به نام خدا

درس مبانی برنامه‌سازی

فاز اول پروژه

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال اول ۰۲ـ۰۳

استاد:

دکتر محمدامین فضلی

مسئول پروژه:

محسن قاسمی

طراحان فاز ۱ :

محمدحسین اسلامی، یوسف سدیدی، سهند اسماعیل‌زاده، مرتضی ترابی، آرمین خسروی، محمدعارف زارع‌زاده، آرین افضل‌زاده

فهرست

معرفی پروژه

اهداف پروژه

- هدف این پروژه ساخت یک نرم‌افزار کنترل نسخه (Version Control System) است. نمونه‌های مختلفی برای این نرم‌افزار وجود دارد. ابزار گیت (Git) که در کارگاه کامپیوتر با آن آشنا شده‌اید، معروف‌ترین و پراستفاده‌ترین نرم‌افزار کنترل نسخه است.
- در این پروژه انتظار می‌رود که با دانشی که تا به اینجا در درس مبانی برنامه‌سازی کسب کرده‌اید، یک نسخه ساده‌سازی شده از این ابزار را پیاده کنید.
- انتظار می‌رود در فرآیند توسعه این پروژه از یک سیستم مدیریت نسخه، Git، استفاده کنید و پروژه را بر بستر یک مخزن Github، نگه‌داری کنید. در این مورد لازم است تغییرات خود را در دوره‌های کوتاه مدت commit کنید تا تاریخچه خوبی از تغییراتی که در طول پروژه ایجاد کرده‌اید، داشته باشید.

ابزار کنترل نسخه چیست؟

ابزار کنترل نسخه، ابزاری برای ردیابی و مدیریت تغییرات سورس کد یک نرم‌افزار است. این ابزار تاریخچه هر گونه تغییر در کد را در یک پایگاه‌داده خاص نگه‌داری می‌کند. قدرت این ابزار هنگامی مشخص می‌شود که توسعه‌دهنده نرم‌افزار در فرآیند توسعه به یک اشتباه در فرآیند توسعه برخورد می‌کند و یا اینکه نیاز به بررسی تاریخچه تغییرات پروژه پیدا می‌کند. به واسطه امکاناتی که این ابزار به ما می‌دهد، بازگرداندن و یا مشاهده تغییرات گذشته به سادگی امکان پذیر خواهد بود و فرآیند توسعه نرم‌افزار با چالش‌های بسیار کمتری مواجه خواهد شد.

توانایی‌های لازم برای پروژه

در فرآیند توسعه پروژه به همه آنچه تا کنون در درس آموخته‌اید، نیاز خواهید داشت. توانایی‌های زیر به طور مستقیم در توسعه پروژه مورد نیاز خواهند بود:

- توانایی کار با فایل.

- توانایی کار با رشته و اعمال عملیات مختلف روی آنها.
- کار با struct ها و طراحی ساختار کلی برنامه به واسطه آنها.
- کار با پوینتر: تخصیص، مدیریت و آزاد سازی حافظه.
- آشنایی با filesystem لینوکس و دسترسی فایل‌ها و نحوه ذخیره‌سازی آنها.

نکات قابل توجه

- پس از اتمام این فاز، در گیت خود یک تگ با ورژن "v1.0.0" بزنید. در روز تحویل حضوری این tag بررسی خواهد شد و کدهای پس از آن نمره‌ای نخواهد گرفت. برای اطلاعات بیش‌تر در مورد شیوه ورژن‌گذاری، می‌توانید به این لینک مراجعه کنید. البته برای این پروژه صرفا رعایت کردن همان ورژن گفته شده کافیست، اما خوب است که با منطق ورژن‌بندی هم آشنا بشوید.
- در فاز اول می‌توانید حداکثر سه روز تاخیر بدون کسر نمره داشته باشید. پس از آن تا سه روز به ازای هر روز با ۱۰ درصد کسر نمره از فاز اول، می‌توانید تاخیر داشته باشید.
- در صورت کشف تقلب نمره کل پروژه صفر خواهد شد.
- هنگام تحویل اجزای مختلف پروژه نمره به آنچه که اجرا خواهد شد تعلق خواهد گرفت و به کد صرفا پیاده‌سازی شده نمره‌ای تعلق نخواهد گرفت.
- در پیاده‌سازی پروژه تا حد امکان از قواعدی که برای کدنویسی تمیز آموخته‌اید استفاده کنید.

  برنامه‌ای که شما پیاده می‌کنید باید مانند دستور git در همه جا قابل فراخوانی باشد. برای این منظور باید فایل کامپایل شده برنامه خود را در PATH سیستم‌عامل قرار دهید.

توضیح بخش‌های مختلف پروژه

انتخاب نام مناسب برای پروژه

در پیاده‌سازی پروژه باید یک کامند جایگزین برای git انتخاب کنید و پروژه را به واسطه آن پیاده‌سازی کنید. ما در ادامه از کامند neogit استفاده می‌کنیم. شما می‌توانیم در پیاده‌سازی پروژه از این کامند یا هر نام دلخواه دیگری استفاده کنید.

تنظیمات اولیه پروژه

با استفاده از این دستور، یک شخص می‌تواند username و email خود را برای تمام پروژه‌هایی که دارد، set بکند. توجه شود که با این دستور username و gmail تمام پروژه‌ها override می‌شود.

```
neogit config –global user.name “ “
```

```
neogit config –global user.email “ “
```

نکته: در صورتی که آپشن global– در کامند نباشد، این موارد صرفا برای خود پروژه تغییر می‌کند.

با استفاده از دستور زیر می‌توانیم به یک دستور دیگر که در پروژه موجود است یک نام نسبت دهیم و پس از آن برای استفاده از آن دستور از کامند زیر استفاده کنیم

```
neogit config (–global) alias.<alias-name> “a command”
```

برای مثال با اجرای دستور زیر می‌توانی از دستور neogit arc برای اضافه کردن همه فایل‌های تغییر یافته در پوشه src استفاده کرد.

```
neogit config alias.arc “git add src/”
```

خطاها:

- در صورتی که کامند نسبت داده شده معتبر نباشد، باید خطای مناسب در خروجی چاپ بشود

ایجاد مخزن گیت

این دستور مخزن git را در پوشه‌ای که در آن اجرا می‌شود، راه‌اندازی می‌کند. به واسطه این دستور که یک پوشه پنهان با نام neogit ساخته می‌شود و تمام اطلاعات لازم برای کنترل ورژن و همچنین تنظیمات اولیه پروژه در آن نگه‌داری می‌شود.

```
neogit init
```

خطاها:

- در صورتی که پوشه پنهان با نام neogit در خود پوشه‌ای که کامند در آن اجرا می‌شود و یا یکی از پوشه‌های بالاتر آن وجود داشته باشد، خطای مناسب در خروجی چاپ بشود.

نکته: در صورتی که دستورهای بعدی در پوشه‌ای که در آن یا پوشه‌های بالاتر آن مخزن گیت ایجاد نشده باشد، باید خطای مناسب در خروجی نمایش داده شود.

دستور add

```
neogit add [file address or directory address]
```

در صورتی که آرگومان ورودی دستور یک فایل باشد،آن فایل به حالت stage می‌رود. در صورتی که یک پوشه به عنوان ورودی داده بشود، باید تمام فایل‌های تغییر یافته داخل آن پوشه به حالت stage بروند.

نکته: اگر یک فایل در حالت stage قرار داشته باشد، با دوباره اجرا شدن این دستور برای آن فایل تغییر به وجود نخواهد آمد.

در صورتی که پوشه یا فایل مد نظر وجود نداشته باشد، خطای مناسب در خروجی نمایش داده شود.

```
neogit add s*c
```

این دستور باید بتواند فایل‌ها و یا پوشه‌هایی که نامشان به واسطه یک wildcard مشخص شده را به حالت stage ببرد.

نکته: تنها لازم است wildcard ستاره تنها برای یک کلمه پیاده بشود.

```
neogit add -f <file1> <file2> <dir1>
```

در صورتی که از آپشن f- در این دستور استفاده شود، می‌توان چندین فایل یا پوشه را با هم به حالت stage برد. در صورتی که تعدای از این فایل یا پوشه‌ها وجود نداشتند، برای آنها در خروجی خطای مناسب نمایش داده بشود و دستور برای سایرین بدون مشکل اجرا بشود.

```
neogit add -n <depth>
```

امتیازی

```
neogit add -n <depth> (depth > 1)
```

این دستور لیستی از تمام فایل‌ها و دایرکتوری‌ها داخل پوشه را باید در خروجی نمایش دهد و مشخص کند که هر کدام آیا در حالت stage قرار دارد یا خیر. مقدار depth مشخص می‌کند که تا چند لایه در پوشه پیش برویم.

امتیازی

```
neogit add -redo
```

در صورتی که این دستور اجرا بشود، تمام فایل‌هایی که ابتدا در حالت stage قرار

داشته‌اند و سپس با اعمال تغییر به حالت unstage رفته‌اند را به حالت stage برمی‌گرداند.

دستور reset

```
neogit rest [file address or directory address]
```

در صورتی که آرگومان ورودی دستور یک فایل باشد،آن فایل به حالت unstage می‌رود. درصورتی که یک پوشه به عنوان ورودی داده بشود، باید تمام فایل‌های stage آن پوشه به حالت unstage بروند.

موارد امتیازی:

- پیاده‌سازی wildcard مانند دستور add
- پیاده‌سازی آپشن f- مانند دستور add
- در صورتی که داخل پوشه، پوشه دیگری وجود داشته باشد، محتویات آن پوشه نیز unstage بشوند

```
neogit reset -undo
```

این دستور، آخرین فایل یا فایل‌هایی که به واسطه دستور add به حالت stage رفته‌اند را به حالت unstage می‌برد.

امتیازی: قابلیت تکرار کردن این دستور تا ۱۰ مرحله

دستور status

```
neogit status
```

این دستور وضعیت فایل‌های داخل پوشه‌ای را که در آن دستور اجرا می‌شود به نمایش می‌گذارد. در هر خط خروجی اسم فایل نمایش داده شود و در کنار آن یه کد وضعیت قرار می‌گیرد. این کد وضعیت به صورت XY است. X دو مقدار + و ـ را می‌پذیرد و که به معنای بودن یا نبودن فایل در حالت stage است. Y نیز حالات زیر را دارد:

- M: محتویات فایل تغییر یافته است.
- A: فایل به پروژه اضافه شده است.
- D: فایل پاک شده است.
- T **(امتیازی)**: نوع دسترسی‌های فایل عوض شده باشد. مثلا دسترسی فایل از ۷۵۳ به ۴۴۴ تغییر پیدا کرده باشد.

در صورتی که فایل تغییری پیدا نکرده باشد، نیازی نیست در خروجی این دستور نمایش داده شود.

امتیازی: با اجرای دستور وضعیت تمام فایل‌های داخل مخزن پروژه بررسی شود و در خروجی نمایش داده شود.

دستور commit

```
neogit commit -m [commit message]
```

با اجرای این دستور تغییرات همه فایل‌های داخل وضعیت stage در مخزن پروژه commit می‌شوند. در صورتی که هیچ فایل در حالت stage نباشد، commit ساخته نخواهد شد.

نکات:

- در صورتی که پیغام commit دارای whitespace باشد، باید پیغام بین دو double qoute باشد.
- پیغام commit نباید بیشتر از ۷۲ کاراکتر باشد و در صورت بیشتر بودن خطای مناسب در خروجی چاپ شود.
- در صورتی که پیغام commit مشخص نشده بود، خطای مناسب در خروجی نمایش داده شود.
- به هر commit موفق یک id منحصر به فرد باید نسبت داده شود و در خروجی id ، زمان و پیغام commit در خروجی نمایش داده شود.

```
neogit set -m ”shortcut message” -s shortcut-name
```

با اجرای این دستور یک shortcut برای پیغام ایجاد می‌شود و می‌توان برای commit با آن پیغام از shortcut استفاده کرد.

در صورت اجرای دستور بالا، عملکرد دو دستور neogit commit -s shortcut- name و ”neogit commit -m ”shortcut message. یکسان خواهد بود

در صورتی که shortcut وجود نداشته باشد، خطای مناسب در خروجی نمایش داده شود.

```
neogit replace -m ”new shortcut message” -s shortcut-name
```

با اجرای این دستور محتوای پیغام shortcut تغییر می‌کند.

در صورتی که shortcut وجود نداشته باشد، خطای مناسب در خروجی نمایش داده شود.

```
neogit remove -s shortcut-name
```

با اجرای این دستور shortcut از قبل مشخص شده حذف خواهد شد.

در صورتی که shortcut وجود نداشته باشد، خطای مناسب در خروجی نمایش داده شود.

دستور log

```
neogit log
```

این دستور تمام commit های داخل مخزن گیت را به ترتیب تاریخ از جدید به قدیم نمایش می‌دهد. در هر log مربوط به یک کامیت باید موارد زیر وجود داشته باشد:

- تاریخ و ساعت commit
- پیغام commit
- شخصی که commit را انجام داده است
- id یا hash مربوط به commit
- branch که در آن commit انجام شده است.
- تعداد فایل‌هایی که commit شده‌اند

```
neogit log -n [number]
```

این دستور به تعداد مشخص شده commit آخر را نمایش می‌دهد.

```
neogit log -branch [branch-name]
```

این دستور تاریخچه commit هایی را نشان می‌دهد که در branch مشخص شده انجام شده اند.

در صورتی که branch-name وجود نداشته باشد، در خروجی خطای مناسب نمایش داده شود.

```
neogit log -author [author-name]
```

این دستور commit هایی را که نویسنده‌شان در ورودی داده شده است، در خروجی از جدیدی به قدیم نمایش می‌دهد.

```
neogit log -since <date>
```

```
neogit log -before <date>
```

این دو دستور به ترتیب تاریخچه commit ها را از تاریخ مشخص شده به بعد و از تاریخ مشخص شده به قبل از جدید به قدیم در خروجی نمایش می‌دهند.

```
neogit log -search <word>
```

این دستور تاریخچه تمام commit هایی که پیغامشان شامل کلمه ورودی است، در خروجی از جدید به قدیم نمایش می‌دهد.

موارد امتیازی:

- پشتیبانی جست و جو از حالت wildcard در یک کلمه.
- پشتیبانی جست و جو از چند کلمه(در صورتی که هر کدام از کلمات داخل پیغام بود، در خروجی نمایش داده شود.

دستور branch

```
neogit branch <branch-name>
```

این دستور از آخرین commit پروژه یک branch با نام مشخص شده می‌سازد.
در صورتی که branch با نام مشخص شده وجود داشته باشد، خطای مناسب در خروجی نمایش داده شود.

نکته: branch اصلی هر مخزن master نام دارد.

```
neogit branch
```

این دستور لیست تمام branchهای مخزن را نمایش می‌دهد.

دستور checkout

```
neogit checkout <branch-name>
```

این دستور وضعیت فایل‌های داخل پروژه را به آخرین commit از branch مشخص شده می‌برد.

در صورتی که تغییری commit نشده وجود داشته باشد، دستور نباید کار کند و در خروجی خطای مناسب نمایش داده شود.

```
neogit checkout <commit-id>
```

این دستور وضعیت فایل‌های داخل پروژه را به commit مشخص شده می‌برد.

نکات قابل توجه:

- در صورتی که تغییری commit نشده وجود داشته باشد، دستور نباید کار کند و در خروجی خطای مناسب نمایش داده شود.
- هنگامی که به یک commit در گذشته chekout می‌کنیم، نمی‌توانیم تغییر در فایل‌ها ایجاد کنم و آن‌ها را commit کنیم. به عبارت دیگر تنها محل مجاز commit کردن HEAD پروژه است.

```
neogit checkout HEAD
```

این دستور وضعیت فایل‌های داخل پروژه را به HEAD پروژه برمی‌گرداند.

امتیازی

```
neogit checkout HEAD-n
```

این دستور وضعیت فایل‌های داخل پروژه را به n امین commit قبل از HEAD می‌برد.