（財）武田計測先端知財団
2006／12／20

りーなす とーばるず

リーナス トーバルズ

(リナックス 2･7 カーネル･メンテナー、オープン･システム･デベロップメント･ラボ)

リナックス開発リーダのプロファイル

ヘルシンキ大学の学生の時にリナックスの最初のバージョンを開発し、インターネットを通して公開。オペレーティングシステム開発に興味を持つ人々がリナックス･コミュニティをつくり、十五年間以上活発な開発活動を続けている。今日では、組み込み用からスーパーコンピュータ用まで幅広く使われ、サーバ用としては、ウィンドウズに次ぐ第二位のオペレーティングシステムとなった。

リーナス トーバルズとリナックス開発

リーナス トーバルズとリナックス開発に関しては何冊かの本が書かれている。主なものとしては、リーナス トーバルズとディビッド･ダイアモンド共著の「それが僕には楽しかったから」[1]、グレン･ムーディ著の“rebel code”[2]がある。また、産業技術総合研究所からの委託でリナックス開発の中でマーケットメカニズムがどう働いたかの調査を行った報告[3]を筆者が書いた。今回は、リナックスの開発経過ではなくて、リーナス トーバルズという人間に焦点をあてた話を書いてみたい。

産業技術総合研究所から委託を受けた調査を行った2003年11月に、五人のリナックス開

1 『それが僕には楽しかったから』風見潤訳、小学館プロダクション刊、
原著は“Just for Fun”, Linus Torvalds and David Diamond, Waterside Productions
2 “rebel code”, Glyn Moody, Persus Publishing（日本語訳はない）
3 『MOT 事例研究　注目先端技術　成功の理由』、赤城三男他著、工業調査会刊

発のキイパーソンにインタビューすることができた。リーナス・トーバルズ氏本人、リナックス2・6のカーネル・メンテナーであるアンドリュー・モートン氏、オープン・ソース・デベロップメント・ラボ（ＯＳＤＬ）ＣＥＯのスチュアート・コーエン氏、ＩＢＭ社の General Manager of e-business on demand のロス・マウリ氏、インテル社シニアフェロー兼 General Manager of Software & Solutions Group のリチャード・ヴァート博士の五名である。これらの人々のリーナス・トーバルズについてのコメントやリーナス・トーバルズ氏自身のコメントを主な情報源とした。また前述の二冊の本からも貴重な情報を得た。始めに感謝しておきたい。

誰にでも開かれているリナックス・コミュニティ

リナックスの開発は、リナックス・コミュニティの技術者によって行われている。コミュニティとは、ソフトウェアの開発者たちのゆるやかな集まりのことである。ソフトウェアを開発した人がそのソフトウェアを、他の人が解析したり、修正したり出来るように、ソフトウェアのソースコードをインターネット上で公開する。別の開発者が、そのソースコードを解析したり、修正したり、自分の必要とする機能を開発したりして、自分が開発したソフトウェアを同じように公開する。このようにして、ソフトウェアの開発が行われる。コミュニティには、ソフトウェアの開発はしないが、自分で使って障害を見つけたらその障害を報告する人もいるし、単に使ってみるだけの人もいる。このようにさまざまな人間が公開されたソフトウェアを中心にして自由に開発を進めるのが、コミュニティによるソフトウェア開発である。ソフトウェアのソースコードを公開するオープンソース・ソフトウェアの開発においては、このコミュニティが重要な役割をはたす。

リナックスの場合はどのようにして、コミュニティが出来たのだろうか。リーナスは、コミュニティを作ろうと思って作ったのではない、と言っている。ただ、リナックス 0・01をリリースする前から、数人の人と電子メールのやり取りはしていた。コミュニティができた背景に、電子メールがあったことは確かである。リナックス・コミュニティについて、リーナスはこう言っている。「コミュニティを作るのは、簡単ではない。ある人がプログラムを開発しようとして、コミュニティを作ろうとしても、うまくいかない。リナックスの場合は、本当にオープンであるように気をつけた。リナックスを使い始めた人や、何か質問をしたい人でも、開発に参加していると感じられるようにした。他のオープンソース・ソフトウェアの開発プロジェクトの場合は、よく、そのプロジェクトのために一生懸命に働くコアグループというのを作る。コアグループは、一人のこともあるし、五人から十人のこともある。そういうものを作ると、コアグループ以外の人は、自分が重要な開発メンバーの一人だと思えなくなってしまう。」

また、誰でもパッチ[4]をリーナスに送ることが出来るように気をつけ、だれからでもパッチを受

[4] プログラムを修正した場合、前のプログラムとの差をパッチという。大まかには、新し

けつけた。送られたパッチについての議論は、公開の電子メールリスト[5]を使って行い、みんなが参加できないコアグループのようなところでは、議論しなかった。リーナスは障害を見つけることは重要であり、ある意味では開発者よりも障害を見つけるユーザの方が重要だと考えている。だから、コミュニティがユーザにも開かれていることが重要だと考えている。

参加している人みんなが、自分のプロジェクトだと思えるように注意しないとうまくいかない事を、モジラ（Mozilla）・プロジェクトの初期の例をあげて説明してくれた。モジラ・プロジェクトは、ネットスケープ社がブラウザ[6]のソースコードを公開し、オープンソース方式で開発をしたプロジェクトである。ネットスケープ社は、商品化したバージョンの独占的な権利を維持しようとした。このため、ネットスケープ社以外のコミュニティメンバーは、第一級市民ではないと感じて、やる気をなくしてしまった。やる気を持ち続けてもらうには、びっくりさせないこと、アンフェアだと言われるようなことをやらないように気をつけなければいけない、と言っている。

このようなやりかたができたのは、リナックスの開発を始めた時に、リーナスは学生であり自分が一番良く知っているとは、とても思えなかった。まったくその逆で、自分の周りの人の方が良く知っていると思うことがしばしばあったことも手伝っている、と言っている。事実リーナスは、障害を指摘してそれを修正するパッチを受け入れ、自分が書いたコード[7]にこだわることは無かった。

わかりやすく、保守しやすく

しかし、同時にリーナスはリナックスのコードを、よりきれいに、よりわかりやすく、より保守しやすいようにしておくことに気を使っていた。ユニックスのシステムとファイルをやり取りするためのＮＦＳ（ネットワーク・ファイル・システム）という機能をスラドキーという人が開発して、リナックスの下で動かそうとした時に、リナックスの方の障害が見つかったことがあった。その時リーナスはできるだけ簡単に解決する方法を選ばずに、正しく解決するやり方を選んだ。つまり、リーナスとスラドキーの仕事を増やすことになっても、リナックスのシステムを強化する方のやり方を選んだのである[8]。

この話は、リナックスの立ち上がりの時から、リーナスがリナックス・プロジェクトをどのように運営するかについて、よく考えており、その考え方をずっと貫いてきたことを示している。一方では、自分がコミュニティのメンバーにどうして欲しいかよりも、コミュニティのメンバーがどうし

く開発された部分的なプログラムと考えればよい。
[5] 電子メール送り先のリスト。関心のある人は、このメールリストに登録すれば、全部のメールのやり取りを見ることができる。
[6] インターネットホームページを見るためのソフトウェア
[7] プログラムのことをコードという。
[8] rebel code, P61

て欲しいのかを、まず考えた。同時に、リナックスのコードを保守しやすく、将来の機能拡張にも耐えられるようにしておくことを考えて意思決定をしていた。

このような開かれたコミュニティは、現在でも維持されている。例えば、リナックス 2・6で一番多くコードを書いたのは、アンドリュー・モートンだが、その半分ぐらいは、多くの人が書いた小さなコードをまとめたものである。おそらく五百人以上の人たちがこのような形でリナックスの開発に参加している。コアグループのメンバーを固定した場合は、そのメンバーしか開発に参加しないが、リナックスの場合は、このような形で多くの人が開発に参加している[9]。

リーナス自身は、就職するときに、できるだけ中立的な立場が取れるような会社を選んだ。リナックスのディストリビューション[10]をやっている会社に入ることもできたが、そうすると自分ではいくら中立な立場をとっても、そのディストリビューション寄りの立場だと思われてしまう。だから、ディストリビューションとは関係がない、トランスメタ社に入った。今は、ＯＳＤＬに入ったので、もっと中立的な機関の人間になった。

なぜ、才能のあるエンジニアがリナックス・コミュニティに参加するのか

沢山の才能のあるエンジニアがリナックス開発に参加している。これがリナックス成功の一つの鍵となっている。才能のあるエンジニアがリナックス・コミュニティに参加するのは当然だ、とリーナスは考えている。

沢山の技術者が、フルタイムの自分の仕事をした後で、面白くてチャレンジングなことをやりたいと思っている。ソフトウェアは、面白くてチャレンジングであり、何かのソフトウェアを開発したいと思う人は多い。しかし、実際にプロジェクトを始めるのは大変である。なぜなら、開発のための基盤つくりをはじめとして、すべてを一から作らなければならないからだ。そうするよりは、もう始まっているプロジェクトに参加して、自分が思うような変更を加える方がずっとやり易い。だから、本当にそのプロジェクトに興味があれば、それを手伝うのは容易なことだ。実際、リナックスに興味を持ったある人は、オペレーティングシステムを作るプロジェクトを始めようとしたが、やらなければならないことが多すぎたので、一からはじめることをあきらめてリナックスの開発に参加したとリーナスに言っている。リナックスの場合は、基盤となることは全部できている。だから、技術的に面白いことをやりたいと思って、沢山の人がリナックス開発に参加した。

モチベーションの高い人がいたら、その人が毎日フルタイムで開発に参加する必要はない。

---

9 Doc Searls , Linux Journal November 24, 2003
http://www.linuxjournal.com/article.php?sid=7272

10 リナックスそのものはインターネット上で入手できるが、たくさんのプログラムが必要であり、素人には難しい。誰でも使えるように、リナックス関係のソフトウェアを集めてCD などに入れてパッケージとして販売する団体や会社をディストリビューションという。

例えばリナックスのユーザとしてフルタイムの仕事をして、その後で毎日一時間をリナックスのテストに使うのでも良い。障害を見つけてそれを解決するのに、毎日一時間ぐらい使って三週間かかったとしても、そのことに興味を持ってくれれば、大変に助かる、とリーナスは考えており、人々をモチベートするのは、そんなに難しくないと言っている。

難しいのは、やりたくないと思っていることを頼んで、その人を疎外してしまわないようにすることだ。もし、何かやって欲しいことがあるときには、どうするのかと聞いたら、「丁寧に頼むようにしている。例えば、何か問題があったときには、その問題の報告書を送って、『直してよ』、とは言わないで、『ちょっと見てくれないかなー』、と丁寧に頼む。もっとも、よく知っているやつなら、そんな気を使わないで、『直せよ』と言ってもいいけど」、と言っている。

コミュニティにおける社会的秩序

リーナスはリナックス・コミュニティには、ソーシャルオーダー（Social order、社会秩序）があると言っている。ソーシャルオーダーというのは、エチケットのようなものでコミュニティの人間は容易に理解できるものであるという。

「顔を会わせるか、電子メールを使うかにかかわらず、ソーシャルオーダーというものがある。それは押し付けるのではなくて、存在しているものだ。マネージャのような人はいらない。まったく自然に組織される。ソーシャルオーダーはあるが、この人がリーダです、などと紙にかけるものではない。ソーシャルオーダーを維持するのは、活動以外にない。時々、どこかわからないところから出てきた人が、数ヶ月のうちにとても重要な開発者になったりする。こういうことは会社ではとても難しい。会社のなかでは、暗黙のソーシャルオーダーの他に、固定した構造があるからだ。誰かがリーダで、あなたはこの間のどこかに位置するという風にソーシャルオーダーを書き下ろすと、人々はとても防御的になる。だれかをソーシャルオーダーの順序で上位におくと他の誰かがびっくりする。ソーシャルオーダーは固定的なものではなく、もっと動的なものだ。人々は、自分は何かあることが得意な人間として、みんなに知られているかがわかっている。その状態を続けるためには、あることが得意な人間であり続けられるようにしなければならないこともわかっている。」

新しい人がコミュニティに加わって、このソーシャルオーダーのダイナミクスを知らなくて、何かふさわしくないことをしたら何が起こるのだろうか、という問いに対しては、次のように答えている。「ソーシャルオーダーは、なにか固定的なものではなく、エチケットのようなものだ。だから、おかしなことをやると、ソーシャルオーダーが動き出し議論が起こる。物事がどう行われるべきかの小さな議論が沢山起こる。例えば、前にうまくいかなかったことを、やるべきだと新しい人が言ったとすると、その人はそうではないという沢山の電子メールを貰うことになるだろう。どこでも同じようなことが起きる。結婚式にふさわしくない服装で参加したら、人々があなたを

見る目つきで、ふさわしい服装をしていないと気がつくだろう。だから、ソーシャルオーダーは、書き下ろされたルールによるのではなくて、どう物事が働くかによるものなのだ。」

大きな絵と細かな絵

リーナスは、二つのタイプの開発者が必要だと考えている。大きな絵が得意な人と、細かな絵が得意な人だ。オペレテーティングシステムの開発では、それぞれの部分をきちんと作る必要があるから、その部分の詳細が得意で熟知している人が必要だ。だが、そういう人にプロジェクトを任せてはならない。大きな絵を理解していないからだ。ハードウェアを動かすプログラムであるドライバを例に取ってみよう。ある特定のハードウェアのドライバを考えると、そのハードウェアができることを実行させなければならない。あいまいなところはない。だから、細かな絵が得意な人が必要だ。同時に、大きな絵が得意な人も必要である。大きな絵が得意な人は、細かな絵が得意な人に、何がしたいかを話す必要がある。細かな絵が得意な人は、そうしたいならこうすればうまくいくという事を、言ってくれるかも知れないからである。大きな絵が得意な人と、細かな絵が得意な人はお互いに近くにいなければならない。

実際には、いろんなタイプの人がいる。狭い領域の大きな絵が得意な人もいる。リーナス自身も、自分は大きな絵が得意だと思っているが、いくつかの小さな領域も受け持っている。この両者は同じように重要である。

どうしてコミュニティの活動が十五年以上も活発に続いているのか

コミュニティの活動が十五年以上も活発に続いているのは、次から次へとやることがあったからだと、リーナスは考えている。オペレーティングシステムの中核部分は、ハードウェアと他のソフトウェアの間を取り持つソフトウェアだ。プロセッサのハードウェアは次から次へと新しいものが開発される。新しいハードウェアが出来るたびに、どこかを直さなければならなくなる。リナックスの場合は、このように、次から次へとやらなければならないことがあった。それに対して、コンパイラ[11]や他のソフトウェアは、ハードウェアに依存することがないので、何年かすると完成してしまい、やることがなくなる。エキサイティングなことがなくなるから、プロジェクトに留まる動機が少なくなり、活動が停滞していく。

新しいハードウェアに対応しなければならない、というのはひとつの例で、常に新しいチャレンジングなことがあることが、コミュニティの活動が活発に長く続く要因である。使い方が変わり、ユーザ層が変わるというのも新しいチャレンジである。だから、リナックスに限らず、長く続いて

[11] コンピュータのプログラムを人間が書きやすくするために開発されたコンピュータ用の言語をコンピュータで動くように変換するプログラム。FORTRN とか C 言語などがある。

いるプロジェクトは他にも沢山ある。

また、リーナスは、コミュニティの中ではポリティカルな議論をしないようにした、とも言っている。技術的な議論は、結論を出しやすい。例えば、どちらのバージョンが高速に動作するかは、実際にやってみればはっきりする。ポリティカルな議論は、どちらが正しいかは、はっきりしない。延々と議論を続けて、お互いに嫌いになってしまうようなことが起きる。そうするとコミュニティの活動はうまくいかなくなる。フリーかオープンかというような議論がそうだ。

副官たち

リーナスは、全部の決定を自分でやっているわけではない。彼は、自分はリナックスの中核部分であるカーネルには興味を持っているが、ユーザレベルのプログラムには興味を持っていない、と言っている。最初のバージョンでは、自分で全部決めなければならなかったが、次のバージョンからユーザレベルのプログラムに興味を持っている何人かの人に、その分野を任せるようになった。つまり、副官(Lieutenants)たちに任せたのである。その意味では、意図的に副官をつくった。カーネルでも、ドライバのようにリーナスが興味をもてない分野もあり、そこは副官たちに任せた。

副官たちは本当にその分野に興味を持っているのだろうか、と聞いたら、リーナスは、興味を持っていなければ良い仕事はできないと信じている、と答えた。長い目でみたら、良い仕事をしてもらうにはそうしなければならない。ある分野に興味がある人はすぐにわかるし、そういう人は、その分野の仕事を自分で始めてしまう。だから、意図的に決めなくても決まる。人を選ぶのではなくて、人が何をするかを選ぶのだ、と言っている。

通常の企業の開発プロジェクトでは、これだけの仕事をしなければならないが、この部分は良い人がいるが、この部分には良い人が見つからないというようなことが起きるが、という質問に対してはこう答えている。「リナックスでも同じことが起こる。例えば、新しいバージョンをリリースするときに、古いハードウェアでの動作検証をする人がいない、というような場合がある。そういう場合に、古いハードウェアは古いバージョンで動かして貰おう、新しいバージョンを古いハードウェアで動かすことはあきらめよう、と決めることもある。もし、古いハードウェアで新しいバージョンを動かすことが本当に大事であれば、それは広くわかっているはずであり、その仕事をやることに興味がある人が現れるはずだ。企業の開発の場合は、だれも興味を持っていないから、やらなくてもいいはずだとは言えないが、リナックスの場合は、ユーザからのフィードバックループがコミュニティにあるから、だれも興味をもっていないならきっと大事ではないのだろう、と言うことができる。」

安定版と開発版

リナックスには、いつも二つのバージョンがある。安定版と開発版である。安定版は、実際の応用に使うためのバージョンで、開発版はその上で開発を行うバージョンであり、新しく開発された機能を順次追加してリリース[12]する。現在は、安定版は、2・6で、開発版は2・7である。安定版を担当するエンジニアを見つけるのは難しいのではないかとの質問に対しては、こう答えている。「そうは思わない。新しい安定版をリリースしたあとしばらくは、二つのグループは一緒に仕事をする。こうすることによって、担当する部分について、共通の見方をするのに役立つ。」しかし、メンテナンスの仕事をする人は、バグの修正をしないわけにはいかないのだから、彼らにも夢を与えなければならないのではないかと聞いたら、彼は、「だから自分は安定版の仕事はしない。私は新しいことをすることで、動機付けられている。安定版の仕事をすると、動機付けを失い、悪い仕事をしてしまう。」と答えた。

さらに、「安定版を商用に使う人は、開発途上の開発版には興味がない。その人たちによって安定版は良くメンテナンスされる。彼らがビジネスをするためには、安定版は良くメンテナンスされていなければならないし、彼らがメンテナンスするのは、必要だと思う人がやるのだから一番良い仕事ができる。それは私の哲学にもあっている。多くのメンテナンスの仕事は、プログラムを書くのではなく、問題を追跡することや、その問題を経験したユーザと話すことや、その問題のパターンを捕まえることだ。また、安定版を使った開発もある。例えばドライバの場合は、開発版は不安定なので、安定版を使って開発し、それを開発版に移植するという開発方法を取ることが多い。」と言っている。リーナスの言う興味を持った人が興味を持ったことを担当するのが一番良い仕事が出来るということも確かであるが、開発版をやりたいが、安定版の仕事をせざるを得ない、それでも面白いから、やっているという人もいるようである。

リーナスの意思決定スタイル

コミュニティの人間がリーナスの意思決定を受け入れるのはどうしてか、と聞いたら「受け入れないよ」という彼一流の言葉の後、以下のように話してくれた。「意思決定をする時の、ひとつのガイドラインは、シンプルということだ。シンプルとパフォーマンスのトレードオフだ。」また、彼は「自分の決定が間違っていれば、変えることにまったく異存はない。人々はいつも自分の決定を受け入れるわけではないが、技術的な面では自分を信頼している。また、GPL[13]によって、ソースコードは全部オープンになっているから、自分が混乱してしまったら、別の人がその人自身のバージョンを作ることも出来る。つまり、システムとしてそういうことができている

[12] バージョン番号をつけて誰でも使えるようにインターネット上に公開する。
[13] The GNU General Public License で、ソースコードを公開するソフトウェア開発では、一番よく使われているライセンス契約。リナックスも GPL を使っている。

のだ。自分が彼らに自分の決定に従うように強制しているわけではない。誰がリナックスの意思決定をするかは選挙のようなものだ。彼らが好まない人を再選したりしない。」とも言っている。

リナックスの場合の意思決定というのは、開発されたある部分のコードをリナックスソースツリーに入れるかどうかを決めることである。リーナスは、こう言っている。「投票のようなものはない。私はマーケットの信者だ。私がマーケットと言うときは、お金のことではなく人々が実際に使ってくれるかどうかということである。あることが今までよりもうまくできるという新しい機能を売り込んできた時には、実際に人々がそれを使い始めたら賛成する。私はそういう意味でマーケットを信じている。民主的な投票ではない。私は、マーケットに基づいて自分自身の意思決定をする。どう動くべきかを強要するのではなく、プラグマティックにどう物事が動いているかによって決めている。多くの人たちがこう動くべきだと思っていても、実際は違う動きをすることが多い。現実を取らなければならない。」

本当の意味でリーナスの意思決定を必要とするのは次の二つの場合だと言っている。ひとつの場合は、ある種のマシンでリナックスを使おうとすると、そのマシンの環境ではうまく動くように設計されていないため良い性能で動作しないということが起こる。そこで、完全に設計のやり直しが必要であると決める。実行するのは大変だが、やり直しをしなければならないことは明白だから、選択としてはわりに簡単な選択である。

もうひとつは、どう物事を実現するかである。始めのころは、パッチを受け取るといつも自分で書き直しをしていた。パッチを見れば何をやりたいかがわかるし、何をやりたいかがわかればそれを実現するよりよい方法を思いつくからだ。これは、カーネル[14]をどのように維持するかについての意識的なシステム上の意思決定だった。ある時からは、書き直しをする時間がなくなったので、ある人のパッチはそのまま受け入れることにした。この人とこの人とこの人はそのまま入れても大丈夫、そのまま入れよう、と言うわけだ。

この二種類のことに対しては、リーナスの意思決定が重要である。どちらも、やってみれば優劣がすぐわかり、自然に決まるもというものではない。最初の場合は、書き直しにどのくらいの努力が必要かを考えて、どのバージョンで書き直すかを決めなければならない。例えばバージョン2・6で、ディスクドライブのためのプログラムである、ブロックレイヤーを書き換えた場合がその例で、沢山のドライバがあり、沢山の箇所を書き換えなければならなかったのでリーナスの意思決定を必要とした。もうひとつはリナックスのカーネルをどのように維持するかについての意識的なシステム上の決定だが、それが良かったということはいろいろな開発が行われた後で実証される。この二種類のことに対して、リナックス開発では大体は正しい決定をし

[14] リナックスの中核となる部分をカーネルという。

てきた。このことが、リナックスを今日のようなポジションにしたと言えるだろう。

それ以外の意思決定プロセスは、自然なものである。「意思決定自身が意思決定する」とでも言うべきプロセスだ。例えば、誰かがスケジューラ[15]のベンチマークテストをやって、ある場合には性能が落ちるというような面白い結果を出したとする。スケジューラに関心を持っている人は沢山いるから、誰かが新しいコードを書いてその結果を示す。ここまでくれば、もう意思決定というものではなくなる。もちろん、新しいスケジューラが良いものだから、それを使うことになる。このように、スケジューラを改良するとか、SMP[16]のロック機構を変えるとか、は一人かあるいは少人数のグループが OK ということができる。メモリシステムは、沢山の人が関与しているが、担当する部分に分かれているので、それぞれの人間が自分の担当する所について決めることができ、鍵となる部分の性能がよくなったことを示すことができる。このほうが良いと言うことを誰かが示しているので、意思決定というようなものでなくても決めることができる。

他の人のリーナスについてのコメント

今回インタビューした4人が、リーナスについてどのように思っているのだろうか。以下はその簡単なまとめである。

リナックス 開発グループのナンバーツーでリナックス2・6のカーネル・メンテナーである、アンドリュー・モートンは、以下のように言っている。「みんながリーナスの決定を受け入れるのは、彼を尊敬しているからだ。たまには、リーナスも間違えることがあるが、あまり間違った決定をしない。平均すれば非常に良い打率をあげている。間違った決定をした時も、彼を説得して正しい決定に直すことができる。みんなは、プロジェクト全体がうまくいくと思ってリーナスの決定を受け入れている。私は、リーナスほどの個人的なオーソリティはないが、私の中にリーナスの影をみているので、私の決定を尊重するのだと思う。」

ＯＳＤＬのＣＥＯである、スチュアート・コーエンは、リーナスはアントレプレナーか、と言う質問に対して、このように言っている。「リーナスは、ビル・ゲイツのようなビジョナリーになることには興味がない。リーナスは、将来のビジョンや方向性について話すのは好まない。リーナスは、オペレーティングシステムを作ることを好む開発者であり、将来のオペレーティングシステムを作りたいと思っている。強い意志を持っており、オープンソースとリナックスを強く信じている。」

---

15 コンピュータがやるべき仕事の優先順位を見ながら、次に実行する仕事を決めていくプログラム。

16 Symmetric Multiple Processor のことで、複数のプロセッサが同等の立場で仕事を分担する手法。ある仕事を特定のプロセッサに割り当てるというようなやり方よりは柔軟性があり、さまざまな応用に対応できるが、技術的には高度な手法。

IBM社のロス・マウリは、同じ質問に対して、以下のように答えている。「私のリーナスに対する印象は、優秀なコンピュータ科学者でソフトウェアエンジニアであるばかりでなく、仮想的なコミュニティであるリナックス・コミュニティをリードする能力を持っているということだ。リーナスは、多分コミュニティの５パーセントぐらいの人間としか、顔を会わせたことがないだろう。それでも、意思決定をしてコミュニティをリードしていく彼の能力は並外れたものだ。最も感心するのは、電子メールを使ってやり取りしながら、プログラムをレビューし、ある人のプログラムを受け入れ、他の人のプログラムを受け入れないと言う意思決定をしている点だ。リーナスは大変難しい意思決定をしなければならないのに、コミュニティ全体が彼と共にある。例えば三人の別々の人間が同じ機能を開発したとする。最終的には、一つを取り、他の二つを取らないという決定をしなければならない。リーナスの意思決定のやりかたは、コミュニティがリーナスについてきて、コミュニティが強くなっていくやり方だ。非常に面白い価値のあるリーダーシップを持っていると思う。

インテル社のリチャード・ヴァートは、同じ質問に以下のように答えている。「リーナスは、確かに新しい世界にいる。他にも同じような人間がいる。例えば、アパッチを開発した人間がそうだ。アパッチは、リナックスのインターネットサーバ用のアプリケーションプログラムで、ウインドウエズNT[17]の上でも動く。リーナスは、この新しい開発スタイルにおいて、もっとも良く知られた、もっとも影響力のある人間だ。ちょっと残念なのは、オペレーティングシステムが無償だと言うこと、ビジネスモデルがないと言うことだ。最終的には成功なのだが。彼は、賞とかグラントなどの成功だけをみており、ビル・ゲイツのようなビジネスモデルと反対の方向に行くように勇気づけられている。このような二つの開発のやり方が、いつどうなるかはわからない。起こりそうなことは、オペレーティングシステムは無償となり、アプリケーションとサービスを売るようになることだ。マイクロソフトにも、カーネルをあきらめ、アプリケーションとサービスを売るというような同じビジネスモデルを取らせるかもしれない。」

プラグマティックな技術の見方

リーナスは、オペレーティングシステムについての、学問的な研究は、もう必要ないのではないかと考えている。

リナックスは、POSIX[18]に基づいており、POSIXは長い間変わっていない。1960年代や1070年代には盛んにオペレーティングシステムの研究が行われたが、もう誰もやっていない。1960年代や1970年代にユニックスが出来た時に、ユニックスを開発した人たちは非常にプ

[17] マイクロソフト社が開発したオペレーティングシステム
[18] Portable Operating System Interface for UNIX のことで、IEEE で決めた UNIX が最低限持つべき仕様。

ラグマティックだった。動くものは動く。ところがある時点からオペレーティングシステムは、とても抽象的になり過ぎた。人々が何を望んでいるか、何が現実の世界で実際に動いているかを気にしなくなってしまった。つまり、オペレーティングシステムを、技術的なメリットに基づくのではなくて、どう動くべきかに基づいて設計し始めた。だから、リナックス開発の初期段階で、リーナスがタンネンバウム教授と、マイクロカーネルかモノリシックか[19]の論争をしなければならなかった。

リーナスは、ユニックスは大きなサーバに重点を置きすぎた、と考えている。リナックスは、大きなサーバのためだけでなく、小さな組み込みシステムのためにも開発されなければならない。小さな、組み込みシステムの場合でも、どんどん機能を組み込むことが必要となり、大きなシステムの能力が必要になる。リーナスは、「携帯電話の場合が良い例だ。無線網を通じて話をするというひとつのニッチな要求に答えるものだった。今では、それだけではなくて、インターネットにアクセスしたり、画像を送ったり、電子メールを送ったり、様々なことが出来るようになった。だから、小さなニッチに特化しているわけにはいかない。携帯電話の世界の人は、無線ネットワークは得意だが、他の事は不得意かもしれない。電話の世界から来た人には、リナックスは大きすぎると思われるかもしれない。リナックスは、そういう人が必要としないことも沢山出来る、これは事実だ。けれども、長い目で見れば、どの道ほとんどの機能が必要になる。ハードウェアは安くなり続けるし、特定のニッチマーケットから出発した組み込みシステムが、実際にはリナックスを使うようになる。これはリナックスだけでなく、ウィンドウズCEでも同じだ。」

彼の新しい技術に対する見方はプラグマティックである。技術そのものが難しいかどうかよりも、リナックス上にインプリメントすることが難しいかどうかを考えている。SMP[20]は難しいだろうと言ったら、彼は「SMPは難しい技術だが、沢山の文献があり良くドキュメント化されている。だからインプリメントそのものは難しくない。もちろん、注意深いインプリメントが必要だが。」と答えた。リナックス 2･0では、いわゆるビッグカーネルロックをつかった。ひとつがすべてをロックしてしまうやり方だ。スケーリング[21]はあまり効率よくいかないが、それは段階的に改良していけばよい。SMPは面白いので沢山の人が開発に参加した。沢山の人が、他の人のやり方は良くないと考えていた。沢山の人が、SMPを追加するとシングルプロセッサのパフォーマンスが落ちると心配していた。事実シングルプロセッサのパフォーマンスが落ちることが良くあった。ユニックスの場合だが、システムVのユニックスとソラリスにSMPサポートを追加した時の

---

19 どちらもオペレーティングシステムを開発するやり方。この論争については、注 3 の『注目先端技術』　ページ 180 参照。

20 注 15 参照

21 プロセッサの数を増やしたときに、数に比例して性能が向上するかどうかの度合い。

パフォーマンスの低下はひどかったことは良く知られている。特にファイングレイン[22]をサポートした時がひどかった。だから、最初は大きなカーネルロックから始めるのは自明の選択だった。しかし、その後で実際にファイングレインを始めたときは、非常に注意深くやった。何人かの人がベンチマークを書き、ベンチマークをやりながら進めた。

リーナスもシングルソースツリーでいいかどうか考えたことがあった

リナックス 2・0で始めてSMPをインプリメントした頃、シングルソースツリーでいいのかどうか確信が持てなかった。六十四から三十二CPUのマシンをサポートする開発をしていたときに、ツリーを分けなければならないのではないかと思っていた。ゴールがちょっと違っているなら、シングルソースツリーを維持するのはもっとやさしいし、違いを調和させることもできるだろうが、ゴールが違いすぎると思った。

だが、現在では、シングルソースツリーを維持すべきだと思っている。シングルソースツリーでスケールアップとスケールダウンで対応するべきだと思っている。現在では二百五十六CPUまでのスケールアップできる。ソースコードツリーを分離する必要はない。

もう一度、分離することを考えたことがある。メモリ管理機構のないCPUでリナックスを動かす時に、分離したほうが良いのではないかと思った。実際CEリナックスというツリーを作った。2・6ではこれをマージして、六十四CPUのモンスターマシーンから、メモリ管理機構を持たないマシンまでをサポートするようになった。

人となり

リーナス トーバルズ氏とのインタビューの最後に、財団スタッフからの質問をいくつかした。その答えが、彼の人物をあらわしていると思うので、以下に収録した。

問:「武田賞受賞後の一番大きな業績はなんですか。」

答:「人々を幸福にし、やる気にさせようと思ってきた。2・6のリリースがその一つの例です。」

問「ユビキタスコンピューティングの世紀にはいったと言われますが、これについてのコメントはありますか。」

答「面白いと思うのは、コンピュータがどんどん小さくなっていくことだ。メインフレーム、ミニコンピュータ、パソコン、で今はどこにでも隠せるぐらい小さくなった。これが、われわれの行動

[22] SMP のプログラミングのやり方の一つ。ユーザのプログラムを実行する時間を出来るだけ多くとるために、カーネルの動作を細かく分けて、ユーザプログラムの実行を禁止する時間を少なくするようなプログラミングのやり方。カーネルの制御が複雑になるため、作り方によっては、効率が落ちることがある。

をどう変えるかだね。」

問:「情報技術のエンジニアとして至福を感じるのは、どんな時ですか。」
答:「家族のことを除いて言うとすると、なにか有効なことを出来たと思ったとき。例えば、バグを直して、『直ったよ』と言うとき。」

問:「日本やアジアで働きたいと思いますか。」
答:「あんまり思わない。ネットワークでどこにいても仕事は出来るし。ベイエリアはアメリカの中でも暮らしやすい。カルチャーがアメリカのほかのところとは違う、フィンランドの方に近いと思う。」

自然体のリーダ

リナックス開発の中心人物であるが、責任者としての気負いのようなものをまったく感じさせない人だ。自分の思う方向に引っ張って行くのではなくて、自然に全体が良い方向に流れていく、そんな感じである。が、オペレーティングシステム開発の考え方のような肝心な部分での見解の違いがあると、グッと切り込んでくる。相当に厚いソフトな表面層の中に、しっかりとした芯が感じられた。

仕様は明確であるが、内部の論理構造は非常に複雑であるというオペレーティングシステムのカーネルの性質そのものがこういうやり方に適しているのであろう。複雑な論理構造のプログラムは作ってみれば出来が良いかどうかはわかってしまうわけだから。自分がやりたいと思うことをやる時と、自分が使いたいと思う機能を開発する時が、一番良い仕事が出来る。そういうやり方でプロジェクトが運営できている、ということ自体が驚きである。普通は、どこかで肩に不必要な力が入るものだが、そういうことはまったくない、やはり天性のリーダである。

2004年8月にサンフランシスコで開かれたリナックス・ワールドExpoには、ビジネススーツを着た人たちが沢山参加し、今までとは違った雰囲気になったと報じられている。リナックスが、商用に本格的に使われるようになった証左だ。企業の技術者も沢山開発に参加するようになり、リナックスが企業に乗っ取られてしまうのではないかと心配する人も多い。しかし、リーナスは企業の技術者が必要だと思う機能を開発し、ソースコードをオープンにして、リナックスのソースツリーに組み込むことは良いことだと言っている[23]。また、リナックスをある企業が独り占めしようとすれば、膨大なメンテナンスの作業を抱え込むことになるにで、誰もそんなことはしない、とも言っている。このようなやり方でどこまでリナックスが発展していくか、楽しみである。

[23] http://www.linuxjournal.com/article.php?sid=7280

オープンソース・ソフトウェアの広がり

オープンソース・ソフトウェアはますます広がっている。リナックスやアパッチのような基本的なソフトウェアから、開発ツール、データベース管理システム、アプリケーションサーバまで広がっている。大企業に席を置く技術者がフルタイムで開発に参加することも多くなったし、大企業が開発したソフトウェアをオープンソース・ソフトウェアに出してしまうことも行われている。業界標準を定めるためにオープンソース・ソフトウェアを開発するようなことも行われ始めた。オープンソース・ソフトウェア開発コミュニティは、いわばソフトウェアベンダーにとって共同研究部門であり共同開発部門のようなものとなっている。同じオープンソース・ソフトウェアであるが、コミュニティのあり方はそれぞれ特徴があるようだ。

一人の大学生が作り始めたリナックスが、ウィンドウズに対抗できるオペレーティングシステムにまで成長した。最初のバージョンを公開したときには、とても考えられないような大きな波及効果をもたらした。こういうソフトウェア開発のやり方があることを実証して見せたことがリーナスの一番大きな貢献である。「自分が興味を持ったことをやるときが一番良い仕事が出来る」ということが、今日のオープンソース・ソフトウェア開発に生かされていると思われる。

リナックス、結婚、アメリカ移住

本稿では、リーナス トーバルズという人物の焦点をあてたため、リナックスを開発するようになった経緯や、リナックスそのものの開発については、あまり記述していない。ごく簡単な要約を以下に書く。詳細は、注1から3の書籍を参照して頂きたい。

幼いころは、おじいさんが使った古い計算機を貰ってそれで遊んでいた。これがコンピュータプログラムに興味を持った始まりである。

大学生だった1991年のお正月に三台目のコンピュータとして、インテル386というプロセッサを使ったコンピュータを買った。これを大学の大型計算機と電話回線でつないで使うために、必要な機能を自分で開発し始めた。自分が必要だと思う機能を次々に開発したので全体を見るとオペレーティングシステムに近くなった。そこで、本格的にオペレーティングシステムを開発しようと決心して、開発に取り組んだ。開発をはじめて出来上がるころに、オペレーティングシステムを開発しています、ということを電子メールで公表したら、いろいろな人から反響があった。それに勇気付けられたこともあり、1991年9月にそれまでに開発したものをバージョン0・01として大学のサーバ上に公開した。フリークスと言う名前にするつもりだったが、サーバに公開する場所を用意してくれた教育助手の人が、リナックスという名前をつけてしまった。

1992年の秋には、リーナスはヘルシンキ大学の教育助手になった。翌年の1993年の秋

から半年は、「コンピュータ・サイエンス入門」を担当した。コンピュータの入門コースのようなものだったので、学生に電子メールを送る宿題を出したら、トーベと言う女子学生が「デートをしてください」というメールを送ってきた。この女性とリーナスは結婚した。

1994年3月にバージョン1・0を公開した。オペレーティングシステムとして一通りの完成をしたわけである。1・0は、ヘルシンキ大学のコンピュータ・サイエンス学科の大講義室で発表した。リーナスはここで講演した。この模様はフィンランド・テレビで放映された。この間に、リナックス開発コミュニティがだんだん作られていった。

1994年8月にノベル社が謝礼を出すからユタ州に来てユニックスについて話をしないかと招待された。カリフォルニアにも行けるなら、という条件で初めてアメリカの地を踏んだ。この時いつかアメリカで暮らすようになるだろうという予感を持った。

1995年にヘルシンキ大学の研究助手になり、リナックスの開発に自分の時間を使えるようになった。1996年にバージョン2・0を公開した。その後、1999年に2・2を、2001年に2・4を、2003年に2・6を公開している。

1996年から、修士課程も修了に近づいたので就職することを考え始めた。アメリカのトランスメタという会社で働いていた仲間から、トランスメタに来ないかという誘いがあった。リーナスがトランスメタに就職する事を考えているという噂が広がると、他にもいろいろなところから誘いがあった。リナックスのディストリビューションをやっている会社からは、トランスメタよりも良い条件で来ないかと言われた。リナックスの関連企業で働く気はなかったので、トランスメタに就職することにした。

1997年2月にビザも下りて、前年に生まれた娘とトーベと一家三人でアメリカに渡った。トランスメタでは、日中はトランスメタの仕事をし、それ以外の時間はリナックスのために使った。トランスメタとの契約では、勤務時間中にリナックス関係の仕事をしてもよいという保証を貰っていた。

アメリカに渡ってから、スターたちに会わなければならなかった。アップルのスティーブ・ジョブスに一緒にやらないかといわれたが、ジョブスは自分の目標、なかでもマーケティングに興味があるのに、リーナスは技術面に興味があるので断った。サンのビル・ジョイにも会ったが、サンのオープンソースの考え方には賛成できなかったので断った。

2001年に武田計測先端知財団から、「オープンなコンピュータ基本ソフトウェア開発モデルの提唱と実践」に対して武田賞を受賞した。

2003年6月にトランスメタを退職して、ＯＳＤＬのフェローになり、リナックスに専念するようになった。

この間、1999年にストックホルム大学から名誉博士号を授与され、その後ヘルシンキ大学からも名誉博士号を授与されている。

略歴

1988 年　ヘルシンキ大学入学

1991 年　Linux　バージョン 0.01 を開発、公開

1997 年　トランスメタ社入社

1999 年　ストックホルム大学より名誉博士号を授与

2000 年　ヘルシンキ大学より名誉博士号を授与

2001年　武田賞受賞

2003年　オープンソースデベロップメントラボ　フェロー