1 Lecture 2a: Open-Source Software Development Flow

1.1 Messages

- About 99% projects fail.
- Software is “soft”; Hardware is “hard”
- Automation is hard
- Nightly build concept (Microsoft)
- Agile software development
- Pair programming
- Extreme programming
- Opensource projects - Continuous Integration

1.2 Platforms

- https://github.com
- gitpod.io - cloud base
- Github’s Codespaces - cloud base
- Lubuntu
- Windows - MSVC++
- FydeOS (ChromeOS) - g++-13
- Android’s Termux - clang-17

1.3 Open-source Work Flow (Python)

1.4 Open-source Work Flow (C++)

1.5 Pull Request

1.6 GitHub, Git

```
git clone https://github.com/luk036/csdigit
cd csdigit
(edit)
git status
git diff
git diff README.md
git pull
git add .
git commit -m "message"
git push
git tag
git branch # list all branches
git branch develop # create a new branch
git switch develop
git switch master
```

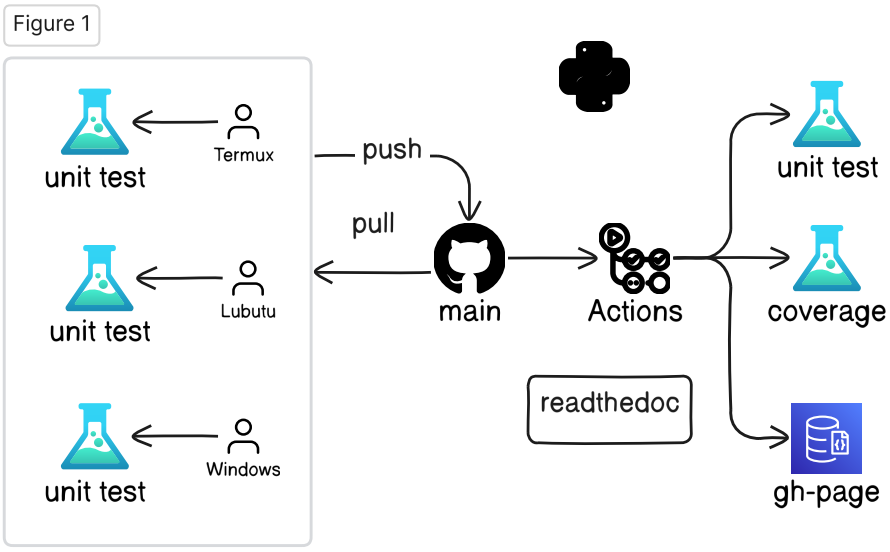

Figure 1: img

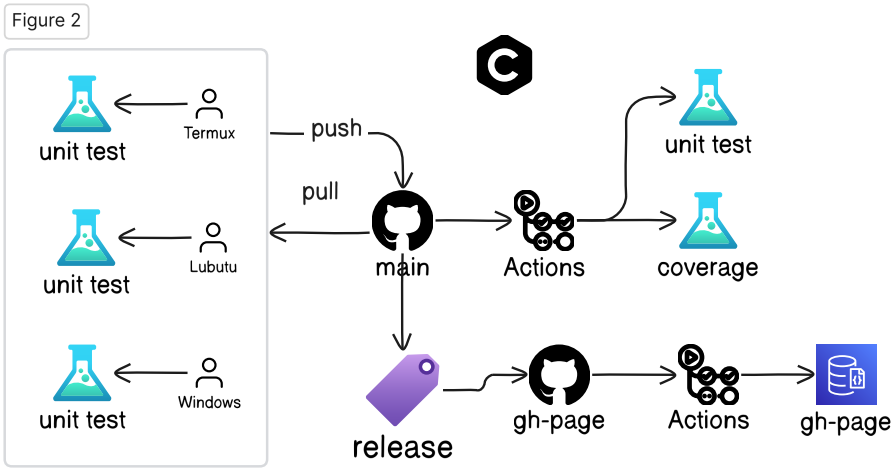

Figure 2: img

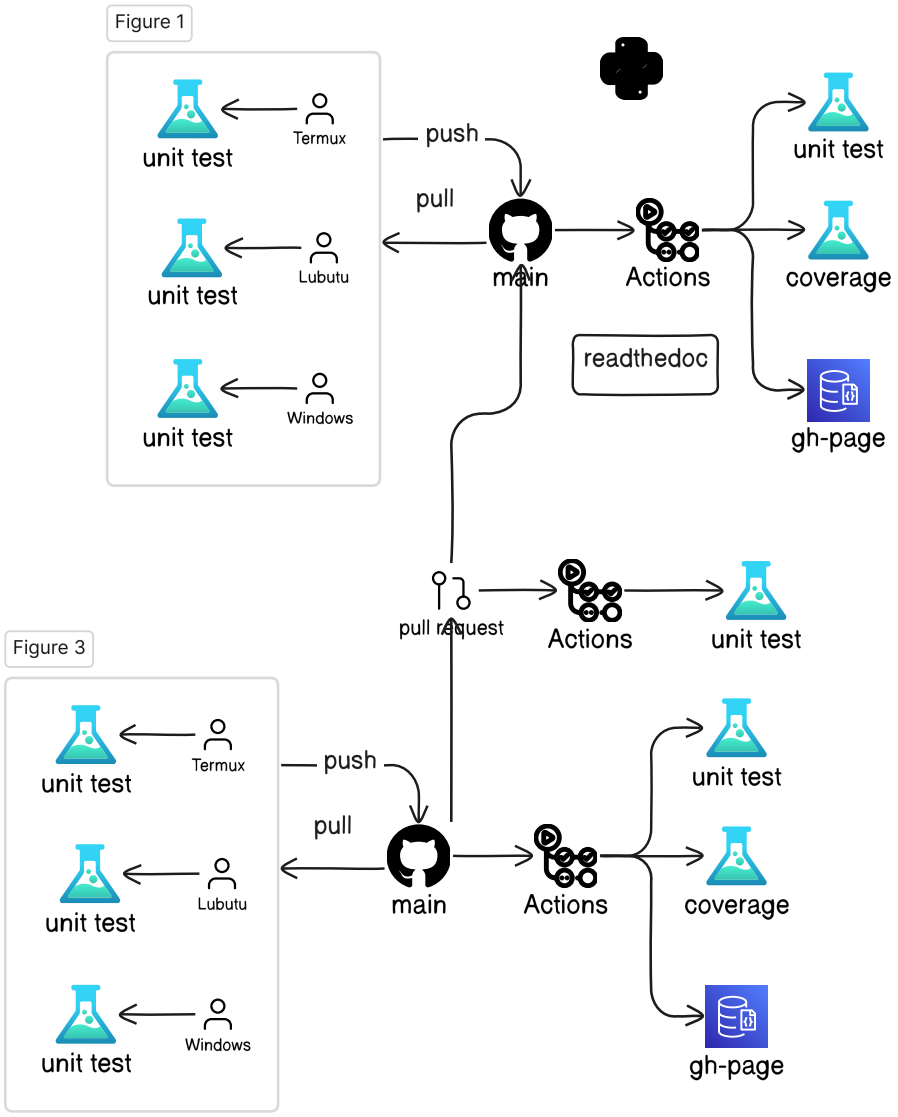
Figure 1
unit test
Termux
unit test
Lubutu
unit test
Windows
push
pull
main
Actions
unit test
coverage
readthedoc
gh-page
pull request
Actions
unit test
Figure 3
unit test
Termux
unit test
Lubutu
unit test
Windows
push
pull
main
Actions
unit test
coverage
gh-page

Figure 3: img

1.7 GitHub, gh

```
gh repo create csdigit --public
gh repo clone csdigit
gh run list
gh run view
gh release list
gh release create
gh issue list
gh issue create
gh search repos digraphx
```

1.8 Python

- Create a new porject

```
pip install pyscaffold[all]
putup -i --markdown --github-actions csdigit
```

- Setup

```
cd csdigit
pip install -e .
pip install -r requirements.txt
```

- Unit Testing

```
pytest
pytest --doctest-modules src
```

- Code Coverage

```
pytest --cov=src/csdigit
```

1.9 Python

- Formatting and static check

```
pip install pre-commit
pre-commit run --all-files
```

- Documentation

```
pip install -r docs/requirements.txt
cd docs
make html
python -m http.server
```

- Benchmarking

```
pytest benches/test_bench.py
```

1.10 Rust

- Create a new project

.font-sm.mb-xs[

```
cargo install cargo-generate
cargo generate -o --init --git https://github.com/rust-github/template.git
```

]

- Setup

```
cd csd-rs
cargo build
```

- Unit Testing

```
cargo test
cargo test --lib
cargo test --doc
```

- Code Coverage

```
cargo tarpaulin (Windows)
```

1.11 Rust

- Formatting and static check

```
cargo fmt
cargo clippy
cargo clippy --fix
```

- Documentation

```
cargo doc
cd target/doc
python -m http.server
```

- Benchmarking

```
cargo bench
```

1.12 C++ (CMake + CPM)

- Create a new project

  Use GitHub's ModernCppStarter template,

- Setup

```
cd csd-cpp
cmake -Sall -Bbuild -DCMAKE_BUILD_TYPE=Release
cmake --build build
```

- Unit Testing

```
cmake --build build --target test
```

- Code Coverage

```
??
```

1.13 C++ (CMake + CPM)

- Formatting and static check

```
pip install cmake-format clang-format
cmake -Sall -Bbuild -DCMAKE_BUILD_TYPE=Release
cmake --build build --target fix-format
```

- Documentation

```
cmake --build build --target GenerateDocs
```

- Benchmarking

```
./build/bench/BM_switch
```

1.14 C++ (XMake)

- Create a new project

```
xmake create -t static lds-cpp
xmake create -t console csd-cpp
```

- Setup

```
xmake f -m debug
xmake
```

- Unit Testing

```
xmake run test_csd
```

- Code Coverage

```
??
```

1.15 C++ (XMake)

- Formatting and static check

```
xmake format
```

- Documentation

```
xmake doxygen
```

- Benchmarking

```
xmake run test_bench
```

2 Lecture 2b: Programming in the Age of AI

2.1 Coding Tips

- Test, test, test!!!
- Write cleaner code
- Refactor repeat codes
- Object oriented programming
- Generic programming
- Design Pattern
- Coroutine is your friend
- Learn from good codes, not bad ones.
- The last rescue: Google search.

2.2 Code generation

- AWS CodeWhisperer (VSCode's extension)
  - generate testing code

2.3 Documentation generation

Mintlify (VSCode's extension)

- Naming
- a, i, p, n
- A x = b
- x: unknown, x_axis
- x, y, z

2.4 Use better variable names

- p: point, polygon, polynomial, prev
- t: time, target, temp
- c: cost, cycle, coefficient
- d: distance, distribution
- e: edge
- v: vertex
- u, v, w: vertex1, vertex2
- i: index
- i, j: row, col
- i, j, k
- l, m: line1, line2
- n: dimension, node, next
- n, m: ndim, mdim

- w: weight, frequence (omega)

2.5 Performance Tips

- Avoid string comparison
- Use sentinel
- Use cheaper measure, avoid `sqrt()`, `sin()`, `cos()`
- Lazy evaluation
- Table lookup
- Avoid sequence search:
  - Backward pointers
  - Hash Table/Dictionary/Map

2.6 Avoid string comparison

.pull-left[

Bad :

```
if pin == "input":
    # ...
elif pin == "output":
    # ...
elif pin == "in_out":
    # ...
elif pin == "dont_care":
    # ...
else:
    # ...
```

]

.pull-right[

Better :

```
pin_type = dict({"input":0},
  {"output":1}, {"in_out":2},
  {"dont_care":3})
...
id = pin_type.get(pin, -1)
if id == 0:
    # ...
elif id == 1:
    # ...
elif id == 2:
    # ...
elif id == 3:
    # ...
```

```
else:
    # ...
```

]

2.7 Use Sentinel

.pull-left[

Bad :

.font-sm.mb-xs[

```
max = 0
bckt = [Dllist() for _ in range(high)]
# ...
def popleft():
    res = bckt[max].popleft()
    while max >= 0 and bckt[max].empty():
        max -= 1
    return res
```

]

] .pull-right[

Better :

.font-sm.mb-xs[

```
max = 0
sentinel = Dllink()
bckt = [Dllist() for _ in range(high+1)]
bckt[0].append(sentinel)  # sentinel
# ...
def popleft():
    res = bckt[max].popleft()
    while bckt[max].empty():
        max -= 1
    return res
# Saved a boundary check `max >= 0`
```

]

]

2.8 Use cheaper measure

.pull-left[

Bad :

```
mind = 10000
maxd = 0
for u, v in G.edges():
    t = vec[u] - vec[v]
*   d = sqrt(t.dot(t))
    if mind > d: mind = d
    if maxd < d: maxd = d
*return maxd - mind
```

] .pull-right[

Better :

```
minq = 10000
maxq = 0
for u, v in G.edges():
    t = vec[u] - vec[v]
*   q = t.dot(t)
    if minq > q: minq = q
    if maxq < q: maxq = q
*return sqrt(maxq) - sqrt(minq)
```

]

2.9 Another Example

.pull-left[

Bad :

```
mind = 10000
maxd = 0
for u, v in G.edges():
*   t = 1 - vec[u].dot(vec[v])
*   d = arcsin(sqrt(t))
    if mind > d: mind = d
    if maxd < d: maxd = d

*return maxd - mind
```

] .pull-right[

Better :

```
minq = 10000
maxq = 0
for u, v in G.edges():
*   q = 1 - vec[u].dot(vec[v])
    if minq > q: minq = q
    if maxq < q: maxq = q
```

```
*return arcsin(sqrt(maxq)) \
*        - arcsin(sqrt(minq))
]
```

2.10 Optimization Tips

- Convex optimization
- Network optimization
- Primal-dual paradigm