

به نام خدا



مهارت‌های پیشرفته کار با کامپیوتر (بهار ۱۴۰۴)

تمرین کامپیوتري ۲

مهلت ارسال: ۱۴۰۴/۰۲/۱۱

استاد درس: دکتر دوستی

دستیاران طراح: محمدرضا ولی - محمدعلی احمدی زارعی

بازبینی: علی خرمفر

قوانين و ملاحظات

نحوه ارسال تمرین:

- تمامی فایل‌ها باید در یک فایل فشرده با نام ECS-CA2-StudentID ارسال شوند.
- کدهای مربوط به هر بخش را با نام مناسب بر اساس جدول انتهای همین فایل ذخیره کرده و همراه گزارش ارسال کنید.
- تمامی کدهای ارسال شده باید امکان اجرای مجدد داشته باشند. اگر تنظیمات خاصی برای اجرا نیاز است، آن را ذکر کنید.
- کدهای ارسال شده باید توسط خودتان اجرا شده باشند و نتایج اجرا در فایل ارسالی مشخص باشد.

رعایت اصول آکادمیک و صداقت علمی:

- این تمرین باید به صورت **فردی** انجام شود. هر گونه همکاری یا نوشتن تمرین به صورت گروهی ممنوع است.
- در صورت مشاهده تشابه در پاسخ، تمامی افراد در گیر نمره صفر دریافت خواهند کرد و موضوع به استاد گزارش خواهد شد.

استفاده از ابزارهای هوش مصنوعی:

استفاده از ابزارهایی مانند ChatGPT، Gemini، Copilot و موارد مشابه مجاز است، اما تحت شرایط زیر:

- نحوه استفاده از این ابزارها را در گزارش خود توضیح دهید (ابزارهای استفاده شده، کاربردهای مشخص و موارد مرتبط).
- تمامی پرامپت‌ها و لینک‌های استفاده شده را در انتهای گزارش قرار دهید.
- عدم ارائه این اطلاعات به منزله سرقت علمی محسوب شده و منجر به نمره صفر خواهد شد.

مهلت ارسال و جریمه تأخیر:

- امکان ارسال تمرین با **تأخير تا ۲ روز** و به ازای **هر روز تأخير ۱۰ درصد جریمه** وجود دارد.
- تأخير به صورت ساعتی محاسبه شده و پس از دو روز تأخیر، تمرین پذیرفه نخواهد شد.

ارزیابی حضوری:

- ارزیابی تمرین به صورت حضوری انجام خواهد شد.
- محل ارزیابی: آزمایشگاه NLP، طبقه منفی یک، دانشکده مهندسی برق و کامپیوتر شماره ۲.

لطفاً پیش از شروع کار بر روی تمرین، به نکات زیر توجه فرمایید.

- حتماً ویدئوی راهاندازی کلاستر را به دقت مشاهده کنید و مطمئن شوید به کلاستر درس دسترسی دارید.
- آدرس‌های کلاسترها درس به شرح ذیل است :

dml0: 172.18.32.200

dml1: 172.18.32.201

dml2: 172.18.32.202

dml3: 172.18.32.203

- برای راحتی در توسعه و تست کد، از ماشین مجازی لینوکس خود استفاده نمایید تا ترافیک کلاستر (به خصوص در ساعت آخر مهلت تمرین) افزایش نیابد. پس از اطمینان از عملکرد کد، می‌توانید آن را روی کلاستر اجرا کنید.
- در صورت بروز مشکل با ایمیل‌های زیر در ارتباط باشید:

m.ahmadizarei@gmail.com : سوال ۱ و ۲ و ۳

mohammadrezavali78@gmail.com : سوال ۱ و ۳ و ۴ و ۵

سوال ۱. ادیتور VIM - ۲۰ نمره

اهمیت یادگیری و کار با ادیتور Vim

موقع کار با سرورهای عملیاتی خبری از محیط گرافیکی نیست و احتمالاً این جمله روشنیدین که «توی لینوکس همه چیز فایله!» پس خیلی مهمه که توی محیط کامندلاین هم، دست شما برای کار با فایل‌ها باز باشه. هدف این تمرین هم اینه که شما مهارت کار با یکی از کاربردی‌ترین ادیتورهای محیط CLI رو کسب کنین.

Vim یکی از ویرایشگرهای متین قدرتمند و سبکه که با یادگیری اون می‌تونین سرعت و کارایی خودتون رو تو ویرایش فایل‌های متنه افزایش بدین. تسلط بر میانبرهای Vim به شما این امکان رو می‌ده تا بدون نیاز به ماوس، به سرعت، متون رو ویرایش کنین و از امکانات پیشرفته‌ای مثل جستجو، کپی، پاک کردن و مدیریت پنجره‌ها بهره ببرین.

۱. توانایی کار با Vim

این قسمت از تمرین hands-on  هست و از مواردی مشابه اون چه که در ادامه اومده، هنگام تحويل حضوری از شما پرسیده می‌شه. هدف این تمرین اینه که شما در عمل از ادیتور Vim استفاده کنین.

در ادامه مواردی از کارهایی که می‌تونین انجام بدین رو توضیح دادیم. در گزارش کار برای هر قسمت تنها یک توضیح کوتاه از میانبرها و کلیدهایی که استفاده کردین بنویسین و نیازی به توضیح نیست؛ مثلاً برای قسمت کلیدهای جابجایی، تنها کافیه که بنویسین (hjkl) یا برای ذخیره فایل کافیه بنویسین (w). هدف این قسمت از تمرین اینه که شما بتونین با Vim کار کنین!

برای تمرین می‌تونین یک کپی از فایل کانفیگ ssh رو به مسیر دلخواه‌تون بیارین و با استفاده از ادیتور Vim مواردی که در ادامه گفته می‌شه رو تمرین کنین. دقت کنین که در هر بخش از سوئیچ‌ها، میانبرها یا دستوراتی استفاده کنین که مستقیماً همون کار رو انجام می‌دن. برای مثال اگر گفته شده «به ابتدای خط بروید و در حالت نوشتن قرار بگیرید» منظور این نیست که با کلیدهای جهت به ابتدای خط ببرید!

○ جابجایی در ادیتور Vim

کلیدهایی که برای جابجایی در Vim به کار می‌رن رو پیدا کنین و سعی کنین به جای استفاده از کلیدهای جهت، از اون‌ها بهره ببرین.

در یکی از خطوط قرار بگیرین و کرسر رو چهار کلمه به جلو ببرین، طوری که کرسر در ابتدای کلمه‌ی چهارم قرار بگیره.

در یکی از خطوط قرار بگیرین و کرسر رو چهار کلمه به جلو ببرین، طوری که کرسر در انتهای کلمه‌ی چهارم قرار بگیره، سپس چهار کلمه به عقب برگردین.

به خط اول فایل ببرین.

به خط آخر فایل ببرین.

به خط هجدهم فایل ببرین.

○ نوشتن و پاک کردن

به خط پانزدهم ببرین و با میانبر به انتهای خط در حالت نوشتن ببرین و شماره دانشجویی خودتون رو یادداشت کنین.

به خط دهم بروین و یکی از کلمات اون خط رو انتخاب کرده و به صورت حروف بزرگ (uppercase) تغییر بدین.

insert به ابتدای همون خط برگردین و دو کلمه رو پاک کنین؛ طوری که بلافاصله پس از پاک شدن، وارد حالت نوشتن (mode) بشین.

یک کلمه رو به دلخواه cut کنین و اون رو در ابتدای خط هشتم paste کنین.

شاید قبله دیده باشین که در نرم افزار Word در صورتی که تنظیمات انجام شده باشن با زدن دکمه insert بر روی کیبورد اصطلاحاً به overtype mode می‌رویم، یعنی از جایی که کرسر هست، روی حروف موجود می‌تونیم تایپ کنیم و هر حرفی که می‌زنیم حرف قبلی رو پاک کنه و جایگزین اون بشه. همین قابلیت رو در Vim هم داریم؛ با استفاده از کلید میانبر این کار رو انجام بدین.

به خط سوم ببرین و از ابتدای خط، سه کلمه به جلو حرکت کنین، سپس با استفاده از میانبر، از محل فعلی تا انتهای خط رو پاک کنین.

دو خط متوالی رو به دلخواه کپی کنین، سپس به انتهای فایل ببرین و اون‌ها رو سه بار پیست کنین.

دو کلمه‌ی متوالی رو به دلخواه کپی کنین، سپس به ابتدای فایل ببرین و اون‌ها رو پیست کنین. می‌توانیم از visual mode استفاده کنیم.

چند خط متوالی رو انتخاب کنین و اون‌ها رو کامنت کنین و بعد کامنټ‌ها رو حذف کنین. (ابتدا خطوط رو در Visual mode انتخاب کنین بعد به ابتدای اون‌ها کاراکتر # رو اضافه کنین و حذف کنین).

فرض کنین املای یک کلمه ایراد داره و نیازه که یکی از حروف اون تغییر کنه. بدون وارد شدن به Insert mode اون حرف رو جایگزین کنین.

○ سرچ کردن و اجرای کامند

یک کلمه رو در متن جستجو کنین و بین نتایج اون حرکت کنین. برای مثال در فایل کانفیگ ssh، کلمه‌ی key یا host رو سرچ کنین.

مجدداً همون کلمه رو جستجو کنین؛ به طوری که حروف بزرگ و کوچیک در نظر گرفته نشن.

دستوری رو وارد کنین که در کل فایل تمام جاهایی که کلمه مورد نظر نوشته شده رو با new_word جایگزین کنه. این جایگزینی باید به صورت case-insensitive انجام بشه و برای هر تغییر ابتدا از شما تایید گرفته بشه.

تنظیماتی را به Vim اضافه کنین که نتایج جستجو رو به صورت هایلات شده نمایش بده.

بدون خارج شدن از Vim یک کامند لینوکسی اجرا کنین. برای مثال موقع ویرایش کردن فایل کانفیگ، یک ابزار نیاز دارین که آی‌بی خودتون رو بدونین و می‌خواین توی ادیتور Vim دستور -c ip -br a رو اجرا کنین.

○ ذخیره، مقایسه و کار با window و tab

تغییراتی که توی فایل ایجاد کردین رو ذخیره کنین.

شش خط متوالی از فایل رو کپی کنین و سپس توی یک تب جدید Vim، اون‌ها رو پیست کنین. فایل جدید رو با نام newtab ذخیره کرده و از اون خارج بشین.

یک پنجره‌ی جدید در تب اول ایجاد کنین و پانزده خط ابتدایی فایل رو کپی و در اون پیست کنین.

به پنجره‌ی جدید بربین، چند کلمه به دلخواه پاک کنین و چند کلمه رو تغییر بدین، سپس فایل رو با نام for-diff ذخیره کنین و از Vim خارج شین.

فایل اصلی و فایل for-diff رو با استفاده از دستور vimdiff مقایسه کنین و خطوط متفاوت بین دو فایل رو با استفاده از دستور diffput و diffget انتقال بدین.

۲. کانفیگ Vim

فرض کنین شما قصد دارین Vim رو به عنوان ویرایشگر اصلی توسعه خودتون، راهاندازی کنین تا Productivity خودتون رو به حداکثر برسونین. تو این سوال می‌خوایم یک پیکربندی سفارشی ایجاد کنیم که نه تنها Usability رو افزایش می‌ده بلکه منجر به انجام سریع‌تر کارهای متدالو هم می‌شه. همچنین در انتهای این بخش با یکی از افزونه‌های Vim به نام NERDTree آشنا می‌شین. در قسمت سرچ، هایلات رو اضافه کردین؛ اما اگر از Vim خارج بشین و فایل دیگه‌ای رو باز کنین، این کانفیگ پاک می‌شه. برای اینکه به صورت دائم تغییر ایجاد کنیم، بایستی که فایل کانفیگ Vim رو تغییر بدیم.

○ ایجاد فایل **.vimrc**

ابتدا لازمه یک فایل با نام `~/.vimrc` ایجاد کنیم. این فایل شامل تمام پیکربندی‌های سفارشی شما خواهد بود و به شما اجازه می‌ده **Options** تعیین کنیم، **Custom behaviors** تعریف کنیم و **Shortcut** هایی ایجاد کنیم که به صورت خودکار با هر بار اجرای Vim بارگذاری می‌شون.

○ تعریف کلیدهای میانبر سفارشی (**Key mapping**)

Key mapping به فرآیند اختصاص دادن فرمان‌ها یا عملیات‌های مشخص به توالی‌های کلیدی اشاره دارد و این امکان رو به شما می‌ده تا **Shortcut** های سفارشی بسازین تا کارهای تکراری سریع‌تر انجام بشن یا روند کار شما بهبود پیدا کنه.

(الف) یک کلید میانبر تعریف کنیم که فایل رو با یک دستور ذخیره کنه. به عنوان مثال، می‌تونیں کلید `w <leader>` رو برای ذخیره‌ی فایل تنظیم کنین.

```
nnoremap <leader>w :w<CR>
```

(ب) Window splitting به تقسیم رابط کاربری Vim به بخش‌های متعدد اشاره دارد. به طوری که می‌تونیں به‌طور همزمان با چندین فایل یا بخش‌های مختلف یک فایل کار کنین. کلیدهایی برای تقسیم‌های عمودی و افقی پنجره‌ها تعریف کنین.

“ **Vertical split (open new window on the right)**

```
nnoremap <leader>v :vsp<CR>
```

“ **Horizontal split (open new window below)**

```
nnoremap <leader>h :sp<CR>
```

○ کانفیگ کردن ادیتور

- کانفیگ مربوط به نمایش شماره‌ی خطوط رو به `vimrc`. اضافه کنین.
- کانفیگ مربوط به نادیده گرفتن حروف بزرگ و کوچک در جستجو رو اضافه کنین.
- کانفیگ مربوط به هایلایت در جستجو رو اضافه کنین.
- سایز `tab` رو در ویرایشگر معادل چهار فاصله تنظیم کنین.
- قابلیت `Syntax highlighting` رو فعال کرده و یک `Color scheme` رو به دلخواه تنظیم کنین.

۳. افزودن پلاگین به Vim

(الف) Vim-`plugin manager` یک سبک و کارآمد برای Vim هست که فرآیند نصب، به‌روزرسانی و مدیریت افزونه‌ها رو ساده می‌کنه. به صفحه‌ی [این پلاگین](#) بین و با توجه به توضیحات `readme` اون رو نصب کنین. سپس پلاگین [NERDTree](#) رو نصب کرده و درمورد نحوه‌ی کار با اون توضیح بدین.

ب) یک Key mapping برای Vim در Toggle File explorer تعریف کنین. NERDTree یک افزونه‌ی محبوب برای File explorer هست که یک File explorer به صورت درختی ارائه می‌دهد.

```
nnoremap <leader>n :NERDTreeToggle<CR>
```

ج) یک پلاگین دیگه به دلخواه به Vim اضافه کنین و در مورد هر کدام، عملکرد و نحوه استفاده از آنها را توضیح بدین. سعی کنین پلاگین‌هایی را انتخاب کنین که از نظر شما کاربردی باشند. برای این قسمت می‌توانید از پلاگین‌هایی که تو [این لینک](#) معرفی شده‌اند استفاده کنید.

۴. استفاده از ماکروها در Vim (امتیازی)

متن زیر را در یک فایل قرار بدین.

```
Ferdowsi,940-1020,Shahnameh  
Omar Khayyam,1048-1131,Rubaiyat  
Nezami Ganjavi,1141-1209,Khamsa  
Attar,1145-1220,Conference of the Birds  
Rumi,1207-1273,Masnavi  
Saadi,1210-1292,Gulistan and Bustan  
Hafez,1315-1390,Divan of Hafez
```

سپس یک ماکرو با نام t بنویسید که با استفاده از اون بشه یک خط متن را به شکل مرتب زیر در آورد.

```
Poet: Ferdowsi | Years: 940-1020 | Famous Work: Shahnameh  
Poet: Omar Khayyam | Years: 1048-1131 | Famous Work: Rubaiyat  
Poet: Nezami Ganjavi | Years: 1141-1209 | Famous Work: Khamsa  
Poet: Attar | Years: 1145-1220 | Famous Work: Conference of the Birds  
Poet: Rumi | Years: 1207-1273 | Famous Work: Masnavi  
Poet: Saadi | Years: 1210-1292 | Famous Work: Gulistan and Bustan  
Poet: Hafez | Years: 1315-1390 | Famous Work: Divan of Hafez
```

برای این کار می‌توانید از دستور زیر که به نوعی زدن کامند sed در ادیتور Vim هست استفاده کنید. این دستور بخش‌های مختلف متن اول را بر اساس ویرگول جدا می‌کند و اونها را در ساختار جدید می‌چیند.

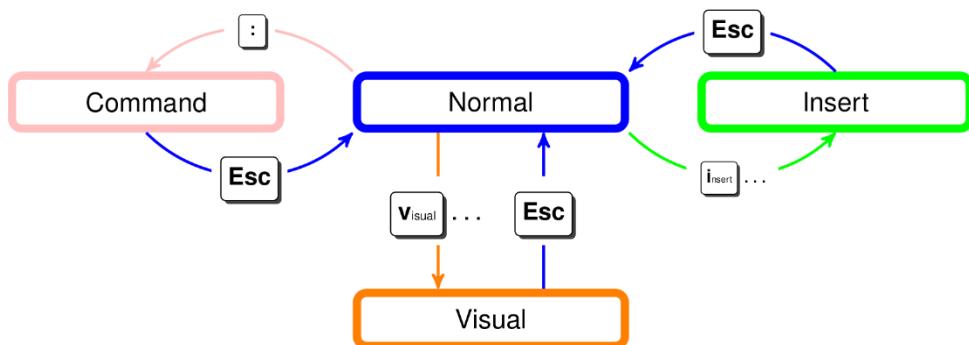
```
:s/\(^,[^,]*\),\(^,[^,]*\),\(.*/Poet: \1 | Years: \2 | Famous Work: \3/
```

حال با استفاده از دستور زیر، ماکرو را روی تمام خطوط فایل اجرا کنید.

```
:%norm! @t
```

۵. سوالات تشریحی (امتیازی)

- ۱) درباره‌ی اهمیت استفاده از یک افرونه File explorer مانند NERDTree در مدیریت فایل‌های پروژه توضیح بدین.
- ۲) درباره‌ی Trade-off میان Performance و Functionality در vimrc configuration توضیح بدین.
- ۳) Nano یکی دیگر از Text editor هایی که هر چند یادگیری اون در مقایسه با Vim ساده‌تره، ولی این سادگی تا حد قابل توجهی منجر به سلب آزادی عمل از کاربر می‌شود. فرض کنیم همکاری به نام نقی دارین که سال‌های زیادیه که به عنوان یک Sysadmin فعالیت می‌کنه و همیشه نیاز داره که از یک Text editor استفاده کنه ولی به دلیل تعصب بر روی Nano از استفاده از سایر Text editor ها امتناع می‌کنه. نقی رو با نوشتمن یک نامه با ذکر دلایل منطقی قانع کنین که Vim در بسیاری از سناریوها می‌توانه انتخاب بهتری نسبت به Nano باشه.
- ۴) مزایای استفاده از ماکرو در Vim و Text editing را شرح بدین
- ۵) چه مشکلاتی ممکن است حین Recording و Replaying ماکروها ایجاد بشن؟ آیا در انجام مراحل این سوال به مشکلی در این قسمت‌ها برخورد کردین؟
- ۶) طبق یک دسته‌بندی، Editor ها به دو دسته‌ی Modal و Modeless تقسیم می‌شن. درباره‌ی تفاوت اون‌ها توضیح بدین و بیان کنین Vim در کدام دسته‌بندی قرار می‌گیرن.
- ۷) درباره تفاوت Mode های Vim طبق تصویر زیر توضیح بدین.



سوال ۲. ابزار مدیریت منبع کد و کنترل ورژن (MERCURIAL & GIT) – نمره ۲۰

اهمیت یادگیری و کار Git

Git یک سیستم کنترل نسخه‌ی توزیع شده (Distributed Version Control System) هست که به توسعه‌دهنده‌ها کمک می‌کنه تا تغییرات کد رو را دیابی کنن و به صورت گروهی روی یک پروژه کار کنن. بدون اینکه تداخلی ایجاد بشه.

از سال ۲۰۰۲ توسعه دهنگان هسته لینوکس از ابزار BitKeeper برای مدیریت منبع کد پروژه استفاده می‌کردند. اما در سال ۲۰۰۵ این پروژه از حالت متن باز خارج شد و لینوس توروالدز (خالق لینوکس) تصمیم گرفت سیستم کنترل نسخه‌ی برای توسعه کرنل لینوکس به صورت متن باز بسازه، به این شکل یکی از محبوب‌ترین ابزارهایی که امروزه در مدیریت منبع کدهای پروژه استفاده می‌شه، شکل گرفت. در این تمرین سعی می‌کنیم بیشتر با ابزار Git و قابلیت‌هایی که به ما میده آشنا شیم. در هر قسمت پاسخ سوالات رو به اختصار توضیح بدین و دستوراتی رو که استفاده کردین در گزارش کار تمرین بیارین.

۱. تو پروژه چه خبره!

گیت به ما کمک می‌کنه که تمام تغییرات روی پروژه رو بتونیم رداپی کنیم و ببینیم چه افرادی چه کارهایی انجام دادن! سوالات این قسمت رو می‌تونین با استفاده از دستور log حل کنین.

[منبع کد ابزار گیت](#) رو clone کنین و سعی کنین به کمک دستور WC تعداد کامیت‌های این پروژه رو تا الان بشمرین. آفای لینوس توروالدز توی پنج سال اخیر چندتا کامیت روی این پروژه زده؟ از شانزده سال پیش به قبل ایشون چند کامیت روی این پروژه زدن؟

به شکل مشابه سری به [منبع کد کرنل لینوکس](#) بزنین. توی این پروژه فایلی با نام c در دایرکتوری kernel وجود داره. در مورد تعداد کامیت‌های این فایل و زمانی که این فایل به پروژه اضافه شده تحقیق کنین. با توجه به اینکه حجم پروژه برای اینکه روی لپتاپتون این مورد رو بررسی کنین، زیاد هست، تنها توضیح کامند در این قسمت کافیه و نیازی به clone کردن پروژه نیست.

۲. کانفیگ Git

یک دایرکتوری با نام yourName-studentId ایجاد کنین و با استفاده از دستور git init گیت رو در این دایرکتوری فعال کنین. سپس نام و ایمیل خودتون رو به صورت local در کانفیگ گیت اضافه کنین.

تفاوت کانفیگ گیت به صورت local، global و system رو توضیح بدین.

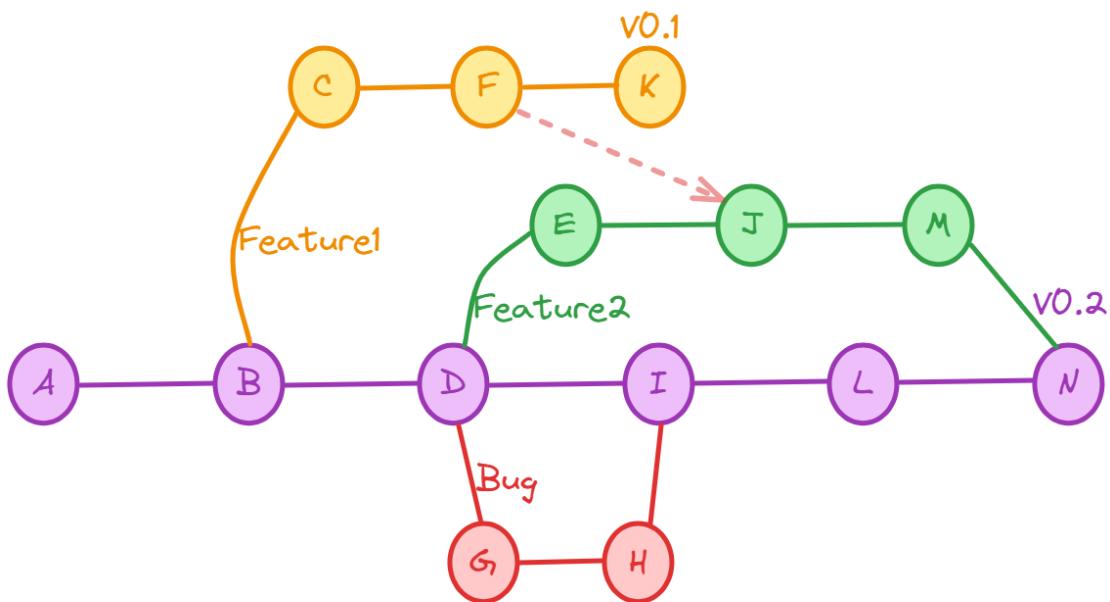
به کانفیگ گیت دو alias برای دستور git log --oneline --all --graph و دستور git bisect bad به صورت global اضافه کنین.

۳. سناریو بازی!

در این قسمت می‌خوایم اتفاقاتی که ممکن است در یک پروژه رخ بده رو تمرین کنیم.

دقیق کنین که ایجاد یک فایل جدید، یا اضافه کردن یک خط پرینت ساده به یکی از فایل‌ها یا هر تغییر کوچیک دیگه‌ای در این قسمت برای ایجاد یک کامیت جدید کافیه و هدف این تمرین کار کردن با گیت است.

در این سناریو کامیت‌ها رو به ترتیب حروف الفبای انگلیسی به پروژه اضافه می‌کنیم. با توجه به تصویر زیر برای هر کامیت ابتدا در مسیح کامیت، نام کامیت رو وارد کنیں و سپس پیام خودتون رو اضافه کنیں. برای مثال مسیح کامیت اول می‌تونه به صورت A: باشه یا کامیت نهم I: Bug fixed باشد.



در کامیت دوم B تضمیم می‌گیریم که با اضافه کردن یک فیچر به پروژه ورژن اولیه‌ی اون رو ایجاد کنیم. برنچی با نام Feature1 رو به پروژه اضافه کنین و یک فایل برای این فیچر در کامیت C اضافه کنین.

سیسیز، یه پرنج اصلی پرگ دین تا وند رو ادامه بدم: کامیت D و سیسیز یک پرنج پایی Feature2 اضافه کنید.

حالا فرض کنیم که متوجه اشتباهی در مسیج کامیت C می‌شیم. بنابراین به برنج Feature1 بین مسیج این کامیت رو با استفاده از git commit --amend تغییر دهد.

کامیت F ایه اشتیاه در ادامه‌ی پر نج Feature1 اضافه کنیز. (فرض کنیز فراموش کردین که به پر نج Feature2 پر گردیدن!)

حالا به برنج اصلی برین و برای اصلاح باگی که متوجه اون شدین، یک برنج ایجاد کنین و بعد از فیکس کردن باگ، اون رو مرج کنین: در اینجا توضیح بدین: که مر ج انجام شده از نوع fast forward هست ما نه و در مو د انواع مر ج به اختصار، توضیح بدین.

سپس با استفاده از قابلیت cherry-pick در گیت، کامیت F را به برنج درست منتقل کنیم و سپس به کمک git reset کامیت رو حذف کنیم. توضیح بدین که از کدوم حالت git reset استفاده می‌کنیم و به اختصار در مورد تفاوت soft و hard کامیت رو حذف کنیم.

به کامیت I برگردین تا توسعه‌ی برنج اصلی رو ادامه بدیم. یک فایل جدید به پروژه اضافه کنین و چهار خط به این فایل اضافه کنین. فرض کنین که تصمیم دارین دو خط اول رو در کامیت L به پروژه اضافه کنین اما در مورد دو خط دیگه مطمئن نیستید و می‌خواید در working directory اون‌ها رو نگه دارین. بدون حذف کردن این خطوط، کامیت L رو به پروژه اضافه کنین. در این قسمت از patch استفاده کنین.

حالا به برنج Feature2 بین. برای کامل کردن این فیچر اما، قبل از رفتن، فرض کنین همچنان می‌خواین دو خط مرحله‌ی قبل رو نگه دارین تا بعدا در مورد اون‌ها تصمیم بگیرین. در این قسمت از stash استفاده کنین و در مورد تفاوت pop و apply در git stash توضیح بدین.

پس از اضافه کردن کامیت M و مرج کردن برنج Feature2 با برنج اصلی، روی کامیت N تگ v0.2 رو بزنین.

نهایتا در مرحله‌ی آخر، یک ریپازیتوری در اکانت گیت‌هاب خودتون با نام ECS-Git-Practice ایجاد کنین و اون رو به عنوان ریپازیتوری ریموت به پروژه اضافه کنین و برنج‌ها و تگ‌های مختلف پروژه رو به ریپازیتوری ریموت اضافه کنین.

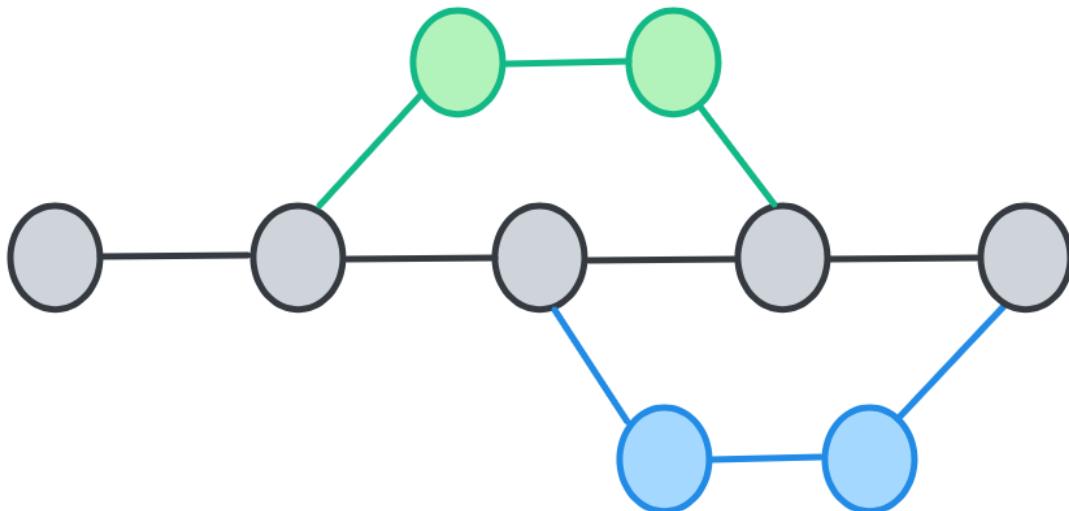
۴. آشنایی با Mercurial

ابتدا با استفاده از دستورات زیر این ابزار رو نصب کنین.

```
sudo apt update
sudo apt install mercurial
hg -version
```

حال به کمک دستورات زیر ساختاری مشابه تصویر زیر رو در mercurial ایجاد کنین. (هر دایره یک کامیته.)

```
hg init, hg add, hg commit, hg branch, hg update
```



در مورد ابزارها و دستوراتی که یک گراف از کامیت‌های شما در mercurial نمایش می‌دان مثل hg log --graph تحقیق کنین و سعی کنین که گراف پروژه رو بکشین. دقیقاً برای ایجاد کامیت، مشابه قسمت قبل، اضافه کردن یک فایل ساده کافیه.

۵. سوالات تشریحی (امتیازی)

- ۱) در گیت، دو تا ابزار `bisect` و `blame` رو برای خطایابی و دیباگ داریم. یک سناریوی عملی کوچیک برای استفاده از این ابزارها پیاده‌سازی کنین و کامندهایی که استفاده کردن رو در گزارش کار توضیح بدین.
- ۲) در مورد `postmortem blameless` به اختصار این مفهوم رو توضیح بدین.
- ۳) در مورد `rebase` کردن در گیت به اختصار توضیح بدین. تفاوت مرج کردن به صورت `rebase` با حالت دیگه چیه؟ در یک محیط عملیاتی که افراد در پروژه انجام می‌دن، حساس و مهمه، توصیه می‌کنین که از `rebase` استفاده بشه یا نه؟ علت توصیه خودتون رو توضیح بدین.
- ۴) در مورد تفاوت `git pull` و `git fetch` به اختصار توضیح بدین.
- ۵) در مورد تفاوت‌های اصلی `git` و `mercurial` توضیح بدین آیا می‌شه با `mercurial` به ابزاری مثل `Github` یا `Gitlab` متصل شد و با یک ریپازیتوری ریموت کار کرد؟

سوال ۳. مدیریت سیگنال‌ها در لینوکس: کنترل فرآیندها و بازپس‌گیری ترمینال - ۱۰ نمره

شهین خانم که به تازگی با به عرصه برنامه نویسی گذاشته، برخلاف توصیه علمای غرب و شرق سراغ پایتون و هوش مصنوعی نرفته و مسیر قدیمی ذهنی خودش یعنی "از بیخ یاد گرفتن" رو سرلوحه کار قرار داده. به همین منظور رفته دنبال سیستم کال (system call) و کد زدن‌های سطح پایین و کرنل و بساط !!!

ولی واسه شروع که نمیشه قدم بلند برداشت به همین خاطر بهش گفتن علی الحساب با سیگنال‌ها توی لینوکس کار کن،
شهین خانم هم گفته باشه و کد زیر رو زده:

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

void handle_sigint(int sig) {
    printf("\nCtrl+C detected! (signal %d)\n", sig);
}

int main() {
    signal(SIGINT, handle_sigint); // Handle Ctrl+C

    while (1) {
        printf("Working... Press Ctrl+C to send SIGINT\n");
        sleep(2);
    }

    return 0;
}
```

اما بعد از کامپایل کردن و اجرای کد، برای خروج از برنامه به مشکل خورده و نمیتواند با استفاده از Ctrl+C برنامه متوقف کند.
این اتفاق باعث شد تا ایده‌ای در ذهن شهین خانم برای کمک به برادرش که اخیراً پا به عرصه‌ی تولید ویدیوهای موسوم به prank گذاشته بزند تا بلکه سهمی هم از درآمد برادرش به او برسد.

۱. به شهین خانم کمک کنید که با استفاده از دستورات کامن‌دلاین در همان terminal که برنامه در حال اجرا هست، shell را پس بگیرد و پراسس مربوط به برنامه را terminate کند.

۲. در گام بعدی به ایده شهین خانم جامه عمل پیوشناید و کد را به شکلی تغییر دهید که به کمک بقیه کلیدها هم نتوان از برنامه خارج شد و در مورد هر سیگنال توضیح دهید.

Ctrl+c, Ctrl+z, Ctrl+\, ...

و توضیح دهید که در این شرایط راه پس گرفتن ترمینال چیست؟

۳. سوال را اینبار با استفاده از `sigaction` به جای `signal` حل کنید و در مورد تفاوت این دو مورد توضیح دهید. (امتیازی)

به عنوان مقدمه‌ای بر برنامه‌نویسی کرنل، در این سوال قصد داریم یک ماژول کرنل لینوکس بنویسیم که هنگام Insert و Remove از کرنل، پیام‌هایی را در لاغ کرنل ثبت کند. این تمرین شما را با مفاهیم اولیه برنامه‌نویسی کرنل و مدیریت ماژول آشنا می‌کند.

۱. نوشتن ماژول کرنل

- الف) یک فایل جدید با نام hello_kernel.c در دایرکتوری خود ایجاد کنید.
- ب) در این فایل، برنامه‌ای به زبان C بنویسید که شامل موارد زیر باشد:
 - یک تابع Initialization با استفاده از ماکروی `__init` که با استفاده از تابع `printf` پیام زیر را چاپ کند:
"Hello, Kernel!"
 - یک تابع `exit` با استفاده از ماکروی `__exit` که با استفاده از `printf` پیام زیر را چاپ کند:
"Goodbye, Kernel!"
 - اطمینان حاصل کنید که هدرهای زیر را اضافه کرده‌اید.
`<linux/init.h>`, `<linux/kernel.h>`, `<linux/module.h>`
 - از ماکروهای زیر برای افزودن اطلاعات مربوط به توضیحات، مجوز و نویسنده (Metadata) استفاده کنید.
`MODULE_AUTHOR`, `MODULE_LICENSE`, `MODULE_DESCRIPTION`

۲. ایجاد Makefile

در همان دایرکتوری یک فایل به نام `Makefile` ایجاد کرده و قوانین لازم برای کامپایل ماژول با استفاده از سیستم ساخت کرنل لینوکس را در آن قرار دهید. برای انجام این کار می‌توانید از این مثال الگو برداری کنید.

```

ifneq ($(KERNELRELEASE),)
obj-m := hello_kernel.o
else
KERNEL_DIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)
all:
$(MAKE) -C $(KERNEL_DIR) M=$(PWD) modules
clean:
$(MAKE) -C $(KERNEL_DIR) M=$(PWD) clean
endif

```

۳. فرایند کامپایل

ابتدا یک ترمینال باز کنید و به دایرکتوری حاوی فایل‌های Makefile و hello_kernel.c بروید؛ سپس دستور زیر را اجرا کنید تا فایل hello_kernel.ko ایجاد شود.

`make`

۴. Remove و Insert مژول

ابتدا از دستور زیر برای Insert کردن مژول به کرنل استفاده کنید و سپس با دستور dmesg لاغ کرنل را چک کنید.

`sudo insmod hello_kernel.ko`

اکنون از دستور زیر برای Remove کردن مژول از کرنل استفاده کنید و سپس مجدداً با دستور dmesg لاغ کرنل را چک کنید.

`sudo rmmod hello_kernel.ko`

۵. سوالات تشریحی

- ۱) درباره تفاوت میان Microkernel و Monolithic kernel به اختصار توضیح دهید.
- ۲) درباره Kernel panic تحقیق کرده و بیان کنید چگونه آن را هندل می‌کند.
- ۳) برای Debug کردن یک مژول کرنل یا درایور از چه ابزارهایی می‌توان استفاده کرد؟ درباره یکی از آن‌ها به اختصار توضیح دهید.

سوال ۵ - CHARACTER DEVICE DRIVER

Device driver ها به صورت LKM (Loadable Kernel Modules) ساخته می‌شوند که می‌توانند در زمان اجرا به کرنل اضافه یا از آن حذف شوند. برای شناسایی یک Device خاص، لینوکس به آن شماره‌های major و minor اختصاص می‌دهد. شماره Device major را نشان می‌دهد و شماره minor را شناسایی می‌کند. درایور دستگاه تماس‌های سیستمی را از کاربران دریافت کرده و سپس به طور مناسب دستگاه را مدیریت می‌کند که به عنوان یک فایل نمایش داده می‌شود. از این رو، درایور دستگاه باید توابع سیستمی مربوط به فایل‌ها مانند open، read، write، close، mmap، lseek وغیره را پیاده‌سازی کند. این عملیات در فیلدهای ساختار struct file_operations تعریف شده‌اند.

۱. صورت مسئله

- در این تمرین می‌خواهیم یک Device driver کاراکتری برای یک Virtual LIFO (Last In, First Out) که اندازه آن نامحدود است (می‌تواند بسیار بزرگ باشد، مثلاً محدود به ۱ مگابایت) پیاده‌سازی کنیم.
- شما باید دو Device driver مجازی LIFO ایجاد کنید که از یک Device استفاده می‌کنند. یکی از این Device ها باید فقط برای خواندن (read-only) باشد و دیگری فقط برای نوشتن (write-only).
- کاراکترهایی که به دستگاه write-only نوشته می‌شوند، تنها از دستگاه read-only قابل خواندن هستند.
- اولین کاراکتری که از دستگاه read-only خوانده می‌شود، متناظر با آخرین کاراکتری است که به دستگاه write-only نوشته شده است.
- خواندن از یک دستگاه LIFO خالی باید منجر به خروجی EOF شود. در غیر این صورت، فرایندی که عملیات خواندن را فراخوانی می‌کند، باید مسدود (blocked) شود و پس از در دسترس قرار گرفتن کاراکترها، این فرایند بیدار شود.

۲. شبه کد

شبه کد Reader به صورت زیر است:

```
int main (int argc , char * argv [ ]){  
    char device [ MAX_BUF_SIZE ];  
    char user_msg [ MAX_BUF_SIZE ];  
    strcpy ( device , argv [1]);  
    fd = open ( device , O_RDONLY );  
    memset ( user_msg , 0 , sizeof ( user_msg ) );  
    ret = read (fd , user_msg , MAX_BUF_SIZE );  
    close (fd);  
}
```

همچنین شبه کد Writer در ادامه ارائه شده است:

```

int main (int argc , char * argv [ ]){
char device [ MAX_BUF_SIZE ];
char user_msg [ MAX_BUF_SIZE ];
strcpy ( device , argv [1]);
strcpy ( user_msg , argv [2]);
fd = open ( device , O_WRONLY );
memset ( user_msg , 0 , sizeof ( user_msg ) );
ret = write (fd , user_msg , strlen ( user_msg ) );
close (fd);
}

```

۳. صحت عملکرد

حال لازم است یک اسکریپت ساده بنویسید که از طریق تعامل با درایور آن را Insert و Remove کرده و صحت عملکرد آن را (طبق صورت مسئله) بررسی می‌کند.

۴. سوالات تشریحی

- ۱) در این تمرین با Character device driver آشنا شدید. درباره انواع دیگر Device driver ها تحقیق کنید و دلیل این تنوع را به اختصار شرح دهید.
- ۲) در این تمرین به صورت کلی با مفهوم Major number و Minor number آشنا شدید. درباره اهمیت آن‌ها به اختصار توضیح دهید.

نحوه تحويل تمرین کامپیووتری ۲

فایلها را به صورت زیر نامگذاری کرده و همه را در یک فایل zip در سامانه ارسال کنید.

نام فایل‌ها	بخش	سوال
تمامی موارد مربوط به این قسمت را در یک پوشه با نام P1 قرار دهید: - فایل .vimrc	تمام بخش‌ها	۱
- گزارش قدم به قدم انجام تمرین به همراه Screenshot‌ها، شرح دستورات و پاسخ PDF سوالات تشریحی در قالب یک فایل	تمام بخش‌ها	۲
تمامی موارد مربوط به این قسمت را در یک پوشه با نام P2 قرار دهید: - گزارش قدم به قدم انجام تمرین به همراه Screenshot‌ها، شرح دستورات و پاسخ PDF سوالات تشریحی در قالب یک فایل	تمام بخش‌ها	۳
تمامی موارد مربوط به این قسمت را در یک پوشه با نام P3 قرار دهید. - فایل‌های hello_kernel.c و hello_kernel.ko و Makefile	تمام بخش‌ها	۴
- گزارش قدم به قدم انجام تمرین به همراه Screenshot‌ها، شرح دستورات، پاسخ PDF سوالات تشریحی و باگ‌ها (در صورت وجود) در قالب یک فایل	تمام بخش‌ها	۵
تمامی موارد مربوط به این قسمت را در یک پوشه با نام P5 قرار دهید: - فایل‌های lifo_driver.c (به همراه Header در صورت نیاز) و test_lifo.sh	تمام بخش‌ها	
- گزارش قدم به قدم انجام تمرین به همراه Screenshot‌ها، شرح دستورات، پاسخ PDF سوالات تشریحی و باگ‌ها (در صورت وجود) در قالب یک فایل		