GNU / Linux and Introduction to Git

Seyed Mohammad Mousavi

Richard Stallman

Harvard University Student in 1970

Programmer at the MIT Artificial Intelligence Laboratory in 1971

Quit his job in 1984 to work full-time on the GNU project

GNU Manifesto

Written in 1985 by Richard Stallman to ask for support in developing GNU operation system.

The GNU project is part of Free Software Movement. It is a mistake to associate GNU with the term “open source”.

The term “open source” was coined in 1998 by people who disagree with the free software movement.

What's GNU? Gnu's Not Unix!

GNU is the name for complete Unix-compatible software system.

A new portable C compiler may release this year. An initial kernel exists but many more features are needed to emulate Unix.

We will use free, portable X Window System as well.

Why I Must Write GNU

I consider the Golden Rule requires that if I like a program I must share it with other people who like it.

So that I can continue to use computers without dishonor, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free. I have resigned from the AI Lab to deny MIT any legal excuse to prevent me from giving GNU away.

Why GNU Will Be Compatible with Unix

Unix is not my ideal system, but it is not too bad. The essential features of Unix seem to be good ones, and I think I can fill in what Unix lacks without spoiling them. And a system compatible with Unix would be convenient for many other people to adopt.

How GNU Will Be Available

GNU is not in the public domain. Everyone will be permitted to modify and redistribute GNU, but no distributor will be allowed to restrict its further redistribution. That is to say, proprietary modifications will not be allowed. I want to make sure that all versions of GNU remain free.

Why Many Other Programmers Want to Help

The fundamental act of friendship among programmers is the sharing of programs; The purchaser of software must choose between friendship and obeying the law.

By working on and using GNU rather than proprietary programs, we can be hospitable to everyone and obey the law.

Why All Computer Users Will Benefit

Once GNU is written, everyone will be able to obtain good system software free, just like air.

This means much more than just saving everyone the price of a Unix license. It means that much wasteful duplication of system programming effort will be avoided. This effort can go instead into advancing the state of the art.

You have to charge for the program to pay for providing the support.

If people would rather pay for GNU plus service than get GNU free without service, a company to provide just service to people who have obtained GNU free ought to be profitable.

If your problem is not shared by enough people, the vendor will tell you to get lost.

If your business needs to be able to rely on support, the only way is to have all the necessary sources and tools then you are not at the mercy of any individual

It's no use advertising a program people can get free.

If this is really so, a business which advertises the service of copying and mailing GNU for a fee ought to be successful enough to pay for its advertising and more. This way, only the users who benefit from the advertising pay for it.

On the other hand, if many people get GNU from their friends, and such companies don't succeed, this will show that advertising was not really necessary to spread GNU. Why is it that free market advocates don't want to let the free market decide this?

proprietary operating system to get a competitive edge

GNU will remove operating system software from the realm of competition. You will not be able to get an edge in this area, but neither will your competitors be able to get an edge over you. You and they will compete in other areas, while benefiting mutually in this one. If your business is selling an operating system, you will not like GNU, but that's tough on you. If your business is something else, GNU can save you from being pushed into the expensive business of selling operating systems.

Don't programmers deserve a reward for their creativity?

If anything deserves a reward, it is social contribution.

Creativity can be a social contribution, but only in so far as society is free to use the results.

If programmers deserve to be rewarded for creating innovative programs, by the same token they deserve to be punished if they restrict the use of these programs.

Shouldn't a programmer be able to ask for a reward for his creativity?

Extracting money from users of a program by restricting their use of it is destructive because the restrictions reduce the amount and the ways that the program can be used.

The reason a good citizen does not use such destructive means to become wealthier is that, if everyone did so, we would all become poorer from the mutual destructiveness.

Won’t programmers starve?

Restricting copying is not the only basis for business in software. It is the most common basis.

If it were prohibited, or rejected by the customer, software business would move to other bases of organization which are now used less often. There are always numerous ways to organize any kind of business.

Don't people have a right to control how their creativity is used?

“Control over the use of one's ideas” really constitutes control over other people's lives; and it is usually used to make their lives more difficult.

The idea of copyright did not exist in ancient times, when authors frequently copied other authors at length in works of nonfiction. This practice was useful, and is the only way many authors' works have survived even in part.

Don't people have a right to control how their creativity is used?

The copyright system was created expressly for the purpose of encouraging authorship. In the domain for which it was invented—books, which could be copied economically only on a printing press—it did little harm, and did not obstruct most of the individuals who read the books.

The case of programs today is very different from that of books a hundred years ago.

Competition makes things get done better

The paradigm of competition is a race: by rewarding the winner, we encourage everyone to run faster.

If the runners forget why the reward is offered and become intent on winning, no matter how, they may find other strategies—such as, attacking other runners. If the runners get into a fist fight, they will all finish late.

Won't everyone stop programming without a monetary incentive?

Actually, many people will program with absolutely no monetary incentive. Programming has an irresistible fascination for some people, usually the people who are best at it.

Pay for programmers will not disappear, only become less. So the right question is, will anyone program with a reduced monetary incentive? My experience shows that they will.

Programmers need to make a living somehow.

This way is customary now because it brings programmers and businessmen the most money, not because it is the only way to make a living. It is easy to find other ways if you want to find them. Here are a number of examples.

A manufacturer introducing a new computer will pay for the porting of operating systems onto the new hardware.

The sale of teaching, handholding and maintenance services could also employ programmers.

Programmers need to make a living somehow.

People with new ideas could distribute programs as freeware, asking for donations from satisfied users, or selling handholding services.

Users with related needs can form users' groups, and pay dues. A group would contract with programming companies to write programs that the group's members would like to use.

Programmers need to make a living somehow.

Suppose everyone who buys a computer has to pay x percent of the price as a software tax. The government gives this to an agency like the NSF to spend on software development.

But if the computer buyer makes a donation to software development himself, he can take a credit against the tax. He can donate to the project of his own choosing—often, chosen because he hopes to use the results when it is done. He can take a credit for any amount of donation up to the total tax he had to pay.

GNU Manifesto

In the long run, making programs free is a step toward the postscarcity world, where nobody will have to work very hard just to make a living.

We have already greatly reduced the amount of work that the whole society must do for its actual productivity.

Linus Torvalds

Creator of Linux.

He released the first prototype in 1991.

Version 1.0 was released in 1994.

Linux? GNU? Or GNU / Linux?

Linux is just a Kernel not an Operation System.

Richard Stallman worked On GNU years before Linux was written.

What is Git?

Git is a distributed version-control system for tracking changes in source code

It is designed for coordinating work among programmers

Its goals include speed, data integrity, and support for distributed, non-linear workflows

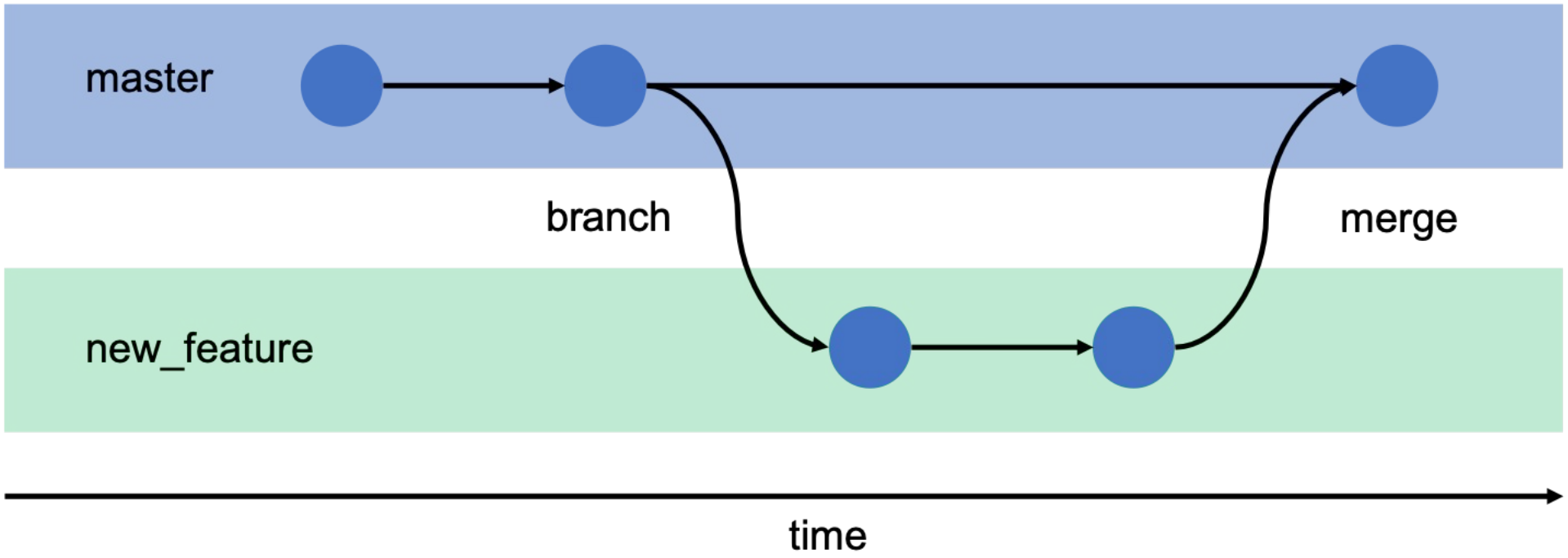
Git History

From 2002 till 2005 people used BitKeeper as a source-control management.

The copyright holder of BitKeeper, Larry McVoy, had withdrawn free use of the product.

Git development began in April 2005, after many developers of the Linux kernel gave up access to BitKeeper.

Visualization of Git



How to install Git

Download mac and windows version from here:

https://git-scm.com/downloads

Download the Linux version from your official repository

Leave all options on default.

Setting Your Name and Email

```
git config --global user.name "FIRST_NAME LAST_NAME"
```

```
git config --global user.email "MY_NAME@example.com"
```

Setting the Default Editor

```
git config --global core.editor emacs
```

Init Command

git init

This command creates an empty Git repository - basically a .git directory with subdirectories

Clone Command

git clone

Clones a repository into a newly created directory.

Status Command

git status

Displays paths that have differences between the index file and the current HEAD commit.

Add Command

git add

This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit.

Commit Command

git commit

Create a new commit containing the current contents of the index and the given log message describing the changes. The new commit is a direct child of HEAD.

Log Command

git log

Show commit logs

Branch Command

git branch

List, create, or delete branches

Checkout Command

git checkout

Switch branches or restore working tree files.

Merge Command

git merge

Join two or more development histories together

Remote Command

git remote

Manage the set of repositories ("remotes") whose branches you track.

Pull Command

git pull

Incorporates changes from a remote repository into the current branch.

Push Command

git push

Updates remote refs using local refs, while sending objects necessary to complete the given refs.

Basic Terminal Commands

cd	Change Directory
pwd	Print Work Directory
ls	List
touch	Creating new File
mkdir	Make Directory
cp	Copy
mv	Move
rm	Remove Here

References

Stallman, R. (1985). The GNU Manifesto- GNU Project - Free Software Foundation. Retrieved from https://www.gnu.org/gnu/manifesto.en.html

Git - Reference. Retrieved from https://git-scm.com/docs