

ITP20004 – Open-Source Software Labs

Project Management

Charmgil Hong

charmgil@handong.edu

Spring, 2023
Handong Global University



Announcements

- Weekly schedule

Week	Mon	Week	Thur
1	Course overview, motivation, administrivia	1	CPR: C Programming Reinforcement - Functions
2	Computer organization and Linux environment (1)	2	CPR: C Programming Reinforcement - Strings
3	Computer organization and Linux environment (2)	3	CPR: C Programming Reinforcement - User-defined types, and memory allocation
4	Basic Linux commands + Writing code on Linux (vim)	4	Getting started with Linux / Hands-on Linux command-line tools
5	More Linux commands	5	CPR: C Programming Reinforcement - Understanding compilation and build process
6	Project management (1) Proj 1 출제	6	Project management (2)
7	-	7	Project: BASIC interpreter
8	- Tuesday night: Midterm exam	8	Project QnA (Optional)
9	CPR: C Programming Reinforcement - Accessing files and directories	9	Debugging with GDB + Unit testing with gtest
10	Shell script	10	Shell script
11	Code review GNU utilities Proj 2 출제	12	Writing an application in C
12	Github and open-source community 	13	Using Github
13	Computer network basics	14	Linux network commands
14	Project: Text-based Game	15	Socket programming
15	Project: Multi-user game Proj 3 출제	16	Project: Multi-user game
16	Final exam		

Announcements

- For each lab
 - Before a lab, **every student** submits a pre-lab report (worksheet-type assignment) – **individual work**
 - After a lab, **each team** sees and reports to the TA with the results – **team work**
- Up-coming schedule
 - There are no post-lab session on May 16 (Week #12)
 - We DO have a post-lab session on May 23 (Week #13)

Last Lab: File Permissions

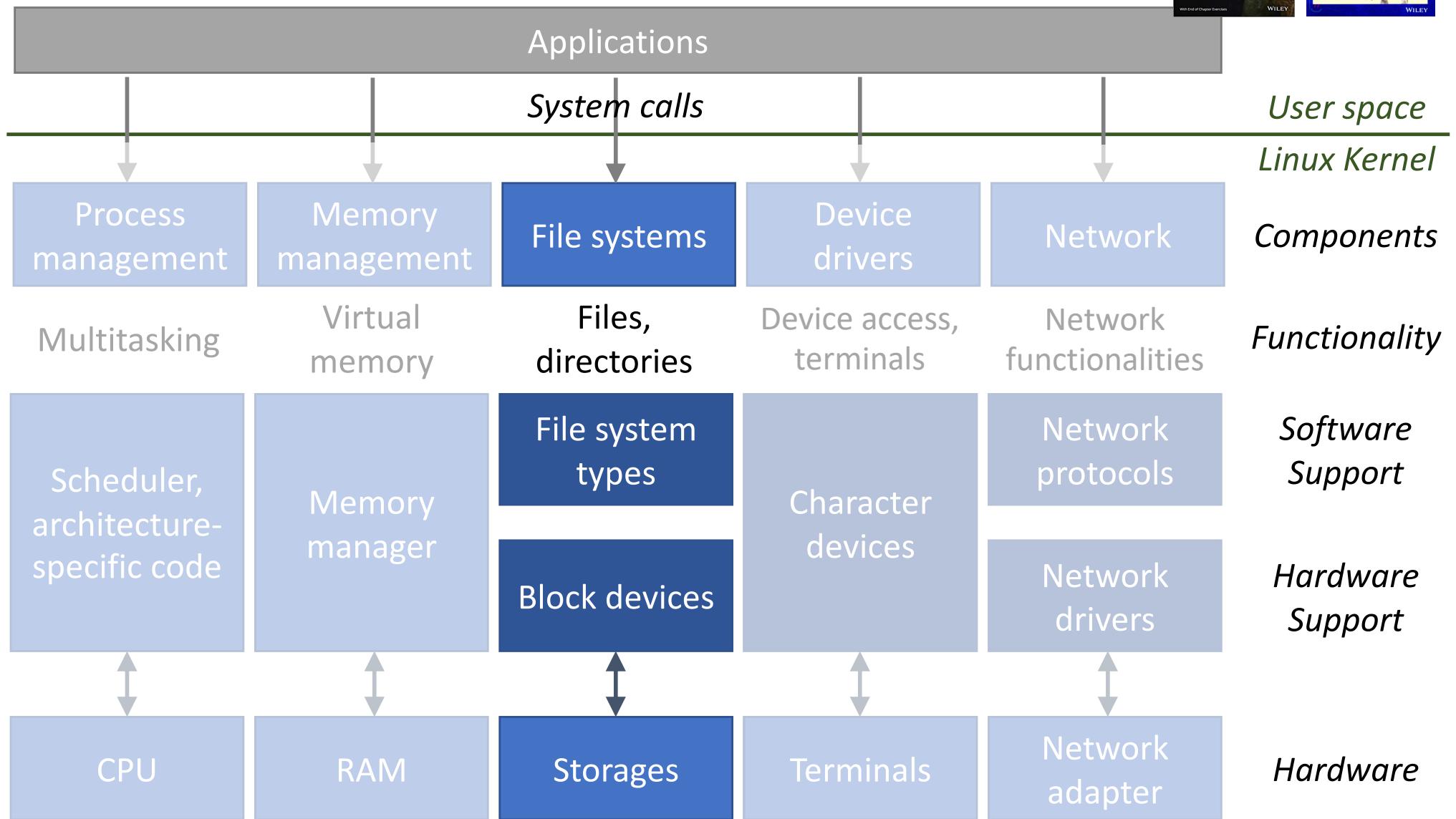
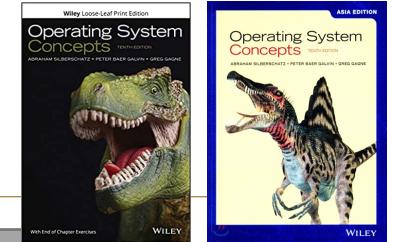
- Permissions

```
charmgil@vm-VirtualBox:~/private/scratch/lab3$ ls -l
total 488
-rw-r--r-- 1 charmgil charmgil 74556 Mar 18 21:52 CRICKET.WAV
-rw-r--r-- 1 charmgil charmgil 31730 Mar 18 21:50 hgu_logo.jpg
-rw-r--r-- 1 charmgil charmgil 1440 Mar 10 17:16 kernel5.txt
-rw-r--r-- 1 charmgil charmgil 217993 Mar 18 23:08 lab3data.zip
-rw-r--r-- 1 charmgil charmgil 47207 Mar 18 21:50 linux_PNG1.png
-rwxr-xr-x 1 charmgil charmgil 114496 May 14 2018 x64-intro.pdf
```

File System

- A **file** is a logical **unit of storage** for a sequence of records
 - A storage is recording medium that holds information
 - A file must have a unique name (**identifier**) in its directory
 - Consists of one or more **streams of bytes** and a set of **attributes** (properties)
- A **file system** is the methods and data structures that an OS uses to keep track of files on storage devices
 - Provides a way to separate data into pieces (files and directories)
 - *C.f.*, Linux kernel treats every I/O device as a file (/dev directory) – Linux simplifies the I/O communications by making (abstracting) them as file accesses

Linux Kernel



History

- Electronic Recording Machine Accounting (ERMA) Mark 1 (1958)
- Microsoft's FAT (8-bit) (1977)
- Apple's DOS 3.x (1978)
- Rémy Card's ext, ext2 (1992, 1993)



Corbis

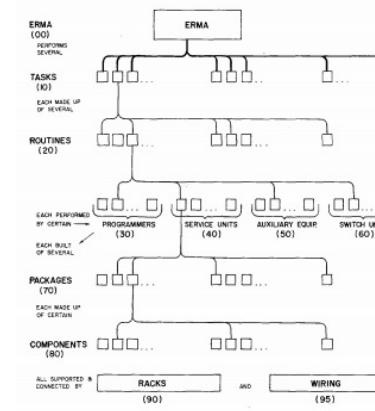


Fig. 2. The ERMA Mark I hierarchical structure

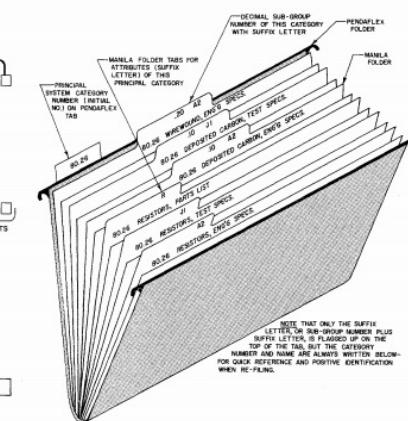
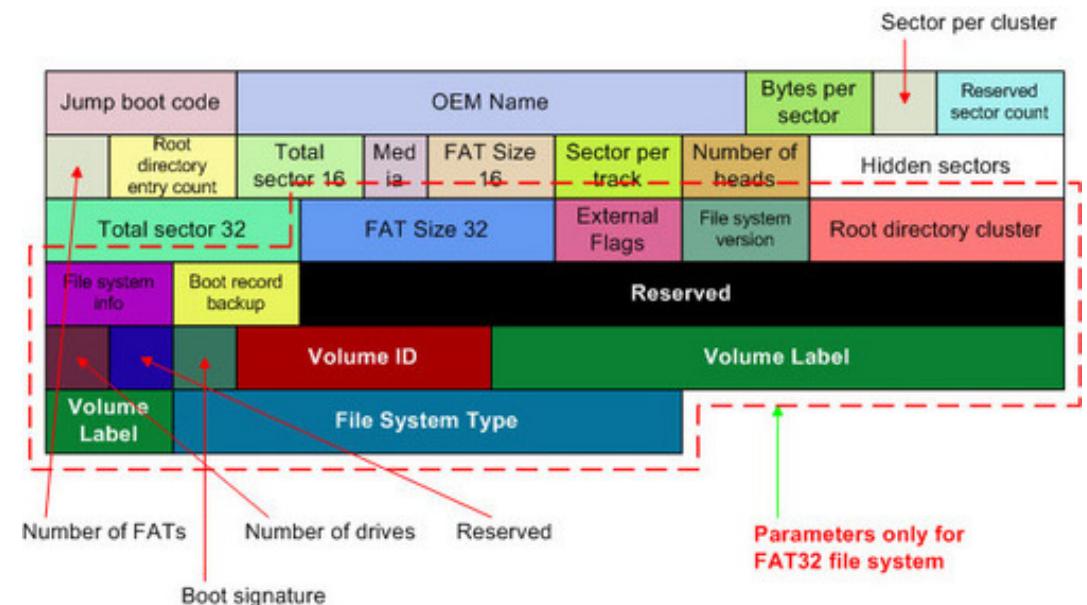
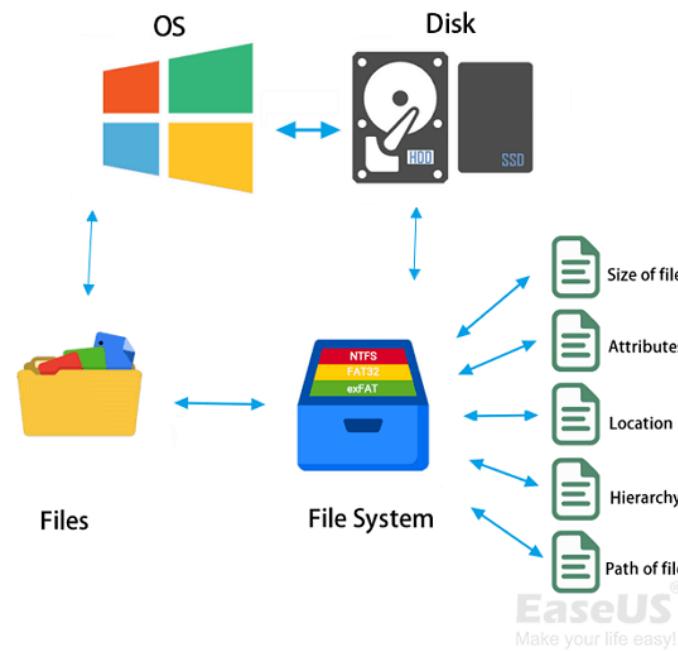


Fig. 3. Records storage and arrangement

File Systems Galore

- ext4 (Linux; 2006)
- APFS (Apple; 2016)
- NTFS (Microsoft; 1993)
- exFAT (Microsoft; 2006)), FAT32 (Microsoft; 1996)
- ZFS (Sun; since 2004)

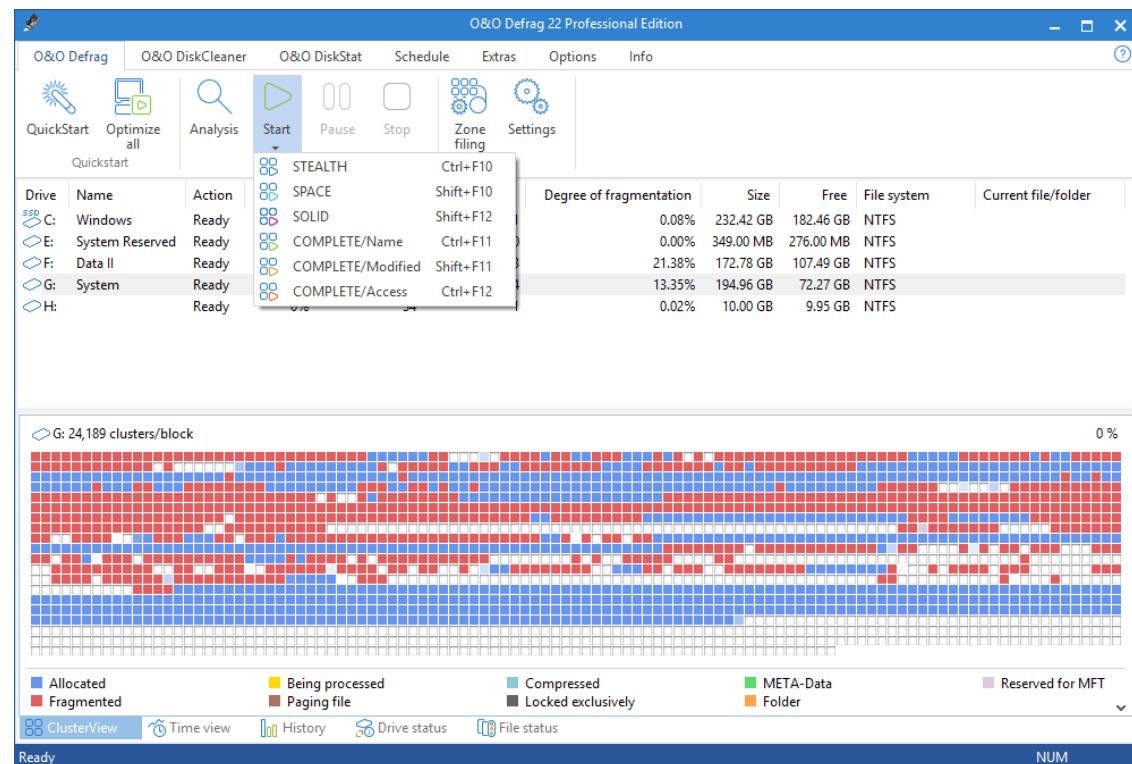
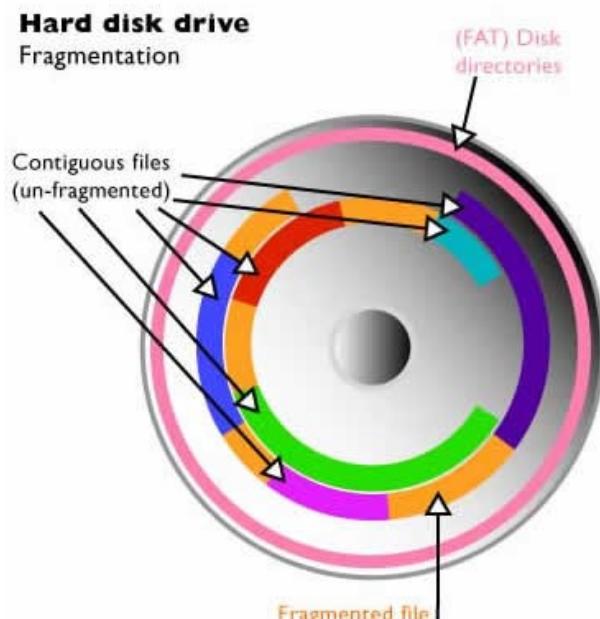


The items of FAT32 boot record

by iprinceps, <iprinceps@gmail.com>

File Fragment

- FAT*, NTFS (with older versions of Windows)



*Image obtained from <http://www.planetoftunes.com/computer/hard-disc-drives.html#.XJP4hBMzaL4>

Files on Linux

- A file and its metadata
 - Metadata: OS allocates a few bytes of storage to carry some basic information about each file
- `stat` – Display file status

```
charmgil@vm-VirtualBox:~/private/scratch$ stat hello.c
  File: hello.c
  Size: 90          Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 552269      Links: 1
Access: (0644/-rw-r--r--) Uid: ( 1000/charmgil)  Gid: ( 1000/charmgil)
Access: 2019-03-18 21:28:08.105502374 +0900
Modify: 2019-03-05 04:02:54.566528235 +0900
Change: 2019-03-05 04:02:54.570530236 +0900
 Birth: -
```

stat – Display file status

```
charmgil@vm-VirtualBox:~/private/scratch$ stat hello.c
  File: hello.c
  Size: 90          Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 552269      Links: 1
Access: (0644/-rw-r--r--) Uid: ( 1000/charmgil)  Gid: ( 1000/charmgil)
Access: 2019-03-18 21:28:08.105502374 +0900
Modify: 2019-03-05 04:02:54.566528235 +0900
Change: 2019-03-05 04:02:54.570530236 +0900
 Birth: -
```

- Identifier: filename and path
- Size, (physical) location, type
 - Size: #bytes, #blocks
 - Location: Inode (holds the mappings to device blocks)
 - Type: {regular, directory, symbolic link, pipe, socket, block/character device}

stat – Display file status

```
charmgil@vm-VirtualBox:~/private/scratch$ stat hello.c
  File: hello.c
  Size: 90          Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 552269      Links: 1
Access: (0644/-rw-r--r--) Uid: ( 1000/charmgil)  Gid: ( 1000/charmgil)
Access: 2019-03-18 21:28:08.105502374 +0900
Modify: 2019-03-05 04:02:54.566528235 +0900
Change: 2019-03-05 04:02:54.570530236 +0900
 Birth: -
```

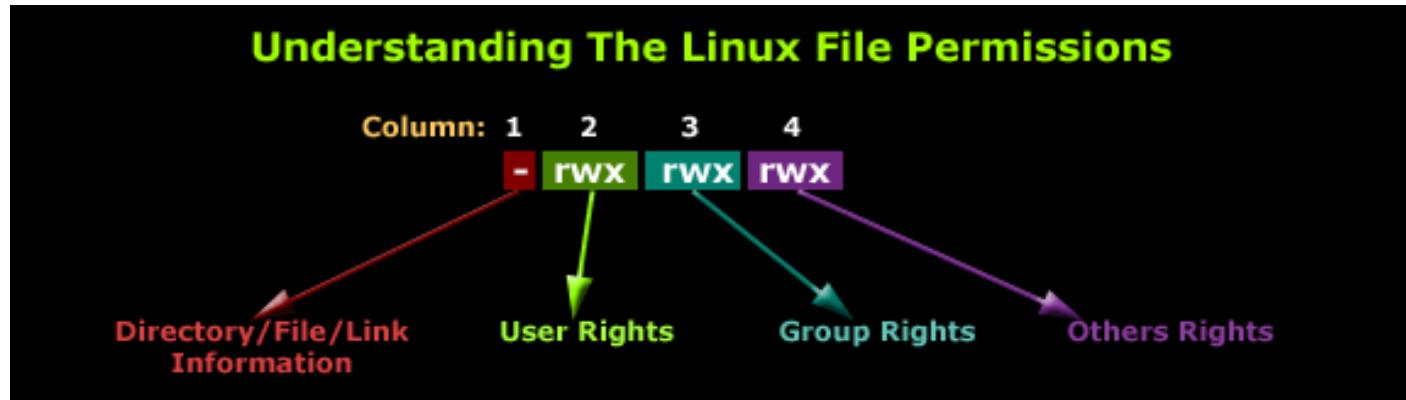
- Access control
 - Permissions (read/**w**rite/**e**xecute)
 - Owner Uid (user) and Gid (group)
- Time
 - Last access time
 - Last modification (content update) time
 - Last change (content or metadata update) time

Permissions (Access Control)

- \$ ls -l

```
charmgil@vm-VirtualBox:~/private/scratch/lab3$ ls -l
total 488
-rw-r--r-- 1 charmgil charmgil 74556 Mar 18 21:52 CRICKET.WAV
-rw-r--r-- 1 charmgil charmgil 31730 Mar 18 21:50 hgu_logo.jpg
-rw-r--r-- 1 charmgil charmgil 1440 Mar 10 17:16 kernel5.txt
-rw-r--r-- 1 charmgil charmgil 217993 Mar 18 23:08 lab3data.zip
-rw-r--r-- 1 charmgil charmgil 47207 Mar 18 21:50 linux_PNG1.png
-rwxr-xr-x 1 charmgil charmgil 114496 May 14 2018 x64-intro.pdf
```

Permissions (Access Control)



- Column 1: Type identifier (directory, file, link, ...)
- Column 2: Permissions (r/w/x) for the owner (Uid)
- Column 3: Permissions for the group (Gid)
- Column 4: Permissions for everyone else

* Image obtained from: <http://tecdistro.com/file-permissions-in-linux/>

How to Modify Permissions?

- **chmod** – Change file modes or access control lists
 - Use-case #1: `chmod <target>{+,-,=}<permission> <file name>`
 - Target: u (user), g (group), o (others), a (all)
 - {+, -, =}: + (add), - (remove), = (set)
 - Permission: A combination of r, w, and x
 - Examples

Notation	Meaning
<code>u+x</code>	Add execute permission for the owner.
<code>u-x</code>	Remove execute permission from the owner.
<code>+x</code>	Add execute permission for the owner, group, and world. This is equivalent to <code>a+x</code> .
<code>o-rw</code>	Remove the read and write permissions from anyone besides the owner and group owner.
<code>go=rw</code>	Set the group owner and anyone besides the owner to have read and write permission. If either the group owner or the world previously had execute permission, it is removed.
<code>u+x, go=rx</code>	Add execute permission for the owner and set the permissions for the group and others to read and execute. Multiple specifications may be separated by commas.

How to Modify Permissions?

- **chmod** – Change file modes or access control lists
 - Use-case #2: `chmod <octal digits> <file name>`
 - Octal digits: Treat the permission mnemonics as a binary number

- - -	Not accessible	000	0
- - x	Executable	001	1
- w -	Writable	010	2
- w x	Executable & Writable	011	3
r - -	Readable	100	4
r - x	Readable & Executable	101	5
r w -	Readable & Writable	110	6
r w x	Fully accessible	111	7

- Examples:
751 = rwx r-x --x
644 = rw- r-- r--

Agenda

- Coding in collaboration
- Version control systems
- Git
- GitHub

Agenda

- Coding in collaboration
- **Version control systems**
- Git
- GitHub

Version Control Systems

- Systems that help the **management of changes** to computer programs, documents, and other collection of information
 - Systems that **record changes to a file or set of files over time**, so that **one can recall specific versions later**
 - A.k.a. revision control or source control
 - Provide **shared repositories** for projects
 - Support and promote **code sharing and collaboration**
 - Allow to maintain **multiple versions/branches** of projects
 - Help **simultaneous work on files**
 - Offer easy ways to **merge** files with conflicts
 - Keep track of development projects; provide ways to “undo”

So, Version Controlled...

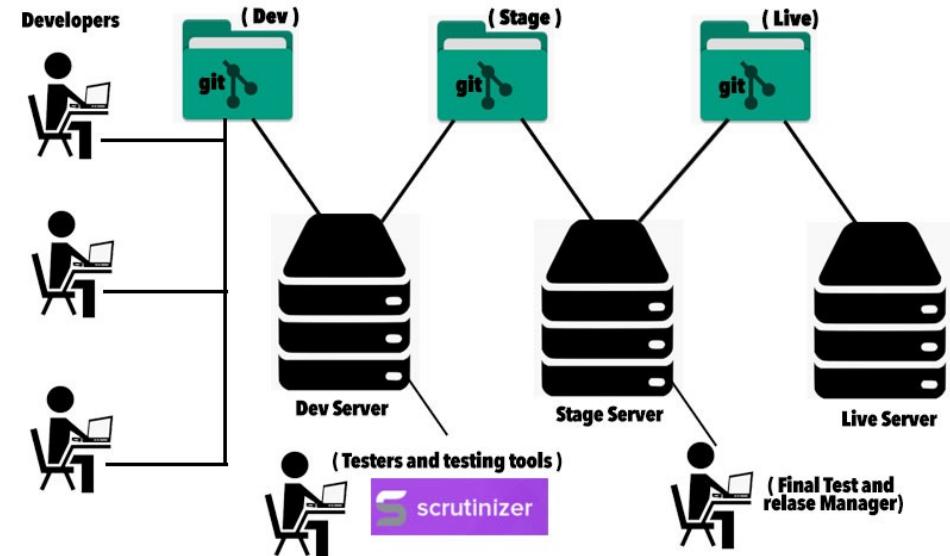
- It is not wrong to manually control versions using
 - Clever filenames: .bak, .orig, .old, .older, .oldest, .old.old, ...
 - Version numbers: v1, v1a, v1aa, ...
 - Date: 20200401_1223.docx, 20200403_0155.docx, ...



* Image src: <https://www.slideshare.net/visual28/version-control-58434966>

Benefits of Version Control

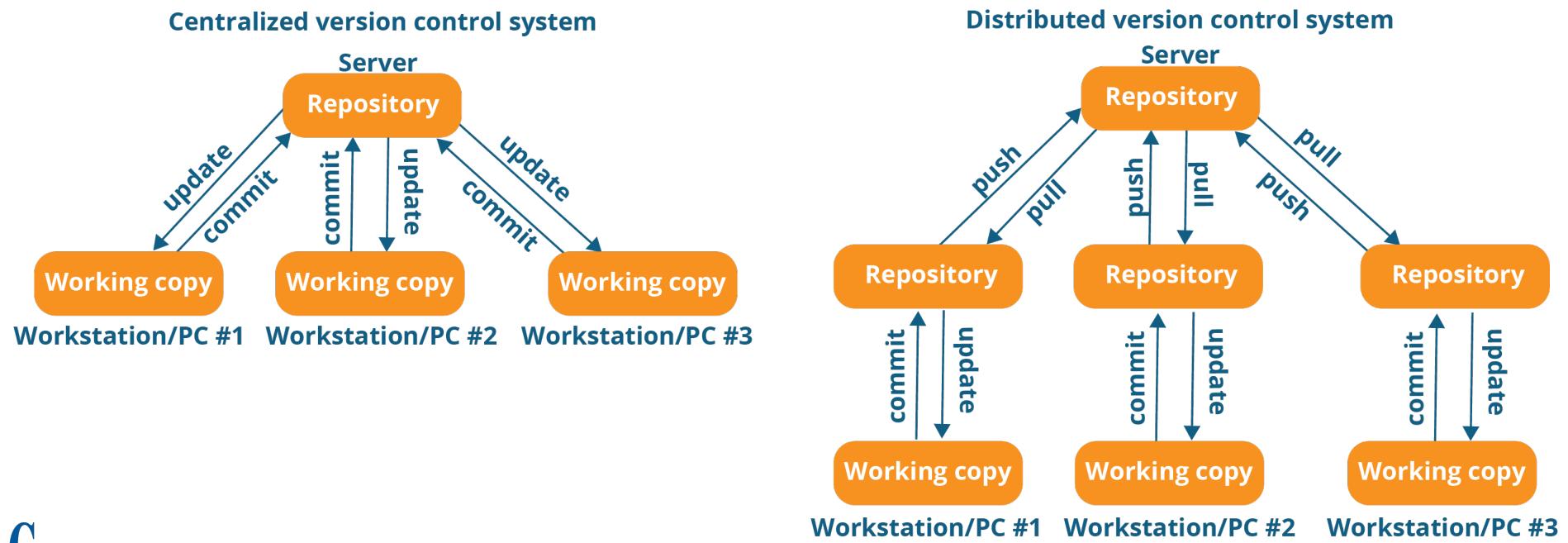
- Backup and restore
- Sync with multiple computers (multiple developers)
- Working in a group
- Safely create and test new features
- Ownership, credits, blame



* Image src: <https://www.helpzysam.com/tech/database-version-control-dbv/>

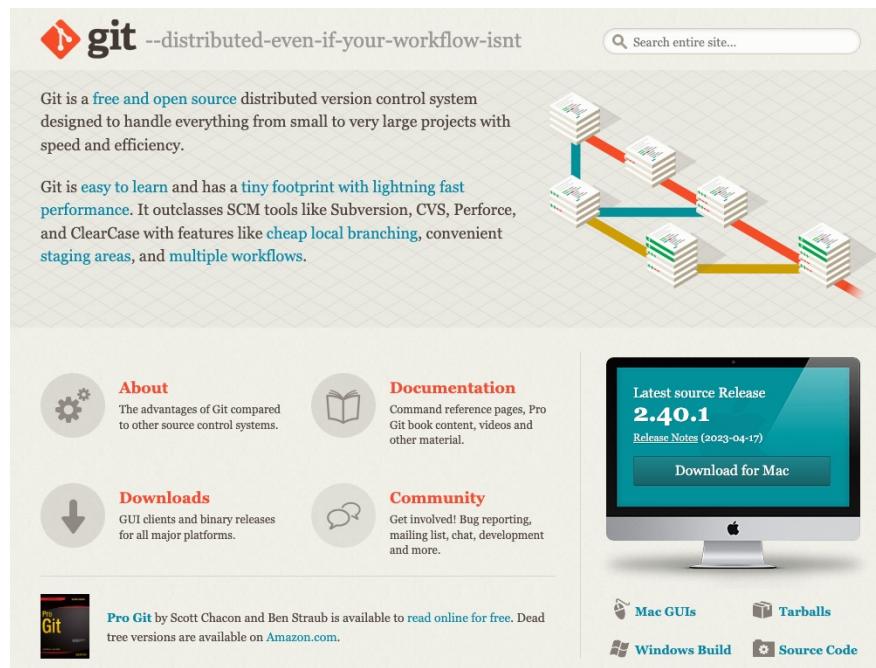
Version Control Systems

- Centralized VCS (cvs, svn) – a single server (repository)
- Distributed (decentralized) VCS (git)
 - No main server
 - All of the full history of the project is available once you cloned the project



git

- The mostly widely used, decentralized VCS
 - The Linux kernel project is running on Git
- Available at: <https://git-scm.com/>



git

- Open-source project originally developed in 2005 by Linus Torvalds
 - After BitKeeper broke
 - The Linux Kernel project was hosted by BitKeeper
 - *A “git” is a cranky old man; Linus meant himself*
- A command line utility
- Sits on top of your file system and manipulates files
- A distributed version control system (DVCS)



* Image src: BitKeeper; <https://insights.dice.com/2018/09/18/linus-torvalds-apology-highlights-soft-skills-necessary/>
<https://twitter.com/talitapagani/status/51135345443626496>

C.f., BitKeeper

[Why?](#)[Download](#)[Try it!](#)[Documentation](#)[Community](#)

Why use BitKeeper when there are lots of great alternatives?

For many projects, the answer is: you shouldn't. For instance, Git is an excellent solution for many use-cases. But it's not ideal for every situation. Here are some instances where BitKeeper will likely be a better solution.

git

- Git is
 - A distributed version control system (DVCS)
 - Free and open source
 - Designed to handle everything from small to very large projects
 - Easy to learn and maintain
- Git takes a snapshot of your repository at a given point in time
 - This snapshot is called a **commit**
- Git is optimized for local operation
 - Clone a repository = a copy of the entire repository + its history
- Branches are lightweight and cheap
- Git is explicit: No auto-saves or auto-syncing with the remote



- An online hosting service (of Microsoft) for software development and version control using Git; provides:
 - A collaboration platform (Distributed version control of Git)
 - Multi-user access control
 - Bug tracking
 - Software feature requests
 - Task management
 - Continuous integration
 - Wikis and bulletin board

The screenshot shows the GitHub repository page for `torvalds/linux`. The top navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. Below the header, there's a search bar and a "Watch" button. The main content area displays the repository name `torvalds / linux` and the description "Linux kernel source tree". Key statistics are shown: 913,964 commits, 1 branch, 0 packages, and 645 releases. A "Create new file" button is also present. The "Code" tab is selected, showing a list of recent commits from the `master` branch. The commits listed are:

- torvalds** Merge tag 'for-v5.7' of git://git.kernel.org/pub/scm/linux/kernel/git... [diff]
- Documentation** Merge tag 'powerpc-5.7-1' of git://git.kernel.org/pub/scm/linux/kerne... [diff]
- LICENSES** LICENSES: Rename other to deprecated [diff]
- arch** Merge tag 'irq-urgent-2020-04-05' of git://git.kernel.org/pub/scm/lin... [diff]
- block** Merge tag 'scsi-misc' of git://git.kernel.org/pub/scm/linux/kernel/gi... [diff]
- certs** .gitignore: add SPDX License Identifier [diff]



- GitHub.com = Git + public repositories + social network
 - Something you want to tell on your resume

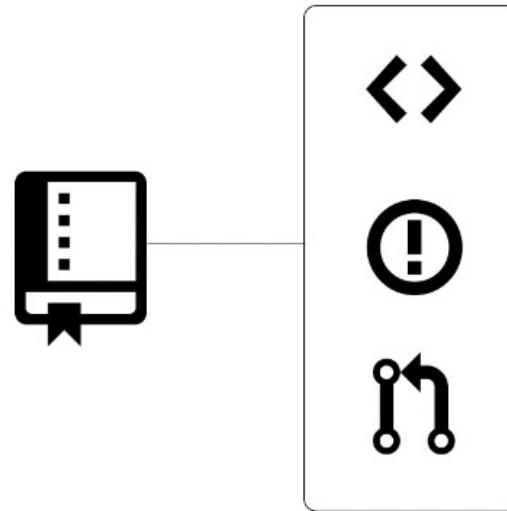
The figure displays six GitHub user profiles arranged in a 2x3 grid:

- Fabien Potencier**: Profile picture, bio, repositories (42), projects (0), stars (168), followers (10.9k), following (0). Pinned repos: symfony/symfony (PHP, 2.3k, 7.5k), twig/twig (PHP, 6.8k, 1.1k). Contributions chart for 2020.
- Andrew Nesbitt**: Profile picture, bio, repositories (288), projects (6), stars (8.6k), followers (2.3k), following (3.8k). Pinned repos: octobox/octobox (Ruby, 1.3k, 2.2k), php/package-managers (Ruby, 6.8k, 1.1k). Contributions chart for 2020.
- Taylor Otwell**: Profile picture, bio, repositories (14), projects (0), stars (187), followers (17.9k), following (0). Pinned repos: laravel/laravel (PHP, 22.1k, 7.2k), laravel/framework (PHP, 20.5k, 1.3k). Contributions chart for 2020, showing 7,121 contributions in the last year.
- Kevin Titor**: Profile picture, bio, repositories (488), projects (8), stars (2.8k), followers (6.8k), following (0). Pinned repos: selenide/selenide (JavaScript, 1.0k, 1.1k), pol/pol (JavaScript, 4.1k, 273). Contributions chart for 2020.
- Hugo Giraudel**: Profile picture, bio, repositories (38), projects (0), stars (0), followers (2.3k), following (0). Pinned repos: ama/ama (HTML, 40, 3), sass-guidelines/sass-guidelines (HTML, 695, 212). Contributions chart for 2020.
- Thibault Duplessis**: Profile picture, bio, repositories (380), projects (0), stars (4.82k), followers (2.3k), following (168). Pinned repos: b6/b6 (JavaScript, 6.6k, 870), searx/searx (Python, 302, 113). Contributions chart for 2020, showing 5,654 contributions in the last year.

* Image src: <https://github.com/fabpot>; <https://github.com/andrew>; <https://github.com/taylorotwell>; <https://github.com/egoist>; <https://github.com/HugoGiraudel>; <https://github.com/ornicar>

A GitHub Repository

- Repository: the most basic element of GitHub
 - Like a project's folder
 - Key components
 - Code
 - Issues
 - Pull requests
 - Wiki
 - Pulse
 - Graphs
 - README.md



Working Locally with Git

Get Your Initial Make Project

- One header file (.h) – mylib.h
- Two source files (.c) – main.c, mylib.c

```
#include <stdio.h>
#include "mylib.h"

int main(void){

    int a = 3, b = 5;
    printf("(initial) a=%d, b=%d\n", a, b);

    swap(&a, &b);
    printf("(swapped) a=%d, b=%d\n", a, b);

    return 0;
}
```

main.c

```
#ifndef _MYLIB_H_
#define _MYLIB_H_

void swap(int*, int*);
```

mylib.h

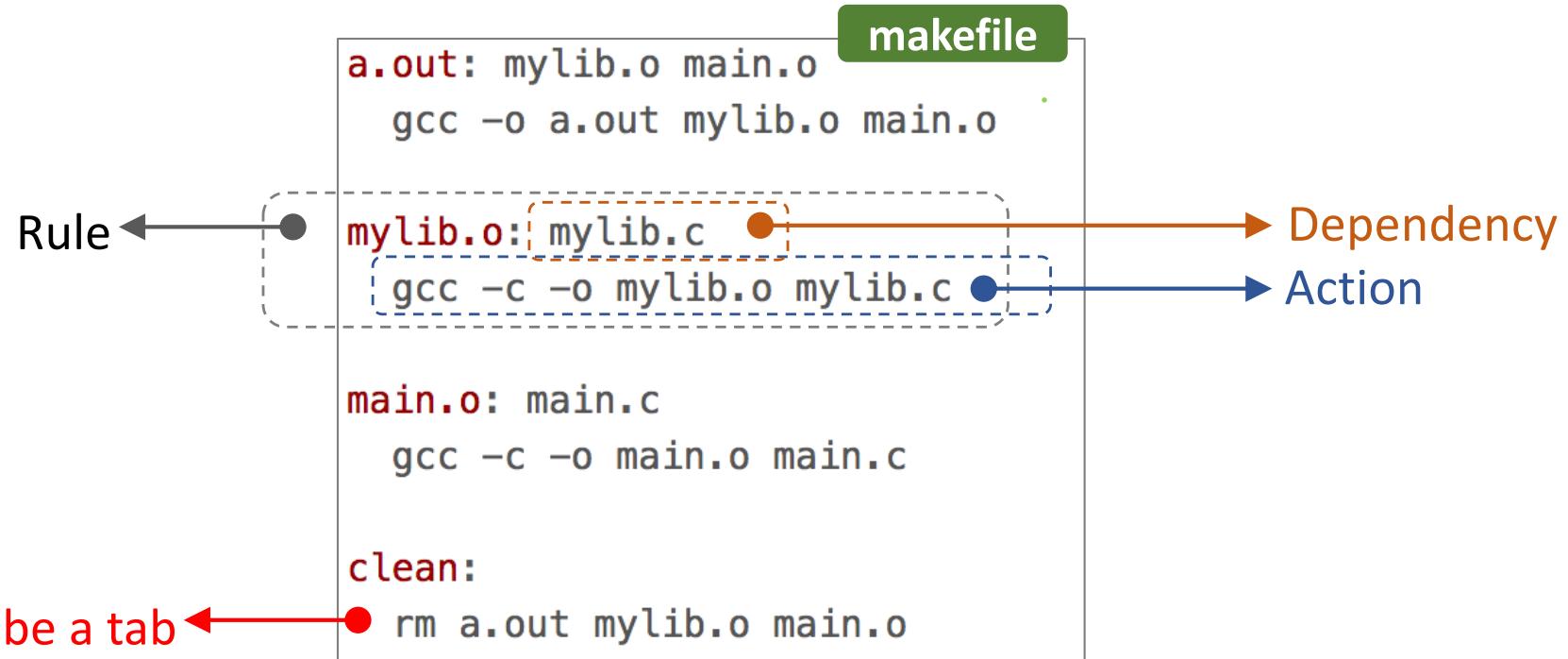
```
#include "mylib.h"

void swap(int* a, int* b){
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```

mylib.c

Get Your Initial Make Project

- An example *makefile*



Check If You Have Git

- Make sure you have a Git client on your machine

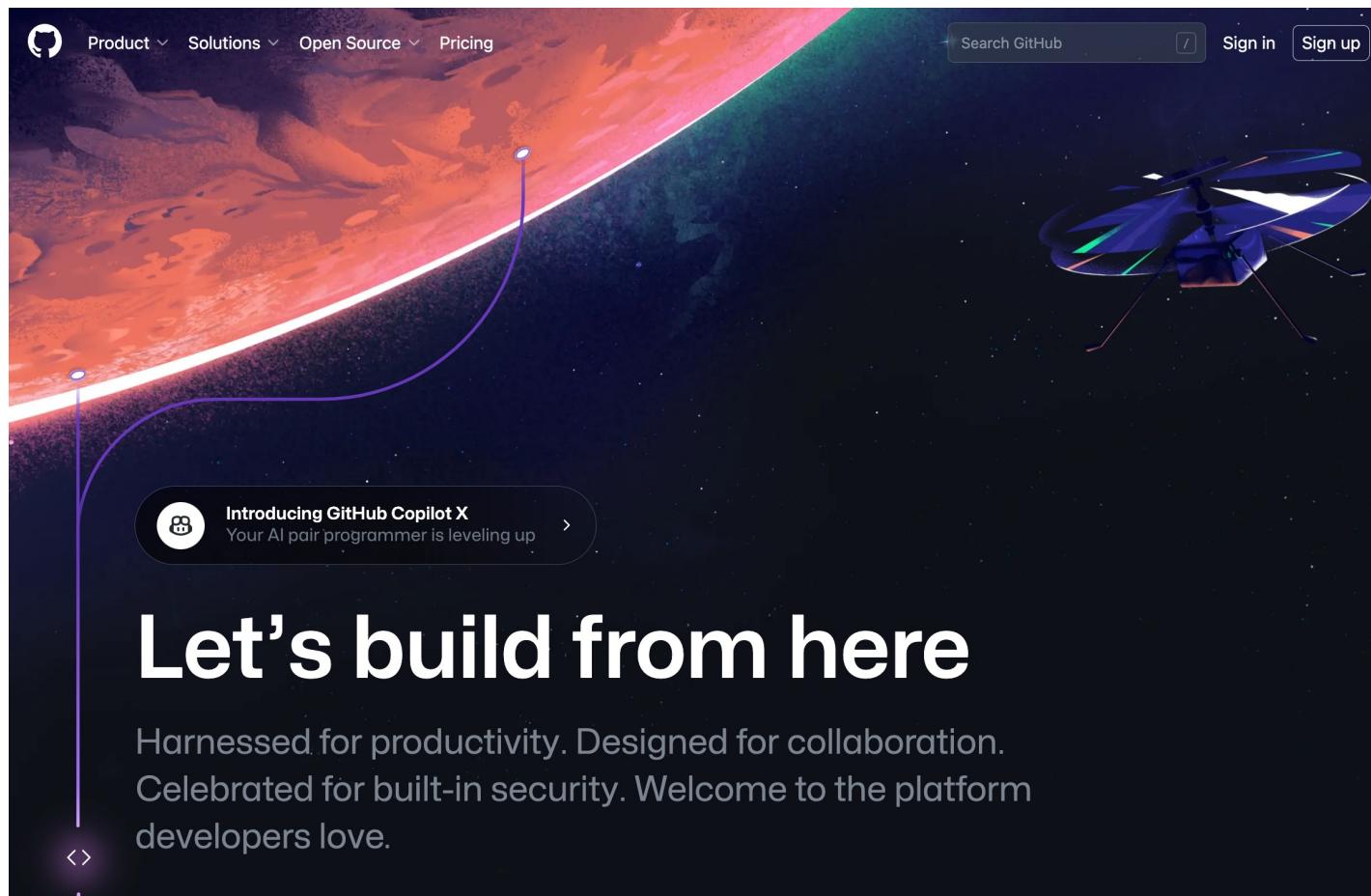
```
$ git --version  
git version 2.7.4
```

- Add your identity

```
$ git config --global user.name "YOUR NAME"  
$ git config --global user.email ACCOUNT@DOMAIN.COM
```

Step 0: First Things First

- Create an account on GitHub.com



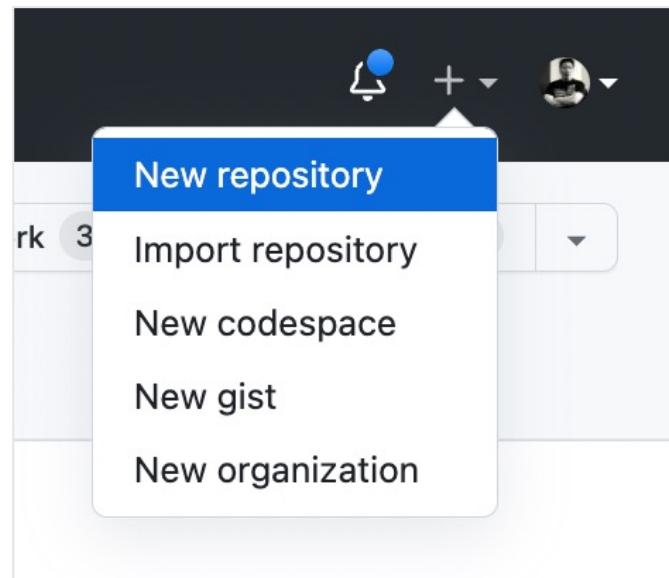
Step 0: First Things First

- Don't forget to get your bonus stuff
 - GitHub.com offers a **free software and service package** for students
 - <https://education.github.com/pack>

The screenshot shows the GitHub Education website with a focus on the Student Developer Pack. The top navigation bar includes links for Students, Teachers, Schools, Benefits, Events, and Sign in. The main heading is "GitHub Student Developer Pack". Below it, a paragraph explains the purpose of the pack: "Learn to ship software like a pro. There's no substitute for hands-on experience. But for most students, real world tools can be cost-prohibitive. That's why we created the GitHub Student Developer Pack with some of our partners and friends." A green button labeled "Sign up for Student Developer Pack" is prominently displayed. At the bottom, there is a link to share the pack: "Love the pack? Spread the word".

Step 1: Create a New Repository

- Click on the + button (upper-right corner) and select "New repository" (repo)



Step 1: Create a New Repository

- Create a new repository (repo) on GitHub.com

Create a new repository

A repository contains all project files, including the revision history.

Owner **Repository name ***

 charmgil ▾ / OSSL2023_make_example ✓

Great repository names are short and memorable. Need inspiration? How about [musical-fortnight](#)?

Description (optional)

 **Public**
Anyone can see this repository. You choose who can commit.

 **Private**
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾ | Add a license: **None** ▾ | ⓘ

Step 1: Create a New Repository

- Create a new repository (repo) on GitHub.com

Quick setup — if you've done this kind of thing before

or git@github.com:charmgil/OSSL2023_make_example.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

URL to the remote repo

...or create a new repository on the command line

```
echo "# OSSL2023_make_example" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin git@github.com:charmgil/OSSL2023_make_example.git  
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:charmgil/OSSL2023_make_example.git  
git branch -M main  
git push -u origin main
```

Step 2: Initialize Your Project Directory

- ‘**git init**’ under your project directory

```
$ ls
main.c      makefile      mylib.c      mylib.h
$ git init
Initialized empty Git repository in
/Users/charmgil/Projects/CODE.c/github_example/.git/
```

- Directory **.git** is created – This is your **local repo**

```
$ ls -a
.          .git      makefile  mylib.h
..         main.c    mylib.c
```

Step 2: Initialize Your Project Directory

- In case you want to stop tracking your project, simply remove the .git directory

```
$ ls -a
.          .git      makefile  mylib.h
..         main.c    mylib.c
$ rm -rf .git
```

Step 3: Check the Status

- ‘**git status**’ – Check the status of repo

```
$ git status
On branch master

No commits yet

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

  main.c
  makefile
  mylib.c
  mylib.h

nothing added to commit but untracked files present
(use "git add" to track)
```

Step 3: Check the Status

- ‘**git status**’ – Check the status of repo

```
$ git status
On branch master
No commits yet
-----
Untracked files:
  (use "git add <file>..." to include in what will be
committed)
  main.c
  makefile
  mylib.c
  mylib.h
-----
nothing added to commit but untracked files present
(use "git add" to track)
```

No commits yet; as we have just initialized it

Step 4: Create .gitignore

- In **.gitignore**, list items that you don't want to track
 - For example, exclude *.o files, which are the intermediate object files after compilation (*your collaborators do not need *.o files*)

```
$ touch .gitignore          # create a file
$ vi .gitignore             # list files/dirs to exclude

$ cat .gitignore
*.o
```

Step 5: Add Local Files to git

- Add the project files, that you want to track, to Git
 - `git add -A` or
`git add .`

```
$ git add -A
```

Step 5: Add Local Files to git

- Commit the changes to your local git repository (repo)
 - `git commit -m "first commit"`

```
$ git commit -m "first commit"
[master (root-commit) ef5376a] first commit
 5 files changed, 38 insertions(+)
  create mode 100644 .gitignore
  create mode 100644 main.c
  create mode 100644 makefile
  create mode 100644 mylib.c
  create mode 100644 mylib.h
```

Step 6: Associate the Repositories

- We have created two repos; We want to associate them
 - One on the local machine
 - Another on GitHub.com

```
$ git remote add origin https://github.com/charmgil/  
OSSL2020_make_example.git
```

Step 6: Associate the Repositories

- We have created two repos; We want to associate them
 - One on the local machine
 - Another on GitHub.com

```
$ git push -u origin master
Counting objects: 7, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 701 bytes | 701.00 KiB/s,
done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/charmgil/OSS2020_make_example.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master'
from 'origin'.
```

Completed: Your Project is on GitHub

The screenshot shows a GitHub repository page with the following details:

- Code tab is selected.**
- Issues: 0**, Pull requests: 0, Projects: 0, Wiki, Insights, Settings.
- No description, website, or topics provided.**
- Edit** button.
- Manage topics** link.
- Statistics:** 1 commit, 1 branch, 0 releases, 1 contributor.
- Branch:** master ▾, **New pull request**.
- Create new file**, **Upload files**, **Find File**, **Clone or download ▾**.
- Commits:**
 - charmgil** first commit (Latest commit ef5376a 19 minutes ago)
 - .gitignore (first commit, 19 minutes ago)
 - main.c (first commit, 19 minutes ago)
 - makefile (first commit, 19 minutes ago)
 - mylib.c (first commit, 19 minutes ago)
 - mylib.h (first commit, 19 minutes ago)
- Add a README with an overview of your project.** button.

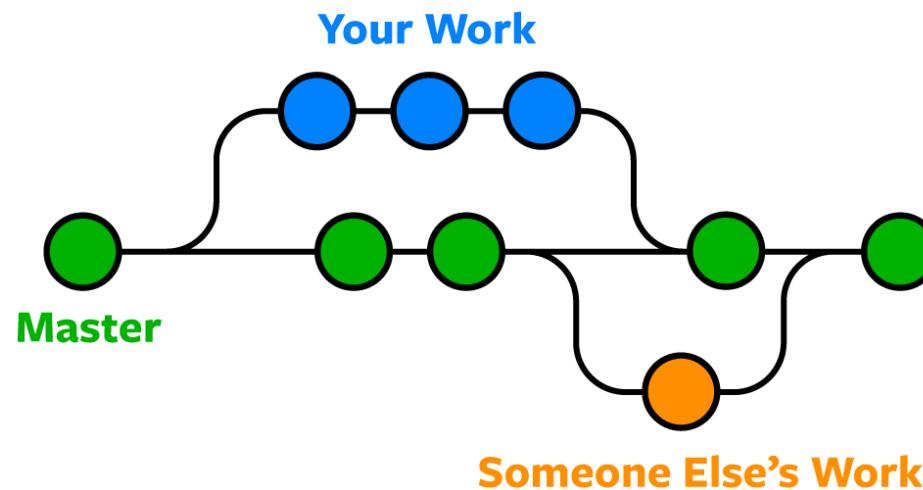
Your Project is on GitHub

- You have established your remote VCS on GitHub
 - You can access your project from anywhere in the world
- Now every developer in the world can watch your development!



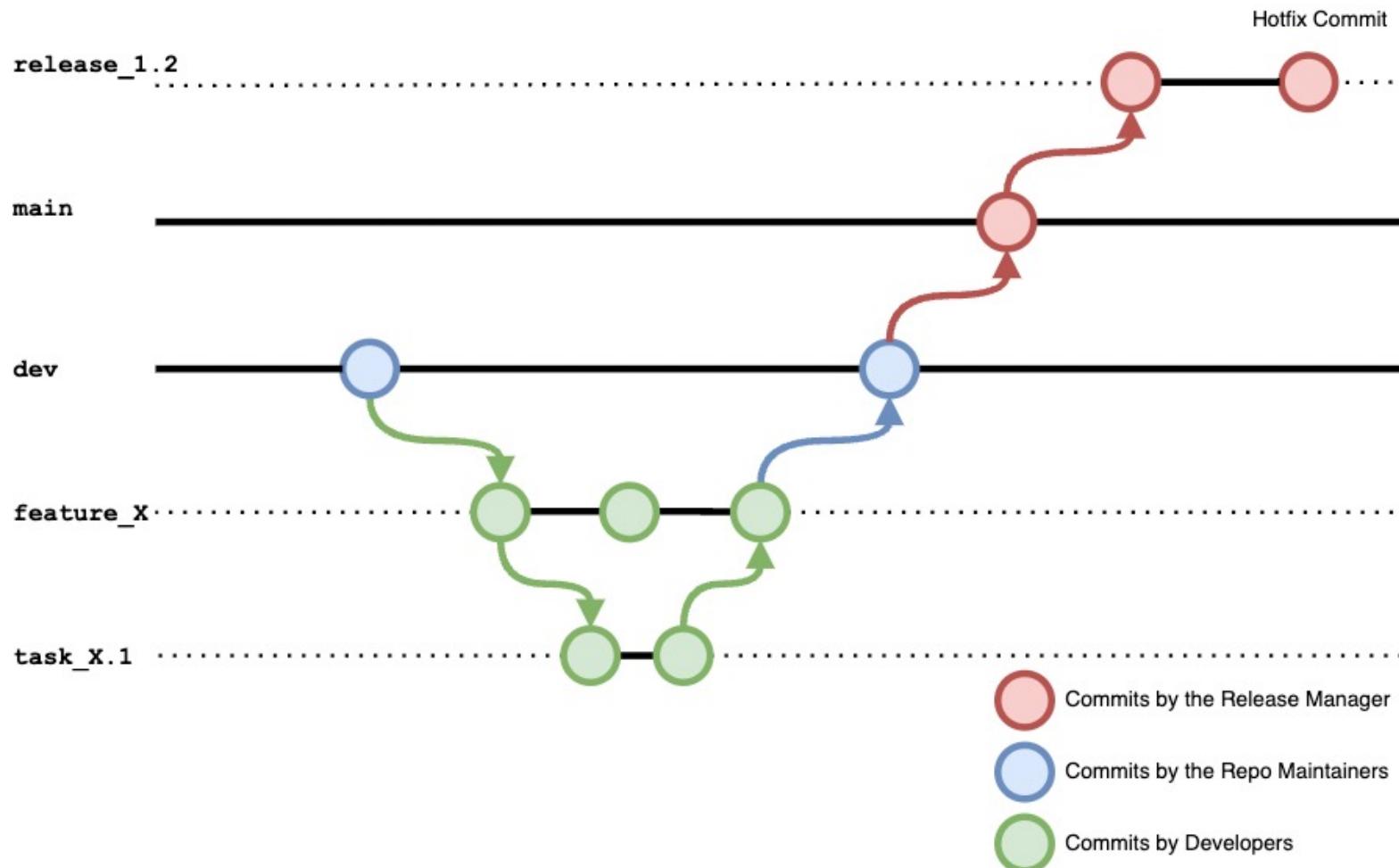
Other Important Functions

- Branching and merging



Other Important Functions

- Branching and merging

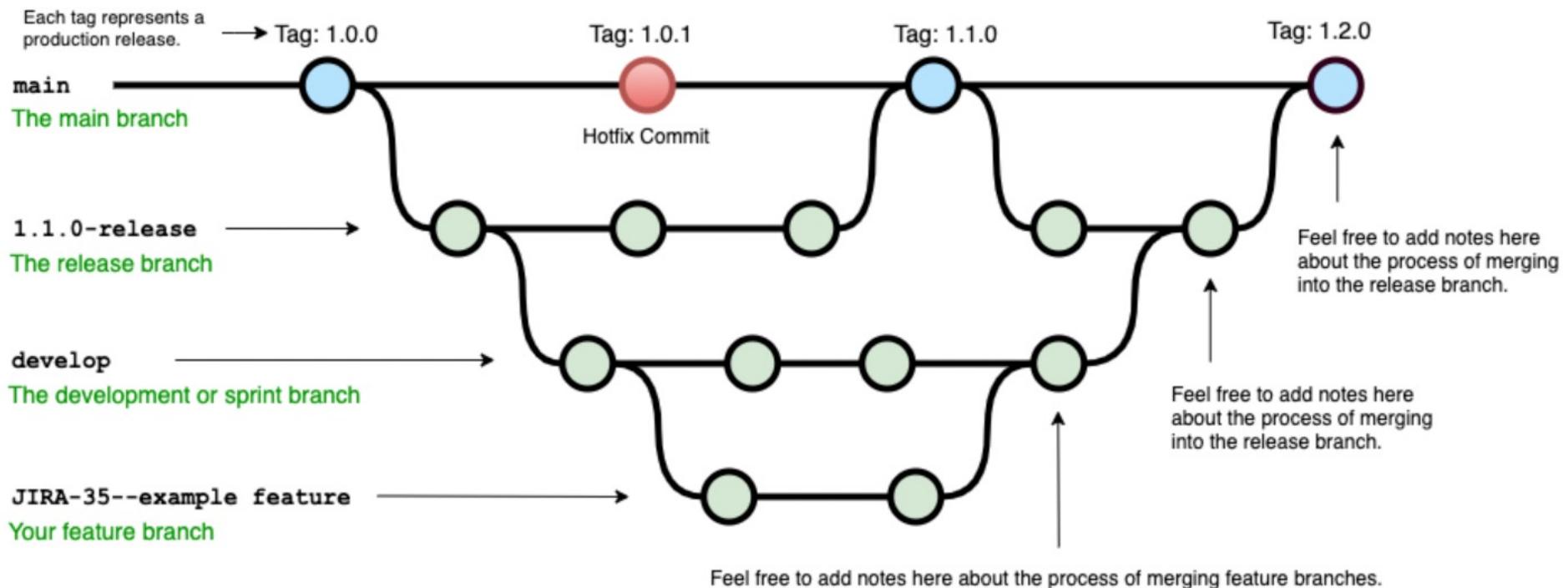


Other Important Functions

- Branching and merging

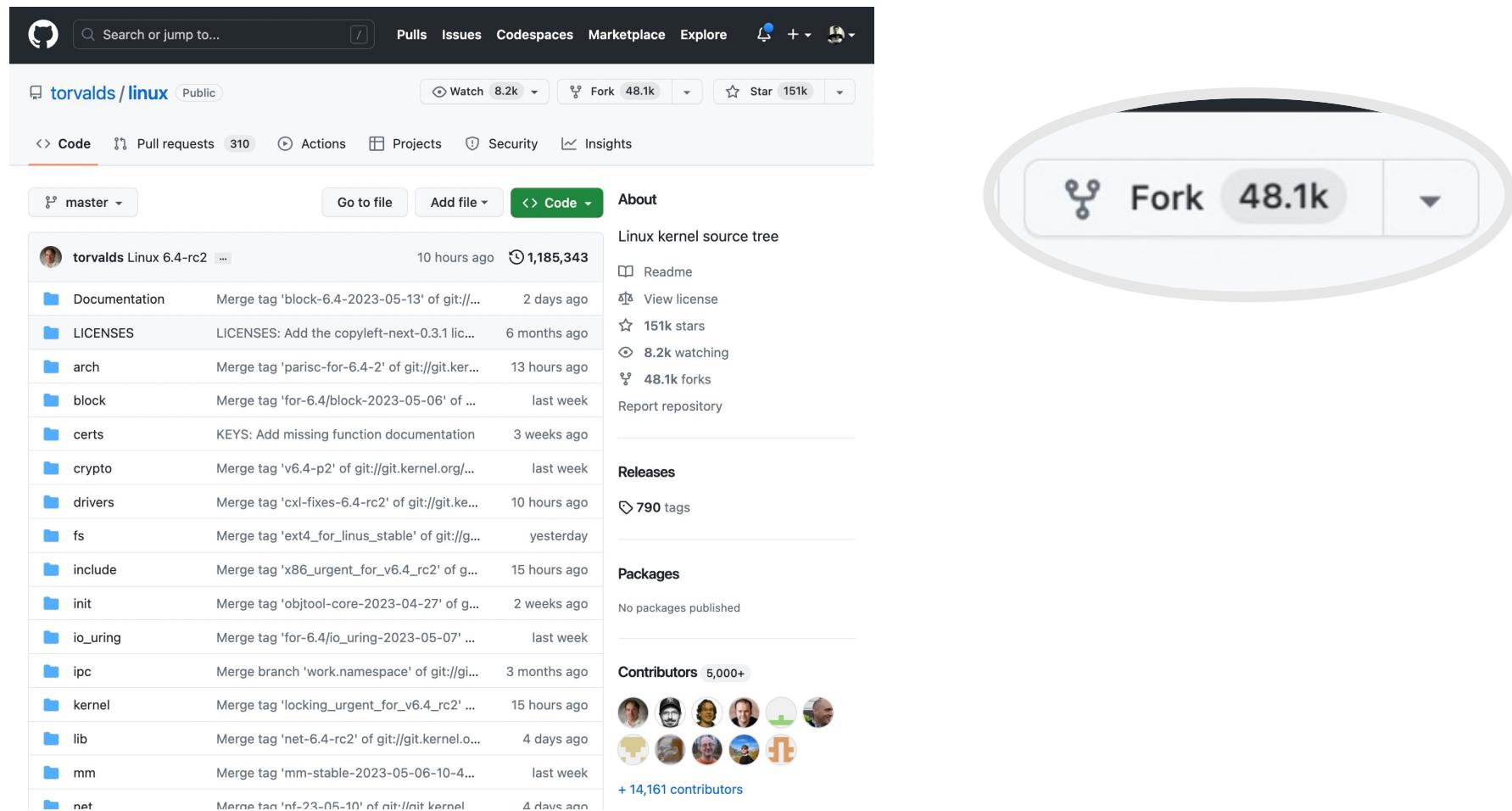
Example diagram for a workflow similar to "Git-flow" :

See: <https://nvie.com/posts/a-successful-git-branching-model/>



Other Important Functions

- Fork: Full copy of a repository residing on a different account



The screenshot shows the GitHub repository page for `torvalds/linux`. The top navigation bar includes links for Pulls, Issues, Codespaces, Marketplace, Explore, and user profile. Below the header, the repository name `torvalds/linux` is shown as public. The main content area displays a list of recent commits, each with a author icon, commit message, date, and a link to the commit details. A sidebar on the right provides summary statistics: 1,185,343 commits, 790 tags, and 5,000+ contributors. The most prominent feature is a large, rounded rectangular button labeled "Fork 48.1k" with a fork icon, which is circled in red.

Author	Commit Message	Date	Link
torvalds	Linux 6.4-rc2 ...	10 hours ago	1,185,343
Documentation	Merge tag 'block-6.4-2023-05-13' of git://...	2 days ago	
LICENSES	LICENSES: Add the copyleft-next-0.3.1 lic...	6 months ago	
arch	Merge tag 'parisc-for-6.4-2' of git://git.ker...	13 hours ago	
block	Merge tag 'for-6.4/block-2023-05-06' of ...	last week	
certs	KEYS: Add missing function documentation	3 weeks ago	
crypto	Merge tag 'v6.4-p2' of git://git.kernel.org/...	last week	
drivers	Merge tag 'cxl-fixes-6.4-rc2' of git://git.ke...	10 hours ago	
fs	Merge tag 'ext4_for_linus_stable' of git://g...	yesterday	
include	Merge tag 'x86_urgent_for_v6.4_rc2' of g...	15 hours ago	
init	Merge tag 'objtool-core-2023-04-27' of g...	2 weeks ago	
io_uring	Merge tag 'for-6.4/io_uring-2023-05-07' ...	last week	
ipc	Merge branch 'work.namespace' of git://gi...	3 months ago	
kernel	Merge tag 'locking_urgent_for_v6.4_rc2' ...	15 hours ago	
lib	Merge tag 'net-6.4-rc2' of git://git.kernel.o...	4 days ago	
mm	Merge tag 'mm-stable-2023-05-06-10-4...' ...	last week	
net	Merge tag 'nf-23-05-10' of git://init kernel	4 days ago	

