

Git:愚蠢的内容跟踪器

约瑟夫·哈利特

2023 年 1 月 10 日



这是怎么回事？

编写源代码很难。
我们需要一种机制来系统地跟踪变化。

幸运的是，
软件工程师早在 70 年代就解决了这个问题。
▶ （并在2000年代完善）

例如

假设Alice编写了以下程序：

```
package uk.ac.bristol.cs.SoftwareTools;
```

```
public class Hello { public static void main(String[]  
args) { if (args.length ==  
    0) {  
  
        args = 新字符串[1]; args[0] =  
            世界 ;  
    }  
    for (最终变量名称: args)  
        System.out.print( 你好  +name+  !\n );  
    }  
}
```

你好世界！

稍后更新

后来她制作了程序的新版本……有什么变化?包 uk.ac.bristol.cs.SoftwareTools; public

```
class Hello2 { public static void main(String[] args)
```

```
{ if (args.length == 0) {
```

```
    args = 新字符串[1]; args[0] =
```

```
        世界 ;
```

```
}
```

```
for (最终变量名称: args)
```

```
    System.out.println( 你好 +name+ ! );
```

```
}
```

```
}
```

你好世界!

一个糟糕的解决方案

我们可以手动跟踪更改.....

► 每个版本的文件都有不同的名称,末尾带有数字

► 编写一套工具来发现文件中的差异... diff uk/ac/bristol/cs/SoftwareTools/Hello*.java 2c2 < public

```
class Hello {    > public class Hello2 { 9c9 < System.out.print( "你好"名字
"! " );> System.out.println( "你好,名字" ! " ); 2c2 <公开课你好{
```

> 公开课 Hello2 { 9c9

```
<          System.out.print( 你好  +name+  !\n );
```

```
>          System.out.println( 你好  +name+  !  );
```

假设 Bob 分叉了代码？

假设鲍勃采用了爱丽丝的原始程序并进行了自己的更改？

包 uk.ac.bristol.cs.SoftwareTools;导入

java.util.*;公共类

Hello2 { 公共静态 void

main(String[] args) {

最终 var people = new LinkedList<String>();

people.addAll(Arrays.asList(args)); if

(people.size() == 0)

people.add(世界);

for (最终变量名称:人)

System.out.print(问候语 +name+ .\n);

}

}

问候世界。

►我们如何将Alice 的更改与 Bob 的更改合并？ ►我们如何应对分歧？

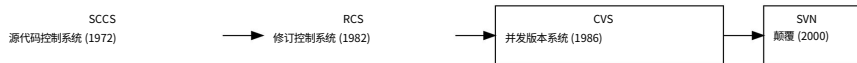
别辛苦了!工作懒!

显然,像这样管理源代码将需要大量的手动工作。
但我们是计算机科学家……我们可以使任何事情自动化。

那么让我们这样做吧!

- 编写软件来为您完成所有软件管理
- 让它跟踪谁更改了什么以及何时更改
- 让程序员介入并修复问题,作为最后的手段

版本控制系统



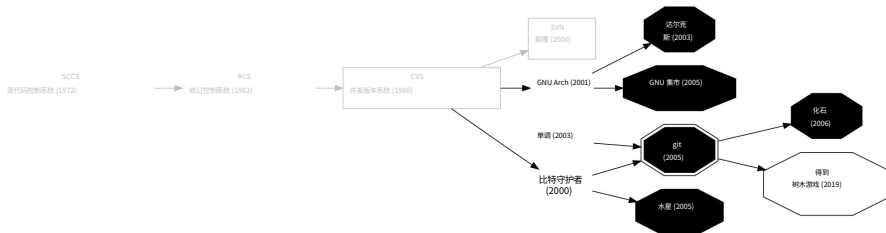
最初所有版本控制系统都是集中式的.....

► 也就是说,他们每个人都有有一个存储最新版本的官方中央存储库。

分散版本控制系统

但在 2000 年左右,我们开始看到从中心化模式向去中心化模式的转变

► 每个用户都有一个源代码管理的主版本 ► 通过合并接受其他人的更改





Linus Torvalds开发 Git 来帮助开发 Linux 内核。

- ▶ 内核是通过将源代码与您想要进行的更改进行比较来开发的▶ 通过电子邮件向负责人发送您想要更改的内核部分以及说明▶ 如果他们进行了更改,他们会将更改通过电子邮件发送给Linus融入他的树

Git 被设计成一个帮助 Linus 完成工作的工具▶ 并不是为了用户友好而设计的▶ 越坏越好

- ▶ 快速处理纯文本文件（源代码）▶ 可以很好地处理大量文件▶ 源代码并不复杂



这仍然是内核的开发方式!

现代 Git

虽然仍然使用基于电子邮件的工作流程……现在有替代方案

- ▶ Git forges提供了基于电子邮件的工作流程的替代方案
 - ▶ 其中最受欢迎的是微软的GitHub
- ▶ 终端命令变得更加有用 ▶ 适合喜欢 GUI 的人
 - ▶ （但也要学习命令行...） ▶ 适合那些喜欢它们的人的编辑器插件
 - ▶ （如果您使用 Emacs,请尝试Magit...）

如有疑问: man 1 git

胃肠道(1)

Git 手册

胃肠道(1)

姓名

git - 愚蠢的内容跟踪器

概要 git [-v |

```
--版本] [-h | --help] [-C <路径>] [-c <名称>=<值>] [--exec-path[=<路径>]] [--html-path] [--man-path] [--info-
path] [-p|--paginate|-P|--no-pager] [--no-replace-objects] [--bare] [--git-dir=<路径>] [--work-tree=<路
径>] [--namespace=<名称>] [--super-prefix=<路径>] [--config-env=<名称>=<envvar>] <命令> [<参
数>]
```

描述 Git 是一个

快速、可扩展、分布式版本控制系统,具有异常丰富的命令集,可提供高级操作和对内部的完全访问。

请参阅 [gittutorial\(7\)](#) 开始使用,然后参阅 [giteveryday\(7\)](#) 了解有用的信息

如果还有疑问……

<https://git-scm.com/book/en/v2> Git官方书籍。免费且写得很好。非常适合了解内部原理

<https://ohshitgit.com>关于如何摆脱 Git 中愚蠢情况的指南。有点脏话了。

好的,让我们开始吧!

要创建Git 存储库,我们可以使用 git init 命令:

```
mkdir 教程 cd 教程  
git init
```

在private/tmp/tutorial.git/中初始化空 Git 存储库

```
ls-a
```

```
。 ..git
```

git 状态

在分支主干上

没有提交,但没有任何要

提交的内容 (创建/复制文件并使用 “git add”进行跟踪)

让我们添加一些代码

```
cat >hello.c <<EOF
```

```
#include <stdio.h>
```

```
int main(void)
```

```
{ printf( 你好,世界\n );返回0;
```

```
}
```

```
EOF
```

```
git add hello.c git
```

```
status
```

在分支 main 上尚未提

交要提交的更改：（使

用 “git rm --cached <file>...”取消暂存)新文件:hello.c

分期

此时,文件 `hello.c` 已暂存,但尚未提交。

当您暂存文件时:

▶ 您说这将是新提交的一部分 ▶ 您正在将更改添加到 Git 的版本控制中 ▶ 但您没有保存任何内容 ▶ 事情仍然可以改变!

当您提交时:

▶ 您迄今为止所上演的所有内容都会作为一次更改写入历史。 ▶ 附上解释说明 ▶ 以及与之相关的您的名字 ▶ 事情不应该改变 ▶
(从技术上讲,他们仍然可以.....但变得更难)

让我们承诺吧！

`git commit -m` 问候程序的初始提交。

向用户打招呼,然后退出。

[main (root-commit) b377fa3] 问候程序的初始提交。 1 个文件已更改,6 次插入(+) 创建模式 100644 hello.c

注意有

时,当您使用新系统时,您会收到设置您的姓名和电子邮件的提示……只需按照提供的说明进行操作即可。所有 Git 提交都需要一个名称和一个电子邮件地址。

```
git config --global user.name Joseph Hallett git config --  
global user.email joseph.hallett@bristol.ac.uk
```

让我们进行一些编辑

```
ed hello.c <<EOS 3c int
```

```
main(int argc, char *argv[]) {
```

```
。
```

```
4c
```

```
    for (int i=0; i<argc; i++) printf( 你好,
```

```
        %s\n  , argv[i]);
```

```
。
```

```
瓦克
```

```
EOS
```

```
83142
```

```
git add hello.c git
```

```
commit -m “向所有通过的人致意。”
```

[main 8f3fafd] 向所有通过的人致意。 1 个文件已更改,3 个插入(+),2 个删除(-)

打个招呼./你好

爱丽丝·鲍勃

```
cc hello.c -o hello 你好, ./hello 你好,爱丽丝 你好,鲍勃
```

又一编辑……

```
ed hello.c <<EOS 4s/0/1/
```

```
wq
```

```
EOS
```

```
git add hello.c git
```

```
commit -m “停止问候程序本身。”
```

142 142 [main 24c96de] 停止问候程序本身。 1 个文件更改、1 个插入
(+), 1 个删除(-)

打个招呼./你好爱

丽丝·鲍勃

make: “你好”是最新的。你好， ./你好你好,爱丽丝你好,鲍勃

那么我们做了什么？

到目前为止,我们已经对代码进行了三处更改:让我们看看它们在 Git 中是什么样子! `git log --oneline | git log --oneline | git log --oneline | git`

`log --oneline | git log --oneline`猫

24c96de 停止问候程序本身。 8f3fafd 向所有经过的人致意。 b377fa3 问候语程序的初始提交。

或者直接使用 gitk

File Edit View Help

● main Stops greeting the program itself.	Joseph Hallett <joseph.hallett@bristol.ac.uk>	2022-11-21 11:04:51
● Greets all the people passed.	Joseph Hallett <joseph.hallett@bristol.ac.uk>	2022-11-21 11:04:51
● Initial commit of the greeting program.	Joseph Hallett <joseph.hallett@bristol.ac.uk>	2022-11-21 11:04:51

SHA1 ID: `c595b87ea597ab03a7d73424f4ed9a53adb71593` ← → Row 1 / 3

Find containing: **Exact** **All fields**

Search

◆ Diff ◆ Old version ◆ New version Lines of context: 3 ☐ Ignore space change **Color words**

Author: Joseph Hallett <joseph.hallett@bristol.ac.uk> 2022-11-21 11:04:51
 Committer: Joseph Hallett <joseph.hallett@bristol.ac.uk> 2022-11-21 11:04:51
 Parent: [31512901735cdfc594c829a384f63eb58c97d9d6](#) (Greets all the people passed.)
 Branch: [main](#)
 Follows:
 Precedes:

Stops greeting the program itself.

----- hello.c -----

index dc9d0cc..9d6f129 100644
 @@ -1,7 +1,7 @@
 #include <stdio.h>

```
int main(int argc, char *argv[]) {
    for (int i=0; i=1; i<argc; i++)
        printf("Hello, %s\n", argv[i]);
    return 0;
}
```

◆ Patch ◆ Tree

Comments
 hello.c

tutorial: All files - gitk

标签、分支和 HEAD...

提交都是通过它们的哈希来标识的... ►但是您可

以使用 `git tag` 命令来命名特定的提交 ►（这对于标记代码的发布或提交版本很有用）

所有提交都对一个分支进行,该分支是一个标签 ►当提交完成时,分支标签会更新以指向顶部的新提交的分支机构。

►默认分支通常称为 `main`（或 `master`） ►（这在所有重要方面都是错误的;但这是一个不错的简化）

还有一个名为 `HEAD` 的特殊标签 ►始终指向您的代码当前所在的位置 ►减去任何未暂存的工作

使用提交

假设您对文件进行了一系列更改,但未提交它们。您想放弃所做的更改: `git checkout HEAD -- hello.c`

或者,如果您更改了很多内容并想要返回干净: `git reset --hard HEAD # 删除所有更改 git clean`

`-dfx # 删除所有未跟踪的文件 HEAD 现在位于 24c96de 停止问候程序`

本身。删除 hello 假设您想返回到上次提交之前的代码状态: `git checkout HEAD~1`

假设您已经查看完旧状态的代码,并且想要返回到主分支上工作:

git checkout 主要

假设一次提交是一个可怕的错误,并且您想反向应用它并撤消它的所有更改: `git revert HEAD [main 3d0eaae] Revert “停止问候程序本身。”`日期: 星期二 Jan 10 11 :33:04

2023 0000 1 个文件已更

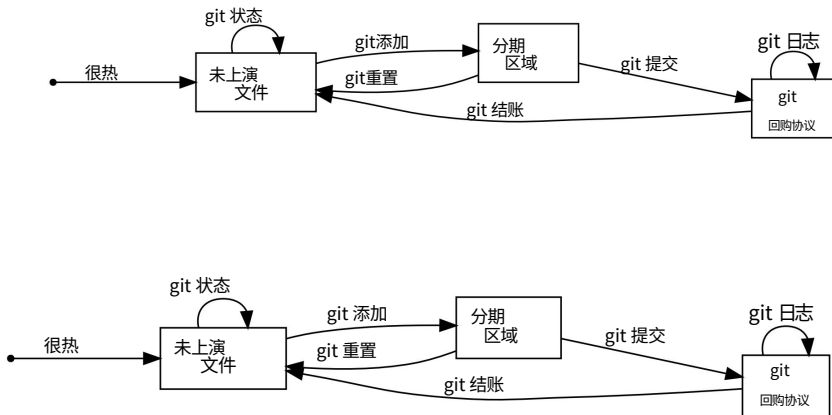
改,1 个插入(+),1 个删除(-)

主要提示1.编

写良好的描述性提交消息（更新不是一个好的消息）

2.永远不要提交损坏的代码（如果它至少不完整,请不要提交）

3.阅读 man git 页面（git 很繁琐！）



还有一件事...

```

2022-11-20 16:02 -0800 Linus Torvalds
2022-11-20 15:31 -0800 Linus Torvalds
2022-11-18 10:15 +0900 Masani Hiramatsu (Google)
2022-11-18 10:15 +0900 Rafael Mendonca
2022-11-18 10:15 +0900 Li Huafei
2022-11-18 10:15 +0900 Yi Yang
2022-11-18 10:15 +0900 Rafael Mendonca
2022-11-18 10:15 +0900 Shang Xiaojing
2022-11-18 10:15 +0900 Shang Xiaojing
2022-11-20 15:25 -0800 Linus Torvalds
2022-11-17 21:42 -0500 Steven Rostedt (Google)
2022-11-14 18:46 +0800 Zheng Yejian
2022-11-18 00:44 +0800 Qiujun Huang
2022-11-17 09:23 +0800 Shang Xiaojing
2022-11-17 09:23 +0800 Shang Xiaojing
2022-11-16 09:52 +0800 Xiu Jianfeng
2022-11-14 17:31 +0300 Daniil Tatianin
2022-11-09 09:44 +0000 Wang Wensheng
2022-11-09 09:44 +0000 Wang Wensheng
2022-11-07 21:35 +0530 Aashish Sharma
2022-11-07 19:04 +0800 Wang Yufen
2022-10-21 12:30 -0400 Steven Rostedt (Google)
2022-10-20 23:14 -0400 Steven Rostedt (Google)
2022-11-20 10:47 -0800 Linus Torvalds
2022-11-10 12:44 +0000 Mel Gorman
2022-10-05 00:59 +0200 Borys Poplawski
2022-11-20 10:43 -0800 Linus Torvalds
2022-10-26 13:43 +0200 Peter Zijlstra
2022-11-02 09:06 -0400 Mathieu Desnoyers
2022-11-20 10:41 -0800 Linus Torvalds
2022-11-12 17:15 +0200 Adrian Hunter
2022-11-14 10:10 +0530 Ravi Bangoria
2022-09-08 10:33 +0530 Sandipan Das
2022-10-31 10:35 +0100 Marco Elver
2022-11-20 10:39 -0800 Linus Torvalds
2022-11-08 14:01 +0800 Guo Jin
2022-11-20 09:47 -0800 Linus Torvalds
2022-11-16 14:39 +0000 Nicholas Piggini
2022-11-19 15:51 -0800 Linus Torvalds
2022-11-10 03:37 +0000 Zhou Guanghui
2022-11-16 11:50 +0100 Benjamin Block
2022-11-17 08:44 +0000 Yuan Can
2022-11-15 09:50 +0800 Yang Yingliang
2022-11-11 10:44 +0900 Shin'ichiro Kawasaki
2022-11-19 09:08 -0800 Linus Torvalds
2022-11-16 13:15 +0800 Tina Zhang
2022-11-16 13:15 +0800 Tina Zhang
2022-11-19 09:03 -0800 Linus Torvalds
2022-11-15 22:04 +0000 Marc Zyngier
2022-11-12 09:07 +0100 Nicolas Schier
2022-11-12 09:07 +0100 Nicolas Schier
2022-11-12 09:07 +0100 Nicolas Schier
2022-11-19 08:58 -0800 Linus Torvalds
2022-11-16 17:10 +0300 Anastasia Belova
2022-11-15 19:39 +0800 Zhao Yixuan

```

```

* [master] (origin/master) (origin/HEAD) -v6.1-rc6 linux 6.1-rc6
Merge tag 'trace-probes-v6.1' of git://git.kernel.org/pub/scm/linux/kernel/git/trace/linux-trace
- tracing/eprobe: Fix eprobe filter to make a filter correctly
- tracing/eprobe: Fix warning in filter creation
- kprobes: Skip clearing aggrprobe's post_handler in kprobe-on-fttrace case
- rethook: fix a potential memleak in rethook_alloc()
- tracing/eprobe: Fix memory leak of filter string
- tracing: kprobe: Fix potential null-ptr-deref on trace_array in kprobe_event_gen_test_exit()
- tracing: kprobe: Fix potential null-ptr-deref on trace_event_file in kprobe_event_gen_test_exit()
Merge tag 'trace-v6.1-rc5' of git://git.kernel.org/pub/scm/linux/kernel/git/trace/linux-trace
- tracing: Fix race where eprobes can be called before the event
- tracing: Fix potential null-pointer-access of entry in list 'tr->err_log'
- tracing: Remove unused __bad_type_size() method
- tracing: Fix wild-memory-access in register_synth_event()
- tracing: Fix memory leak in test_gen_synth_cmd() and test_empty_synth_event()
- ftrace: Fix null pointer dereference in ftrace_add_mod()
- ring_buffer: Do not deactivate non-existent pages
- ftrace: Optimize the allocation for mcount entries
- ftrace: Fix the possible incorrect kernel message
- tracing: Fix warning on variable 'struct trace_array'
- tracing: Fix memory leak in tracing_read_pipe()
- ring-buffer: Include dropped pages in counting dirty patches
- tracing/ring-buffer: Have polling block on watermark
Merge tag 'x86_urgent_for_v6.1-rc6' of git://git.kernel.org/pub/scm/linux/kernel/git/tip/tip
- x86/fpu: Drop fpregs lock before inheriting FPU permissions
- x86/sgx: Add overflow check in sgx_validate_offset_length()
Merge tag 'sched_urgent_for_v6.1-rc6' of git://git.kernel.org/pub/scm/linux/kernel/git/tip/tip
- sched: Fix race in task_call_func()
- rsoc: Use pr_warn_once() when deprecated/unknown ABI flags are encountered
Merge tag 'perf_urgent_for_v6.1-rc6' of git://git.kernel.org/pub/scm/linux/kernel/git/tip/tip
- perf/x86/intel/pt: Fix sampling using single range output
- perf/x86/amd: Fix crash due to race between amd_pmu_enable_all, perf NMI and throttling
- perf/x86/amd/uncore: Fix memory leak for events array
- perf: Improve missing SIGTRAP checking
Merge tag 'locking_urgent_for_v6.1-rc6' of git://git.kernel.org/pub/scm/linux/kernel/git/tip/tip
- locking: Fix qspinlock/x86 inline asm error
Merge tag 'powerpc-6.1-5' of git://git.kernel.org/pub/scm/linux/kernel/git/powerpc/linux
- powerpc: Fix writable sections being moved into the rodata region
Merge tag 'scsi-fixes' of git://git.kernel.org/pub/scm/linux/kernel/git/jejb/scsi
- scsi: iscsi: Fix possible memory leak when device_register() failed
- scsi: zfcp: Fix double free of FSF request when qdio send fails
- scsi: scsi_debug: Fix possible UAF in sdebug_add_host_helper()
- scsi: target: tcm_loop: Fix possible name leak in tcm_loop_setup_hba_bus()
- scsi: mpt3mr: Suppress command reply debug prints
Merge tag 'iommu-fixes-v6.1-rc5' of git://git.kernel.org/pub/scm/linux/kernel/git/joro/iommu
- iommu/vt-d: Set SRE bit only when hardware has SRS cap
- iommu/vt-d: Preset Access bit for IOVA in FL non-leaf paging entries
Merge tag 'kbuild-fixes-v6.1-3' of git://git.kernel.org/pub/scm/linux/kernel/git/masahiroy/linux-kbuild
- kbuild: Restore .version auto-increment behaviour for Debian packages
- MAINTAINERS: Add linux-kbuild's patchwork
- MAINTAINERS: Remove Michal Marek from Kbuild maintainers
- MAINTAINERS: Add Nathan and Nicolas to Kbuild reviewers
Merge tag '6.1-rc5-smb3-fixes' of git://git.samba.org/sfrench/cifs-2.6
- cifs: add check for returning value of SMB2_set_info_init
- cifs: Fix wrong return value checking smb3_GETELAGS

```