followed breaking news from CNN or something like that, then when something is happening, it's kind of - you can go there and see what's going on. I mean, I find it very useful. There's traffic reports or...

Steve: Well, it sure is popular, whatever it is.

Leo: Whatever it is.

Steve: I mean, this whole Twitter thing. So...

Leo: Well, you know what, get ready because I'm going to tweet that you have started a Twitter account.

Steve: That's cool.

Leo: And that will give you probably a considerable number of followers.

Steve: Well, we'll find out whether anyone cares

Leo: See what happens, yeah. We'll see what happens. Well, you've got 23 followers

in your personal - actually I should refresh because now that we've mentioned it, maybe you've got more. Let's see if people ran out - yeah, 175 followers. So you just gained 150 followers just in the last 30 seconds.

Steve: Wow.

Leo: Those are people listening to you, Steve, who want to know what you think.

Steve: These are the navel lint people.

Leo: Steve, you're going to have to stop doing that. You're going to alienate them.

Steve: Now I'm going to have to think of something useful to say.

Leo: Yeah. You know, once a day, it's a nice little discipline, you get up in the morning - think of it as a 140-character column. Remember, when you were doing that column...

Steve: Yeah, see, that's just not enough space, Leo. That's, you know...

Leo: Well, it can link back. It could blink back to stuff.

Steve: This is why I use, like, "R" and "U" as single letters instead of, like, actually spelling out words?

Leo: That's okay. You're allowed to do that. But this week alone you sent me...

Steve: This is all really going to be bad.

Leo: You sent me links to two very interesting articles this week alone.

Steve: True.

Leo: From now on, just tweet them.

Steve: You can send links?

Leo: Oh, sure.

Steve: Oh, hey, that's pretty cool.

Leo: You just say, hey, here's an interesting article about Vitamin D, paste the link in, and you're done. And that's of great value, I think.

Steve: Hmm.

Leo: See?

Steve: Okay.

Leo: See?

Steve: Yeah. And so the link's got to be less than 140 characters, though; right?

Leo: No, well, people use link shorteners. And in fact Twitter has a built-in link shortener. You'll find most people don't use the Twitter web page. They will use - I'll recommend Brizzly, B-r-i-z-z-l-y, dotcom. You log in there instead of your Twitter page. And it has built-in shortening. It'll, you know, and Twitter says how many characters you've gone. You'll see. It's very easy to do. Brizzly is a good page.

Steve: Well, I'll think of something fun to...

Leo: Oh, I think you could be very valuable on Twitter. I can't wait to see what AgileSynapse has to say.

Steve: Ah, we'll see what AgileSynapse is up to.

Leo: Okay. You ready? I'm pressing the tweet button. I have to actually count these. Because I go from Google Buzz. Maybe I should just...

Steve: How many people are following you, Leo?

Leo: 185,000.

Steve: Oh, goodness.

Leo: You'll catch up.

Steve: And Dvorak?

Leo: I think 30 or 40,000.

Steve: Okay. And Pirillo?

Leo: He may have millions, for all I know. For a while Twitter was promoting people. And the people who Twitter promoted are in the millions. Because when you first signed up for Twitter, it would say, oh, you should follow whoever, Ashton Kutcher.

Steve: This whole thing's a couple years old; right?

Leo: Believe it or not, it's since 2007.

Steve: Wow.

Leo: It's amazing. I mean, technically it's four years old because they launched in April of 2006. But nobody used it then. It really didn't start taking off till 2007. So I am now, okay, I'm going to say "Hell has frozen over. Steve Gibson is on Twitter." And I'm putting the two things in there. And when you do that, see, it shows you the count. And, oh, here's another little tip for you, Steve. If you're giving somebody's Twitter account, you precede it with the @ sign. And then once you tweet that, somebody can click it and go right to your account. So now you see it's a link, @gibsonresearch is a link, and it shows you there. And people can follow you, you see.

Steve: Wow. I'm going to have to play this podcast back, Leo, so I can see if I can...

Leo: I'm teaching Steve something [laughing].

Steve: So I can watch all this.

Leo: All right. But that's good, I'm glad. Any other updates for us?

Steve: Oh, my goodness. I did want to mention, I listened to you and Paul doing your This Week in Google, talking about patent stuff.

Leo: Yes.

Steve: And one of the things that I've been annoyed by - first of all, I'm delighted that HP purchased Palm.

Leo: Really.

Steve: Oh, yeah.

Leo: It saves them; right?

Steve: Exactly. For that reason. And it means that HP's tablets will not be limited to Windows. Clearly Palm's Web OS is oriented toward being a large tablet.

Leo: Right.

Steve: And so we're going to have Apple-based tablets, based on the iPhone OS. We will have Droid-based tablets. We'll have Web OS-based tablets. And you probably have seen that Blackberry is apparently on the way to doing one?

Leo: Oh, I didn't see that.

Steve: So, yeah, there was a - I saw some pictures of a Blackberry tablet. So but on the patent side, one of the things that's frustrated me is, for example, well, the companies that have gotten in early have all laid down various domains of intellectual property. And unfortunately they're competing with each other and being largely unwilling to share this stuff because they regard their various technologies as competitive edges. For example, I love that on my Blackberry I can hold a key down, and it will initially type in lowercase, but then I'm able, if I hold it down, it switches to uppercase. And wouldn't it be nice to have that on the iPad? But I can't. The iPad can't have it because Blackberry patented that. And so a nice feature like that is, sadly, only available on whatever platform's developer came along and created that idea first. Now, do you know about the sliding your finger off of the shift key on the iPad?

Leo: Oh, you mean, well, I know you can slide, like if you press the period, you slide up, or on the exclamation. For instance, apostrophe's not on the main keyboard, and you need it all the time. So if you tap the exclamation mark and slide it up, you get an apostrophe. Is that what you mean?

Steve: Oh, that's one I didn't know about.

Leo: That's really useful because, well, you need apostrophe; right?

Steve: Yes, I agree. And, for example, you notice that the Apple keyboard does not change the case of the letters. But the Palm keyboard does.

Leo: Oh, that is - oh, you're right, they should.

Steve: They should, but they can't because...

Leo: It's patented.

Steve: ...Palm got that.

Leo: See, that's stupid. Excuse me.

Steve: It's, well, this is what - yes, it is. And it's really frustrating. I mean, the problem is, the U.S. Patent and Trademark Office is now giving patents out with reckless abandon.

Leo: And for trivial applications, or even applications with prior art, frankly.

Steve: Yes. And so what's happening is the consumer is now losing because no one manufacturer is able to take a bunch of good ideas and put them together in one platform. If you want upper and lower case to display on your onscreen keyboard, you've got to go to Palm. If you want to be able to hold a button down and have it turn into capitalization, you've got to go to Blackberry. And so on. And it really is the case that this is no longer benefiting us. And, I mean, for 17 years, we have to wait for 17 years for these patents to expire, and then these ideas go into the public domain.

Leo: I agree. It's a real problem. And originally the idea of the patent system was to encourage innovation, to reward innovators. At this point it's the exact opposite.

Steve: Yup, exactly. And it's basically just giving people a license to sue each other is what it boils down to. I did want to mention that one of my - one of the things I'm very excited about is there is a very nice traditional NNTP, which is the original Internet Network News Transfer Protocol.

Leo: Usenet.

Steve: Usenet. There is a reader called NewsTap which is the only one I could find for the iPhone. I wrote to its developer about three weeks ago and said, hey, is there any

chance that you will do one, or upgrade this one, for the iPad?

Leo: Oh, that would be so good.

Steve: And Leo, it's in beta. And it is spectacular.

Leo: Hallelujah.

Steve: Alexander Strauss, I think, or Clauss, is his name. And he wrote back two days ago. And he said, hey, you asked me about a news reader for the iPad. He said, it's in beta. If you're interested, I'll send you the details. He's looking for a few beta testers, not our whole listenership's worth.

Leo: I'd be glad to. The thing that I'm finding I do much more of with the iPad is reading RSS feeds, reading websites, I mean, it's become a great way to keep up with content, yeah.

Steve: Yes.

Leo: And I'd love to add newsgroups to that list. I kind of gave up on newsgroups. It's been a long time.

Steve: Well, GRC has active newsgroups. And in fact I depend upon them to be interactive when I'm developing software. While I was writing the DNS Benchmark last summer, I mean, the work I was doing was so hugely accelerated by the fact that I had a bunch of guys that were hanging around all over the planet, testing versions that I would release, and in some cases with platforms I didn't have. Like we got it running completely under Linux with Wine. And I didn't have to install Linux and Wine in order to make it go. Actually they ended up producing an image that I was able to run under VMware for a couple things where I really did need to see what they were talking about myself. But it's just hugely leveraging. So I'm just very excited that I will have access to GRC's newsgroups and all the other newsgroups on the 'Net for that matter.

Leo: How do I follow the GRC newsgroup? Is it part of the regular newsgroup download on something - I use, is it SuperNews? I can't remember which Usenet...

Steve: I used to use SuperNews, too. No, we have our own server.

Leo: So I'll have to point it to your server.

Steve: We do not propagate. It's been an issue from time to time. Google started picking, when Google Groups happened, there was some leakage from it, which we've plugged. We just want to keep it sort of a separate piece of the 'Net, sort of off to itself,

to limit the amount of exposure. Also, if we propagated outwards, people would see the postings, reply to them, but they would never come back to us. So these are not public groups, although the news server is just news.grc.com.

Leo: Okay.

Steve: So n-e-w-s dot GRC.com.

Leo: Does NewsTap let you follow multiple news servers?

Steve: Yes.

Leo: Oh, wonderful. I can't wait.

Steve: You can subscribe to multiple servers. It'll pull different groups from each of them and pull them together. And he's done, oh, I mean, he's got a view for the thread which is just graphical and beautiful, where you're able to slide around within this tree view and tap on articles and then read them in the pane below. I mean, it's just - this doesn't look like a first shot. This is already beautifully polished.

Leo: Does he support Instapaper? Do you know about Instapaper?

Steve: I do, and I have it. And I'm trying to think what does. Something I ran across the other day.

Leo: A lot of things, I mean, NewsRack, which I used as a lot of, I mean, to me a reader, a newsreader or a newsgroup reader, it's really nice if it does, for a couple of reasons. One is there's a great Instapaper application on the iPad. Instapaper allows you to say, oh, save that, I want to read that later. Save that, I want to read that later, and put it in Instapaper. And Instapaper will send you an email to your Kindle with your agglomerated Instapaper stories. So you could actually get it on the Kindle, too, which is - you didn't know that?

Steve: Ho ho ho ho ho, very nice.

Leo: Yeah, go create an Instapaper account. It's free, but if you create an account, then you can say send me a weekly or daily digest of the Instapapers. And then you have it on the Kindle, which I love. But I used Instapaper a lot because I'm always in a hurry. I never have a chance to read these articles. So I just save them. I put them aside in Instapaper, and then I have that, I have access to them.

Steve: I will ask...

Leo: Be a nice thing for him to do.

Steve: Yeah, that's great. And then finally we have a reader, I mean a listener, a nice little testimonial. In fact, his name is Lance, and he's in Concord. Didn't say where, but maybe Massachusetts. Or maybe California. He says - or any of the other Concords that I'm sure must be dotted around the country. He just said "Another testimonial for your show." He said, "My sister and I don't talk much. So I was surprised when she and her engineer husband called me. They know I'm a computer nut, and they had already tried everything they could think of, including running a different data recovery product, although they didn't recall its name. They had neglected to back up their PC for over three years."

Leo: Oh, good lord.

Steve: "Okay, at this point I really wanted to tease them. He's an engineer, for crying out loud. And now the drive wouldn't boot. I sent them your program, and they promised to purchase it if it saved their PC." Which I think is entirely reasonable. "Needless to say, SpinRite used its necromantic powers to raise their drive from the dead, saving their pictures from the last four Halloweens" - I guess that's more on the necromantic theme - "from the last four Halloweens, as well as all the data on the drive. While I know they will buy the product, I still doubt they'll back up their PC. Thank you for the great podcast and the great product." And thank you, Lance, for your note.

Leo: The multi-verse. This is part of our basics of computing; right?

Steve: Yes. We discussed two episodes ago the concept of a stack, which is one of the fundamental core principles that underlies all current computing. I mean, there just isn't anything you could replace it with. And visually, just to remind our listeners, you can sort of model it on the stack of dishes in a cafeteria, where there's like it's spring-loaded, the idea being if you were to place dishes on this stack, they sort of are absorbed by it. That is, they go down into this well. And as you remove them, you naturally get them in the reverse order that you put them on.

So the way the computer uses this, say that you had four registers, and you needed to put their - you needed to save them somewhere safe while you were doing something else with them, and then later you wanted to restore them. Well, you could simply say, push register one on - the contents of register one on the stack. Push the contents of register two on the stack, the contents of register three on the stack, the contents of register four on the stack. And you don't have to worry about where the contents went. This abstraction, this cool idea of a stack where it's a region of memory with its own pointer, the stack pointer, which automatically advances to always be pointing at the top of the stack.

Then in our example you do anything you want to with these four registers, not worrying about disturbing their contents, because you've saved them on the stack, as the jargon goes. And then when you're done, you would - where "pushing" something is the term for putting it on the stack, "popping" it is the term of removing it from the stack. So you would pop the stack into the contents of register four, pop the stack into the contents of register three, pop the stack into the contents of register two, pop the stack into the

contents of register one. That is, in the reverse order. So just as the plates would go down and come off in the reverse order, so does the contents of the stack.

And the point is that this is sort of a place that, as long as you always back out of what you're doing in the reverse sequence that you went in, the stack is your friend. It always provides you what you're expecting. And so that's a - that's one fundamental principle. The second we talked about in the last episode, which was hardware interrupts, the idea being that, if you wanted to do a couple things at once, if you wanted, for example, if your software program wanted to be printing, well, a printer operates vastly slower than the code running in the computer. And that means the printer can at some level only accept one character at a time.

Say that it was like an old-style teletype where you're going chung-chung-chung-chung-chung-chung-chung, you know, 10 characters per second, typing them out on a yellowish kind of roll of paper. You don't want to sit around in your program waiting for the teletype to say, oh, I finally got through printing the A, now I'm ready for the next character, whatever it is, because that would require that your program could do nothing else at the same time. It would just have to sit there watching the status of the teletype until it's ready to accept the next character.

So this notion of hardware interrupts was generated where you could have the hardware interrupt your program, which is off doing something else. And that interrupt, what's called an interrupt service routine, which services the hardware interrupt, it would yank control away from your program, just right in between the execution of any two instructions. You would never know when it was going to happen. In fact, that program running in the foreground, it would sort of even be unaware that anything had happened because, in between successive instructions, control would get yanked away from it by the hardware interrupt, which would save the registers, whatever they happened to have in them at the time, which would then free up those registers for its own use to, for example, get the next character in an I/O buffer and hand it to the teletype and start that being printed. Then it would restore the previous interrupted state of the machine and jump right back to the instruction that was about to be executed in the normal main foreground process. So hardware interrupts were the beginning of allowing us to do multiple things at once.

Well, the next evolution of that is when we said, wait a minute, we actually do want to do more things with a computer at once, not just printing in the background, but what about timesharing? What about having multiple users who are all able to use the same computer? And of course timesharing was a big innovation that we saw in the early '70s because these computers were many tens of thousands of dollars. And it became clear that, well, we've got this hardware which is much faster than a single user. I mean, even someone, Bill Gates in 1973, at Harvard, using the teletype on one of these early machines, he was typing relatively slowly, I mean, because you couldn't type fast on those machines. You could only input things at 10 characters per second, no matter how fast a typist you were.

And so what they realized was, wait a minute, the computer is sitting around waiting for the next character to be typed in. We could have 10 teletypes connected to this computer, and each user would feel like they had the computer to themselves. The question was, how do you share a machine among multiple users? How do you do timesharing?

And what they realized was, we'll have hardware interrupts generated by a timer. And that was, just that little change, that was a massive innovation because now what this meant was that part of the hardware, part of the fundamental hardware of the computer

would be some piece of hardware, a little timer chip, or a divider which was dividing down the frequency of the main clock crystal, down into something like maybe 100 Hz, or maybe it used the AC line, actually used the AC current, which is 60 cycles in the U.S., 50 cycles in Europe. And every time the AC waveform crossed through zero, which would be 120 times a second, or 100 times a second, that could generate a hardware interrupt.

And the idea was that software, you could essentially have five programs, say, all loaded into memory at the same time. And then you'd have a supervisor. And here we begin to get the concept of an operating system. At that point we'd have a supervisor which would be supervising the execution of more than one program at once. Well, you needed some way to get control, to sort of yank control back from any one of these programs back to the supervisor. And hardware interrupts was the secret for doing that.

So when you started up the computer, you'd have the supervisor, sort of an early form of an operating system, which would initialize the hardware interrupt on the so-called "counter timer," or a periodic interrupt which occurred very often, but not too often, typically several hundred times a second, something like that. Maybe a thousand in later, faster machines. But initially you were basically slicing up time based on these interrupts. And every time the timer expired or the AC line crossed through zero volts, like 100 or 120 times a second, that would generate an interrupt that would give control back to the supervisor. It would look to see how much time had gone by and decide if the same program should keep executing, or maybe if it was time to give somebody else a chance.

And so essentially all of the programs which were running at once, sort of - as we know, computers don't actually run things, well, actually they do later on, when you have multi-core. They actually are running more than one thing at once, which we'll get to in a second. But back then they were just - they were time slicing. They were giving, often in a round-robin fashion, that is, program one, then program two, then program three, then program four, then program five, then back to program one, they would - this supervisor would get control and then hand control back to the next successive program. Meanwhile, each of those five programs had essentially been interrupted.

So execution would start on the program. And after this period of time, this clock interrupt would occur, yanking control away from that program, so putting it in sort of an interrupted state, which was fine. It would just sort of patiently sit there until the supervisor decided it was time for it to get another time slice, to get some more slice. And so the supervisor would essentially return from that interrupting event back to the program, having restored the state of the machine, the way the program had left it, and then it would continue executing just like nothing had ever happened. So this was sort of the first notion of there being sort of a nascent operating system, the supervisor, that would start up programs. And the supervisor was able to keep track of the passage of time by these periodic interrupts which had the power, thanks to the hardware architecture, of yanking control away from anything going on at any time.

Now, later on, programmers said, okay, well, I've got a program here, and it's nice to be able to have other programs running at the same time, but I'd like my own one program to be able to do multiple things at once. And what was invented was the notion of a thread, the idea being that a traditional original sort of older style program could only be inside of itself, could only be doing one thing at a time. That is, you'd have - you'd call it a single-threaded program, which is to just sort of say a non-threaded program. That is, it's only doing one thing at a time.

But what we realized was you could - the same abstraction which allowed us to apparently be running five programs at a time, could also give you the ability to apparently be actually in many places of a single program at a time, each one of those

places being a thread. And so essentially it's like timesharing or multi-tasking, which is timesharing, but within a single program. Thus that's called multi-threading. And so what happens is, when the program starts, it always starts with just one thread. That is, you know, it starts at the beginning of the program. And so it goes along until it decides it wants to sort of branch out. It wants to be able to, for example, maybe it wants to be in charge of sending characters out to a device while it's doing something else, much like I was talking about before. And so it's literally able to spawn a thread that is, like, it's able to, within the program, it's able to say, okay, create another thread and start executing it at this location.

So because in a traditional single-core system the CPU actually only has a single program counter, which can only be in one location at a time, we know that we're not actually executing multiple threads, or in this case these two threads at the same time. But once again we have a timer as part of the hardware which is able to get control back to the supervisor. And so whereas before the supervisor was dividing all time up into, for example, five main programs, now it's more granular. Now it's dividing time up, not only among whatever programs may be running, but within those programs, among whatever threads each individual program may have created. So this is a tremendous benefit and very powerful thing that modern programs are able to do because it sort of gives the individual programmer within an individual application the ability to set different things running and let them sort of handle themselves.

You have to be careful, I mean, and this is where some of the skill of programming comes in because we're, as we do this, we're escalating complexity, in some cases dramatically. For example, you need to be careful about not having the threads collide with each other or not having multiple threads trying to do something that conflicts with each other because essentially the threads don't know, they have no awareness of when they're running and when they're not. Each thread sees itself as if it's running all the time, when in fact that time is being chopped up among all the threads in a system. You could have many different threads in different processes, that is to say, different programs, and from sort of a thread-centric view they just sort of see that they're running along executing, unaware that chunks of time are missing during which time other threads in the system have the actual access to the processor, and they're running moving forward.

So Intel at one point said, you know, this is all well and good, but what if we actually supported multiple threads in the hardware? What if, instead of this threading being a complete abstraction, what if we actually created hardware that could do sort of more things at once? And so this was before multi-core, but this was what they called "hyper-threading." And the idea was that you could have multiple program counters and share a lot of the resources of the processor; but, for example, have multiple sets of registers. And the idea would be that, so long as the programmers set these things up right, you could actually be doing multiple things at once. That is, the hardware could.

And so we went for a couple years like that. And then of course Intel went to the next stage and gave us multiple cores, where now you actually had full processors which, again, had their own program counters; they had their own registers; they had all their own resources. They shared memory. So that was the common thing that they shared, as well as the hardware, like the display and so forth. That is, they had common access to the system's I/O and also to its memory resources. But otherwise they were independent.

Well, by the time that this happened, operating system technology had evolved substantially. I mean, we already had Windows and multiple tasking and multiple threads. And what was so cool about this notion, first of hyper-threading and then of this

multiple core, was that you'll notice that our operating system software and the applications we were running, nothing changed. That is, we didn't have to have different applications. We didn't have to have a dramatically different operating system. That is, there was this, sort of this notion of multi-core support. But from the user's standpoint, our systems just got faster. Things worked better.

Well, and the reason that happened was that we had already, before that time, there was this mature concept of multiple processes and multiple threads. And the operating system was given the task of deciding what the computer was actually doing at any one instant. And the operating system was jumping the program counter all over the place, giving all these different threads a chance to run until the next hardware interrupt yanked control back to the operating system, where it would decide what to do. And it was the so-called "scheduler," the operating system scheduler is, I mean, it alone, the optimal strategy for that is the subject of doctoral theses and whole books on computer science, how to maximally schedule all of these things that are going on simultaneously in the computer. Do you want to allow some things to run with high priority? Do you want some things to run in the background? Do you - how do you manage the simultaneous demand across all of this?

But what was so interesting was that the abstraction of all these threads essentially running at the same time, that is, in a given program, the threads were set up so they thought they were all going at once. And individual programs in the operating system thought they were all going at once. So when we added multiple cores, they actually could be going at once. And the way we got away with this was there was already the discipline in place about not having them conflict with each other. That is, there was already some interprocess containment so that processes couldn't stop on each other's memory. And within a process there was already inter-thread discipline to make sure that these threads that conceptually were all running at the same time behaved themselves and didn't mess with each other's business, essentially.

So when we added hardware that actually was able to run multiple things at once, first through this hyper-threading that was sort of poor man's multi-core, and then real multiple core, well, everything just got faster. It scaled beautifully because everything we already had, the whole structure was oriented toward the simultaneity which initially we were faking out, but later we were actually able to do it.

So that's basically the multi-verse of contemporary computers. And what we're all sitting in front of right now is all of that going on at once, all behaving itself, for the most part. When it doesn't, things definitely go sideways. But when everybody plays their part and behaves themselves, we end up with just an incredibly powerful facility for computing.

Leo: It's been fun watching Intel slowly add these capabilities, as you say, with hyper-threading and multi-cores. And they also use something called QPI, interprocess communications now in the new i3, 5, and 7 chips. They've really gotten very sophisticated about how to do this. It's no longer just time slicing. They're really getting sharp about it. And it does make a difference. And I think the reason that they put so much effort into it is they realized that they couldn't put more megahertz out, that they were getting all sorts of problems past 3 GHz. And so they backed off and said, well, we're not going to have a 6 GHz processor. How else can we improve speed? Multiple processors.

Steve: Well, and Leo, I mean, 3 GHz.

Leo: Is a lot.

Steve: Three billion. Three billion cycles per second. I mean, it's nuts. The first IBM PC was 4.77 MHz. And we were all thinking, whoa, this puppy just - it cooks.

Leo: I remember very well when we first started doing TV for Ziff-Davis in the early, I guess it was between '92, '93, '94. And Bill Machrone came in, and he brought in the new Pentium. And it was 90 MHz. Megahertz. And we were all saying, oh, it's so fast. And I remember using it, saying, wow, it feels slippery, it's so fast. It's moving around. That was 90 MHz.

Steve: Yeah.

Leo: I mean, we are now more than 30 times faster. In just clock cycles. And then you double the core or quadruple the core until it's got six core processors coming out. Each is hyper-threaded. That's 12 processes, 12 threads at a time.

Steve: Well, and what's really interesting, too, is that while the processor performance has been able to scale generation after generation after generation, main memory stalled, essentially. The so-called DRAM, Dynamic Random Access Memory, its technology turned out to be stubbornly non-scalable. So, and it's just due to the way, just due to the nature of the way DRAM operates. There are, you know, we hit essentially the physical fundamental limits of its speed much sooner than we did processors. And so what's been done is, we've made it wider, that is, so that one memory access is fetching many more bits, sort of in width, at a time.

And then we've also had - we've gotten very sophisticated with caching. Because one of the things that the early researchers noticed was that computers would sort of go over here in memory for a while and work in some loops and do some work in a certain region. And then they'd go over somewhere else for a while and do some work in a different region. I mean, it wasn't like they were, like the actual execution was randomly occurring throughout all of memory. There was a regionality to it.

And so what they realized was that, if you had a much smaller but much faster memory, a so-called cache, a stack memory that was able to be made to be kept at the same speed as the registers of the computer, where they could run in a single cycle, what they realized was, well, you might have to wait, essentially, to read what's in main memory into the cache to get it into the cache. And that would take a while. The processor would essentially, it might be stalled while you were painfully slowly pulling main memory into the cache. But once you got the cache loaded, then the processor could run again at full speed, full sort of local speed, as long as there were, as the term is, a high percentage of cache hits as opposed to misses. So that while the processor was doing things locally, staying there, it was able to execute again at full speed.

And so that's - and they extended this notion, not to just a single level of cache, but to then two levels of cache, having a very high speed, but smaller; and then a semi high speed, but larger; and then, like, finally, main memory, where we just can't make it any faster, unfortunately.

Leo: Yeah, in fact we know L2 cache, that Level 2 cache, the amount of that makes a huge difference in overall processor speed.

Steve: Yes. And when people are selecting processors, it's also expensive because it takes up chip real estate, and it takes up power. And so one of the things that purchasers do is they decide, oh, yeah, how much speed do I really want, because the chips with the larger Level 2 cache, the L2 cache, they're going to be more pricey.

Leo: Right. It's a great subject. At some point maybe, I don't know if there's enough for a whole show, but I'd love to talk or hear you talk about cache, the concepts behind caching. Because we see cache all over computing. We used to see hard drive caches, there's browser caches, and of course there are these processor level caches. And it's all the same idea, I would guess.

Steve: It's absolutely, it's very similar.

Leo: And there's a lot of computer science into how you do this; right?

Steve: Oh, well, yes. And there's also write-back versus write-through. And there's this notion of the cache being dirty. That is, the processor also wants to be able to write into the cache. It's not only that data from main memory is pulled into the cache, but the processor may be modifying the cache. So the question is, do you start writing back to main memory when the cache is changed? Or do you wait until you have, like, a different flush event? That is, for example, the cache may be changed a whole bunch of times, in which case you wouldn't want to be writing it back all the time. But then you've got a problem, for example, with multiple cores, multiple processors, because they need to have a coherent view of what's in memory. And if one processor's got modified memory in its local cache on its chip, then the other processor can't see into that. It's seeing main memory. And so it gets very tricky.

Leo: Hmm. Love to hear more about that someday. Very tricksy. By the say, Steve, you've crossed the 1,000 follower count on both GibsonResearch and AgileSynapse on Twitter. So a thousand people now are waiting for your first tweet, AgileSynapse.

Steve: I will try not to disappoint anyone.

Leo: And I'm sure that next we come back, after this podcast comes out, there'll be several thousand more.

Steve: Cool.

Leo: Who knows. Who knows. See how high we can get it.

Steve: Well, I will use the GibsonResearch Twitter account to let people know what's going on with Gibson Research, and AgileSynapse to let people know what's going on with me. I don't know why anyone cares, but...

Leo: We do, though.

Steve: ...we'll see how that goes.

Leo: We care, Steve. We really do. Steve Gibson is the man in charge at GRC.com, the Gibson Research Corporation. Go there for, of course, this show, both 64 and 16KB versions for people with low bandwidth. There's also transcripts there, show notes and more. And GRC.com is the home of ShieldsUP!, Shoot The Messenger, DCOMbobulator, Wizmo, all those free things Steve does, Perfect Paper Passwords. And that one, one thing that he charges you for, SpinRite, the world's best hard drive maintenance and recovery utility. If you don't have it yet, and you have hard drives, you need it. SpinRite from GRC.com. Steve, we'll see you next week.

Steve: We'll do a Q&A.

Leo: Oh, yes.

Steve: And answer any questions. So do have our listeners - GRC.com/feedback will take you to a web form where you can tell me what's on your mind. I will read them, and we'll choose a bunch of good ones and talk about them next week.

Leo: And they're reminding me in the chatroom that you might want to turn off email notifications of new followers on Twitter because you just got 2,000 emails. Congratulations.

Steve: The good news is I did understand that much, at least.

Leo: Steve's no fool.

Steve: Yeah, I had that turned off before I'd breathe a word of it to Leo.

Leo: You know who we really hurt with that was the wonderful woman in Auckland, New Zealand, LisaTickledPink, who we just randomly choose on TWiT as somebody that everybody should follow, and she got something like 50,000, I don't know, some huge number of emails almost instantly. And I felt bad. She thought she'd broken the Internet. She thought she did something wrong. I feel terrible. Thanks, Steve. We'll see you next week.

Steve: Thanks, Leo.

Leo: On Security Now!.