2020年度版
Linuxカーネルの歴史レポート

v5.8 出版元：The Linux Foundation（電子書籍版）| 2020年8月

Signed-off-by: Kate Stewart <kstewart@linuxfoundation.org>
Signed-off-by: Shuah Khan <skhan@linuxfoundation.org>
Signed-off-by: Daniel M German <dmg@uvic.ca>
Reviewed-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
Reviewed-by: Jon Corbet <corbet@lwn.net>
Reviewed-by: Konstantin Ryabitsev <konstantin@linuxfoundation.org>
Reviewed-by: David A Wheeler <dwheeler@linuxfoundation.org>
Reviewed-by: Jason Perlow <jperlow@linuxfoundation.org>
Reviewed-by: Steve Winslow <swinslow@linuxfoundation.org>
Reviewed-by: Mike Dolan <mdolan@linuxfoundation.org>
Reviewed-by: Craig Ross <ccr@linuxfoundation.org>
Reviewed-by: Alison Rowan <arowan@contractor.linuxfoundation.org>

www.linuxfoundation.org

THE LINUX FOUNDATION

もくじ

概要

2020 年 8 月 2 日に 5.8 がリリースされ[1]、5.9 のマージ ウィンドウが終了したことで、過去 29 年間に記録された 100 万件以上の Linux カーネルのコミット履歴を分析できるようになりました。このレポートでは、Linux カーネルの歴史について、および最大規模と謳われるソフトウェア コラボレーションを可能にしたベストプラクティスとツール基盤の影響について振り返ります。5.8 カーネルは、いくつかの新たな記録も生み出しているため[2]、開発が遅くなっている兆候は見られません。

はじめに

過去数十年にわたって、Linux は着実に成長し、最も広く使用されているオペレーティング システム カーネルになりました。 センサーからスーパーコンピューターまで、ロケットの打ち上げ、自動車、スマートフォン、時計をはじめ、日常生活で使用されるさまざまなデバイスで使用されています。Linux Foundation が 2008 年に Linux Kernel Development Reports の発行を開始して以来[3]、わたしたちは、ある時点から次の時点までの Linux カーネル開発の経過を観察してきました。

ただし、1991 年の最初のリリース以来、開発の全容を共有することはできていません。最初のリリース以来、Linux は 29 年間で 2 万人を超えるコントリビューターが参加し、2020 年 8 月の時点で 100 万件以上のコミットがあり、史上、最も成功したコラボレーションの 1 つになっています。最近発表された 5.8 リリースはこれまでで最大規模のものであり、この最新のリリースは 1 時間に 10 件を超えるコミット数という新記録も達成し、開発スピードが低下している兆候は見られません[2]。

このレポートでは、1991年9月17日の最初のリリースから2020年8月2日までのLinuxを見ていきます。この歴史的な分析は、Dr. Daniel Germanの取り組みと彼の開発したツール`cregit`のおかげで可能になりました[4, 5]。このツールを使用すると、1991年9月17日の最初のリリース[6]から、2020年8月2日の最新の5.8リリースまで、コントリビューターによるトークン レベルでのカーネル コミット履歴を調べることができます。`cregit`を使用して、Linux開発の3つの異なる開発環境（バージョン管理使用前、`BitKeeper`、`Git`）を連続的に見ることができました。バージョン管理使用以前の段階では、Linuxの各リリースを開発のスナップショットとして扱っています。これは1991年9月から2002年2月4日まで続きました。2002年2月4日は、`BitKeeper`が使用され始めたときであり、Linuxカーネルのコミット履歴が収集できるようになりました。現在のバージョン管理システムである`Git`は、2005年4月16日に使用が開始されました。これらの3つの段階の開発履歴が関連付けられ、カーネルソースコードの進化を追跡できるようになりました[7]。

`Git`は、ラインレベルでLinuxの進化を追跡することができます。`Git`を使えば、コード行の挿入や最終的な変更を特定できます。`cregit`を使用することで、トークン レベルでカーネルの進化を追跡することが可能となりました。ソースコード プログラムのトークンは、プログラム内の最小要素です（プログラミングにおけるトークンは、人間の言語の単語の概念に似ています）。したがって、各行のすべてのトークンについて、誰がいつそれを挿入したかを追跡することができました。

linux-0.01.tar.Z カーネルがリリースされた時、それは 88 個のファイルと 10,239 行のコードからなり、単一のハードウェア アーキテクチャ i386[8] で実行されていました。今日、v5.8 リリースは 69,325 ファイルと 28,442,673 行のコード（**110,354,844 トークン**）で構成されており、30 以上の主要なハードウェア アーキテクチャ [9] で動作します。

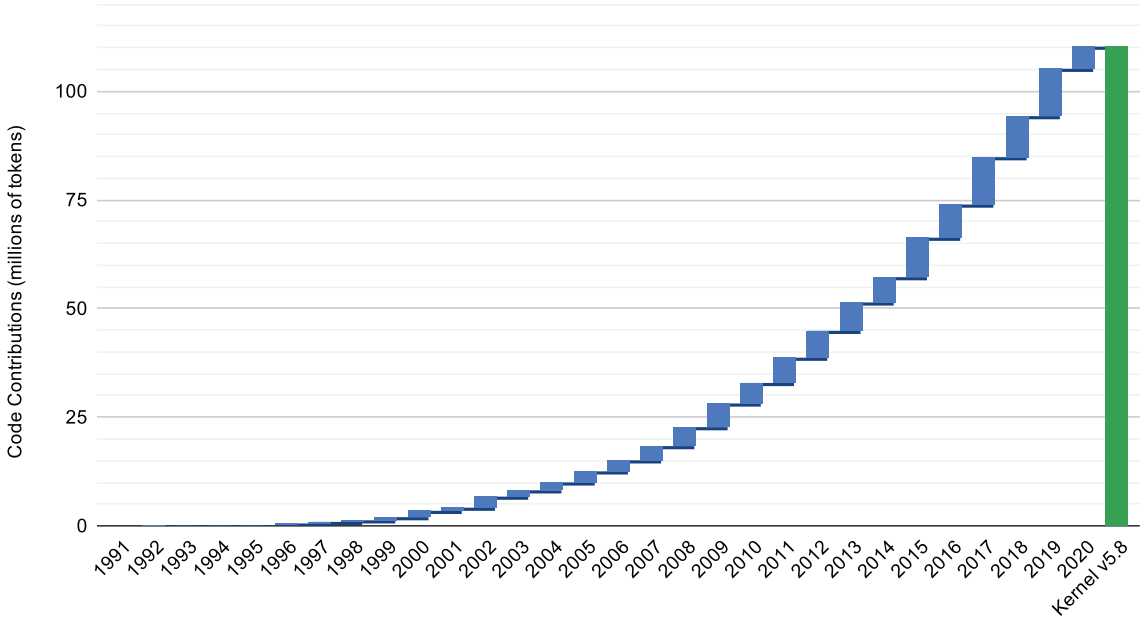
カーネル考古学

その最初のリリースの統計に関する Linus のメモ[8] では、最初のリリースは「**88 ファイルで約 1 万行あり、Lars Wirzenius と共同開発した vsprintf ルーチンを除いて、私が書いたものです**」と要約されています。今日 vsprintf ファイルを見ると、最初のコミットからのソースコードが含まれている数少ないファイルの 1 つであることがわかります。

出典 : https://creGit.linuxsources.org/code/5.8/lib/vsprintf.c.html

2,964 のトークンは 1991 年までさかのぼることができ、5.8 のカーネルにまだ残っています。実際、それ以降のすべての年のトークンが 5.8 カーネルに残っています。表 1 は、現在カーネルに残っているトークンが挿入された年の内訳を示しています。

表 1 が示すように、5.8 カーネルのコードの半分以上は過去 7 年間に作成されたものですが、過去すべての年の痕跡が存在し、現在もバージョン 5.8 のコードに貢献しています。



カーネル v5.8 コードベースへの年度別コード貢献

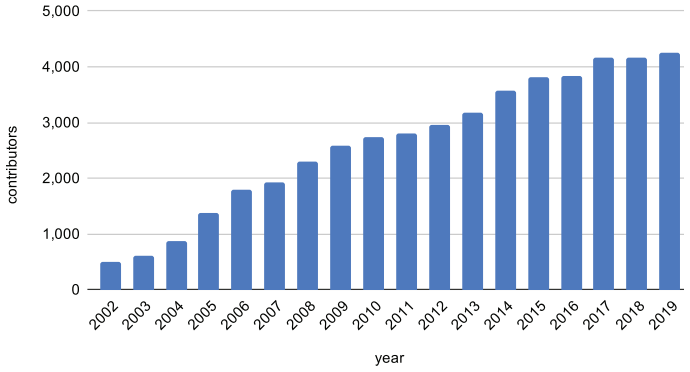
年	トークン数	比率	累積
1991	2,964	0.00%	0.00%
1992	30,740	0.03%	0.03%
1993	46,910	0.04%	0.07%
1994	38,443	0.03%	0.11%
1995	95,498	0.09%	0.19%
1996	204,573	0.19%	0.38%
1997	370,193	0.34%	0.72%
1998	436,941	0.40%	1.11%
1999	775,706	0.70%	1.81%
2000	1,469,450	1.33%	3.15%
2001	828,530	0.75%	3.90%
2002	2,475,002	2.24%	6.14%
2003	1,310,598	1.19%	7.33%
2004	2,023,705	1.83%	9.16%
2005	2,575,195	2.33%	11.49%
2006	2,449,966	2.22%	13.71%
2007	3,118,829	2.83%	16.54%
2008	4,410,921	4.00%	20.54%
2009	5,461,240	4.95%	25.49%
2010	4,805,525	4.35%	29.84%
2011	5,859,906	5.31%	35.15%
2012	6,072,494	5.50%	40.65%
2013	6,440,436	5.84%	46.49%
2014	6,091,508	5.52%	52.01%
2015	8,968,298	8.13%	60.14%
2016	7,622,786	6.91%	67.04%
2017	11,028,463	9.99%	77.04%
2018	9,248,269	8.38%	85.42%
2019	10,843,056	9.83%	95.24%
2020*	5,248,662	4.76%	100.00%

表 1. Linux v5.8 のトークンが作成された年（ 2020 年 8 月 2 日現在）

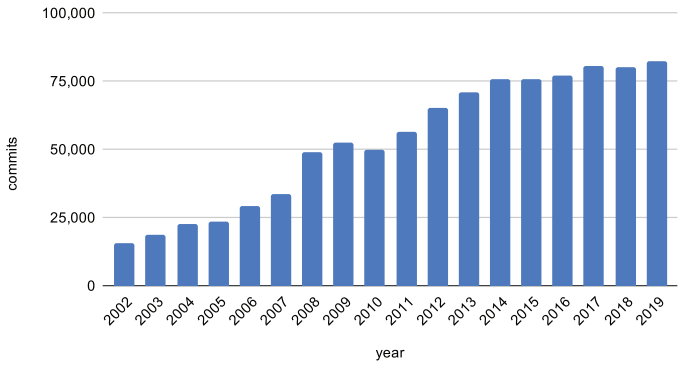
開発プロセスのベストプラクティスの影響

2015 年、The Linux Foundation の Core Infrastructure Initiative (CII) プログラムは、オープンソース プロジェクトの「ソフトウェア開発」と「セキュリティ実践」のレベルを公式に認定するためのベストプラクティス バッジを導入しました [10]。Linux カーネルは、バッジを取得した最初のプロジェクトの 1 つでした。時が経つにつれ、追加のプラクティスも特定され、それらは、プロジェクトが目指すより高度なバッジ レベルに組み込まれてきました [11]。今年の 6 月、Linux カーネルは、少数のプロジェクトのみが持つゴールド バッジ (バッジの最高レベル) [12] のメンバーになりました。これは、長年継続して存在していたものが視覚的に認知されたものです。Linux カーネル コミュニティは、大規模な「ソフトウェア エンジニアリング」と「セキュリティの高い開発」の分野におけるベストプラクティスの確立をリードし続けています。

カーネルのメンテナーは、毎年 (今年を除いて) Maintainer Summit で一堂に会し、共に協力して、作業する方法をいかに改善していくかについて話し合っています。サミットのトピックは、開発者が `ksummit-discuss` メーリングリスト [13] に提起することで決定されます。健全な議論の後、解決されていないものについては、直接会って話し合う方法が選択されます。これらのミーティングは、コミュニティのプロセスの継続的な改善に不可欠です。カーネルへのコントリビューター数とコミットメント数は毎年着実に増加しています。これらのプロセス改善会議の影響を定量化することは容易ではありませんが、それらは良い効果を生み出す要因になっていると考えられます。



Linuxカーネルのコントリビューター数 (年度別)



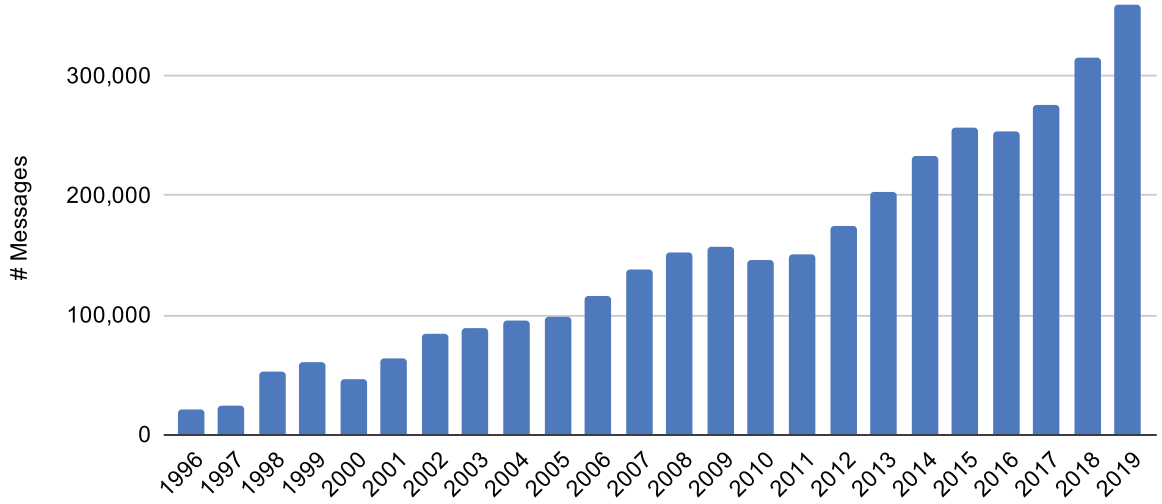
Linuxカーネルへのコミット数 (年度別)

メンテナー階層の採用

カーネルが時間とともに拡大するのを助けてきたベストプラクティスの1つは、メンテナー階層の採用です。カーネル リリース アーカイブを振り返ると、MAINTAINERSファイルの最初のものは、v1.3.68の一部として1996年1月にコミットされたものです[14]。ファイルの長さはわずか107行で、3人のメンテナー（Alan Cox、Jon Naylor、Linus Torvalds）がリストされていました。そのバージョンのMAINTAINERSファイルの最後は、以下のようになっています。

```
REST:
P:     Linus Torvalds
S:     Buried alive in email
```

このファイルの形成につながった議論を理解するために、初期のLinux開発メーリングリストが調査されました。残念ながら、Linuxの開発は複数のメーリングリストとUSENETグループのメールで行われていたため、1997年以前に公開された議論の記録は一部のみです。linux-kernelメーリングリストがvger.kernel.orgでホストされる前のものは、vger.rutgers.eduに存在します。これは、議論が行われた数あるリストの1つにすぎず、重要な議論はまだUSENETで行われていました。1つのアーカイブ ソースは http://lkml.iu.edu/hypermail/linux/kernel/ で、1995年にさかのぼるlinux-kernelメーリングリスト（「LKML」）のアーカイブが含まれていますが、いくつかの主要な議論が抜けています。本レポートを読んでいる方で、初期のLinuxディスカッション スレッドにアクセスできる方がいれば、lore.kernel.orgの管理者[15] に連絡してください。それにより、アーカイブ[16]に初期の開発ディスカッションを追加できるようになります。



年度別のLKML上のメール数

それとは対照的に、今日の v5.8 MAINTAINERS ファイルは19,033 行になり、1,501 人[17] のメンテナーがリストアップされています[18]。v5.8 MAINTAINERS ファイルの最後は以下のようになっています。

```
THE REST
M: Linus Torvalds <torvalds@linux-foundation.org>
L: linux-kernel@vger.kernel.org
S: Buried alive in reporters
Q: http://patchwork.kernel.org/project/LKML/list/
T: Git Git://Git.kernel.org/pub/scm/linux/kernel/
Git/torvalds/linux.Git
F: *
F: */
```

メンテナーの数が増加するにつれて、LMKL 上のメッセージの数も増えましたが、少なくとも Linus はそれを問題として言及しなくなりました。

年	バージョン管理システム	コミット数	コントリビューター数
2002	BitKeeper	15,474	497
2003	BitKeeper	18,609	609
2004	BitKeeper	22,520	882
2005	BitKeeper/Git	23,553	1,372
2006	Git	29,256	1,788
2007	Git	33,761	1,928
2008	Git	48,851	2,304
2009	Git	52,576	2,593
2010	Git	49,809	2,734
2011	Git	56,405	2,806
2012	Git	65,432	2,950
2013	Git	70,966	3,167
2014	Git	75,650	3,580
2015	Git	75,827	3,817
2016	Git	77,041	3,840
2017	Git	80,826	4,175
2018	Git	80,097	4,168
2019	Git	82,371	4,249

表2：コミット数とコントリビューター数の推移

バージョン管理システム

バージョン管理システムの選択は、コミュニティの共同作業能力に大きな影響を与えてきました。これは2000年代初頭、多くの人が不満を持っていたトピックであり、カーネルの開発からの離脱を生み出し、Linus TorvaldsによるGitの作成に拍車をかけました[19]。2005年4月10日にGitの開発が開始されてから15年以上が経過していますが、Gitはカーネル コミュニティだけでなく、他の多くのプロジェクトにも非常に効果的なものであることが証明されています。

BitKeeperを使用する前は、誰が貢献しているのか、貢献数はどれくらいか、貢献がどのような経路で流れていったのかなどの透明性に欠けていました。このレポートでは、コミット履歴は2002年に開始されたと見なしています。それ以前の状況については、Linux Kernel Historic Tree[20]とDave Jonesのhistory.Git[21]に保存されているリリース履歴からリリース情報を確認できます。CREDITS v1.0 ファイルには、初期のコントリビューターのリストが記載されており、その一部には、今日でもよく知られている名前が登場します[22]。

BitKeeperへの移行後、目に見えるかたちで、コミット数とコントリビューター数が毎年大幅に増加しています。その後、バージョン管理をGitに移行した後も、着実にその数は増加しています。

バージョン管理システムがBitKeeperからGitに変更されてからも、コントリビューター数とコミット数は毎年着実に増加しています。

```
commit 1da177e4c3f41524e886b7f1b8a0c1fc7321cac2

Author: Linus Torvalds <torvalds@ppc970.osdl.org>

Date: Sat Apr 16 15:20:36 2005 -0700

Linux-2.6.12-rc2

Initial Git repository build. I'm not bothering
with the full history, even though we have
it. We can create a separate "historical" Git
archive of that later if we want to, and in the
meantime it's about 3.2GB when imported into
Git - space that would just make the early Git
days unnecessarily complicated, when we don't
have a lot of good infrastructure for it.

Let it rip!
```

出典：https://Git.kernel.org/pub/scm/linux/kernel/Git/torvalds/linux.Git/commit/?h=v2.6.12-rc2&id=1da177e4c3f41524e886b7f1b8a0c1fc7321cac2

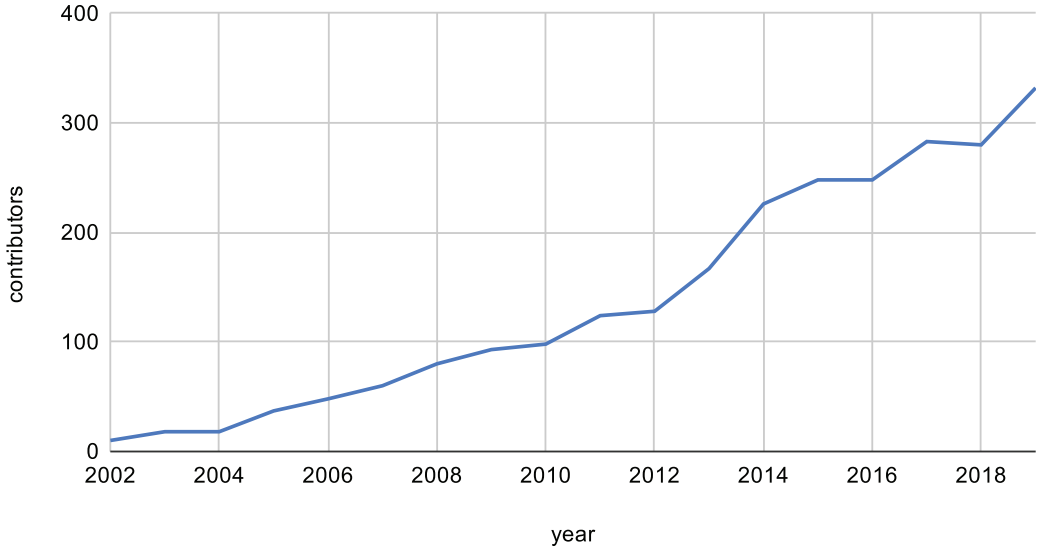
カーネルレポートが 2008 年[3] に始めて発行され、そのレポートの図 3 は、2005 年から 2008 年までのリリースの 1 時間あたりの変化を示していました。このレポートによれば、15 年前の 2005 年 5 月の 2.6.12 リリースでは、1 時間あたり平均 2 つのコミットがあったことがわかります。15 年後の 2019 年、昨年のカーネルでは 1 時間あたりの平均コミット数が 9.4 です。最新の 5.8 カーネルでは、1 時間あたり平均 10.7 コミットでした。

タスクに適したツールを持つことで、違いが生まれました。

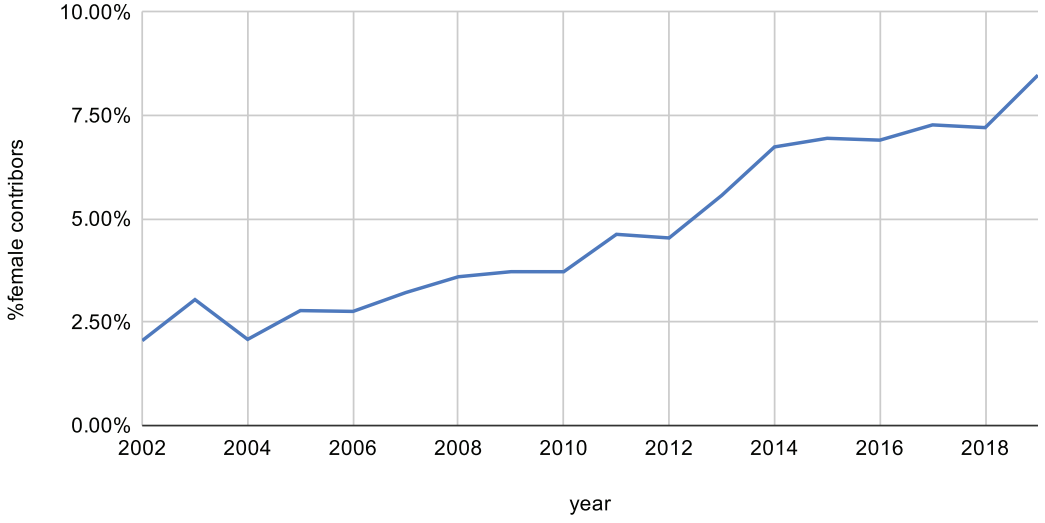
Linux カーネルへのコントリビューターの性別を特定することは必ずしも可能ではありませんが、近年、コントリビューターの多様性が多少改善されていることを示唆するデータがあります。

2019 年末には、コントリビューターの 8.5%が女性であると推定できることがわかりました。また、全体的のコントリビューターの中で、女性のコントリビュータの比率は時とともに増加しています。

コントリビューターの性別を特定するために、Eindhoven 工科大学と Carnegie Mellon 大学の研究者が開発した GenderComputer[3] を使用しました。しかし、カーネルへのコントリビューター全体を見ると、参加者の性別の多様性を向上させる余地がまだ残っていることは明らかです。



Linuxカーネルへの年度別女性コントリビューター数推定



女性と推定されるLinuxカーネル コントリビューターの比率

Linux カーネル コミュニティのメンバーは、コントリビューターの多様性を向上させる方法を模索し続けており、このトピックスは過去の Kernel Summits[24] でも取り上げられています。カーネル開発者は、Outreachy[25] や Linux Foundation の Kernel Mentorship（LKMP）プログラム [26] のようなプログラムに参加している新しい開発者を指導するために、自分の時間をボランティアで提供しています。LKMP は ommunityBridge[27] のメンターシップ プログラムの一部であり、オープンソースの初心者がオープンソース コミュニティを体験し、学習し、貢献するようになるための支援をしています。結果として、オープンソース コミュニティの専門家やかれらの雇用者とつながりができることにも役立っています。このプログラムは、プロジェクトに資金を提供し、コードを保護し、さまざまなバックグラウンドを持つ有能な開発者とつながりを作り、私たちが気にかけ、依存しているオープンソース ソフトウェアの健全なエコシステムを構築することを目的としています。

また The Linux Foundation は、新しい女性のコントリビューターに対し、新しいコントリビューターが利用できる資料やガイダンスを改善するための提案を求めています。その過程で得たいくつかの重要なフィードバックを以下に示します。

- オンライン リソースが不十分だったり古かったりするため、更新が必要です。
- サブシステム パッチ送信ガイドラインに関するFAQがもっと多ければ、より役立つでしょう。
- Linuxカーネルに貢献するためのベストプラクティスに関するFAQやブログがもっと多ければ、より役立つでしょう。
- パッチの受け入れ状況に関する情報交換の方法は改善の余地があります。
- Shuah KhanのLFD103[28]やGreg Kroah-Hartmanの “Write and Submit your first Linux kernel Patch (はじめてのLinuxカーネル パッチの作成と提出)”[29]のような初心者向けのコースやビデオがもっとあれば、リソースの制約が多い学生に役立つでしょう。

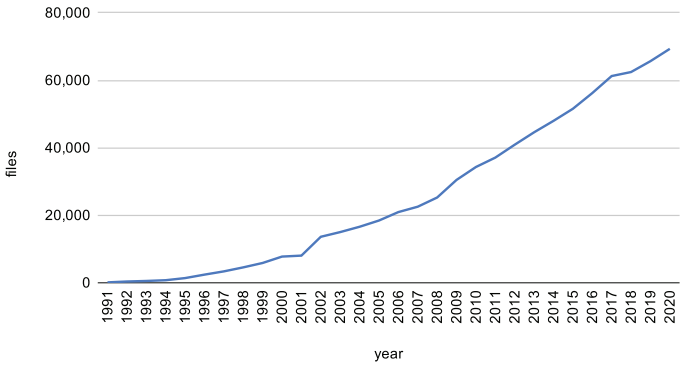
未使用のコードの削除

予想されるように、コミット数は時間とともに増加しているため、カーネルはファイル数とコード行数も増加しています。ただし、各カーネルのビルドでは必要な一部のソースコードのみが組み込まれます。 Greg Kroah-Hartmann が彼のブログで述べているように、「平均的なラップトップ環境は、正常に機能させるために 5,000 ファイルの約 200 万行のカーネル コードを使用しますが、Pixel スマートフォンは SoC の複雑さが増しているため、6,000 ファイルの 320 万行のカーネル コードを使用します。」[30]

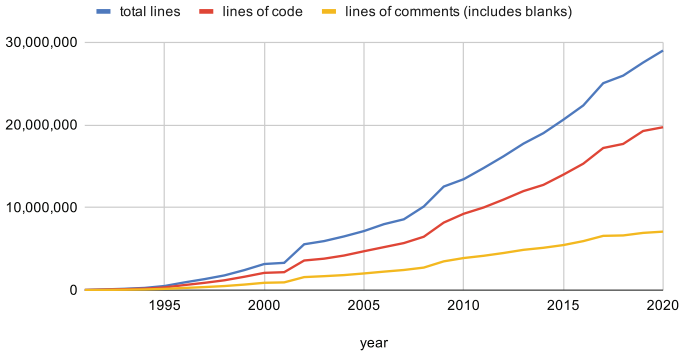
未使用のコードやファイルを削除することには、引き続き重点的に取り組まれていますが、機能とインフラストラクチャが追加されるにつれて、ファイル数は全体では増え続けています。カーネル内のファイルを時間の流れに沿って見てみると、2018 年の v4.17 のように、8 つのアーキテクチャ、18 万行が削除されて、大規模な「剪定」が行われたリリースもあり[31]、左下のグラフの線にも表れています。

ファイルと同様に、カーネルを構成するコード行も増え続けています。増えているのはソースコードだけではありません。コメントは理解するために重要であり、空白行も読みやすさを保つために重要です。次のグラフは、cloc[32] によって計算されたコード行数、コメント行数、および空白行数を示しています。

以前 Daniel German が行った研究によると、コードの各行は平均約 7 トークンで構成されており、時間が経過しても、コードの行数とトークンの総数は密接な相関関係を維持しています[33]。



Linuxカーネル内のファイル数



コードとコメントの行数

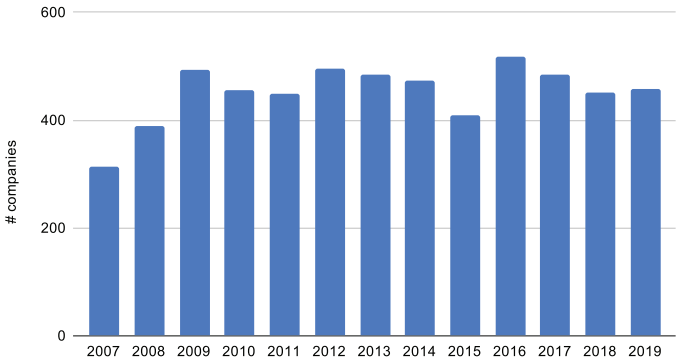
多様性に富んだ企業コントリビューター

Linux カーネルには、長年にわたってさまざまな組織が貢献してきました。Git でコミット データを見ると、カーネルに貢献している組織は、非常に長いロングテールの傾向を見せています。2007 年から 2019 年にかけて、1,730 の組織からの 780,048 件のコミットが Linux カーネルに受け入れられました。リストに示した上位 20 の組織がコミットの 68%を占めていますが、単一のコミットしか行わなかった企業のロングテールも見ることができます。コミット数上位 20 の組織は、左の表のとおりです。

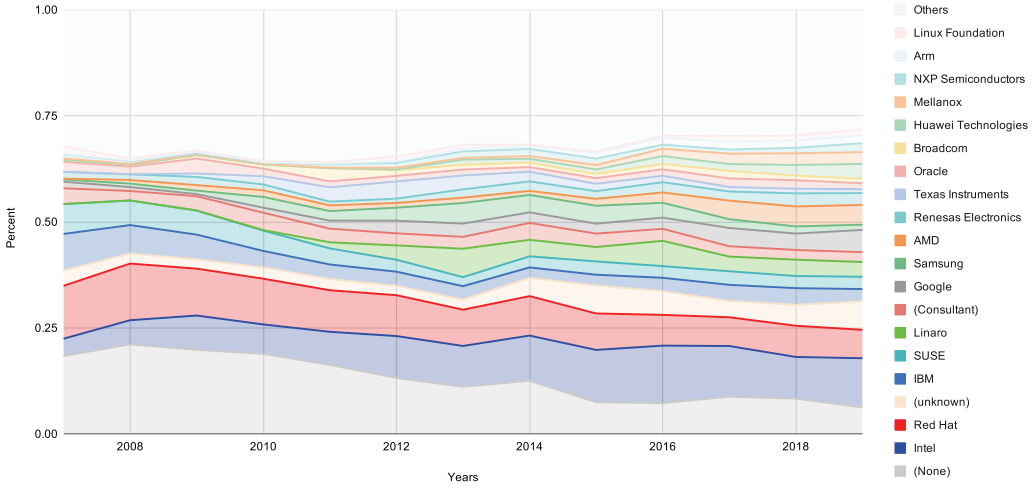
これらの表やグラフで「不明 (Unknown)」のラベルで示しているものは、サポートしている雇用主を識別できなかったパッチを示しています。「なし (None)」が示されている場合は、ボランティアで、自身の時間に取り組んでいることがわかっている開発者からのパッチを示しています。この慣例は、これらのテーブルを生成するために使用された Gitdm ツール [34] で使用され、2.6.20 リリースの統計で初めて文書化されました [35]。

企業	コミット件数 2007〜2019年	%
None	93,225	11.95%
Intel	78,068	10.01%
Red Hat	69,443	8.90%
(Unknown)	31,919	4.09%
IBM	29,538	3.79%
SUSE	27,239	3.49%
Linaro	24,740	3.17%
(Consultant)	23,081	2.96%
Google	21,779	2.79%
Samsung	20,160	2.58%
AMD	17,781	2.28%
Renesas Electronics	15,542	1.99%
Texas Instruments	13,855	1.78%
Oracle	13,295	1.70%
Broadcom	9,572	1.23%
Huawei Technologies	9,379	1.20%
Mellanox	9,267	1.19%
NXP Semiconductors	9,223	1.18%
Arm	7,646	0.98%
Linux Foundation	6,109	0.78%

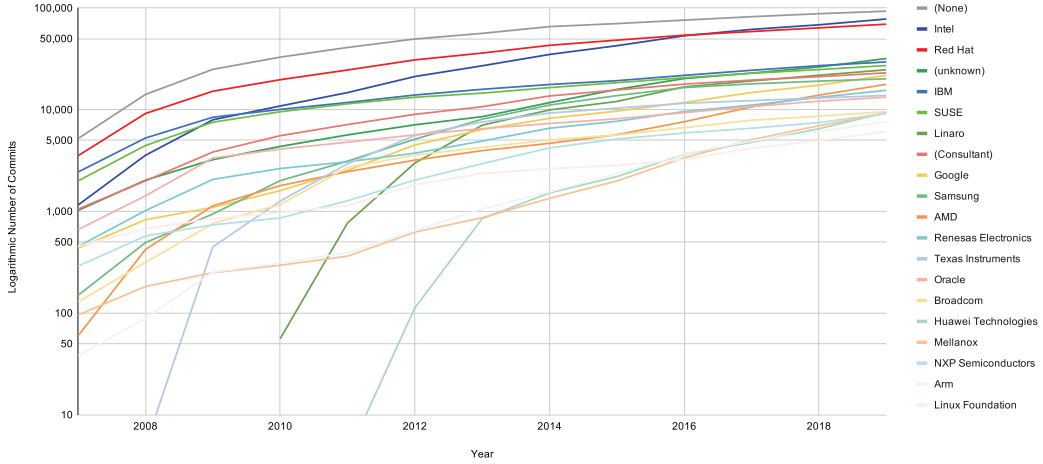
表3：上位20コミッター



Linuxにコミット貢献している企業数（年度別）



企業によるLinuxコミット数の比率（年度別）



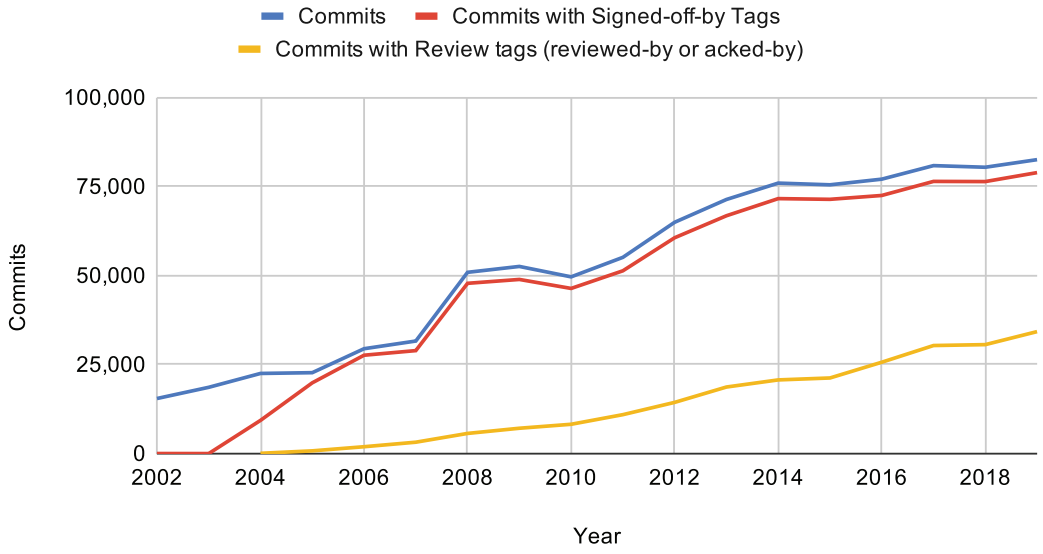
企業によるLinuxカーネル コミット数（累積）

過去 10 年間、毎年 400 以上の組織が Linux カーネルに貢献してきました。 これらのうちのかなりの数が、1 つのコミットしかサブミットしていないようです。各年のコミット企業の比率を見ると、ロングテール部分が、全コミットの約 3 分の 1 を占めています（左のグラフでは「Others（その他）」とラベルが付けられている部分）。

企業からの貢献は、ビジネスニーズや戦略に応じて、時間の経過とともに変化しています。たとえば上位 20 の貢献企業のいくつかは、2007 年にはカーネルにまったく貢献していませんでした。また、2007 年には強力な貢献企業であったものの、買収や合併などで参加しなくなった企業もあります。コントリビューターの多様性は強みであり、プロジェクトを活性化し続けています。

開発者証明書（Developer Certificate of Origin）のタグ付けと採用

Linux カーネルの歴史におけるその他の重要なプロセスの変更の 1 つに、2004 年に開発者証明書（DCO）[36] が標準化されたことがあげられます [37]。DCO は、開発プロセスに大きな負担をかけることなく、開発者とユーザーにさらなる法的保護を提供するために追加されました。DCO により、パッチがカーネルに受け入れられるまでにたどった経路を追跡する機能が大幅に改善されました。バージョン管理システムの Git への移行とともに、過去 15 年間にわたって、DCO は開発者の貢献負担を軽減してきました。開発者の間で人気が高いことが証明され、他のオープンソース プロジェクトにも採用されています。

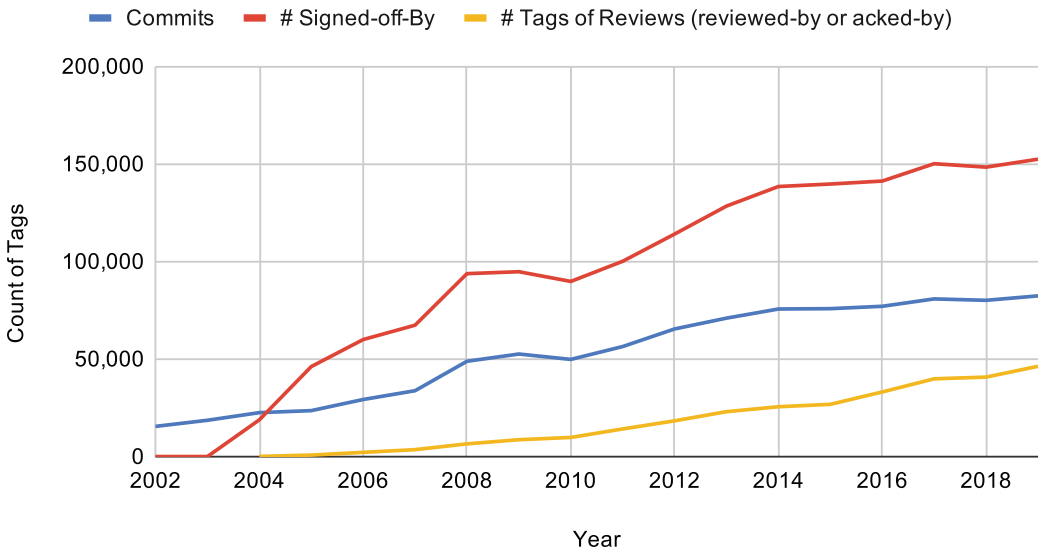


タグ付きのLinuxカーネル コミット数（年度別）

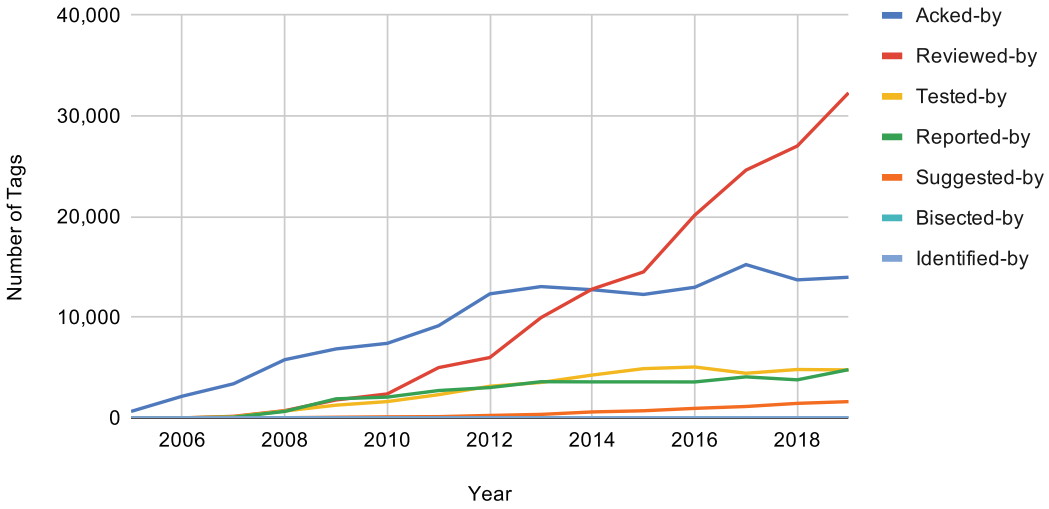
カーネル コミュニティにより、2004年にDCOの使用が標準化され、ほぼすべてのコミットに`Signed-off-by`タグが付けられるようになりました。各コミットに対する目に見えるレビューも着実に増えています。レビューを示すタグ（`Reviewed-by`および`Acked-by`）を追加することで、その領域に関心があり、コメントしたり、懸念事項を表明したりする機会があった人を追跡するのに役立ちます。パッチは、関連するコードのメンテナーが変更を受け入れるまで、数サイクルのレビューが必要な場合もあります。変更がコミットされる前のこのレベルの精査が、カーネルの品質確保に重要です。

`Signed-off-by`タグにより、パッチがカーネルに受け入れられるまでの経路に対する理解が向上しました。平均して、通常、各コミットには2つの`Signed-off-by`タグが関連付けられ、コードがマージされる前のメンテナーの階層を反映します。レビューが行われたことを示すためのタグ（「レビュー（`Reviewed-by`）」または「承認（`Acked-by`）」のいずれか）の使用も、この機能が2007年に導入されて以来、着実に増加しています。

ただし、導入された他のタグは、当初の期待ほど広く採用されたり使用されたりしていません。タグにより、開発者の創造性を観察することができます。Gitリポジトリを見てみると、より興味深いものを見つけることができます。たとえば、"`Enthusiastically-ack'd-by`"、"`Thanks-to`"、"`Based-on-the-original-screenplay-by`"、"`Catched-by-and-rightfully-ranted-at-by`"などがあります。今後は、"`Tested-by`"の使用が増えれば素晴らしいですね。



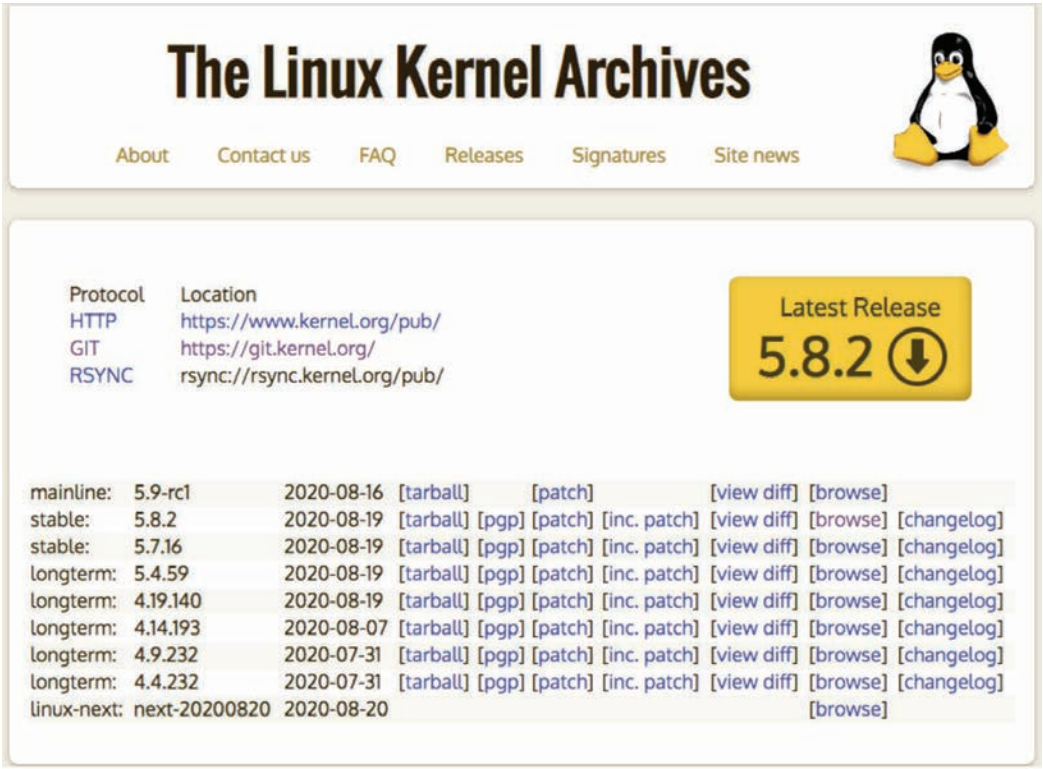
Linuxカーネルのタグの種類（年度別）



Linuxカーネル内のタグ数（signed-off-byタグを除く：年度別）

リリース サイクルが予測可能なリリース モデル

Linuxカーネルのリリース モデルは進化し、リリースは4カテゴリに分類されています。プレパッチ(Prepatch)（または「-rc」）、カーネル（Kernel）、メインライン(Mainline)、安定版(Stable)、および長期安定版(Long Term Stable)です[38]。最新のカーネル リリースの詳細情報は、https://www.kernel.org/で公開されています。

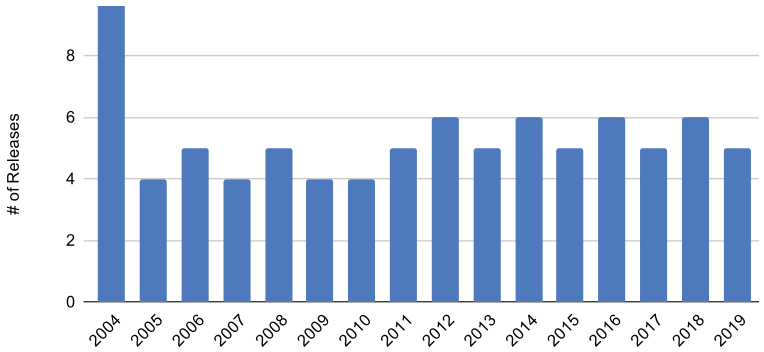


左：出典：https://www.kernel.org/ on 2020-08-20

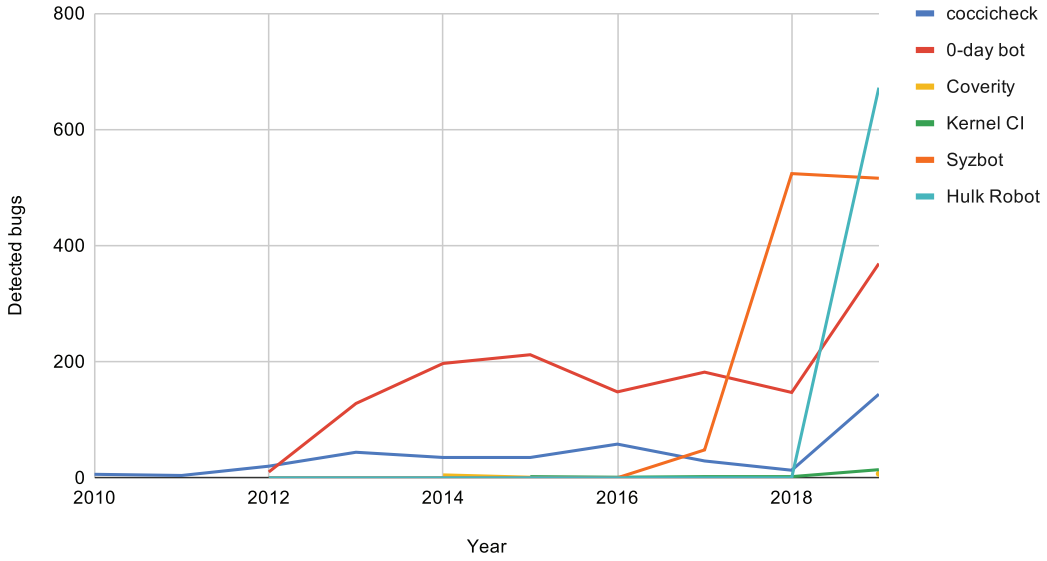
リリース サイクル頻度については、コミュニティで、多くの議論と実験が行われてきました。2011 年以降、コミュニティは、彼らが求める品質をサポートし、機能するリリース モデルを見つけたようです。実際、リリースの時期は精度高く予測可能になり、正確な予測ツール[39]も開発されたため、開発者は企業のマネージャーの「いつリリースされるのか」という質問にもすばやく答えられるようになりました。。

適切なレビューがなされた新機能が次のリリース向け Git リポジトリに格納されると、各リリース サイクルは 2 週間の「マージ ウィンドウ」をスタートさせます。rc1 のタグが付けられると、統合テスト、デバッグ、および安定化のサイクルが始まります。週ごとの rc の候補は、目標の品質と安定性に達するまでタグ付けされます。それがリリースされると、次のマージ ウィンドウでそのサイクルが再び開始されます。

カーネル リリース モデルとその実現方法の詳細については、コミュニティがこのモデルを考案した経緯を紹介している Greg Kroah-Hartmann のブログをご覧ください[40]。



Linux リリース回数（年度別）



ボットが検出したLinuxカーネルのバグ数

カーネルの自動テストの改善

Linux Kernelのテストは、コミュニティによる取り組みの1つです。`sparse` (semantic parser)[41], `smatch` (source matcher)[42] 、および `coccicheck` (特定のバグをテストするセマンティック パッチ) [43] のような静的解析ツールは、Linux カーネル ツリー上で、自動テスト用ボット(autotest bots)[44] の `0-day`[45]や `Hulk Robot`[46] などによって実行されます。さらに、 `Trinity` (Linuxシステムコール ファザー) [47]などのファズ テスターや他のテストが実行され、問題を積極的に検出します。`syzbot`などのファズ テスト[48] ツールは、さまざまなカーネル ツリー上で実行されます。これらのボット（bot）やツールは、どのようにして問題を発見しているのでしょうか。左のグラフは、さまざまなボットとツールによって検出されたカーネル バグの修正数を示しています。

Linuxカーネル5.4開発リリースの統計は、ボットがバグを見つけ、それに対する修正を提供できることの重要性を明快に示しています[49]。ボットによる“`Reported-by`”タグは、各リリース サイクルにおけるリグレッション防止にテストの自動化が果たしている役割の重要性について、認識を高めさせています。

安定版リリース プロセス

最新の安定版リリースは、約 1 週間に 1 回リリースされます。Linux 5.7.9 を例に、リリース プロセスについて説明します。リリース候補（release candidate: rc）は、安定版リリース メーリング リスト（stable release mailing list）[50] へのメールで公開されます。このメールには、疑似的なコミットの短いログと、レビューやテスト向けのリリース候補のダウンロード元情報が添えられます [51]。このリリース候補には 166 個のパッチがあり、これらのパッチはすべて別のメールでメーリングリストに送信されます。個々の開発者と自動テスト リング（Kernel CI や 0-day など）がリリース候補をテストし、リリース候補のメールに返信する形で結果を報告します。rc1 で見つかった問題は、後続の rc2、rc3 候補で修正されます。リリース候補のテストで問題が発生せず、rc2 がリリースされることはまれです。では、誰がテストを実施しているのでしょうか。

Buildbot (for Linux kernel hwmon and stable-queue)[52] が、サポートされている 30 以上のすべてのアーキテクチャ（arm64 と arm のような同一アーキテクチャの改版も含む）でリリース候補を複数のカーネル構成用にビルドし、その結果を報告します [53]。安定版リリースのレビュー サイクルでは、31 のアーキテクチャと 56 の構成 (configs) のすべてで、ビルド テストが行われます。LKFT（Linux カーネル機能テスト）リングは、arm および arm64 ハードウェアでリリース候補をビルドおよびテストし、結果を報告します [54]。実行されるテストは、Linux Test Project (LTP)、Linux Kernel Selftests、Linux Perf などです。このリングで実行されるテストの完全なリストについては、LKFT Test ドキュメント [55] で公開されています。Kernel CI[56] リングは、複数のシステムでブートテストを実行し、その結果を報告します。

リリース候補が十分にテストされると正式にリリースされ、リリースを通知するための別メールが送信されます。

長期リリース カーネル

安定版リリースをベースにした長期リリース カーネル（Longterm release kernel）の提供により、製品メーカーによる Linux の人気がさらに高まり、新しい市場セグメントでの Linux の採用が増えています。Greg Kroah-Hartmann は、どのカーネルを長期リリース カーネルに選択するかについて、2018 年に優れたブログを書いています[57]。プロジェクトにどのカーネルを選択すべきか調べている開発者に役立つ情報です。サポートされている長期安定版リリース（long term stable release）と、サポートが終了するまでの期間（予測 EOL）の詳細については、kernel.org で確認できます。

長期リリース カーネルは、長期間保守することが約束された安定版カーネルとしてスタートします。SUSE、Ubuntu、Red Hat などのディストリビューションがこの概念の開拓を支援し、2010 年[58]、「長期」カーネルの正式化と stable@kernel.org メーリングリストの採用により、正式に提供されるようになりました。

既存の Linux 安定版カーネルで見つかったバグは、アップストリームで修正され、安定版カーネルに反映されます。修正が長期リリースカーネルの 1 つに適用可能であると判断された場合、それはバックポートされ、適用されます。

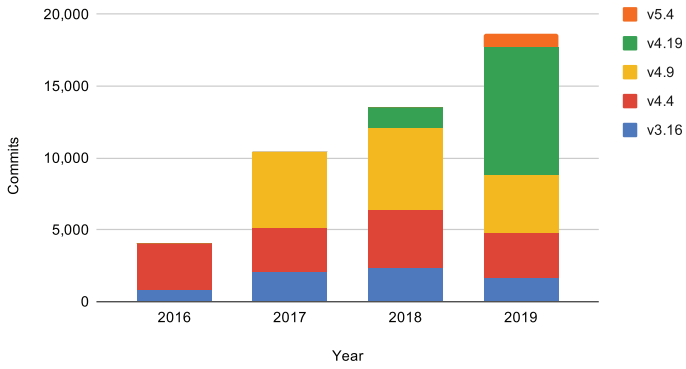
修正が長期安定版カーネルにいつ適用できるかを検出するためのツールの機能向上は（有用な修正のように見えるパッチを特定するために機械学習技術を使用するツールを作成した Sasha Levin の作業のように）、近年大幅に進んでおり、長期リリースのカーネルに適用できるバグ修正が増加しています。

2019 年には、さまざまな LTS カーネルに対して 18,668 件のコミットが行われました。1 時間あたりのコミット数は 2.15 以上です。**2019 年の LTS リリースには、15 年前の 2.6.12 カーネルよりも多くの変更が加えられました。**

Linux を使用しているアプリケーションをサポートするために、さらに長いサポート期間を求める市場の要請があります。たとえば、Civil Infrastructure Platform（CIP）プロジェクト[60] のメンバーは、長期間保守が必要なインフラストラクチャを持つことに共通の関心を持っています。このプロジェクトに参加している開発者は、4.4 と 4.19 を超長期安定版（Super Long-term Stable: SLTS）カーネル リリースとしてサポートすることを決定しました[61, 62]。これらの開発者が、今後何年にもわたってこのビジョンを実現するためにどのようなツールやプロセスを作成するのかを見るのは興味深いことです。

バージョン	メンテナー	リリース日	予測EOL
5.4	Greg Kroah-Hartman & Sasha Levin	2019-11-24	December 2025
4.19	Greg Kroah-Hartman & Sasha Levin	2018-10-22	December 2024
4.14	Greg Kroah-Hartman & Sasha Levin	2017-11-12	January 2024
4.9	Greg Kroah-Hartman & Sasha Levin	2016-12-11	January 2023
4.4	Greg Kroah-Hartman & Sasha Levin	2016-01-10	February 2022

出典：https://www.kernel.org/category/releases.html



ツールが長期Linuxカーネルのより多くの修正を検出

まとめ

2020 年に、Linux カーネルはゴールド CII ベストプラクティス バッジ [63] を獲得しました。これは、プロジェクトが品質の向上、セキュリティの向上、および欠陥の防止に、多くのプラクティスを適用していることを立証しています。Linux カーネルは、医療機器 [64] から自動運転車 [65]、宇宙船 [66] まで、セキュリティと安全性を最重視する製品で使用されています。これらの安全性重視のアプリケーションで実際に Linux を使用する前に、適切なレベルのセキュリティ & 安全性分析を自信を持って行えるようにインフラストラクチャを改善していくことは、今後、取り組むべき大きな課題の 1 つです [67]。

カーネル コミュニティが、リグレッションのない高品質なオペレーティング システム実現という共通の目標を持ち、必要に応じて新しいプロセスやツールを作成してさらなる効率化をはかろうとすることで、新たな市場に導入される Linux カーネルの信頼性が継続的に高まります。カーネルのテスト基盤で起きているツールの進化により、開発者はアップストリーム カーネルの速度変化に対応でき、また、その後のプロセスを見通すことができるため、安定化カーネルや将来のリリースの改善を続けることができます。カーネル開発者は、改善のために、疑問を投げかけたり、議論したり、多様な視点を歓迎したりする意思があることを示してきました。現在、Linux カーネルには、オープンソース ソフトウェア業界全体を支援するベストプラクティスの作成方法をリードし続ける優れた基盤が存在します。

謝辞

著者は、何万人ものカーネル貢献者に感謝したいと思います。彼らがいなければ、このような論文は、誰にとっても興味深いものではなかったでしょう。

また、Linuxカーネル開発の履歴を保存するための基盤構築に時間と労力を費やし、自分たちのベストプラクティスから他の人が学ぶことができるようにしてくれた開発者の方々にも感謝します。

参考文献

[1] https://lore.kernel.org/lkml/CAHk-=wj+mDPbj8hXspXRAksh+1TmPjubc9RNEbu8EVpYyypX=w@mail.gmail.com/

[2] https://lwn.net/Articles/827735/

[3] https://www.linuxfoundation.org/publications/2008/04/linux-kernel-development-report-2008/

[4] https://creGit.linuxsources.org/

[5] https://Github.com/creGit/creGit

[6] https://Github.com/dmgerman/linux-pre-history/commit/9417d4148f0ddc5ee2cc1114ce97c71c5e4cb4b7

[7] https://archive.org/details/Git-history-of-linux

[8] https://repo.or.cz/davej-history.Git/commit/bb441db1a90a1801ef4e6546417a8d907c55d92f

[9] アーキテクチャ数は、以下のページのブート テストの対象をカウント https://Github.com/groeck/linux-build-test/tree/master/rootfs

[10] https://www.coreinfrastructure.org/announcements/linux-foundations-core-infrastructure-initiative-seeks-community-input-on-new-security-focused-badge-program/

[11] https://www.linuxfoundation.org/blog/2020/06/why-cii-best-practices-gold-badges-are-important/

[12] https://www.linuxfoundation.org/blog/2020/06/linux-kernel-earns-cii-best-practices-gold-badge/

[13] https://lore.kernel.org/ksummit-discuss/

[14] https://repo.or.cz/davej-history.Git/commit/8bf26ec84e6362618e1abe641ac7f026c2674372

[15] https://www.kernel.org/lore.html

[16] https://korg.docs.kernel.org/lore.html

[17] `grep “M:” MAINTAINERS | sort | uniq | wc -l`

[18] https://Git.kernel.org/pub/scm/linux/kernel/Git/stable/linux.Git/tree/MAINTAINERS?h=v5.8.2

[19] https://www.linuxfoundation.org/blog/2015/04/10-years-of-Git-an-interview-with-Git-creator-linus-torvalds/

[20] https://Git.kernel.org/pub/scm/linux/kernel/Git/history/history.Git/

[21] https://repo.or.cz/davej-history.Git

[22] https://Git.kernel.org/pub/scm/linux/kernel/Git/history/history.Git//?h=1.0&id=13f97bf0ff18e367f94a5ebcf6e89998ecedb80f

[23] https://Github.com/tue-mdse/genderComputer

[24] https://lwn.net/Articles/662911/

[25] https://www.outreachy.org/

[26] https://wiki.linuxfoundation.org/lkmp

[27] https://communitybridge.org/

[28] https://training.linuxfoundation.org/training/a-beginners-guide-to-linux-kernel-development-lfd103/

[29] https://www.youtube.com/watch?v=LLBrBBImJt4

[30] http://www.kroah.com/log/blog/2018/02/05/linux-kernel-release-model/

[31] https://lwn.net/Articles/756031/

[32] https://Github.com/AlDanial/cloc

[33] German, D.M., Adams, B. & Stewart, K. creGit: Token-level blame information in Git version control repositories. Empir Software Eng 24, 2725–2763 (2019). https://doi.org/10.1007/s10664-019-09704-x

[34] Git://Git.lwn.net/Gitdm.Git

[35] https://lwn.net/Articles/222773/

[36] https://developercertificate.org/

[37] https://www.wired.com/2004/05/linux-whose-kernel-is-it/

[38] https://www.kernel.org/category/releases.html

[39] http://phb-crystal-ball.org/

[40] http://www.kroah.com/log/blog/2018/02/05/linux-kernel-release-model/

[41] https://sparse.docs.kernel.org/

[42] http://smatch.sourceforge.net/

[43] https://bottest.wiki.kernel.org/coccicheck

[44] https://bottest.wiki.kernel.org/

[45] https://01.org/lkp

[46] mailto:hulkci@huawei.com

[47] https://Github.com/kernelslacker/trinity

[48] https://lwn.net/Articles/536173/

[49] https://lwn.net/Articles/804119/

[50] https://lore.kernel.org/stable/

[51] https://lore.kernel.org/lkml/20200714184115.844176932@linuxfoundation.org/

[52] https://kerneltests.org

[53] https://kerneltests.org/one_line_per_build

[54] https://lkft.linaro.org/

[55] https://lkft.linaro.org/tests/

[56] https://kernelci.org/

[57] http://kroah.com/log/blog/2018/08/24/what-stable-kernel-should-i-use/

[58] https://lwn.net/Articles/418580/

[59] https://lwn.net/Articles/789225/

[60] https://www.cip-project.org/

[61] https://www.cip-project.org/blog/2020/08/17/cip-kernel-team-helping-cip-sustain-industrial-grade-systems

[62] https://lwn.net/Articles/749530/

[63] https://www.linuxfoundation.org/blog/2020/06/linux-kernel-earns-cii-best-practices-gold-badge/

[64] https://www.techdesignforums.com/practice/technique/medical-linux-security/

[65] https://www.automotivelinux.org/about/

[66] https://www.zdnet.com/article/from-earth-to-orbit-with-linux-and-spacex/

[67] https://elisa.tech/

The Linux Foundation は、オープンソースがクローズドな
プラットフォームと競い合う上で必要な統合されたリソースや
サービスを提供し、Linux の普及推進、保護、および
標準化に努めています。

The Linux Foundationやその他のプロジェクトの詳細については、
www.linuxfoundation.org をご覧ください。