

크래프톤 정글 10기 WEEK10-11

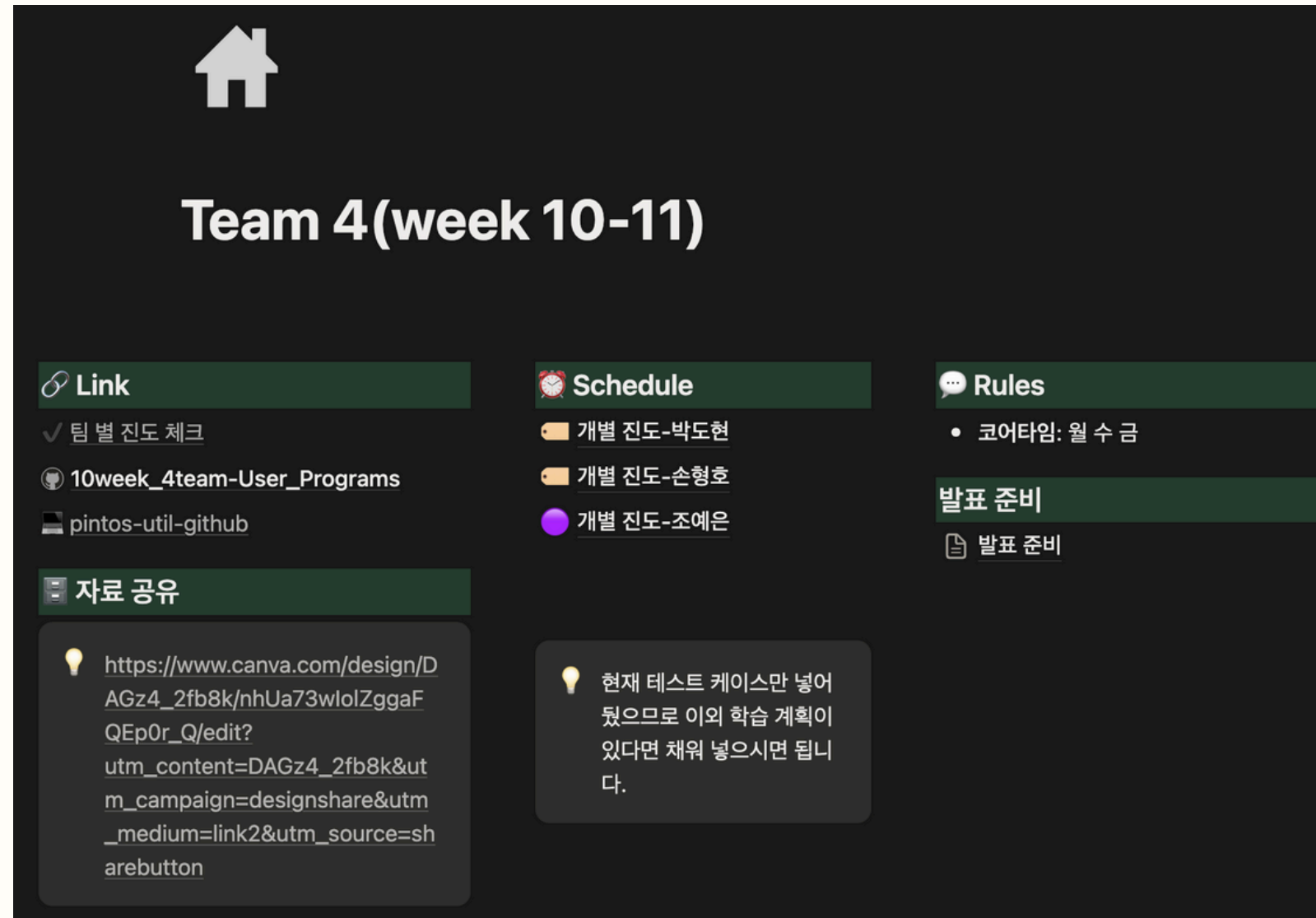
# Pintos Project II

## 협업 과정과 이슈

---

TEAM04 손형호, 조예은, 박도현

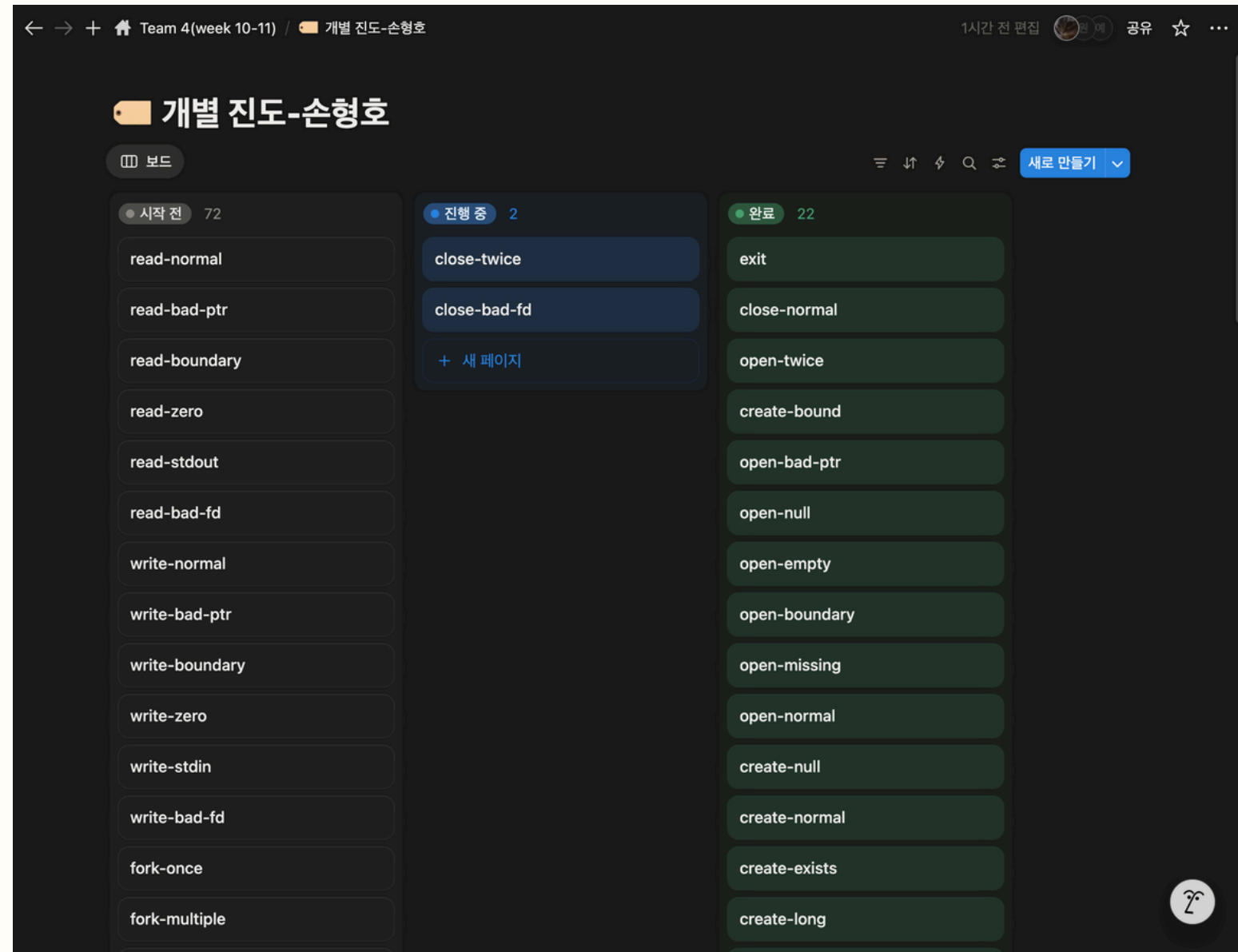
# TEAM NOTION



팀 노션 페이지를 제작

각종 링크나 자료, 칸반보드와 같은  
팀 활동과 관련된 요소들을  
한 곳에서 효율적으로 관리

# 개별 칸반보드 관리

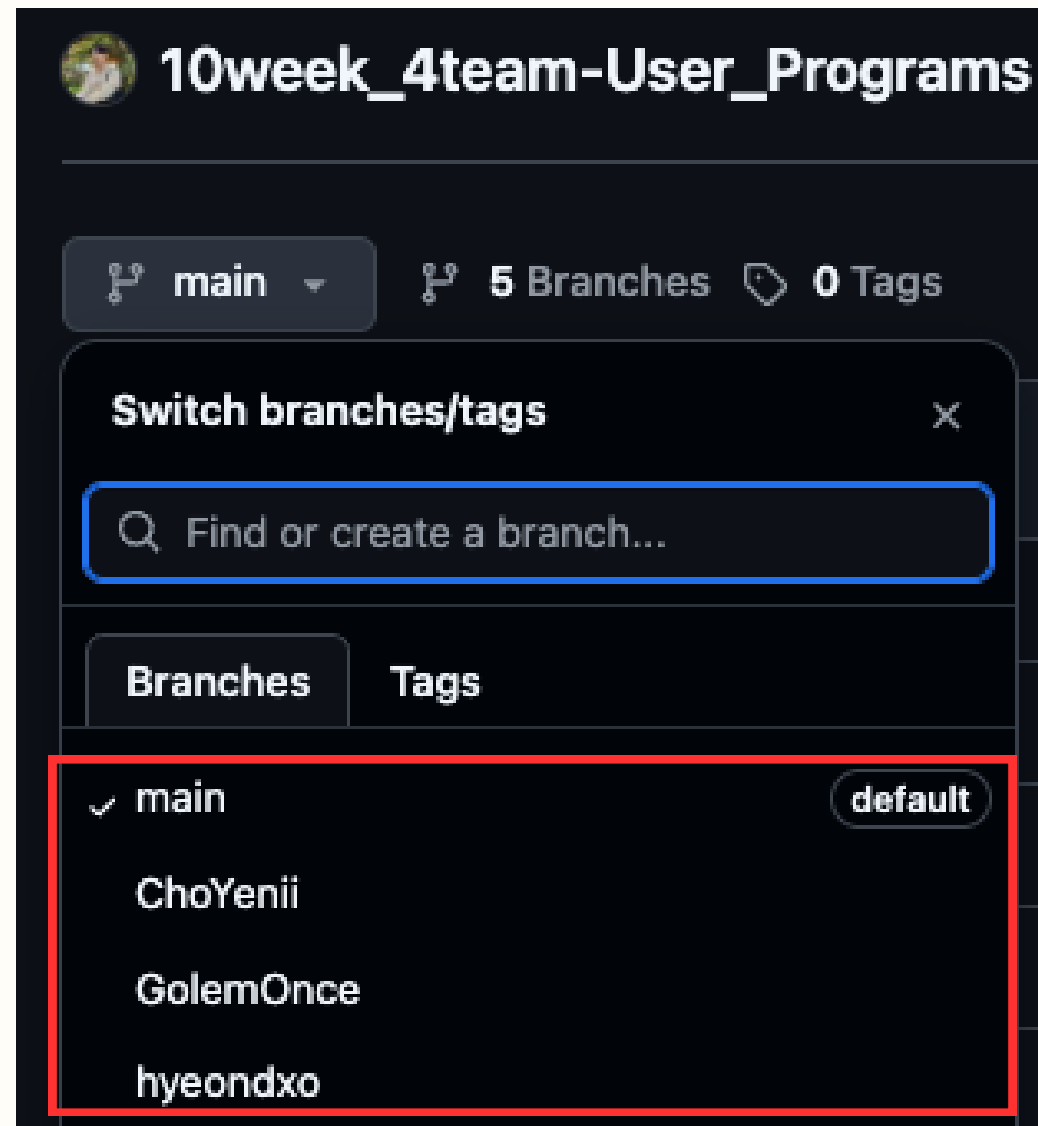


개별 진도를 칸반보드로 시각화

시작 전, 진행 중, 완료된 할 일들을  
직접 수정하고 서로 공유

팀원 간 진행 상황의 흐름을 한 눈에 확인  
효율적으로 작업의 우선순위 선정

# Github Team Repository



Project 1까지의 완성된 소스코드를  
initial commit

→ 팀원 모두가 같은 소스코드로부터 시작

이후엔 각자의 브랜치를 만들고  
각 브랜치에서 개별 작업을 진행

# CORE TIME

🔒 10주차-4조

📧 메시지 📁 캔버스 추가 ⚙️ 고장 📄 파일 +

```
for (int fd = 2; fd < FD_TABLE_SIZE; fd++) { // fd 0,1(STDIN/STDOUT) 제외하고 부모의 fd와 같은 fd는 제외
    struct file *pf = parent->fd_table[fd]; // 부모 fd 슬롯 확인
    if (pf != NULL) { // 실제로 열려있는 파일이라면
        struct file *dup =
            file_duplicate(pf); // 동일 inode를 참조하는 새 file 구조체 생성 (pos/deny_write 복제)
        if (dup == NULL) { // 복제 실패 시 더이상 진행 불가
            succ = false;
            break;
        }
        current->fd_table[fd] = dup; // 자식 fd 테이블의 동일 번호 슬롯에 저장
    }
}

if (succ && parent->running_file != NULL) { // 부모가 실행 파일 핸들을 보유 중이면
    current->running_file = file_duplicate(parent->running_file); // 자식도 실행 파일 핸들 복제
    if (current->running_file == NULL)
        succ = false; // 복제 실패 시 오류 처리
    else
        file_deny_write(current->running_file); // 자식 실행 파일에도 write 금지 유지 (ROX 보호)
}
lock_release(&filesys_lock); // 파일 복제 작업 끝났으니 락 해제
}
```

손형호(정글10기-17) 오후 9:27

0Kernel PANIC at ../threads/thread.c:329 in thread\_current(): assertion 'is\_thread(t)' failed.  
Call stack: 0x80042194cc 0x80042071f9 0x800420af87 0x800421cb7e 0x800421cd63 0x800421cd43 0x80042177ba 0x800421772a 0x8004216e88 0x800421cbe2 0x800421681f .  
The 'backtrace' program can make call stacks useful.  
Read "Backtraces" in the "Debugging Tools" chapter  
of the Pintos documentation for more information.  
Kernel PANIC recursion at ../threads/thread.c:329 in thread\_current().  
Interrupt 0x0d (#GP General Protection Exception) at rip=800422177b  
cr2=0000000000000000 error= 0  
rax ccccccccccccccc rbx 0000000000000000 rcx 00000080042228b5 rdx 0000000000000027  
rsp 0000008004240f10 rbp 0000008004240f20 rsi 0000000000000149 rdi ccccccccccccccc  
rip 000000800422177b r8 000000800422294e r9 00000080042193ea r10 0000000000000000  
r11 0000000000000000 r12 0000000000000000 r13 0000000000000000 r14 0000000000000000  
r15 0000000000000000 rflags 00000002  
es: 0010 ds: 0010 cs: 0008 ss: 0010  
Interrupt 0x0d (#GP General Protection Exception) at rip=8004221237

박도현(10기-34) 오후 10:02

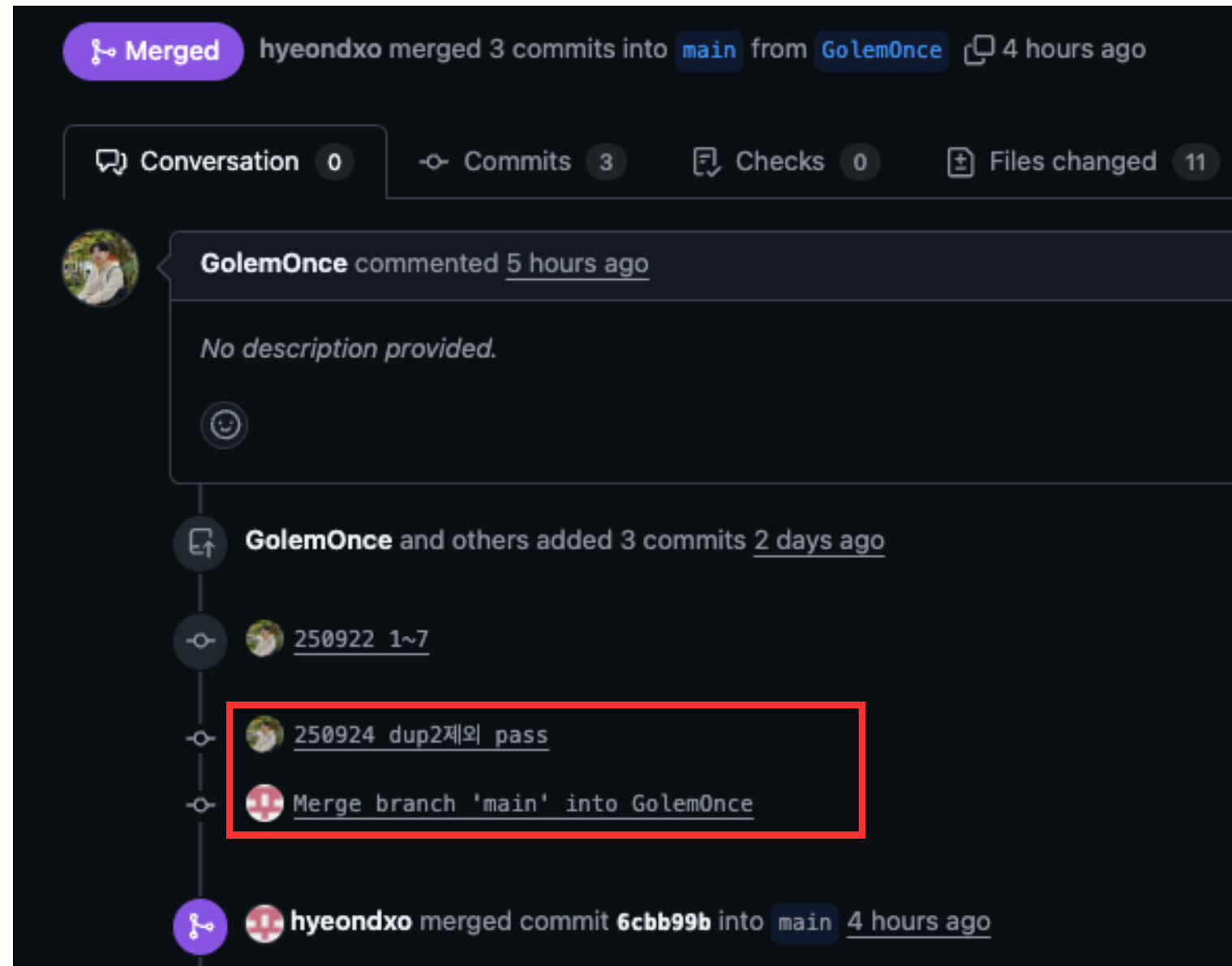
<https://github.com/hyeondxo/Pintos/commit/a279ac88e2383255331b7c6a4d553de36aa82353>  
argument passing 성공 커밋 링크입니다..!

GitHub

서로의 진행상황, 코드, 오류 로그 등을 공유  
→ 팀 전체의 진행 상황을 동기화

한 팀원의 오류를 함께 해결하거나  
서로 질문과 답변을 주고받으며  
프로젝트를 점진적으로 완성

# force push로 인한 git 충돌과 소스코드 꼬임 발생



눈에 보이는 소스코드의  
충돌을 해결한 후 main merge하였지만

히스토리 간의 충돌은 해결되지 않았음

→ git pull 이후 빌드 시  
함수 중복 정의 등 소스코드가 꼬이며  
수많은 빌드 오류 발생

# Conflict Issue 해결 과정

## Revert "dup2 제외 all pass" #3

Merged GolemOnce merged 1 commit into `main` from `revert-2-GolemOnce` 4 hours ago

```
jungle@0cdf587793ce:/workspaces/team4/pintos/userprog$ git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
jungle@0cdf587793ce:/workspaces/team4/pintos/userprog$ git reset --hard GolemOnce
HEAD is now at de1f486 250924 dup2제외 pass
jungle@0cdf587793ce:/workspaces/team4/pintos/userprog$ git push --force-with-lease origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/GolemOnce/10week_4team-User_Programs.git
+ cdd99e4...de1f486 main -> main (forced update)
jungle@0cdf587793ce:/workspaces/team4/pintos/userprog$ git log --oneline
de1f486 (HEAD -> main, origin/main, GolemOnce) 250924 dup2제외 pass
633f21b 250922 1~/
1ade653 init
df7367c init
```

1. Revert로 PR을 되돌려 main을 초기화  
(히스토리 보존)

2. Reset으로 로컬의 main을  
백업 브랜치의 커밋 시점과 맞춤

3. 원격 main을 로컬 main과 맞추어  
정상 동작 소스코드로 덮어쓰

# 4팀 협업 결과

TOTAL TESTING SCORE: 100.0%  
ALL TESTED PASSED -- PERFECT SCORE

## SUMMARY BY TEST SET

Test Set	Pts	Max	% Ttl	% Max
tests/threads/Rubric.alarm	7/	7	2.0%/	2.0%
tests/threads/Rubric.priority	25/	25	3.0%/	3.0%
tests/userprog/Rubric.functionality	40/	40	40.0%/	40.0%
tests/userprog/Rubric.robustness	40/	40	30.0%/	30.0%
tests/userprog/no-vm/Rubric	3/	3	10.0%/	10.0%
tests/filesys/base/Rubric	17/	17	15.0%/	15.0%
Total			100.0%/	100.0%

정상적인 빌드 이후  
make test → 모든 인원 All Passed ✓