
دانشکده مهندسی برق و کامپیوتر

**پروژه برای دریافت درجه کارشناسی در رشته**
**مهندسی کامپیوتر گرایش مهندسی نرم افزار**

# ورژن کنترل

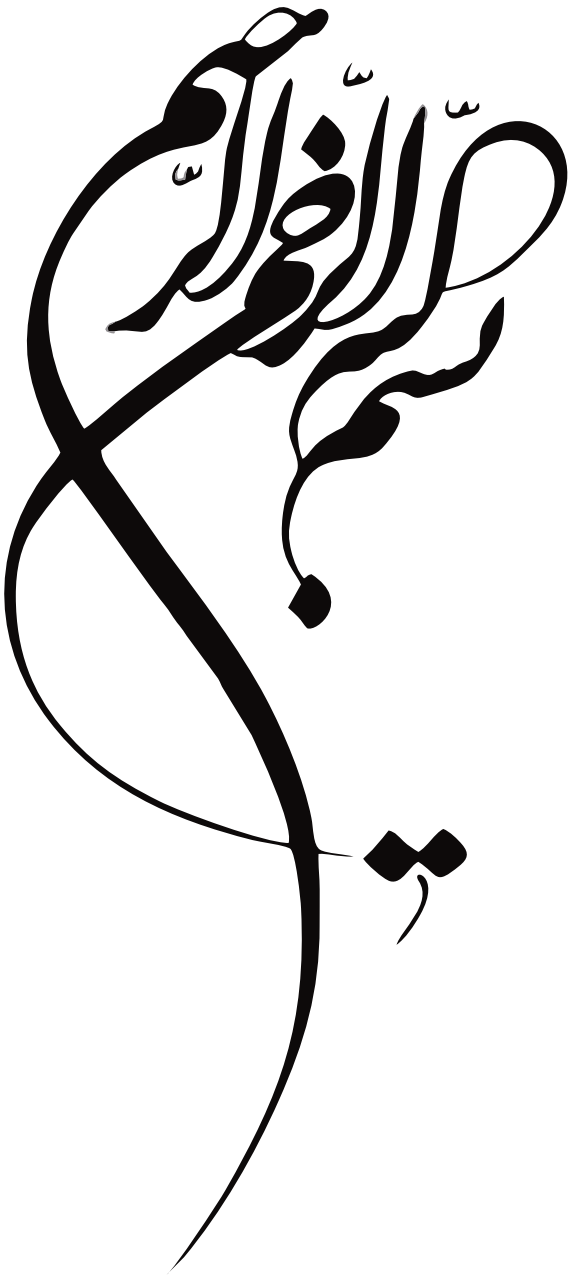
نگارش

الیاس عباسی - کیهان رعیتی

استاد راهنما

**دکتر حمید جزایری**

دی ۱۳۹۷


بسم الله الرحمن الرحیم

دانشگاه صنعتی نوشیروانی بابل

**دانشکده مهندسی برق و کامپیوتر**

**شناسنامه پروژه کارشناسی**

عنوان: **ورژن کنترل**

نام و نام خانوادگی دانشجو: **الیاس عباسی ـ کیهان رعیتی**

شماره دانشجویی: **۹۵۳۱۲۰۰۵۲-۹۴۳۲۲۰۰۵۹**

تاریخ ارایه (دفاع): **۱۳۹۷/۱۰/۲۰**

استاد راهنما: **دکتر حمید جزایری** ..... استادیار دانشگاه صنعتی نوشیروانی بابل

# چکیده

*یک سیستم کنترل نسخه (VCS) یک ابزار برای مدیریت مجموعه ای از کد برنامه است که به شما سه قابلیت مهم را ارائه می کند: برگشت پذیری، همزمان سازی و حاشیه نویسی.*
*اولین VCS ها به عنوان فن آوری پشتیبانی برای برنامه نویسان کامپیوتر ساخته شده اند و هنوز هم استفاده اصلی آنها است، اما آنها می توانند برای محتوای وب یا هر نوع سند یا فایل داده ذخیره شده در کامپیوتر استفاده شوند. VCSes مدرن، به طور خاص، به طور کامل پشتیبانی از فایل های باینری مانند تصاویر را پشتیبانی می کند.*
*اساسی ترین توانایی که از VCS دریافت می کنید، برگشت پذیری است، توانایی پشتیبان گیری از یک حالت ذخیره شده، شناخته شده خوب هنگامی که شما متوجه می شوید که برخی از اصلاحات شما یک اشتباه یا ایده بد بود.*
*VCS ها همگام سازی را پشتیبانی می کنند، توانایی بسیاری از مردم در اصلاح همان مجموعه ای از کد یا اسناد می دانند که تغییرات متضاد می تواند شناسایی و حل شود.*
*در نهایت، VCS ها به شما داده های شما را حاشیه نویسی می کنند، ضمیمه توضیحات توضیحی درباره قصد پشت هر تغییر به آن و رکوردی که مسئول هر تغییر است. حتی برای یک انفرادی که برنامه نویس است، تغییرات تاریخ، کمک مهم به حافظه است؛ برای یک پروژه چند نفره، آنها تبدیل به یک شکل بسیار مهم ارتباطی میان توسعه دهندگان می شوند.*

*هدف ما در این مقاله آموزش آسان پایه ی ورژن کنترل گیت است.*



**واژه‌های کلیدی:** *گیت ، ورژن کنترل*


# تأییدیه صحت و اصالت نتایج و مالکیت مادی ومعنوی

اینجانب **الیاس عباسی ـ کیهان رعیتی** به شماره دانشجویی ۹۵۳۱۲۰۰۵۲-۹۴۳۲۲۰۰۵۹ رشته **مهندسی کامپیوتر** دانشکده **دانشکده مهندسی برق و کامپیوتر** در مقطع **کارشناسی** تأیید می‌نمایم که کلیه نتایج این **پروژه** تحت عنوان **ورژن کنترل** به استاد راهنمایی **دکتر حمید جزایری** حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداري‌شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتي، ضوابط و مقررات آموزشي، پژوهشي و انضباطي ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب مي‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویي به اشخاص اعم از حقیقي و حقوقي و مراجع ذي‌صلاح (اعم از اداري و قضایي) به عهده‌ي اینجانب خواهد بود و دانشگاه هیچ‌گونه مسؤولیتي در این خصوص نخواهد داشت. در ضمن تمام دستاوردهای مادی و معنوی حاصله از **پروژه** متعلق به **دانشگاه صنعتی نوشیروانی بابل** می باشد و اینجانب هیچ گونه ادعایی در قبال آن ندارم.

نام و نام خانوادگی: **الیاس عباسی ـ کیهان رعیتی** ..... امضا


**دانشکده مهندسی برق و کامپیوتر**

# مجوز بهره‌برداری از پایان‌نامه

بهره‌برداری از این پایان‌نامه در چهارچوب مقررات کتابخانه و با توجه به محدودیتی که توسط استاد راهنما به شرح زیر تعیین می‌شود، بلامانع است:

☐ بهره‌برداری از این پایان‌نامه برای همگان و با ذکر منبع، بلامانع است.

☐ بهره‌برداری از این پایان‌نامه با اخذ مجوز از استاد راهنما و با ذکر منبع، بلامانع است.

☐ بهره‌برداری از این پایان‌نامه تا تاریخ _____ ممنوع است.

استاد راهنما: **دکتر حمید جزایری** . . . . . امضا

# فهرست مطالب

# فهرست شکل‌ها

# فصل ۱

# آموزش ورژن کنترل

## ۱ـ۱ مقدمه

ورژن کنترل چیست ؟ و چرا باید به آن اهمیت دهیم ؟ ورژن کنترل یک سیستم ذخیره سازی یک یا چند فایل در طول زمان است [۱] ورژن کنترل سیستمی است که به توسعه‌دهندگان نرم‌افزار کمک می‌کند تا علاوه بر امکان مشارکت روی پروژه‌های نرم‌افزاری، بتوانند به تاریخچه‌ای از کدهایی که قبلاً نوشته‌اند نیز دست پیدا کنند و به طور کلی اهداف استفاده از سیستم‌های ورژن کنترل را می‌توان در موارد زیر خلاصه نمود:

ـ فراهم آوردن فرصتی برای توسعه‌دهندگان به منظور کار کردن به صورت هم‌زمان ـ مجزاسازی نسخه‌های توسعه داده شده اختصاصی تک‌تک توسعه‌دهندگان ـ نگهداری تاریخچه‌ای از هر نسخه از هر چیزی که به اشتراک گذاشته شود.

با استفاده از ورژن کنترل شما می توانید ایده های جدید خود را بدون نگرانی آزمایش کنید و در صورت نیاز به ورژن های قبلی برگردید[۲]. ورژن کنترل سیستمی ضروری برای کار گروهی بر روی یک پروژه نرم افزاری است [۳] .

گیت یک نرم‌افزار کنترل نسخه و از مدل نرم‌افزارهای متن‌باز برای بازنگری کدمنبع توزیع شده و مدیریت منبع کد است که برای دنبال کردن تغییر فایلهای کامپیوتری و دنبال کردن کارهای انجام شده روی آن‌ها توسط افراد مختلف است. که در تمامی سیستم عامل های اصلی توسعه داده شده است [۴] . گیت یک راه قدرتمند برای ردیابی و مقایسه نسخه ها، رفع خطاها، کشف رویکردهای جدید به شیوه ای ساختاری است[۵] .

گیت ابتدا برای توسعه لینوکس توسط لینوس تُروالدز به وجود آمد و اکنون پروژه‌های فراوانی از آن الهام گرفته‌اند. هر دایرکتوری کاری در گیت یک مخزن کامل با تاریخچه کامل تغییرها و قابلیت بازنگری آن‌ها است و برای کار با آن نیازی به دسترسی به شبکه یا سرور مرکزی وجود ندارد.

امروزه برنامه نویسی که به گیت مسلط نباشد را عملا برنامه نویس نمیدادند، به عبارت دیگر تسلط به گیت وظیفه ی هر برنامه نویس میباشد. در هرپروژه ای در هر سطحی در دنیا و در هر شرکتی از گیت استفاده میشود حتی شرکت های غیر برنامه نویسی مانند مقاله نویسی ، پایان نامه و هر نوع فایل متنی دیگری میتوان از گیت استفاده نمود؛ مانند همین مقاله که با استفاده از گیت انجام شده است! پس با ما در ادامه این آموزش همراه باشید چون با همین اطلاعات و تمرین آن ها میتوانید آشنایی با گیت را در رزومه خود اضافه کنید.

## ۱ـ۲ تاریخچه

همانند بساری از اتفاقات بزرگ زندگی، گیت با یک خرابی و انفجار بزرگ شروع شد!!!
هسته لینوکس یک پروژه نرم افزاری منبع باز با دامنه نسبتا وسیع است. برای بیشتر عمر تعمیر و نگهداری هسته لینوکس (۲۰۰۲-۱۹۹۱)، تغییرات در نرم افزار به عنوان تکه ها و فایل های آرشیو شده منتقل شد. در سال ۲۰۰۲، پروژه کرنل لینوکس با استفاده از یک DVCS اختصاصی با نام BitKeeper شروع شد.
در سال ۲۰۰۵، ارتباط بین جامعه که هسته لینوکس و شرکت تجاری که BitKeeper را توسعه داده بود، شکست خورده بود و وضعیت رایگان این ابزار لغو شد. این باعث شد که جامعه توسعه لینوکس (و به ویژه لینوس توروالدز، خالق لینوکس) ابزار خود را بر اساس برخی از درسهایی که در هنگام استفاده از BitKeeper آموخته بودند، توسعه دهند. برخی از اهداف سیستم جدید عبارت بودند از:

سرعت
طراحی ساده
پشتیبانی قوی برای توسعه غیر خطی (هزاران شاخه موازی)
کاملا توزیع شده
قادر به پردازش پروژه های بزرگ مانند هسته لینوکس موثر (سرعت و اندازه داده ها)

از زمان تولدش در سال ۲۰۰۵، Git تکامل یافته و بالغ شده است که به راحتی قابل استفاده است و هنوز هم این ویژگی های اولیه را حفظ می کند. این بسیار شگفت انگیز است، بسیار کارآمد است با پروژه های بزرگ، و آن دارای یک سیستم شاخه باور نکردنی برای توسعه غیر خطی است

## ۱ـ۳ کنترل منابع (Source control)

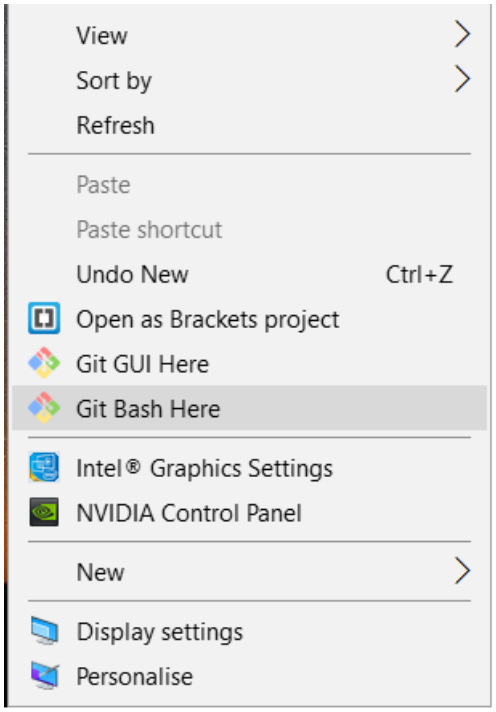
ورژن کنترل های متوعی چون CVN و SVN و bazaar و غیره موجودند اما پدر تمام این برنامه ها Git است! که در تمام شرکت ها و پروژه ها از شما انتظار میرود گیت را بلد بوده و پروژه های خود را ازین طریق ارائه دهید.

## ۱ـ۴ گیت (Git)

در این پژوهش به کارایی گیت و همچنین نحوه اجرای دستورات مختلف آن آشنا خواهید شد.

## ۱ـ۴ـ۱ شروع کار با گیت

برای شروع کار با گیت باید نرم افزار گیت را در سیستم خود نصب نمایید. سپس با با ساختن یک git dash در مسیری که میخواهید پروژه خود را شروع کنید صفحه ای باز میشود و میتوانیم در آن دستورات گیت را اجرا کنیم. برای شروع نیاز به دانستن دو نوع شروع پروژه یا ساخت مخزن داریم:




شکل ۱ ـ ۱ : چگونگی ساختن گیت bash

### init

دستور git init یک مخزن (repository) جدید ایجاد میکند. این دستور برای تبدیل یک پروژه موجود که ورژنش مشخص نشده به یک مخزن گیت یا ساختن مخزن جدید خالی مورداستفاده قرار میگیرد.

بقیه دستورات گیت خارج از محدوده ی یک مخزن گیت ساخته شده نیستند. پس بطور معمول این دستور اولین دستور است که شما آن را در یک پروژه جدید اجرا میکنید.

اجرای دستور git init یک زیرشاخه دستور .git در شاخه ی جاری در حال استفاده میسازد که شامل تمامی متادیتاهایی است که گیت برای ساختن یم مخزن جدید به آن ها نیاز دارد. این متادیتا شامل زیرشاخه ها،مراجع ، قالب فایل هااست.

درکنار شاخه .git در شاخه ریشه پروژه ، یک پروژه موجود باقی می ماند بدون تغییر میماند.

به طور پیش فرض، دستور init پیکربندی Git را به مسیر زیرشاخه .git راه اندازی خواهد کرد. مسیر

دلخواه را می توانید تغییر دهید و سفارشی کنید اگر دوست دارید آن را در جای دیگر اسفاده کنید. شما می توانید متغیر محدوده GIT-DIR را به یک مسیر سفارشی تنظیم کنید و در ابتدا دستور init فایل های پیکربندی Git را در آنجا راه اندازی خواهد کرد. علاوه بر این شما می توانید یک آرگومان separate-git-dir برای نتیجه ی مشابه منتقل کنید. یک مورد استفاده معمول برای یک زیرگروه جداگانه git. این است که پیکربندی سیستم شما را در پوشه اصلی حفظ کند و پوشه git. را در جای دیگر نگه دارد.

```
Comp Center@Keyhan MINGW64 /d/algorithem
$ git init
Initialized empty Git repository in D:/algorithem/.git/

Comp Center@Keyhan MINGW64 /d/algorithem (master)
$
```

شکل ۱ ـ ۲: اعمال دستور init بر روی یک شاخه دلخواه در محیط گیت bash

### clone

این دستور برای دانلود یک ریپازیتوری موجود در شبکه استفاده میشود. در واقع ما میتوانیم با استفاده از این دستور یک فایل git. موجود در شبکه که میتواند سایت هایی مانند gitlab یا github و یا گیت شرکت شما باشد را در یکی از شاخه های سیستم خودمان داشته باشیم. مانندشکل ۱ ـ ۳ این دستور عینا پروژه را در مسیر منتخب شما کپی میکند، و شما میتوانید آن را تغییر داده و ادامه دهید.

گیت کلون تنها یک کپی کاری ساده نیست، بلکه با این دستور شما در واقع یک کپی از تمامی اطلاعاتی که سرور دارد است. تمامی ورژن ها ، تغییرات اعمال شده بر آنها در طول زمان است.

در حقیقت، اگر دیسک سرور شما خراب شود، شما اغلب می توانید تقریبا هر یک از کلون ها را بر روی هر سیستمی در شبکه بکار ببندید تا سرور را به حالت ای که در زمان کپی شدن آن بود، بازگردانید.

```
Comp Center@Keyhan MINGW64 /d/algorithem (master)
$ git clone https://github.com/elyasneo/Latex.git
Cloning into 'Latex'...
remote: Enumerating objects: 106, done.
remote: Counting objects: 100% (106/106), done.
remote: Compressing objects: 100% (70/70), done.
```

شکل ۱ ـ ۳: دانلود روند همین پروژه از گیت لب از پروژه مشترکان

## ۱ ـ ۴ ـ ۲ بررسی و دستیابی

یکی از محاسن گیت این است که ما میتوانیم در هر لحظه تغیررات ایجاد شده بر روی هر فایل را مشاهده کنیم ! فایل های تغییر یافته را بشناسیم و در کل وضعیتی که داریم را مشاهده کنیم. همینطور میتوانیم به گذشته برویم و به ورژن های مختلف دسترسی داشته باشیم. این ویژگی بسیار در کار ما با گیت اهمیت دارد زیرا گاها پیش می آید که فرضا در حین انجام یک بخش از کار نیاز به نصب نرم افزار بر یک سیستم یا ایجاد یک باگ شود که نیاز به دسترسی به نسخه های قبلی برای نصب و یا ایراد یابی باشد.

### status

دستور گیت status وضعیت دایرکتوری کار و منطقه پیمایش را نمایش می دهد. این به شما اجازه می دهد ببینید کدام تغییرات مرتب شده اند و کدام فایل ها توسط Git ردیابی نمی شوند. خروجی این دستور شما هیچ اطلاعاتی را نشان نمی دهد برای این کار باید از گیت log استفاده

کنید. شکل های را ببینید: ۱ـ۴ در این تصویر فایل های قرمز ابتدایی فایل های تغییر یافته و فایل های قسمت دوم فایل های که توسط گیت ردیابی نمیشوند را نمایش میدهد.

```
Comp Center@Keyhan MINGW64 ~/Desktop/latec project/Latex (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Chapters/chapter1.aux
        modified:   Chapters/chapter1.tex
        modified:   Figures/GitInit.PNG
        modified:   MainThesis.aux
        modified:   MainThesis.lof
        modified:   MainThesis.log
        modified:   MainThesis.out
        modified:   MainThesis.pdf
        modified:   MainThesis.synctex.gz
        modified:   MainThesis.toc

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Figures/GitClone.PNG

no changes added to commit (use "git add" and/or "git commit -a")
```

شکل ۱ـ۴: گیت استتوس بر روی روند مقاله فعلی بر شاخه git.

### log

با اجرای این دستور لیستی از کامیت هایی که تاکنون بر روی شاخه انجام شده است مشاهده میشود، شکل را ببینید: ۱ـ۵

```
Comp Center@Keyhan MINGW64 ~/Desktop/latec project/Latex (master)
$ git log
commit 629a2080e60c966f433ce588b47f652e21c16247 (HEAD -> master)
commit 629a2080e60c966f433ce588b47f652e21c16247 (HEAD -> master)
Author: keyhan <keyhan@gmail.com>
Date:   Thu Jan 10 12:50:59 2019 +0330

    commit #31

commit 09fbe3ab9103e910656ee4ce7944ed5cd98eb621 (origin/master, origin/HEAD)
Author: keyhan <keyhan@gmail.com>
Date:   Thu Jan 10 11:12:55 2019 +0330

    avale sob

commit ba0c87e94e2a0b50018f0454759ab57efa07209d
Author: keyhan <keyhan@gmail.com>
Date:   Thu Jan 10 03:14:19 2019 +0330

    shabe nokhost

commit 55a4fc885106e0d3fea98edd6e5427b2014ed7e4
Author: Elyas <elyasneo@gmail.com>
Date:   Thu Jan 10 00:24:03 2019 +0330

    m1

commit 8bbca46ee10965e069b5b8ffbf37fdbd000ad2c8
Author: Elyas <elyasneo@gmail.com>
Date:   Thu Jan 10 00:22:30 2019 +0330

    m1
```

شکل ۱ـ۵: مشاهده کامیت های انجام شده بر روی مقاله کنونی تا بدینجای کاربا دستور گیت log

### checkout

دستور checkout عمل تغییر بین نسخه های مختلف یک موجودیت هدف است. این دستور بر روی سه موجودیت مجزا عمل می کند: فایل ها، commit ها، و شاخه ها. علاوه بر تعریف «checkout»، عبارت «چک کردن» معمولا به معنای اجرای دستور فرمان گیت checkout است. در موضوع لغو تغییرات، ما شاهد چگونگی استفاده از گیت Checkout برای مشاهده اعمال قدیمی بودیم. اکثر تمرکزاین سند بر عملیات checkout در شاخه ها است.

چک کردن شاخه ها مشابه چک کردن قرارداد های قدیمی و فایل ها در آن است که دایرکتوری کاری برای مطابقت با شاخه انتخاب شده به روز می شود؛ با این حال، تغییرات جدید در تاریخ پروژه ذخیره می شود ـ یعنی یک عملیات فقط خواندنی نیست.

در شکل ۱ـ۶ ابتدا به یک نسخه قدیمی میرویم و بعد در شکل ۱ـ۷ با دستور بعدی به ورژن فعلی

برمیگردیم!

```
Comp Center@Keyhan MINGW64 ~/Desktop/latec project/Latex (master)
$ git checkout 8bbca46ee10965e069b5b8ffbf37fdbd000ad2c8
Note: checking out '8bbca46ee10965e069b5b8ffbf37fdbd000ad2c8'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 8bbca46 m1
```

شکل ۱ ـ ۶: به ورژن های قبلی

```
Comp Center@Keyhan MINGW64 ~/Desktop/latec project/Latex ((8bbca46...))
$ git checkout master
Previous HEAD position was 8bbca46 m1
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)
```

شکل ۱ ـ ۷: برگشت به ورژن master

### ۱ ـ ۴ ـ ۳ اعمال تغییرات

پس از تغییر دادن هر فایل اگر از انجام آن تغییر مطمئن هستید میتوانید آن تغییرات را ذخیره کنید. در اینجا روند تغییر هر فایل را با هم یاد میگیریم:

#### add

پس از اعمال تغییر روی هر فایل برای ثبت کردن این تغیررات ابتدا باید آن فایل را در لبه یا همان stage قراردهیم! با این دستور فایل های تغییر یافته در آستانه ی ثبت تغییر قرار میگیرند. ؟؟

#### reset

دستور ریست برای خالی کردن stage استفاده میشود. هنگامی که ما از یک کامیت پشیمان شدیم قبل از اعمالش میتوانیم با این دستور لبه را خالی کنیم.

```
Comp Center@Keyhan MINGW64 ~/Desktop/latec project/Latex (master)
$ git add -A
warning: LF will be replaced by CRLF in Chapters/chapter1.aux.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in MainThesis.aux.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in MainThesis.ind.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in MainThesis.lof.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in MainThesis.log.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in MainThesis.out.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in MainThesis.toc.
The file will have its original line endings in your working directory
```

شکل ۱ ـ ۸: مثالی از گیت add

```
Comp Center@Keyhan MINGW64 ~/Desktop/latec project/Latex (master)
$ git reset
Unstaged changes after reset:
M       Chapters/chapter1.aux
M       Chapters/chapter1.tex
M       MainThesis.aux
M       MainThesis.lof
M       MainThesis.log
M       MainThesis.out
M       MainThesis.pdf
M       MainThesis.synctex.gz
M       MainThesis.toc
```

شکل ۱ ـ ۹: مثالی از گیت reset

**commit**

دستور commit محتویات فعلی فهرست لبه را در یک index جدید ذخیره می کند به همراه یک پیام توصیف تغییرات که توسط کاربر وارد میشود.

```
Comp Center@Keyhan MINGW64 ~/Desktop/latec project/Latex (master)
$ git commit -m "commit #31"
[master 629a208] commit #31
 12 files changed, 65 insertions(+), 35 deletions(-)
 create mode 100644 Figures/GitClone.PNG
 rewrite Figures/GitInit.PNG (99%)
 create mode 100644 Figures/GitStatus.PNG
 rewrite MainThesis.synctex.gz (99%)
```

شکل ۱ ـ ۱۰: مثالی از گیت commit

## ۱ـ۴ـ۴ توزیع

در ابتدای خلقت گیت بیان شد که قرار است گیت بر روی سیستم های توزیع شده در دسترس قرار بگیرد که این اتفاق چندی پس از پیدایش آن افتاد. برای فرستادن و دریافت فایل ها به شبکه از دو دستور زیر استفاده میشود؛

### push

دستور push جهت آپلود فایل جاری بر روی شبکه میباشد !
بعد از به کارگیری این دستور فایل پروژه با فایل قبلی روی شبکه جایگزین میشود.

```
Comp Center@Keyhan MINGW64 ~/Desktop/latec project/Latex (master)
$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 38.77 KiB | 19.38 MiB/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/elyasneo/Latex.git
   589de5f..aac05f8  master -> master
```

شکل ۱ـ۱۱: مثالی از گیت push

### pull

دستور pull دقیقا معکوس کار دستور push است.
پس از بکارگیری دستور گیت pull فایل موجود در شبکه جایگزین فایل کنونی میشود و تمام تغییرات و تمام اطلاعات آن به سیستم کاربر می آید.

```
Comp Center@Keyhan MINGW64 ~/Desktop/latec project/Latex (master)
$ git pull
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 2 (delta 1), reused 2 (delta 1), pack-reused 0
Unpacking objects: 100% (2/2), done.
From https://github.com/elyasneo/Latex
   5e5731e..589de5f  master     -> origin/master
Updating 5e5731e..589de5f
Fast-forward
 ReadMe.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ReadMe.txt
```

شکل ۱ـ۱۲: مثالی از گیت pull

### ۱ـ۴ـ۵ گسترش

یکی از قابلیت ها و شاید مهمترین آن ها در گیت امکان ایجاد شاخه در پروژه هاست: یعنی شما میتوانید با استفاده از شاخه سازی یا branching کارهای خود را دسته بندی کنید و هر قابلیت پروژه را بصورت جداگانه انجام دهید! قابلیت های جدید را اضفه کنید یا نسخه هارا مدیریت کنید. با استفاده از شاخه ها میشود کارهارا بین افراد دخیل در پروژه تقسیم نمود و بدون اینکه لطمه ای به فایل اصلی وارد شود تمام تغییرات را ذخیره نمیود و در آخر نیز با پروژه اصلی ترکیب کرد.

#### branch

شاخه بندی یک ویژگی است که در بسیاری از سیستم های مدرن کنترل نسخه موجود است. شاخه گیری در دیگر VCS ها می تواند عملیات گران در هر زمان و فضای دیسک باشد. در Git، شاخه ها بخشی از روند توسعه روزمره شما هستند. شاخه های Git به طور موثری اشاره گر به عکس فوری تغییرات شما هستند. وقتی میخواهید یک ویژگی جدید اضافه کنید یا یک اشکال را رفع کنید، مهم نیست که چقدر بزرگ یا کوچک است، شاخه جدیدی را برای تغییر دادن تغییرات خود ایجاد می کنید. این باعث می شود که کد ناپایدار به پایه اصلی اصلی ادغام شود و این به شما امکان می دهد تا تاریخ آینده خود را قبل از ادغام آن در شاخه اصلی پاک کنید.
پیاده سازی پشت شاخه های Git بسیار سبک تر از سایر مدل های سیستم کنترل نسخه است. به جای کپی کردن پرونده ها از دایرکتوری به دایرکتوری، Git یک شاخه را به عنوان یک مرجع به یک مرتبه ذخیره می کند. به این معنا، یک شاخه نشان دهنده نوک یک سری از مرتکبین است ـ این یک ظرف برای انجام نیست. تاریخچه یک شاخه از طریق روابط متعهد استخراج می شود.
همانطور که خواندید، به یاد داشته باشید که شاخه های Git مانند شاخه های SVN نیستند. درحالیکه شاخه های SVN تنها برای تسخیر تلاش های گاه به گاه در مقیاس بزرگ توسعه می یابند، شاخه های گیت بخشی جدایی ناپذیر از گردش کاری روزمره شما هستند. محتوای زیر در معماری شاخه داخلی GIT گسترش می یابد.

#### merge

ادغام کردن راه Git برای قرار دادن تاریخچه شاخه دار با هم است. دستور ادغام git اجازه می دهد تا خطوط مستقل توسعه ایجاد شده توسط شاخه git را بسازید و آنها را در یک شاخه واحد ادغام کنید.
توجه داشته باشید که تمام دستورات زیر ارائه شده به شاخه فعلی ادغام می شوند. شاخه فعلی برای انعکاس ادغام به روز می شود، اما شاخه هدف کاملا تحت تاثیر قرار نخواهد گرفت. باز هم این به

این معنی است که ادغام git اغلب در ارتباط با پرداخت git برای انتخاب شاخه جاری وgit branch -d برای حذف شاخه منسوخ استفاده می شود.

**در این رشته آموزش با دستور شاخه بندی و گسترش در گیت آشنا خواهید شد:**

به عنوان مثال میخواهیم به پروژه یک فایل readMe در شاخه ای جدید اضافه کنیم .

ـ شکل ۱ ـ ۱۳ : ساخت شاخه جدید به نام newBranch

```
Neo@NeoPC MINGW64 /d/Project/Latex/943220059_953120052 (master)
$ git branch newBranch
```

شکل ۱ ـ ۱۳

ـ شکل ۱ ـ ۱۴ :رفتن از شاخه اصلی به شاخه newBranch

```
Neo@NeoPC MINGW64 /d/Project/Latex/943220059_953120052 (master)
$ git checkout newBranch
Switched to branch 'newBranch'

Neo@NeoPC MINGW64 /d/Project/Latex/943220059_953120052 (newBranch)
$
```

شکل ۱ ـ ۱۴

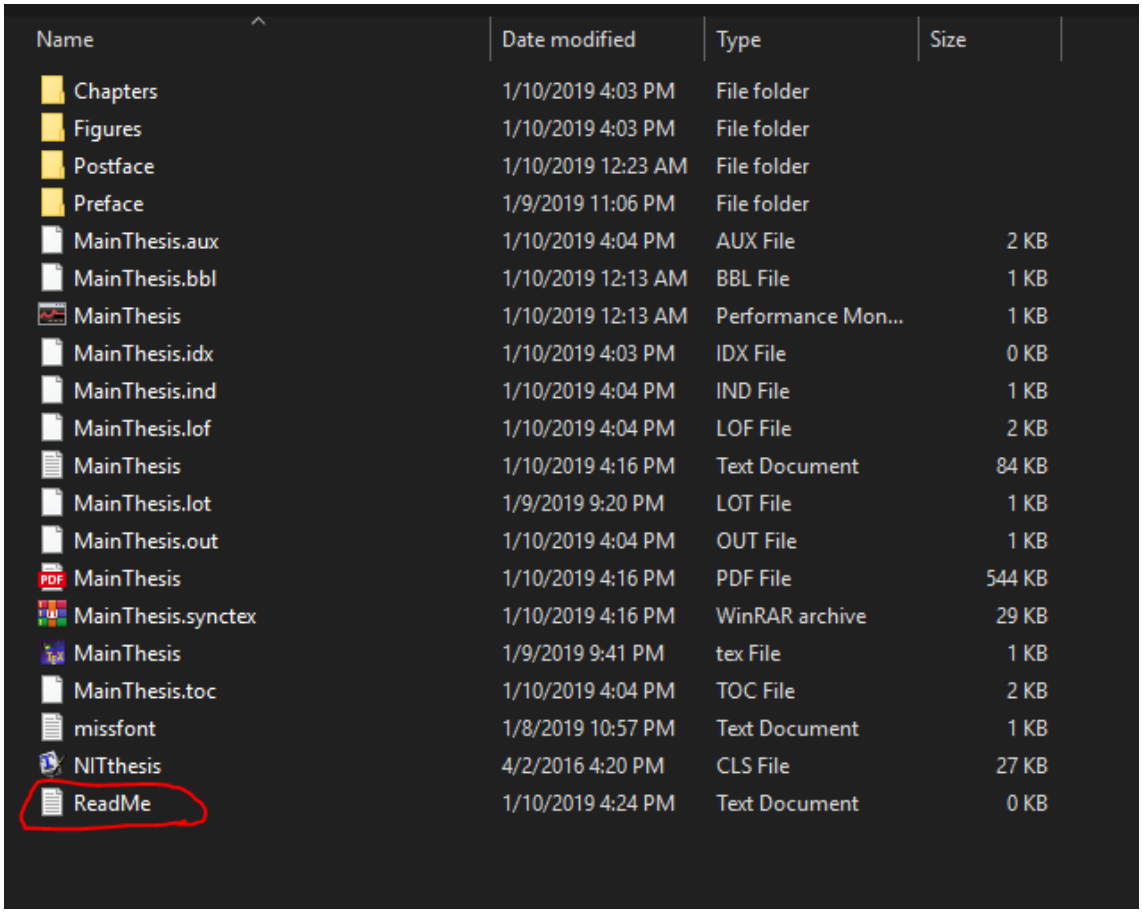
ـ شکل ۱ ـ ۱۵ :اضافه کردن فایل readMe در شاخه newBranch و سپس add و commit

```
Neo@NeoPC MINGW64 /d/Project/Latex/943220059_953120052 (newBranch)
$ git add -A

Neo@NeoPC MINGW64 /d/Project/Latex/943220059_953120052 (newBranch)
$ git commit -m"add readMe in newBranch"
[newBranch 589de5f] add readMe in newBranch
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ReadMe.txt
```

شکل ۱ ـ ۱۵

ـ شکل ۱ ـ ۱۶ :محتویات پوشه پروژه بعد از عملیات گفته شده



| Name | Date modified | Type | Size |
|---|---|---|---|
| Chapters | 1/10/2019 4:03 PM | File folder | |
| Figures | 1/10/2019 4:03 PM | File folder | |
| Postface | 1/10/2019 12:23 AM | File folder | |
| Preface | 1/9/2019 11:06 PM | File folder | |
| MainThesis.aux | 1/10/2019 4:04 PM | AUX File | 2 KB |
| MainThesis.bbl | 1/10/2019 12:13 AM | BBL File | 1 KB |
| MainThesis | 1/10/2019 12:13 AM | Performance Mon... | 1 KB |
| MainThesis.idx | 1/10/2019 4:03 PM | IDX File | 0 KB |
| MainThesis.ind | 1/10/2019 4:04 PM | IND File | 1 KB |
| MainThesis.lof | 1/10/2019 4:04 PM | LOF File | 2 KB |
| MainThesis | 1/10/2019 4:16 PM | Text Document | 84 KB |
| MainThesis.lot | 1/9/2019 9:20 PM | LOT File | 1 KB |
| MainThesis.out | 1/10/2019 4:04 PM | OUT File | 1 KB |
| MainThesis | 1/10/2019 4:16 PM | PDF File | 544 KB |
| MainThesis.synctex | 1/10/2019 4:16 PM | WinRAR archive | 29 KB |
| MainThesis | 1/9/2019 9:41 PM | tex File | 1 KB |
| MainThesis.toc | 1/10/2019 4:04 PM | TOC File | 2 KB |
| missfont | 1/8/2019 10:57 PM | Text Document | 1 KB |
| NITthesis | 4/2/2016 4:20 PM | CLS File | 27 KB |
| ReadMe | 1/10/2019 4:24 PM | Text Document | 0 KB |

شکل ۱ ـ ۱۶

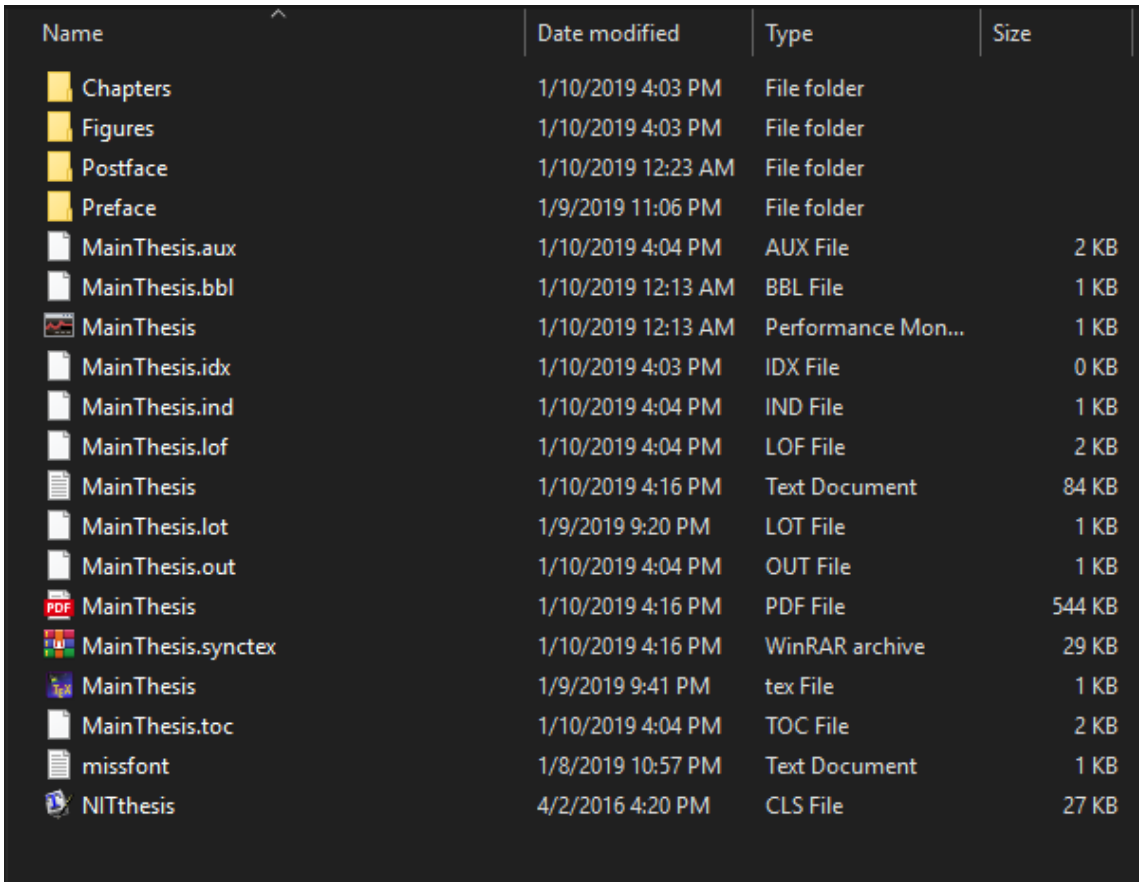
ـ شکل ۱ ـ ۱۷ :برگشتن به شاخه اصلی

```
Neo@NeoPC MINGW64 /d/Project/Latex/943220059_953120052 (newBranch)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

Neo@NeoPC MINGW64 /d/Project/Latex/943220059_953120052 (master)
$
```

شکل ۱ ـ ۱۷

ـ شکل ۱ ـ ۱۸ :محتویات پوشه پروژه در شاخه اصلی



| Name | Date modified | Type | Size |
|---|---|---|---|
| Chapters | 1/10/2019 4:03 PM | File folder | |
| Figures | 1/10/2019 4:03 PM | File folder | |
| Postface | 1/10/2019 12:23 AM | File folder | |
| Preface | 1/9/2019 11:06 PM | File folder | |
| MainThesis.aux | 1/10/2019 4:04 PM | AUX File | 2 KB |
| MainThesis.bbl | 1/10/2019 12:13 AM | BBL File | 1 KB |
| MainThesis | 1/10/2019 12:13 AM | Performance Mon... | 1 KB |
| MainThesis.idx | 1/10/2019 4:03 PM | IDX File | 0 KB |
| MainThesis.ind | 1/10/2019 4:04 PM | IND File | 1 KB |
| MainThesis.lof | 1/10/2019 4:04 PM | LOF File | 2 KB |
| MainThesis | 1/10/2019 4:16 PM | Text Document | 84 KB |
| MainThesis.lot | 1/9/2019 9:20 PM | LOT File | 1 KB |
| MainThesis.out | 1/10/2019 4:04 PM | OUT File | 1 KB |
| MainThesis | 1/10/2019 4:16 PM | PDF File | 544 KB |
| MainThesis.synctex | 1/10/2019 4:16 PM | WinRAR archive | 29 KB |
| MainThesis | 1/9/2019 9:41 PM | tex File | 1 KB |
| MainThesis.toc | 1/10/2019 4:04 PM | TOC File | 2 KB |
| missfont | 1/8/2019 10:57 PM | Text Document | 1 KB |
| NITthesis | 4/2/2016 4:20 PM | CLS File | 27 KB |

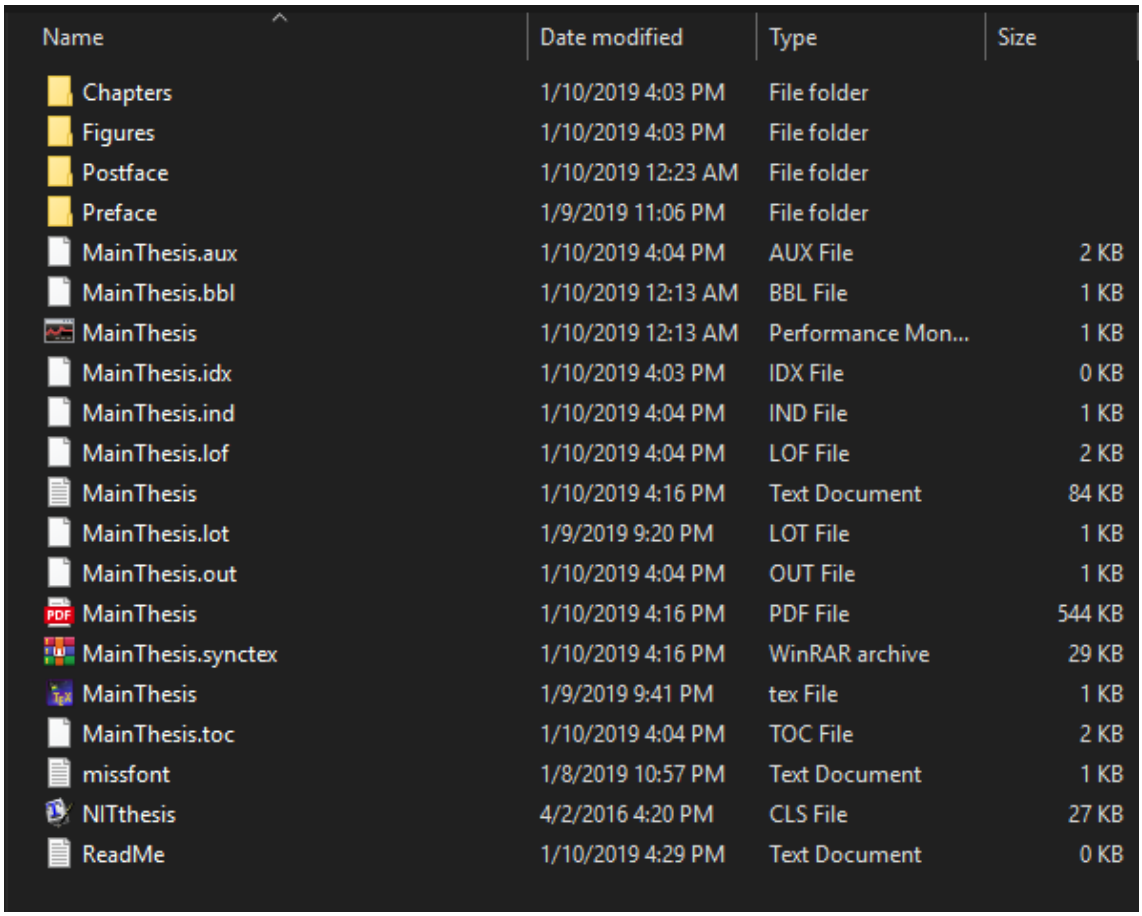
شکل ۱ ـ ۱۸

ـ شکل ۱ ـ ۱۹ :ترکیب (merge) شاخه اصلی و شاخه newBranch

```
Neo@NeoPC MINGW64 /d/Project/Latex/943220059_953120052 (master)
$ git merge newBranch
Updating 5e5731e..589de5f
Fast-forward
 ReadMe.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ReadMe.txt
```

شکل ۱ ـ ۱۹

ـ شکل ۱ ـ ۲۰ :محتویات پوشه پروژه پر شاخه اصلی بعد از ترکیب (merge)



| Name | Date modified | Type | Size |
|---|---|---|---|
| Chapters | 1/10/2019 4:03 PM | File folder | |
| Figures | 1/10/2019 4:03 PM | File folder | |
| Postface | 1/10/2019 12:23 AM | File folder | |
| Preface | 1/9/2019 11:06 PM | File folder | |
| MainThesis.aux | 1/10/2019 4:04 PM | AUX File | 2 KB |
| MainThesis.bbl | 1/10/2019 12:13 AM | BBL File | 1 KB |
| MainThesis | 1/10/2019 12:13 AM | Performance Mon... | 1 KB |
| MainThesis.idx | 1/10/2019 4:03 PM | IDX File | 0 KB |
| MainThesis.ind | 1/10/2019 4:04 PM | IND File | 1 KB |
| MainThesis.lof | 1/10/2019 4:04 PM | LOF File | 2 KB |
| MainThesis | 1/10/2019 4:16 PM | Text Document | 84 KB |
| MainThesis.lot | 1/9/2019 9:20 PM | LOT File | 1 KB |
| MainThesis.out | 1/10/2019 4:04 PM | OUT File | 1 KB |
| MainThesis | 1/10/2019 4:16 PM | PDF File | 544 KB |
| MainThesis.synctex | 1/10/2019 4:16 PM | WinRAR archive | 29 KB |
| MainThesis | 1/9/2019 9:41 PM | tex File | 1 KB |
| MainThesis.toc | 1/10/2019 4:04 PM | TOC File | 2 KB |
| missfont | 1/8/2019 10:57 PM | Text Document | 1 KB |
| NITthesis | 4/2/2016 4:20 PM | CLS File | 27 KB |
| ReadMe | 1/10/2019 4:29 PM | Text Document | 0 KB |

شکل ۱ ـ ۲۰

# مراجع


[1] J. D. Blischak, E. R. Davenport, and G. Wilson, "A quick introduction to version control with Git and GitHub," *PLoS computational biology*, vol. 12, no. 1, p. e1004668, 2016.

[2] S. Chacon and B. Straub, *Pro git*. Apress, 2014.

[3] B. De Alwis and J. Sillito, "Why are software projects moving from centralized to decentralized version control systems?," in *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, pp. 36–39, 2009.

[4] K. Ram, "Git can facilitate greater reproducibility and increased transparency in science," *Source code for biology and medicine*, vol. 8, no. 1, p. 7, 2013.

[5] D. Spinellis, "Git," *IEEE software*, no. 3, pp. 100–101, 2012.