skills个人经验

1.按照仓库配置环境，我个人强烈推荐使用uv，例如uv pip install -e .

2.在.mcp_env设置相关任务的key，openai的key我已经传上去了

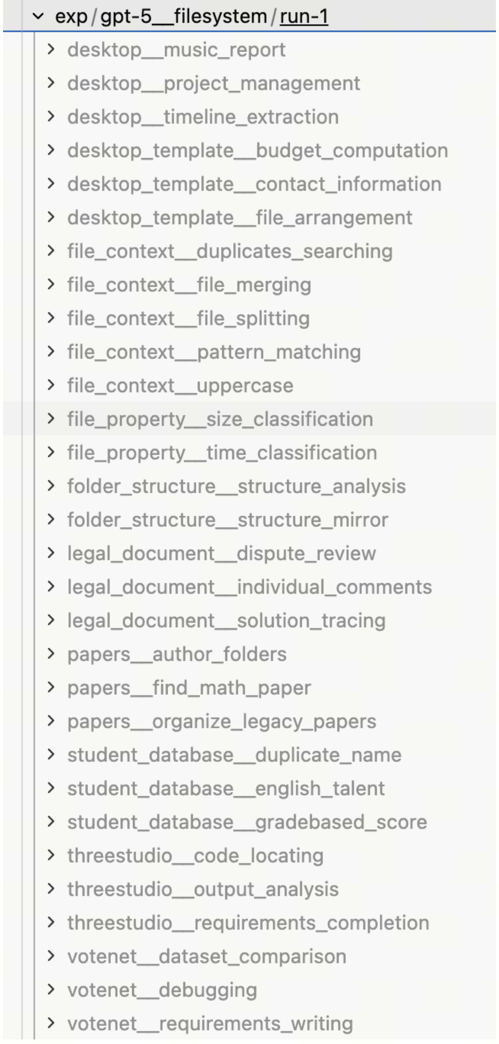
3.运行子领域任务：例如，

代码块

```
python -m pipeline --exp-name exp --mcp filesystem --tasks all --models gpt-5 --k 1
```

如果是其他子领域就把filesystem换一下

4.运行完之后会有下面的轨迹



```
exp / gpt-5__filesystem / run-1
  desktop__music_report
  desktop__project_management
  desktop__timeline_extraction
  desktop_template__budget_computation
  desktop_template__contact_information
  desktop_template__file_arrangement
  file_context__duplicates_searching
  file_context__file_merging
  file_context__file_splitting
  file_context__pattern_matching
  file_context__uppercase
  file_property__size_classification
  file_property__time_classification
  folder_structure__structure_analysis
  folder_structure__structure_mirror
  legal_document__dispute_review
  legal_document__individual_comments
  legal_document__solution_tracing
  papers__author_folders
  papers__find_math_paper
  papers__organize_legacy_papers
  student_database__duplicate_name
  student_database__english_talent
  student_database__gradebased_score
  threestudio__code_locating
  threestudio__output_analysis
  threestudio__requirements_completion
  votenet__dataset_comparison
  votenet__debugging
  votenet__requirements_writing
```

在这里每个文件夹下面的meta json可以看是否pass

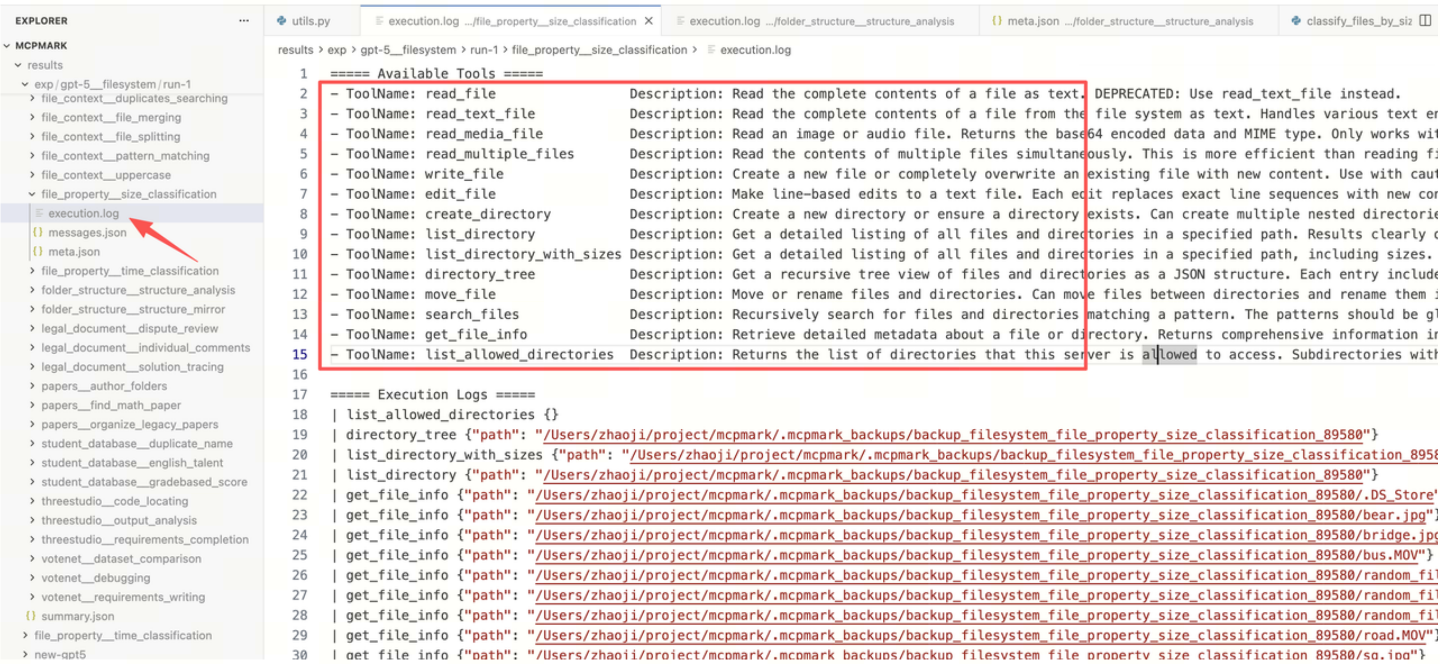
5.对于不通过的子任务，可以拿gpt多运行几次，或者拿claude sonnet 4.5，我们那个key也是可以用Claude的。

例如：

代码块

```
python -m pipeline --exp-name new-gpt5 --mcp filesystem --tasks folder_structure/structure_analysis --models gpt-5 --k 3
```

6.在多次运行之后尽量拿到多的正确的pass结果之后，每个子领域（例如filesystem，github）都会有一些特定的工具，如下：


EXPLORER
MCPMARK
results
exp / gpt-5__filesystem / run-1
file_context__duplicates_searching
file_context__file_merging
file_context__file_splitting
file_context__pattern_matching
file_context__uppercase
file_property__size_classification
execution.log
messages.json
meta.json
file_property__time_classification
folder_structure__structure_analysis
folder_structure__structure_mirror
legal_document__dispute_review
legal_document__individual_comments
legal_document__solution_tracing
papers__author_folders
papers__find_math_paper
papers__organize_legacy_papers
student_database__duplicate_name
student_database__english_talent
student_database__gradebased_score
threestudio__code_locating
threestudio__output_analysis
threestudio__requirements_completion
votenet__dataset_comparison
votenet__debugging
votenet__requirements_writing
summary.json
file_property__time_classification
new-gpt5
utils.py
execution.log .../file_property__size_classification
execution.log .../folder_structure__structure_analysis
meta.json .../folder_structure__structure_analysis
classify_files_by_siz
results > exp > gpt-5__filesystem > run-1 > file_property__size_classification > execution.log

```
===== Available Tools =====
- ToolName: read_file                 Description: Read the complete contents of a file as text. DEPRECATED: Use read_text_file instead.
- ToolName: read_text_file            Description: Read the complete contents of a file from the file system as text. Handles various text er
- ToolName: read_media_file           Description: Read an image or audio file. Returns the base64 encoded data and MIME type. Only works wi
- ToolName: read_multiple_files       Description: Read the contents of multiple files simultaneously. This is more efficient than reading fi
- ToolName: write_file                Description: Create a new file or completely overwrite an existing file with new content. Use with caut
- ToolName: edit_file                 Description: Make line-based edits to a text file. Each edit replaces exact line sequences with new cor
- ToolName: create_directory          Description: Create a new directory or ensure a directory exists. Can create multiple nested directorie
- ToolName: list_directory            Description: Get a detailed listing of all files and directories in a specified path. Results clearly c
- ToolName: list_directory_with_sizes Description: Get a detailed listing of all files and directories in a specified path, including sizes.
- ToolName: directory_tree            Description: Get a recursive tree view of files and directories as a JSON structure. Each entry include
- ToolName: move_file                 Description: Move or rename files and directories. Can move files between directories and rename them i
- ToolName: search_files              Description: Recursively search for files and directories matching a pattern. The patterns should be gl
- ToolName: get_file_info             Description: Retrieve detailed metadata about a file or directory. Returns comprehensive information ir
- ToolName: list_allowed_directories  Description: Returns the list of directories that this server is allowed to access. Subdirectories with

===== Execution Logs =====
| list_allowed_directories {}
| directory_tree {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_89580"}
| list_directory_with_sizes {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_8958
| list_directory {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_89580"}
| get_file_info {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_89580/.DS_Store'
| get_file_info {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_89580/bear.jpg"]
| get_file_info {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_89580/bridge.jpg
| get_file_info {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_89580/bus.MOV"}
| get_file_info {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_89580/random_fil
| get_file_info {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_89580/random_fil
| get_file_info {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_89580/random_fil
| get_file_info {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_89580/road.MOV"]
| get_file_info {"path": "/Users/zhaoji/project/mcpmark/.mcpmark_backups/backup_filesystem_file_property_size_classification_89580/sq.jpg"}
```

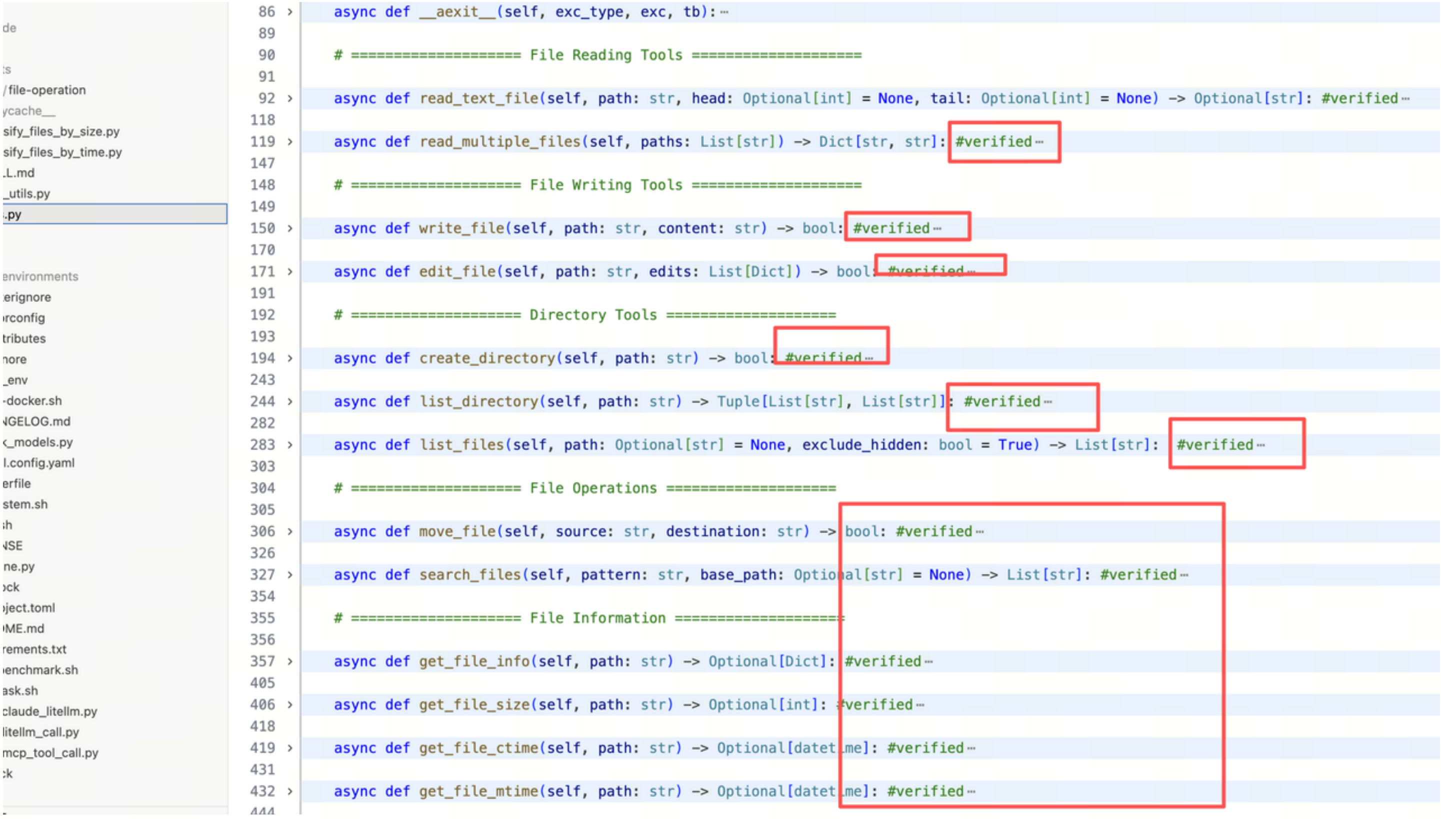
我会新建一个py文件，实现其中的几乎所有tool，然后之后的逻辑就使用这里实现的tool来完成，可以使代码更结构化。例如我的https://github.com/zjtco-yr/open-agent-skills/blob/main/skills/file-operation/utils.py

但是有些坑需要注意，
1）例如上面的工具，其中第一个read_file是已经弃用了的。不用实现。

2）有些工具无论怎么调用都是会报错的，并且这个工具本身没那么重要（即可以使用其他工具的组合来实现其功能），就不用实现。上面的list_directory_with_size以及directory_tree就是这样，观察message.json可以看到这两个tool调用的结果都是failed，不用实现这两个。

3）对于我上面说的utils.py，我写了一个单独的test_utils.py来测试所有函数的功能，这个是很重要，tool的实现如果不对，之后基于这个实现的功能也必然不对。

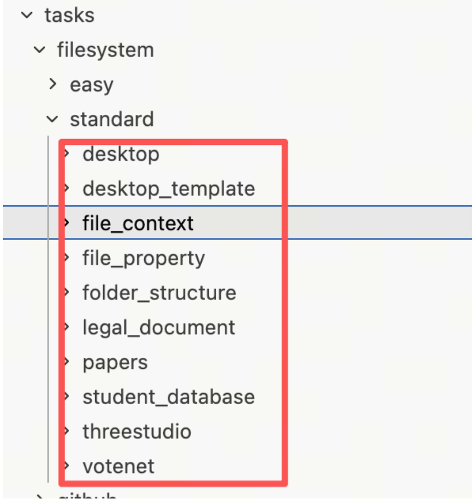
4）对于验证准确的实现，我会在函数后面标一个 #verified



```
    async def __aexit__(self, exc_type, exc, tb):…

    # ==================== File Reading Tools ====================

    async def read_text_file(self, path: str, head: Optional[int] = None, tail: Optional[int] = None) -> Optional[str]: #verified…

    async def read_multiple_files(self, paths: List[str]) -> Dict[str, str]: #verified…

    # ==================== File Writing Tools ====================

    async def write_file(self, path: str, content: str) -> bool: #verified…

    async def edit_file(self, path: str, edits: List[Dict]) -> bool: #verified…

    # ==================== Directory Tools ====================

    async def create_directory(self, path: str) -> bool: #verified…

    async def list_directory(self, path: str) -> Tuple[List[str], List[str]]: #verified…

    async def list_files(self, path: Optional[str] = None, exclude_hidden: bool = True) -> List[str]: #verified…

    # ==================== File Operations ====================

    async def move_file(self, source: str, destination: str) -> bool: #verified…

    async def search_files(self, pattern: str, base_path: Optional[str] = None) -> List[str]: #verified…

    # ==================== File Information ====================

    async def get_file_info(self, path: str) -> Optional[Dict]: #verified…

    async def get_file_size(self, path: str) -> Optional[int]: #verified…

    async def get_file_ctime(self, path: str) -> Optional[datetime]: #verified…

    async def get_file_mtime(self, path: str) -> Optional[datetime]: #verified…
```

5）整个过程都可以借用vibe coding来帮忙。实现完一个子领域下面的一类任务，其他就会很快，因为tool都是一样的。

6）建议对每个子领域下面的一个小类分一个skill：


tasks
filesystem
easy
standard
desktop
desktop_template
file_context
file_property
folder_structure
legal_document
papers
student_database
threestudio
votenet
github