翻訳

Linuxの心

Linus Torvalds
翻訳：安藤　進

原文: "The Linux Edge"
Communications of the ACM, Vol.42, No.4, pp.38-39（Apr. 1999）．原著者より許可を得て翻訳

Linuxは今日，数百万人のユーザ，数千人の開発者に利用されており，市場も大きく成長した．Linuxは，埋め込みシステムやロボット制御装置，スペースシャトルにも使われている．実は「こうなることは分かっていたのさ．俺様だって密かに世界制覇を狙っていたんだ」．もちろん，これは冗談だが，正直に告白すると，いささか驚いている．Linuxユーザが百人レベルになることは予想していたが，百万人レベルまで増加するとは思いもよらなかった．

【原著者はLinuxの開発者で，Linuxという名前は，Linus作のUNIXを意味している．Linuxは今ではオープン・ソース・システムの代表例である．】

Linuxが成功したのは，移植性の向上や広範な普及を当初目標に設定したからではなく，設計方針と開発モデルがよかったからである．土台がしっかりしていたからこそ，結果として，移植性が向上し幅広く普及したのである．

当初，Linuxが念頭に置いていたのは，インテル社の80386 CPUアーキテクチャだけだった．しかし現在では，PalmPilotからAlphaワークステーションに至るまで，どのようなアーキテクチャでも実行できるようになった．LinuxはパソコンのOSとしても広く移植されている．Linuxでプログラムを作成すれば，そのプログラムはLinux以外のさまざまなマシンでも実行できる．Javaの歌い文句ではないが，「1回作成するだけで，どこでも実行できる（write once, run anywhere）」のである．

ここで，Linuxの設計方針とその後の開発経過について述べておきたい．Linuxが当初の思惑とはやや異なるものになっていった経緯が分かるだろう．

現在のLinuxで実現できている機能の中には，マイクロカーネル方式でしか達成できないと思われていたものがたくさんある．開発当初は，マイクロカーネル風のアーキテクチャでなければ無理だというのが一般的な風潮だった．だがわたしは，マイクロカーネルは，(a) あくまで実験的なものであり，(b) いずれもっと複雑になることは確実であり，(c) そうなれば，実行速度は目に見えて遅くなる，と感じていた．

現実の世界でOSのスピードは非常に大切だ．当時，マイクロカーネルの処理速度を高速化する手法が盛んに研究されていたが，それらの手法の中には従来のカーネルの高速化にも適用できるものがたくさんあると，わたしはにらんでいた．

そこで，代表的なアーキテクチャすべてに共通する要素を抽出して，1つの汎用カーネルモデルの作成を試みた．その結果，移植性に優れたLinuxカーネルが実現できた．マイクロカーネルだけで汎用性を追及すると，抽象化層といったものを別個に用意する必要があり，スピードが犠牲になる．

一方，複数のカーネルモジュールを使用して，ハードウェアに固有のコードを特定のモジュールだけに限定すれ

ば，コアカーネルの移植性を高めることが可能になる．カーネルモジュールの効果的な使用例としてデバイスドライバが挙げられる．ハードウェアに固有の処理をデバイスドライバに任せるのである．

ハードウェアに固有の処理をすべてコアカーネルに組み込むと，スピードは速くなるが移植性は悪くなる．ハードウェアに固有の処理をユーザ空間に移すと，スピードが遅くなり，動作が不安定になりやすい．両者の中間に位置付けられるのがカーネルモジュールである．

移植性に優れていることが，Linuxを取り巻く開発者の世界に活気をもたらした．多数の開発者が参集し，Linuxカーネル周辺部の同時開発が可能になった．だが，カーネルの中心部だけはわたしの目の届く範囲に置いた．

Linuxの汎用アーキテクチャは，カーネルの変更をわたしがチェックできる枠組みを提供してくれた．適度な抽象化は，さまざまなアーキテクチャごとに枝別れした別々なコードをわたしがいちいちフォローする作業から解放してくれた．Linuxの開発にどんなに多くの人々が参加してきても，コアカーネルはわたしの手元から離れないですんでいる．また，カーネルモジュールは，本来別個に開発すべき部分をさまざまなプログラマにそれぞれ独自に開発してもらうのを可能にしてくれた．

Linuxのカーネル空間をできるだけ小さくするという方針は正解だったと思う．現時点ではカーネルの大幅な変更はあまりないと思う．ソフトウェア開発プロジェクトは，ある時点で成熟期に達し，それ以降の改良作業はペースダウンするのが普通である．カーネルに関しては新たに大幅な改良を必要とすることはあまりない．今後の課題は，できるだけ多くのシステムをサポートすること，つまり，新たに登場する新システムにLinuxを移植することがポイントになるだろう．

カーネルの拡張を求めるアプリケーション領域がいくつかある．たとえば，Webサービング（Web serving）がそうだ．これは，カーネルの負担が非常に重くなるので，いつも対応に苦慮してきた問題である．対応を誤ると身の破滅になりかねない．LinuxをWebサービング用のプラットフォームにしてほしいという要望がたくさん寄せられている．この要望に沿うには，Webサービング専用にLinuxを最適化しなければならない．確かにWebサービングは重要だが，それだけがすべてではないと考え，わたしは自重している．

わたしの願いは，Linuxに時代の最先端を走ってもらいたい，できれば最先端の少し先を歩んでもらいたいということである．今日最先端の技術であっても，明日にはありきたりの技術になってしまう．将来の課題は，クラスタリング，SMP（Symmetric Multi-Processing），埋め込みシステムだ．しかし現在最も活気のある開発領域は，カーネル空間ではなくユーザ空間にある．カーネルに多少の変更があるとしても，カーネルの外部や周辺で予想される改良のほうが大きくなると思う．Linuxのカーネル開発がどのような方向に行くだろうかなどよりは，Red Hat 17.5やWine（Windowsのエミュレータ）が数年後にどうなるかという話題のほうがはるかに興味をそそる．

今から15年も経てば「Linuxにできることならなんでもできるさ．だけど，おれのは20年間もお相手はできないから，もっと小さくて速いものにしてみせる」と胸張って挑戦する人物が登場するだろう．「Linuxは80386用の設計だった．今の新しいCPUなら，別のやり方でできる．Linuxなんて，もう時代遅れだ」という人も出てくるだろう．実はLinuxの開発を始めたころ，わたしもこんなことを言っていたものだった．将来，だれかが我々のコードやインタフェースを参考にして，バイナリレベルでの互換性も実現するだろう．そうなればわたしとしても本望だ．

謝辞　この翻訳では，いくつかの疑問点について執筆者のLinus Torvaldsから有益な情報をいただいた．また，日立教育部の田口昭二氏に下訳に目を通してもらい貴重なアドバイスをいただいた．この両氏に感謝を申し上げる．

（平成11年6月13日受付）