Git Bash는 Git을 사용하기 위해 설계된 Windows 환경에서의 터미널입니다. Git Bash는 Git 명령뿐만 아니라 대부분의 Unix 기반 명령어를 지원하므로, Windows에서도 Linux/Unix 스타일의 터미널 명령어를 사용할 수 있습니다. 아래는 Git Bash에서 자주 사용되는 기본적인 터미널 명령어에 대한 튜토리얼입니다.'

1. 기본 파일 및 디렉토리 명령어

1.1 디렉토리 이동 및 보기

- `pwd` **(Print Working Directory)**: 현재 작업 중인 디렉토리의 경로를 출력합니다.

```
pwd
```

- `ls` : 현재 디렉토리의 파일 및 폴더 목록을 표시합니다.

```
ls
```

  - `ls -l` : 자세한 정보(파일 권한, 소유자, 크기, 수정 날짜)를 포함하여 출력합니다.
  - `ls -a` : 숨겨진 파일도 함께 출력합니다.

- `cd` **(Change Directory)**: 다른 디렉토리로 이동합니다.

```
cd directory_name
```

  - `cd ..` : 상위 디렉토리로 이동합니다.
  - `cd ~` : 홈 디렉토리로 이동합니다.

1.2 파일 및 디렉토리 관리

- `touch` : 새로운 빈 파일을 생성합니다.

```
touch filename.txt
```

- **`mkdir` (Make Directory)**: 새로운 디렉토리를 생성합니다.

```
mkdir new_directory
```

- `rm` : 파일을 삭제합니다.

```
rm filename.txt
```

  - `rm -r directory_name` : 디렉토리와 그 안의 모든 파일을 삭제합니다.

- `cp` **(Copy)**: 파일이나 디렉토리를 복사합니다.

```
cp source_file destination_file
```

  - `cp -r source_directory destination_directory`: 디렉토리 전체를 복사합니다.

- `mv` **(Move)**: 파일이나 디렉토리를 이동하거나 이름을 변경합니다.

```
mv old_name new_name
```

1.3 파일 내용 보기 및 수정

- `cat` : 파일의 내용을 출력합니다.

```
cat filename.txt
```

- **`nano`** : 간단한 텍스트 편집기입니다. 파일을 열고 편집할 수 있습니다.

  ```
  nano filename.txt
  ```

  - Nano 에디터에서 `Ctrl + X` 를 눌러 종료하고, 저장 여부를 묻는 메시지가 나타납니다.

- **`less`**: 큰 파일을 화면 단위로 볼 수 있게 해줍니다.

```
less filename.txt
```

  - `q`를 눌러 `less`를 종료합니다.

2. 시스템 및 프로세스 명령어

2.1 시스템 정보 및 유틸리티

- **`uname -a`** : 시스템의 모든 정보를 출력합니다.

```
uname -a
```

- `top` : 현재 실행 중인 프로세스 목록과 시스템 리소스 사용량을 실시간으로 표시합니다.

  ```
  top
  ```

  - `q` 를 눌러 `top` 을 종료합니다.

- `df -h` : 파일 시스템의 디스크 사용량을 표시합니다.

```
df -h
```

2.2 프로세스 관리

- **`ps`** : 현재 실행 중인 프로세스 목록을 출력합니다.

```
ps
```

  - `ps aux` : 모든 사용자와 프로세스에 대한 자세한 정보를 출력합니다.

- **`kill`** : 특정 프로세스를 종료합니다.

```
kill PID
```

  - 여기서 `PID` 는 `ps` 명령어를 사용하여 확인한 프로세스 ID입니다.
  - `kill -9 PID` : 프로세스를 강제 종료합니다.

3. Git 명령어

Git Bash에서 Git 명령어도 함께 사용할 수 있습니다. 여기서는 몇 가지 기본적인 Git 명령어를 소개합니다.

3.1 Git 설정

- `git config` : 사용자 정보를 설정합니다.

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

3.2 저장소 초기화 및 상태 확인

- **`git init`** : 새로운 Git 저장소를 초기화합니다.

```
git init
```

- **`git status`** : 현재 저장소의 상태(수정된 파일, 스테이지 상태 등)를 확인합니다.

```
git status
```

3.3 커밋 및 로그

- **`git add`** : 파일을 스테이징 영역에 추가합니다.

```
git add filename.txt
```

  - `git add .` : 현재 디렉토리의 모든 변경 사항을 스테이징합니다.

- **`git commit`**: 스테이징된 파일을 커밋합니다.

```
git commit -m "커밋 메시지"
```

- **`git log`** : 커밋 기록을 확인합니다.

```
git log
```

3.4 원격 저장소

- **`git remote add`** : 원격 저장소를 추가합니다.

```
git remote add origin https://github.com/user/repository.git
```

- **`git push`** : 변경 사항을 원격 저장소에 푸시합니다.

```
git push origin main
```

- **`git pull`** : 원격 저장소의 변경 사항을 가져옵니다.

```
git pull origin main
```

4. 기타 유용한 명령어

4.1 파일 찾기

- **`find`** : 특정 파일을 찾을 수 있습니다.

```
find . -name "filename.txt"
```

4.2 네트워크 관련

- **`ping`** : 네트워크 연결 상태를 확인할 수 있습니다.

```
ping google.com
```

- `curl` : 웹 요청을 보내고 응답을 확인할 수 있습니다.

```
curl http://example.com
```

결론

Git Bash는 Git 명령어뿐만 아니라 Unix 기반의 다양한 명령어를 활용할 수 있는 강력한 도구입니다. 위에서 소개한 명령어들은 Git Bash에서 가장 기본적이고 자주 사용되는 명령어들입니다. 이러한 명령어들을 익혀두면, Git Bash를 통해 효율적으로 파일을 관리하고 Git 작업을 수행할 수 있습니다.