計算機科学実験及演習3 ハードウェア「Gitの使い方」

京都大学情報学科計算機科学コース



実験3ハードウェアでは、バージョン管理と設計デー タ・課題提出にGitHubを用います。

① <u>Gitの概要</u>

バージョン管理システムとGitの基本概念を紹介

② <u>Gitチュートリアル</u>
 Gitの基本的使い方を簡単な例を用いて説明

③ <u>リモートリポジトリ・GitHubの概要</u> 本実験で使用するリモートリポジトリGitHubの紹介

④ <u>GitHubチュートリアル</u> GitHubの使い方を説明

⑤ <u>GitHub Classroom</u> GitHub Classroomの説明



PCでの作業で起こりうること

- エラーを含んだ状態でプログラムを保存してしまった。
- レポートのファイルを間違って削除してしまった
- 1週間ぶりに開くファイル、以前どんな編集をしたのか忘れてしまった。
- 二人で1つのファイルを編集してしまい、一方の人がした 編集が反映されなかった。



Gitによるバージョン管理

- Gitは分散型のバージョン管理システム。
- Linuxのソースコードを管理するためにLinus Torvalds自 身が開発。
- Gitの機能
 - ファイルの変更履歴を保存する。
 - 履歴には、いつ、誰が、どんな変更を行ったかを記録で きる。
 - いつでもファイルを保存した履歴の状態に戻せる。
 - 他人が編集したファイルを上書きしようとすると警告を 出す。

commit

- Gitでファイルの変更履歴をつけていくことをcommitという。
- commitには、いつ、誰が、どんな変更を行ったかを記録 しておける。
- いつでも、任意のcommitに戻ることができる。



リポジトリ

- commitはリポジトリと呼ばれる場所に保存されていく。
- commitするたびに、その時のファイルの状態が保存されて いく。

	\mathbf{i}
(リポジトリ	
コミット3	
コミット2	
コミット1	

Gitの概要

ディレクトリをGitの管理下に置く(初期化する)とワーキングディレクトリ、ステージングエリア、リポジトリの3つの場所が作られる。



● Gitで管理している、実際の作業を行うディレクトリのこと
 をワーキングディレクトリという。



- 変更履歴として保存するファイルを選択し、置いておく場 所をステージングエリアという。
- コミットの前に、ワーキングディレクトリのファイルの中から履歴として保存したいファイルを選びステージングする。

Direct	tory			
	ワーキング ディレクトリ	ステージング エリア	リポジトリ	
		7711LA		
	7 7 1 MB			

Gitの概要

- commitをすると、ステージングエリアに置かれている ファイルの変更履歴が、リポジトリに保存される。
- ステージングエリアがあることで、ワーキングディレクト リの中の保存しておきたいファイルだけを選んで、 commitすることができる。



branch

- 変更履歴の流れを分岐することで並行での編集を支援する 機能をbranchという。
- あるブランチへの変更は、他のbranchに影響しない。
- ブランチ同士の統合 (merge) も可能。



2Gitチュートリアル

Gitの利用

- Windowsの場合
 - Git for Windows をインストールする
 - WSL (Windows Subsystem for Linux) を利用する
 - Cygwin などのLinuxライクな環境を用意する互換レイヤーをインストールする
- Mac の場合
 - Xcode Command Line Tools をインストールする
 - 「git -version」などとした際に未インストールならインストール面面へ
- Linux の場合
 - 「sudo apt install git-all」など

Gitチュートリアル

Gitの初期設定

\$ git config --global user.name "JOHO KYODAI"
\$ git config --global user.email "foo@bar.com"

- Gitの操作はターミナル上で行う。
- git config --globalコマンドによりユーザー名とメールアドレスを登録する。
- 登録したユーザーがcommitの際に、記録される

Gitリポジトリを初期化する

git init

現在のディレクトリをGitリポジトリとして初期化する。

Gitリポジトリを初期化する

```
$ mkdir gittest
$ cd gittest
$ git init
$ git init
Initialized empty Git repository in <ディレクトリ>
/gittest/.git
```

- gittestディレクトリがGitリポジトリとして認識される。
- ディレクトリには.gitディレクトリが作成されている。



ディレクトリ内にテキストファイルを作成する。

\$ echo "bow wow" >> dog.txt



ワーキングディレクトリにdog.txtが追加される。

現在のディレクトリ内の状態を確認する

git status

ディレクトリ内の状態を確認する。

```
Gitチュートリアル
```

現在のディレクトリ内の状態を確認する

- 新しく作成したファイル(dog.txt)がUntracked files:の一覧に表示される。
- Untracked filesには一度もステージングしたことがないファイルが表示される

Gitチュートリアル

指定したファイルをステージングエリアへ追加する

git add <ファイル名>

<ファイル名>をステージングエリアに追加する。

指定したファイルをステージングエリアへ追加する

\$ git add dog.txt

• dog.txtがステージングエリアに追加される。



ステージングエリア追加後の状態を確認する

```
$ git status
...
Changes to be commited:
(Use "git rm -cached <file>..." to unstage)
new file: dog.txt
```

- ステージング後に、再びgit statusコマンドにより状態を確認する。
- Changes to be committed:の一覧にdog.txtがnew fileとして表示されている。ス テージングエリアに新たに加えられたことを示している。

Gitチュートリアル

ステージングされたファイルをcommitする。

git commit -m $< \exists \times \rangle h >$

ステージングされているファイルをcommitする。 -mオプションをつけると1行のコメントをつけることが できる。

ステージングされたファイルをcommitする。

\$ git commit -m "create dog"
[master (root-commit) 2c2adf7] create dog
1 file changed, 1 insertion(+)

- create dogというコメントをつけてcommitする。
- ステージングされていたdog.txtがcommitされる。



dog.txtの変更と、新規ファイルの追加。

- \$ echo "mew" >> cat.txt
- \$ echo "wan wan" >> dog.txt
- dog.txtを編集し、新たにcat.txtを追加する。



dog.txtの変更と、新規ファイルの追加。

```
$ git status
On branch main
Changes not staged for commit:
...
modified: dog.txt
Untracked files:
...
cat.txt
```

- Changes not staged for commitには過去にステージングしたことがあり、 かつ最新のcommitから変更されているファイルが表示される。
- Untracked filesにはcat.txtが表示される。

Gitチュートリアル

dog.txtの変更と、新規ファイルの追加。



全ての変更をコミット。

\$ git add *.txt
\$ git commit -m "commit all files"

- dog.txtの変更とcat.txtの追加が一つの変更としてコミットされる。
- git addのファイル指定ではワイルドカードが使える。
- "."を指定すると変更があった全てのファイルをステージングする。



Gitチュートリアル

commitの履歴を確認する

git log

リポジトリの保存されているcommitの履歴を確認する。

```
Gitチュートリアル
```

commitの履歴を確認する

- commit履歴が新しいものから順番に表示される。
- 1行目の文字列はcommit IDといって、commitを識別するためのもの。

Gitチュートリアル

commitの履歴を確認する



以前の commitの 状態に 戻す

git reset --hard <commit id>

ファイルを<commit id>で指定したcommitの状態に戻す。

以前の commitの 状態に 戻す

```
$ echo "tweet" >> bird.txt
$ echo "meow meow" >> cat.txt
$ git status
// bird.txtの追加とcat.txtが変更されているメッセージが出る。
$ git reset --hard //コミットIDを省略すると最新のコミットに戻る
$ git status
// 変更が何もないというメッセージがでる。
```

- 新たにbird.txtを追加し、cat.txtを変更する。
- git reset --hardコマンドで、ファイルを最新のcommitの状態に戻す。
- lessコマンドなどで、ファイルを確認すると、元に戻っている。

branch一覧を表示する。

git branch

作成されているbranchの一覧を表示する。

branchを利用する-branch一覧を表示する。

- \$ git branch
 * main
- 作成済みのbranchが表示される。
 - main branchは最初から存在している。
 - (デフォルトが master branch の場合も)
- *は現在選択 (checkout) されているbranchを示している。


新しくbranchを作成する

git branch <branch名>

新しくbranchを作成する。

新しくbranchを作成する

- \$ git branch mod-cat
 \$ git branch
- \$ git branch
- * main
 - mod-cat
- mod-catというbranchを作成する。
- git branch で確認するとmod-cat branchが作成されているのがわかる。





branchを選択する

git checkout <branch名>

 ch名>で指定したブランチを選択する。

branchを選択する

\$ git checkout mod-cat
Switched to branch 'mod-cat'
\$ git branch
main
* mod-cat

mod-catブランチを選択して、git branchコマンドでmod-catが選択されていることを確認する。



(参考) branchを作成して選択する

git checkout -b <branch名>

-b コマンドでを使うことで新規ブランチの作成と選択 を同時に行うことも出来る。

mod-catブランチでcommitする

```
$ echo "nya-" >> cat.txt
$ git add cat.txt
$ git commit -m "mod cat"
$ git log --oneline
* 3b503ba mod cat
* 55d0fb1f commit all files
* 2c2adf72 create dog
```

- cat.txtを変更して、commitする。
- git log コマンドに--onelineオプションをつけると履歴を省略形で表示 する。
- 3つのcommitがあることを確認できる。

mod-catブランチでcommitする



- 作成されたcommitはmod-catブランチで更新される。
- mainブランチは変わらずに550dfb..のcommitを保持している。

mainブランチを更新する

```
$ git checkout main
$ echo "waon" >> dog.txt
$ git add dog.txt
$ git commit -m "mod dog"
$ git log
* e6fa391 mod dog
* 55d0fb1 commit all files
* 2c2adf7 create dog
```

- mainブランチを選択してdog.txtを変更しcommitする。
- commit履歴を確認するとmod-catブランチでのcommitは含まれておらず、mainブランチで行なったcommitのみが表示される。

mainブランチでcommitを行う



- 新しく作成されたcommitはmainブランチで更新される。
- mod-catブランチで行われた変更はmainブランチには影響されない。

branchをmergeする

git merge <branch 名>

現在選択されているbranchに<branch名>で指定したbranchをmergeする。

mainブランチにmod-catブランチをmergeする

- \$ git checkout main
 \$ git manage med cat
- \$ git merge mod-cat
- mainブランチにcheckoutして、mod-catブランチをmergeする。



mainブランチにmod-catブランチをmergeする

Merge branch	n 'mod-cat'		
# //省略			

• merge commitを行うためのエディタが立ち上がるので、必要に応じて コメントを修正し、保存・終了する。

mainブランチにmod-catブランチをmergeする



git logコマンドに--graphオプションをつけると、履歴の分岐をグラフィカルに表示する。





branchを破棄する

git branch -d <branch 名>

git branchに-dオプションをつけると<branch 名>で指 定したbranchを破棄する。

branchを破棄する

\$ git branch -d mod-cat
Delete branch mod-cat (was 23e1793)

• 機能追加が完了し、不必要となったbranchは消去する。



リモートリポジトリの概要

ローカルリポジトリとリモートリポジトリ

 自分の端末に存在し、個人で使用するリポジトリをローカ ルリポジトリ、サーバー上に存在し、複数の人が参照する ことのできるリポジトリをリモートリポジトリという。



リモートリポジトリの概要

Push

 リモートリポジトリに自分のローカルリポジトリにある変 更履歴をアップロードすることをpushという。



リモートリポジトリの概要

cloneとpull

 リモートリポジトリにある変更履歴を自分のローカルリポ ジトリにダウンロードする方法に、cloneとpullがある。



リモートリポジトリの概要

clone

 cloneは空のローカルリポジトリにリモートリポジトリの コピーを作成する時に使用。



リモートリポジトリの概要

pull

 ● pullはローカルリポジトリをリモートリポジトリの履歴で 更新する時に使用。



GitHubの概要

GitHub

 GitHubはGitの仕組みを利用して、プログラムのソース コードなどを共有・ホスティングできるサービス。リモー トリポジトリとして使用する。

GitHub

リポジトリの公開設定

- GitHubでは自分のリポジトリをpublicとprivateいずれか に設定できる。
- Publicでは不特定多数の人がアクセス可能となる。
- Privateでは自分とcollaboratorに設定した人しかアクセ スができない。
- 本実験で扱うリポジトリは必ずprivateにすること。

GitHubの概要

collaborator

● 自分のprivateリポジトリに他のユーザをcollaboratorとして登録すると、リポジトリを共有することができる。



GitHubの概要

tagとrelease

- 重要なコミットに分かりやすい名前をつけるための機能を tagという。
- tagがつけられたcommitを、他の人が使用できるように提供する機能をreleaseという。本実験ではreleaseにより課題の提出を行う。



④GitHubチュートリアル

GitHubチュートリアル リモートリポジトリの作成



GitHubチュートリアル

リモートリポジトリの作成



GitHubチュートリアル

リモートリポジトリの作成



GitHubチュートリアル リモートリポジトリのアドレス登録

git remote add <エイリアス> <URL>

<URL>を<エイリアス>という名前でリモートリポジト リとして登録する。

GitHubチュートリアル

リモートリポジトリのアドレス登録

A kazunarikato / le3-test Private	
✓> Code ① Issues 0 ① Pull requests 0 ② Actions Ⅲ Projects 0 Ⅲ Wiki ①	Security 🔟 Insights 🔅 Settings
Quick setup — if you've done this kind of thing hofers Last up in Deskto or HTTPS SSH https://github.com/kazunarikato/le3-test.git	Ê
Get started by creating a new me or uploading an existing me. We recommend every repositivy	Include a READIVIE, EIGENGE, and .gilightire.
or create a new repository on the command line	
<pre>echo "# le3-test" >> README.md git init git add README.md git commit -m "first commit" git remote add origin https://github.com/kazunarikato/le3-test.git git push -u origin master</pre>	GitHubのリポジトリページに リポジトリのURLがある
or push an existing repository from the command line	
git remote add origin https://github.com/kazunarikato/le3-test.git git push —u origin master	Ê
or import code from another repository You can initialize this repository with code from a Subversion, Mercurial, or TFS project. Import code	

GitHubチュートリアル

リモートリポジトリのアドレス登録

\$ git remote add origin <リモートリポジトリのURL>

- 一般的にエイリアス名には "origin"という名前をつける。
- リモートリポジトリとしてURLをoriginというエイリアス名で設定す る。

GitHubチュートリアル リモートリポジトリへのプッシュ

git push <エイリアス> <ブランチ>

<エイリアス>のリモートリポジトリにローカルの<ブ ランチ>をpushする。

GitHubチュートリアル

リモートリポジトリへのプッシュ

\$ git push origin main

● originのリモートリポジトリにmainブランチをプッシュする。



GitHubチュートリアル リモートリポジトリからのpull

git pull <エイリアス> <ブランチ>

<エイリアス>のリモートリポジトリの内容でローカル の<ブランチ>を更新する。 GitHubチュートリアル

ローカルリポジトリを更新

\$ git pull origin main

● リモートリポジトリの開発がローカルリポジトリよりも進んでいると
 き、リモートリポジトリの内容でローカルリポジトリを更新する。


GitHubチュートリアル リモートリポジトリからのclone

git clone <URL>

<URL>のリモートリポジトリの内容をローカルにコ ピーする。

リモートリポジトリからのclone

- \$ git clone <URL>
- <URL>で指定したリモートリポジトリをcloneする。





A kazunarikato / le3-test Pr	ivate
<> Code (!) Issues 0 (!) Pu	ull requests 0 💿 Actions 🔟 Proj
Options	Settings
Manage access	Repository name
Webhooks	le3-test
Notifications	Template repository
Integrations & services	Tensplate repositories let users gene kazunarikato/le3-test can be used
Deploy keys	
Secrets	Social preview
Actions	画面左のメニューから"Manage <u>access"をク</u> リック









|--|

コミットにtagをつける

git tag -a <タグ名> -m <コメント>

最新のコミットに対して、tagをつける。

一般的にタグ名はセマンティックバージョニングに準じてつけるが、本実験

では課題ごとに指定されたタグ名をつける。

-aの後にタグ名、-mの後にコメントを記載

過去のコミットにタグをつけたい場合は、<コメント>の後に、コミットID を書く。

tagの確認

git tag

つけたタグの一覧を確認できる。

git show <タグ名>

指定したタグ名のコミットを確認できる。

tagのpush

git push <エイリアス> --tag

git pushに--tagオプションをつけると、タグの情報がリモートリポジトリに 反映される。

普通にpushしただけでは、リモートリポジトリにtagは反映されない。(逆 にこのコマンドだけではブランチへのpushができない)。

必ず "git push <エイリアス> <ブランチ>"でブランチへのpushも完了して おくこと。

コミットにtagをつける

git tag -a v1.0.0 -m 'add v1.0.0'
git push origin --tag

*必ず "git push <エイリアス> <ブランチ>"でブランチへのpushも完了しておくこと。



releaseの作成

a <u>smnimo</u>	/le3-test Private			③ Unwatch \rightarrow 1 $$ Star 0 $\overset{99}{\circ}$ Fork 0
<> Code	() Issues 24 Pull requests ()	Actions 🛄 Projects 🖽 Wiki	🕕 Security 🗠 Insights 🔅 Settings	
ę	😚 main 👻 🐉 1 branch 💿 1 tag		Go to file Add file ▼	About 🕸
	Commit Second commit Commit Commit README.md Commit test.txt	Initial commit	e3cb35e / minutes ago 10 minutes ago 7 minutes ago	provided.
R	README.md		P	Releases
	le3-test			Create a new release Packaries
			リポジトリのページ "Create a new rele	Workages published Publish your first package ぐで"release"内の ease"をクリック

Releaseの作成



Releaseの作成

Releases Tags Tag version Image: Target: master = Choose an existing tag, or create a new tag on publish Releases title	Releaseのバージョン番号を入力。 バージョンはGit tagでつけたタグ 名から選択。タグ名は課題ごとに 指定されている。
Release lille	
Write Preview	
Describe this release	リリースのタイトルを入力
Attach files by dragging & dropping, selecting or pasting them.	リリースの説明を入力

This is a pre-release

We'll point out that this release is identified as non-production ready.

Publish release Save de

"Publish release"をクリック

Releaseの作成

Releaseの作成



⑤GitHubに公開鍵を登録

SSHで使うRSAキーの公開鍵をGitHub登録すること でターミナルで「git clone git@github.com:kuisisle3hw/....」

詳しくは https://docs.github.com/en/authentication/co nnecting-to-github-with-ssh を参考

ここではざっくり課程を見せる。

ターミナルで「ssh-keygen」ツールを使ってキーを作成

stela@jupiter:~\$ ssh-keygen Generating public/private rsa key pair. Enter file in which to save the key (/home/stela/.ssh/id_rsa): Created directory '/home/stela/.ssh'. Enter passphrase (empty for no passphrase): Enter same passphrase again: Your identification has been saved in /home/stela/.ssh/id_rsa Your public key has been saved in /home/stela/.ssh/id_rsa.pub The key fingerprint is: SHA256:EtwevEOyzifh82x7F4+/A6MIwX4Fr0NTgDXtdl5Ztnc stela@jupiter The key's randomart image is: +---[RSA 3072]---+ 0+0 ..0..0 0 .+ == +. o*oo= . o.E .+oS= o . o +00+. = *0.0 . * *0 0 0 0 .+0 . .00 -[SHA256]----+ stela@jupiter:~\$

Settings

Your profile

Your stars

Your gists

Upgrade

Help

Settings

Sign out

e

g Ca

ii:

ag

е

ag ok

aı 3

公開鍵をGitHubに登録する

\rightarrow SSH and GPG keys \rightarrow New SSH Key



公開鍵をGitHubに登録する

stela@jupiter:~\$ cat ~/.ssh/id_rsa.pub

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCoA9WqejOcpRrdWWI58cJoD6EXy dOm70k5nQPx36FGougS1xj2h2dRMvZw24D7hgXJ8enwWZ5FJSpJ1Kf2jE5VrVX4J2 OzLNPEfVgVtaq3kM5UNguFzVKyYhithMyuKZiWDuAVinMSm4xk9Am1RglsHVVHNGG P/B03L5vSpuE/AzwLApJ3sc OGE16wb/pxtEya7/vU2R1fKugsiT2KqXsi2qB4zUr Cf/+EJSIKUh3Z+/OF4WaThD PoolwCPVUDU6bsEcs LD2TdCDc2dEVELvWtiSclEv n4CAF54ok9ZpBkDg eLtroouPUsEGZXMs 9dYxUXqIGAqA7c4 IM/YAs8RGZA1mSPz Key type Authentication Key \$

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp-ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

id_rsa.pubの中身を コピー&ペースト

ターミナルでgit repositoryをclone

stela@jupiter:~\$ git clone git@github.com:kuis-isle3hw/2023-intro-stelaseo.git Cloning into '2023-intro-stelaseo'... Enter passphrase for key '/home/stela/.ssh/id_rsa': remote: Enumerating objects: 3, done. remote: Counting objects: 100% (3/3), done. remote: Compressing objects: 100% (2/2), done. remote: Total 3 (delta 0), reused 1 (delta 0), pack-reused 0 Receiving objects: 100% (3/3), done. stela@jupiter:~\$

⑤GitHubをVS2022内で アクセス

公開鍵をGitHub登録せずIDE内で使いたい場合

GitHubからrepositoryのHTTPSをコピー

	Go to file	Add file 🔻	<> Code 🗸
Local Codespaces			
>_ Clone ?			
HTTPS SSH GitHub CLI			
https://github.com/kuis-isle3hw/2023-intro-st 🗸			
Use Git or checkout with SVN using the web URL.			
더 Open with GitHub Desktop			
Download ZIP			

VSでcloneを始める



<u>Clone a repository</u> Get code from an online repository like GitHub or Azure DevOps

Clone a repository

Enter a Git repository URL

Repository location

https://github.com/kuis-isle3hw/2023-intro-stelaseo.git

<u>P</u>ath

C:\Users\stela\Source\Repos\2023-intro-stelaseo

Browse a repository

Azure DevOps

GitHub

200

Popupから「Sign in with a code」を選択



https://github.com/login/device のページに コードを入力する

⑤GitHubをVS Code内で アクセス

Open Remote Repositoryを選択 Open Repository from GitHubを選択



×1 v	Open Remote Repository	
	Enter a remote un or colorization provide.	
	🕥 Open Repository from GitHub	remote sources
	Provise & Edit	

ログイン後 Authorize Visual-Studio-Codeを押す



⑥GitHub Classroomの利用

GitHub Classroom

- GitHub Classroom は GitHub を講義で利用するための プラットフォーム
 - 個人用, グループ用の課題の作成が可能
 - 各学生の個人GitHub アカウントで作成したリポジトリ とリンクさせることで課題用のリポジトリを一括管理
 - 本実験ではこのGitHub Classroom を利用し、
 課題は各リポジトリでRelease を作成することで提出

GitHub Classroomの導入

- 招待リンクはPandA内のお知らせを参照
 - リポジトリは2種
 - o 導入課題(個人課題):2025-Intro
 - o プロセッサ課題(グループ課題):2025-Simple
- まずGitHub Classroom と GitHub の連携について authorizationを求められるので承認
 - GitHub アカウントを持っていない人はここで作成を



● 導入課題は各学生1人に対して1つのリポジトリ

GitHub Classroom	GitHub Education ロ) ifie
Join the classroom: kuis-isle3hw-classroom- To join the GitHub Classroom for this co from the list below to associate your Gi school's identifier (i.e., your name, ID, co	2021 course, please select yourself hitHub account or email). can't find your name? Skip to the 今 eAccept でリポジトリが作成される
Identifiers	2025-intro-XXX(Github アカウント名)
1029299434	※自分の学生番号が見つからない
1029302951 1029304955	→ 場合はスタッフまで連絡を
1029307671 1029309236 リポ	ジトリが作成されたら、"git clone"で自分の端
1029310012 末に	ダウンロードし、そのフォルダでquartusのプロジェクトを佐り調覧を進める
1029310040	

プロセッサ課題

● プロセッサ課題は基本的に学生2人に1つのリポジトリ

- 3人グループもある

- グループはこちらで割り振って連絡しますので後ほど PandAを確認してください.
- 1人の学生がチームの作成(連動してリポジトリの作成)
 → もう1人の学生は作成されたチームを探して参加
 - チーム名はグループ分けの番号と対応して「teamXX」 (例:team01)としてください.

o リポジトリは 2025-simple-teamXX が作成されます.

プロセッサ課題

- リポジトリが作成されたら、グループの一人が"git clone" で自分の端末にダウンロードし、そのフォルダでquartus のプロジェクトを作成し、"git push"。
- 別のグループ員は上記作業後に"git clone"して、以後それ ぞれの端末で課題を進める。

課題の提出

- 導入課題、プロセッサ課題の提出はGithubのreleaseを 使って行う。
- **release**については本資料の<u>p61</u>と<u>p81~88</u>を参照。
⑥その他のGitHub機能

fork

● 他の人のリモートリポジトリを自分のリモートリポジトリ
 にコピーする。



forkの方法

octocat / Spoon-Kni	fe		O Watch → 373	3 🖈 Star 1	0.2k YFork 109k
<> Code () Issues 1,010	0 🕅 Pull requests 5,000+	Actions III Projects	o 💷 Wiki 🕕 S	ecurity 🔟 Ins	sights
his repo is for demonstr	ation purposes only.				
-O- 3 commits	[₽] 3 branches	1 0 packages	S 0 release	es	41 contributor
	, C Dranoneo	to packageo	V • Foldas	00	
Branch: master - New pu	ll request	(Create new file Upload	d files Find file	Cion a or downloa t -
Branch: master - New pu	ll request	(Create new file Upload	d files Find file	Clone or downlos 1 - I0dd1f6 on 13 Feb 201-
Branch: master - New pu	uide for forking Pointing to the guide for f	forking	Create new file Upload	d files Find file	Clone or downlos 1 + 10dd1f6 on 13 Feb 2014 6 years ago
Branch: master - New pu	Il request uide for forking Pointing to the guide for f Created index page for fu	forking uture collaborative edits	Create new file Upload	d files Find file	Clon : or downlo: 1 + I0dd1f6 on 13 Feb 2014 6 years ago 6 years ago

コピーしたいリポジトリページの リポジトリ名の右側にある "Fork"をクリック



forkの方法

Insights ♀ Settings ♥ O releases	Le 1 contributor
ি O releases	🚨 1 contributor
♥ 0 releases	L contributor
te new file Upload files Fin	d file Clone or downlo
	ן Pull request 🗈 Com
Latest con	nmit dødd1f6 on 13 Feb 2
	6 years
	6 years
	6 years
	te new file Upload files Fin

Pull request

- 新機能を追加し、ファイルをcommitした時、いきなりリ モートリポジトリのmainにpushすると、そこにバグが含 まれていた時に問題となる。
- Pull requestを使うと、pushする前にローカルリポジトリ での変更内容をリモートリポジトリを共有している他の人 に通知することができる。
- 変更内容を確認後、リモートリポジトリ上でmergeすることで、リモートリポジトリが更新される。
- コード・レビューがしやすくなる。

Pull requestの作成

A kazunarikato / le3-test Private		O Unwatch	▼ 1 ★ Star 0 ¥ Fork 1
<> Code ① Issues 0 ① Pull requ	ests 0 🔘 Actions 🔟 Projects	0 🕕 Security 🔟 Insights	🔅 Settings
No description, website, or topics pro Manage topics	ovided.		Edit
-0- 4 commits	ဖို 1 branch	🗊 0 packages	\bigtriangledown 2 releases
Your recently pushed branches:			
ဖို dev (less than a minute ago)			্ট্ৰী Compare & pull request
Branch: master - New pull request		Create new file Upload fi	iles Find file Clone or download -
🔚 kazunarikato Merge pull request #1 from	n kanda5515/master 🛛 …		Latest commit fe922b5 18 days ago
readme.md	first commit		18 days ago
test.txt	rev test.txt		18 days ago
リポジト "Pull reque	・リページの sts"をクリック		

14

Pull requestの作成

A kazunarikato / le3-test Private		O Unwatch → 1 ★ Star 0 % Fork 1
↔ Code ① Issues 0 Ŋ Pull requests 0	🗘 Actions 🛛 Projects 0 🕕	Security 🔟 Insights 🔅 Settings
Your recently pushed branches:		
ំទ dev (1 minute ago)		្រឹ Compare & pull request
Filters - 🔍 is:pr is:open		C Labels 9 T Milestones 0 New pull request
🔲 🎁 0 Open 🗸 1 Closed	Author - Label - Projects -	Milestones Reviews - Assignee - Sort -
"New pull requets"元 Th You co	ドタンをクリック ere aren t any open pul puld search all of GitHub or try an	I requests. advanced search.

Pull requestの作成

Compare changes

Compare changes across branches, commits, tags, and more below. If you need to, you can also compare across forks.

û base: master ▼ ♦ compare: master ▼	mergeするブランチを決める。
Choose a head ref	"base:"にmerge先ブランチ
11 Create pull request Find a branch	, "compare:"にmerge元ブランチを指定 する。
Branches Tags	
✓ master	default
dev	
Compa	are and review just about anything
Branches, tags, commit r	ranges, and time ranges. この例では"master"←"dev"のmergeを
EXA	AMPLE COMPARISONS リクエストするので、compare:に"dev"
8º c	dev 1 minute ago を選択する。
°D	master@{1day}master24 hours ago

Pull requestの作成

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

ឯ base: master ▼ ← compare: de	v ▼ ✓ Able to merge. These b	ranches can be automatically merged.	
1 Create pull request	d review the changes in this com	"Create pull request する	"ボタンをクリック
- > 1 commit	1 file changed	Commit comments	o
Commits on Apr 10, 2020	adme.md		1a786eb
Showing 1 changed file with 1 addition	n and 0 deletions.		Unified Split
∨ 1 ∎ readme.md			<> ■ …
@@ -1 +1,2 @@	-1		
ユーロー les-testles-testles-te 2 + devで編集	st		

Pull requestの作成



Pull requestの作成

mod readme.md #2

Conversation 0 Commits 1 R Checks 0 D Files changed 1		+1 -	0
kazunarikato commented now	••••		
No department provided		Reviewers	¢
No description provided.		No reviews	
		Assignees	¢
-o- 🚠 mod readme.md	1a786eb	No one—assign yourself	
		Labels	¢
Add more commits by pushing to the dev branch on kazunarikato/le3-test.		None yet	
Continuous integration has not been set up		Projects	¢
GitHub Actions and several other apps can be used to automatically catch bugs and en style.	force	None yet	
This branch has no conflicts with the base branch		Milestone	\$
Merging can be performed automatically.		No milestone	
Merge pull request T You can also open this in GitHub Duriton or view semand line in	structions	Linked issues	¢
		Successfully merging this pu may close these issues.	II request
Write Preview AA B i GG 🖘 🖘 🗄 🖆 @	A * -		
Pull reques	stが作	成された。	
Leave a comment			
		You're receiving notifications you're watching this reposite	s because ery.
Attach files by dragging & dropping, selecting or pasting them.	6M	1 participant	
() Close pull request	Comment		

Edit

Damamhar contributions to this rangeitory should follow our CitHub Community Guidalinas

Pull requestの作成

mod readme.md #2



Edit

Pull requestの作成

mod readme.md #2



Edit

よくある問題と対策

コンフリクト

 他の人が編集したファイルに重複した編集をした状態で pushやpullをするとコンフリクトが起きる。(1人で作業 する場合でもmergeの時に起きる可能性あり)



よくある問題と対策

コンフリクト対策

- コンフリクトを未然に防ぐ
 - .gitignoreを設定して、ログファイルなど不要なファイルはGitの管理対象外にする。
 (https://docs.github.com/ja/get-started/git-basics/ignoring-files)
 - プルリクエストを使う。
 - 二人で1つのファイルを同時に編集しないようにする。

よくある問題と対策

コンフリクト対策

- コンフリクトしてしまったら
 - git statusコマンドでコンフリクトしているファイルを 確認。
 - 直接ファイルを開く。編集が重複している部分が "<<<<HEAD"、"====="、">>>>"でハイライト されているので、不要部分を削除する。
 - 編集したファイルをステージング、コミットする。

参考情報

- サルでもわかるGit入門
 <u>http://www.backlog.jp/git-guide/</u>
- Pro Git book (日本語版) <u>https://git-scm.com/book/ja/v2</u>
- Git Cheat Sheet <u>https://services.github.com/on-</u> <u>demand/downloads/github-git-cheat-sheet.pdf</u>
- GitHubヘルプドキュメント
 <u>https://help.github.com/ja/github</u>