Windows + WSL Codex/PDF Merger 완전 설치 매뉴얼 (한글)

이 문서는 github.com/ffkaka/pdf-merger 프로젝트를 **Windows 환경에서 WSL2 기반으로 안정적으로 실행**하기 위한 전체 절차를 설명한다.
기준 날짜: **2026-02-22**

핵심 원칙:
- Codex CLI의 Windows 네이티브 지원은 실험적일 수 있으므로, 실사용은 **WSL2(Ubuntu)** 기준으로 구성한다.
- 프로젝트의 agent/skill 및 pmerge 실행도 모두 WSL에서 표준화한다.

---

1. 사전 준비 (Windows)

관리자 권한 PowerShell을 열고 아래를 순서대로 실행한다.

```
winget install --id Git.Git -e
winget install --id OpenJS.NodeJS.LTS -e
winget install --id Python.Python.3.12 -e
wsl --install -d Ubuntu
```

설치 후 Windows를 재부팅한다.

확인:

```
git --version
node --version
npm --version
python --version
wsl --status
```

---

2. WSL(Ubuntu) 초기 설정

Ubuntu를 실행하고 Linux 사용자 계정을 만든다. 이후:

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y build-essential python3 python3-pip python3-venv ghostscript poppler-utils
```

권장 설정:

```
git config --global user.name "YOUR_NAME"
git config --global user.email "YOUR_EMAIL"
```

---

3. Codex CLI 설치 (WSL 내부)

공식 기본 설치(2026-02-22 기준):

```
npm install -g @openai/codex
```

인증(권장):

```
codex --login
```

또는 API 키 방식:

```
echo 'export OPENAI_API_KEY="YOUR_OPENAI_API_KEY"' >> ~/.bashrc
source ~/.bashrc
```

동작 확인:

```
codex --version
codex
```

---

4. 프로젝트 설치 (WSL 홈 기준)

요구사항대로 WSL 홈에 설치:

```
cd ~
git clone https://github.com/ffkaka/pdf-merger.git
cd ~/pdf-merger
```

디렉터리 확인:

```
pwd
ls -la
```

기대 경로:
- 프로젝트 루트: ~/pdf-merger
- 입력 PDF 폴더: ~/pdf-merger/japan
- 결과 폴더: ~/pdf-merger/output/pdf
- 임시 폴더: ~/pdf-merger/tmp/pdfs

---

5. Skill/Agent 실행 준비

이 프로젝트는 AGENTS.md 지침에 따라 동작한다.
PDF 관련 작업 시 Codex에 pdf skill을 사용하게 하거나, 필요 시 아래를 실행:

```
codex skill install pdf
```

주의:
- PDF 검색은 텍스트 추출 기반으로 수행한다.
- 최종 병합 산출물은 output/pdf/에 생성한다.
- 중간 산출물/리포트는 tmp/pdfs/를 사용한다.

---

6. pmerge 사용법 (WSL 내부)

프로젝트 루트에서 실행:

```
./pmerge <폴더명> [키워드1 키워드2 ...]
```

예시:

```
cd ~/pdf-merger
./pmerge japan IC-211 IC-212 IC-220
```

키워드 파일 기반(권장):

```
PMERGE_KEYWORDS_FILE=/home/<USER>/pdf-merger/keywords.txt ./pmerge japan
```

드라이런:

```
PMERGE_EXTRA_ARGS="--dry-run" ./pmerge japan IC-211 IC-212
```

옵션 환경변수:
- PMERGE_OUTPUT_DIR (기본: ~/pdf-merger/output/pdf)
- PMERGE_OUTPUT_NAME (기본: merged_keywords.pdf)
- PMERGE_TMP_DIR (기본: ~/pdf-merger/tmp/pdfs)
- PMERGE_KEYWORDS_FILE (키워드 파일 경로)
- PMERGE_EXTRA_ARGS (내부 Python 실행 옵션 전달)

기본 용량 정책:
- 출력 PDF는 기본 2MB 제한을 적용한다.
- 2MB를 넘으면 자동으로 인덱스 분할 파일을 생성한다.
  - 예: merged_keywords_01.pdf, merged_keywords_02.pdf

6-1) PMERGE_EXTRA_ARGS 상세 옵션 전체

PMERGE_EXTRA_ARGS는 keyword_merge.py에 그대로 전달된다.

지원 옵션:
- --dry-run
- --case-sensitive
- --no-recursive
- --match-mode content|filename
- --keywords-file <파일경로>

아래는 각 옵션의 의미와 실사용 예시다.

1. --dry-run
- 기능: 병합 파일을 만들지 않고, 어떤 파일이 매칭되는지만 출력한다.
- 사용 예:

```
PMERGE_EXTRA_ARGS="--dry-run" pmerge japan IC-211 IC-212
```

2. --case-sensitive
- 기능: 대소문자를 구분해서 키워드를 찾는다.
- 기본값: 대소문자 무시(case-insensitive)
- 사용 예:

```
PMERGE_EXTRA_ARGS="--case-sensitive" pmerge japan ic-211
```

3. --no-recursive
- 기능: 하위 폴더를 탐색하지 않고, 지정한 폴더의 최상위 PDF만 검색한다.
- 사용 예:

```
PMERGE_EXTRA_ARGS="--no-recursive" pmerge japan IC-211
```

4. --match-mode content
- 기능: PDF 본문 텍스트를 추출해서 키워드를 매칭한다. (기본값)
- 특징: 정확도는 높지만 PDF가 많으면 시간이 더 걸릴 수 있다.
- 사용 예:

```
PMERGE_EXTRA_ARGS="--match-mode content" pmerge japan IC-211
```

5. --match-mode filename
- 기능: PDF 파일명에 키워드가 포함되는지만 검사한다.
- 중요: 이 모드에서는 PDF 내용(본문 텍스트)을 읽지 않는다.
- 사용 예:

```
PMERGE_EXTRA_ARGS="--match-mode filename" pmerge japan IC-211 IMC-211
```

6. --keywords-file <파일경로>
- 기능: 키워드를 명령행 인자 대신 파일에서 읽는다. (한 줄에 키워드 1개)
- 파일 예 (keywords.txt):

```
IC-211
IC-212
IC-220
```

- 사용 예:

```
PMERGE_EXTRA_ARGS="--keywords-file /home/<USER>/pdf-merger/keywords.txt" pmerge japan
```

같은 기능을 더 간단히 쓰려면 PMERGE_KEYWORDS_FILE을 사용한다.

```
PMERGE_KEYWORDS_FILE=/home/<USER>/pdf-merger/keywords.txt pmerge japan
```

6-2) PMERGE_EXTRA_ARGS 조합 예시

1. 파일명 기준 + 대소문자 구분 + 드라이런:

```
PMERGE_EXTRA_ARGS="--match-mode filename --case-sensitive --dry-run" pmerge japan IC-211
```

2. 본문 검색 + 하위폴더 제외:

```
PMERGE_EXTRA_ARGS="--match-mode content --no-recursive" pmerge japan IC-211 IC-212
```

3. 키워드 파일 + 파일명 기준:

```
PMERGE_KEYWORDS_FILE=/home/<USER>/pdf-merger/keywords.txt PMERGE_EXTRA_ARGS="--match-mode filename" pmerge japan
```

6-3) 자주 하는 실수

1. PMERGE_EXTRA_ARGS를 따옴표 없이 입력
- 공백이 포함된 옵션 문자열은 반드시 큰따옴표로 감싼다.

2. --keywords-file 경로를 Windows 경로로 입력
- WSL 내부 경로(/home/...)를 사용한다.

3. 파일명 기준이 필요한데 --match-mode를 지정하지 않음
- 기본은 content이므로, 파일명 기준이면 반드시 --match-mode filename을 넣는다.

4. 출력 파일명이 기본값으로 덮어써지는 문제
- PMERGE_OUTPUT_NAME으로 작업별 파일명을 지정한다.
- 예:

```
PMERGE_OUTPUT_NAME=shipment_A.pdf PMERGE_KEYWORDS_FILE=/home/<USER>/pdf-merger/keywords.txt pmerge japan
```

---

7. Codex 프롬프트 예시 (머지 + 검증)

아래 예시는 Codex 대화창(프롬프트)에서 그대로 사용할 수 있다.

7-1) 왜 @경로를 붙여야 하나

Codex 프롬프트에서 @경로를 붙이면 해당 파일/폴더를 작업 문맥으로 명확히 지정할 수 있다.

장점:
- 어떤 파일을 기준으로 작업할지 모호성이 줄어든다.
- 현재 디렉터리 기준 상대경로를 그대로 써도 대상이 명확해진다.
- 사용방법 같은 추상 질의에서도 프로젝트 규칙(@AGENTS.md)을 근거로 일관된 답을 받기 쉽다.

예:
- @AGENTS.md
- @japan
- @tmp/pdfs/keywords.txt

7-2) 가장 간단한 실행 예시 (상대경로 + 키워드 파일)

```
@AGENTS.md 규칙대로 병합해.
입력 폴더는 @japan, 키워드 파일은 @tmp/pdfs/keywords.txt 를 사용해.
PMERGE_KEYWORDS_FILE 방식으로 실행하고 결과는 @output/pdf 에 저장해.
머지 후 결과 PDF와 @tmp/pdfs/keyword_merge_report.json 존재 여부, 파일 크기, 첫 페이지 렌더링까지 검증해.
검증 결과까지 요약해서 보고해.
```

7-3) 파일명 기준 머지(본문 미검색) 예시

```
@japan 폴더를 파일명 기준으로만 필터링해서 병합해.
키워드는 @tmp/pdfs/keywords.txt 파일에서 읽고,
PMERGE_KEYWORDS_FILE + PMERGE_EXTRA_ARGS="--match-mode filename" 조합으로 실행해.
2MB 초과 시 분할 파일(_01, _02 ...) 생성 여부까지 검증해.
```

7-4) 드라이런 후 실제 실행 예시

```
먼저 드라이런으로 매칭 개수 확인하고, 그다음 실제 머지를 수행해.
키워드 파일은 @tmp/pdfs/keywords.txt, 입력 폴더는 @japan.
드라이런 결과와 실제 생성 결과를 비교해서 누락 여부를 확인해.
```

7-5) 출력 파일명 지정 + 검증 자동화 예시

```
@tmp/pdfs/keywords.txt 키워드 파일로 병합해.
출력 파일명은 shipment_2026_02.pdf 로 지정하고,
완료 후 생성된 PDF 목록, 파일 크기, 리포트 match_count를 표 형태로 정리해.
```

7-6) "사용방법 알려줘" / "사용방법" 요청 시 권장 프롬프트

아래처럼 @AGENTS.md와 실제 입력 경로를 함께 주면, Codex가 실행 가능한 형태로 안내하기 쉽다.

```
@AGENTS.md 기준으로 이 프로젝트 사용방법 알려줘.
특히 @japan 폴더와 @tmp/pdfs/keywords.txt 를 사용해서
필터링, 병합, 2MB 초과 시 분할 생성, 검증까지 한 번에 하는 절차를
초보자도 따라할 수 있게 명령어 중심으로 설명해줘.
```

검증 최소 체크리스트(프롬프트에 함께 넣기 권장):
- 결과 PDF 존재 여부 (output/pdf/*.pdf)
- 리포트 존재 여부 (tmp/pdfs/keyword_merge_report.json)
- 키워드별 match_count 확인
- 샘플 페이지 렌더링 검사(깨짐/누락/겹침)

---

8. pmerge를 전역 명령으로 등록 (WSL)

WSL 어디서든 pmerge 실행하려면:

```
mkdir -p ~/.local/bin
ln -sf ~/pdf-merger/pmerge ~/.local/bin/pmerge
echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

검증:

```
which pmerge
pmerge japan IC-211
```

---

9. 자주 발생하는 문제

1. codex: command not found
- 원인: 전역 npm bin 경로 미반영
- 해결:

```
echo 'export PATH="$(npm bin -g):$PATH"' >> ~/.bashrc
source ~/.bashrc
```

2. Permission denied: ./pmerge
- 해결:

```
chmod +x ~/pdf-merger/pmerge
```

3. PDF 라이브러리 미설치 오류
- 현재 프로젝트 스크립트는 pypdf/pdfplumber가 없으면 ghostscript(gs) fallback을 사용한다.
- gs 확인:

```
which gs
```

---

10. 운영 권장사항

- 대량 병합 작업은 WSL 내부에서 실행한다.
- 결과물은 output/pdf/만 최종 산출물로 취급한다.
- tmp/pdfs/는 중간 산출물/리포트 용도로 유지한다.
- 키워드가 많으면 파일(--keywords-file 또는 PMERGE_KEYWORDS_FILE) 기반으로 관리한다.

---

11. 공식 문서 링크

- Codex 소개: https://openai.com/codex/
- Codex CLI 시작 문서: https://developers.openai.com/codex/cli