컴퓨터학부 20221494 박찬솔

1. 과제 개요

홈 디렉토리 아래의 파일을 지정된 백업 디렉토리에 백업하고 복원하는 것과 백업된 파일을 삭제하는 것을 주요 기능으로 하고 있다. 파일의 백업과 복원 중에는 sha1 또는 md5 해쉬 함수를 사용하여 같은 파일이면 백업 및 복원 작업을 건너뛰는 기능이 있다. 이외에도 외부 명령어 실행 등의 기능을 내장하고 있다.

2. 기능

2-1. 파일 백업 (명령어 add)

이 명령어의 구문은 **add <FILENAME> [OPTION]**이다. <FILENAME>은 백업하고자 하는 파일 또는 디렉토리의 경로이다. [OPTION]은 필수가 아니며, -d 옵션만 있다. 디렉토리가 주어진 경우, -d 옵션을 사용하여 디렉토리 전체를 재귀적으로 백업할 수 있다.

주어진 파일의 경로가 ~/<a>와 같은 형식일 때(<a>는 홈 디렉터리에 대해 백업할 파일의 상대 경로), ~/backup/<a>에 해당 파일을 백업한다. 이 프로그램은 다음 두 가지 사항을 만족하는 백업하는 기능을 제공한다.

1. 백업 디렉터리에 파일을 백업할 때에는 <date> = <YYMMDDHHMMSS> 식의 시간 포맷을 정의한 후, ~/backup/<a>_<date> 형식의 파일 이름을 사용한다.
2. 파일을 백업하기 전, ~/backup/<a>_<date>를 만족하는 파일 중, 현재 파일과 내용이 동일한 파일이 있다면 백업하지 않는다. (파일과 내용이 동일하다는 것은 해시 함수[sha1, md5]를 사용했을 때, 그 해시값이 동일하다는 것이다.)

2-2. 파일 삭제 (명령어 remove)

이 명령어의 구문은 **remove [FILENAME] [OPTION]**이다. [FILENAME]은 삭제하고자 하는 파일 또는 디렉토리의 경로이다. [OPTION]은 필수가 아니며, -a 옵션과 -c 옵션이 있다. -a 옵션은 백업된 파일이 여러 개 있을 때 모두 삭제하거나, 폴더가 주어졌을 때 해당 폴더 아래의 백업된 파일을 모두 삭제한다. -c 옵션은 백업 폴더(~/backup) 아래의 모든 파일과 디렉터리를 삭제한다.

파일의 경로와 관련된 사항은 명령어 add와 동일하다.

-a와 -c 옵션 없이 단일 파일을 삭제하려 하고 백업 디렉토리 내 해당 파일이 여러 개라면(이름은 같은데 백업된 시간이 여러 개인 경우), 그중 하나만을 선택하여 삭제하는 기능을 제공한다.

2-3. 파일 복원 (명령어 recover)

이 명령어의 구문은 **recover <FILENAME> [OPTION]**이다. <FILENAME>은 복원하고자 하는 파일 또는 디렉토리의 경로이다. [OPTION]은 필수가 아니며, -d 옵션과 -n 옵션이 있다. 디렉토리가 주어진 경우, -d 옵션을 사용하여 디렉토리 전체를 재귀적으로 복원할 수 있다. -n 옵션은 사용 시 <NEW>라는 추가적인 값을 필요로 한다. 주어진 <FILENAME>의 경로가 ~/backup/<FILENAME>이면, ~/<NEW>라는 경로에 복원을 진행하는 것이다.

두 파일 <NEW>와 <FILENAME>의 경로와 관련된 사항은 명령어 add와 동일하다.

2-4. 기타 (명령어 ls, vi, vim)

이 명령어는 외부 명령어를 그대로 실행하는 기능을 제공한다.

3. 설계

프로그램은 크게 명령어를 입력받은 후, 파일의 해시를 비교하며, 백업, 삭제, 복원 작업을 수행한다. 명령어를 입력받고 실행하는 코드는 commands/ 아래에 정의되어 있으며, 파일과 경로의 관리 및 처리를 용이하게 하고자 file/ 아래에 관련 함수들이 정의되어 있다. 이 외에도 명령어 도움말을 출력하는 usage/, 메인 프로그램인 ssu_backup.c, 연결 리스트를 구현한 data/로 구성되어 있다. 또한, 여러 파일들의 컴파일과 링크를 유용하게 하고자 Makefile을 통해 프로젝트 빌드 시스템을 관리하고 있다.

3-1. 파일(모듈) 전체 목록

프로젝트의 파일(모듈) 구성은 하기와 같다.

commands/
- command_add.c
- command_help.c
- command_recover.c
- command_remove.c
- init.{c, h}

data/
- linked_list.{c, h}

debug/
- debug.{c, h}

file/
- file.{c, h}
- hash.{c, h}
- path.{c, h}

usage/
- usage.{c, h}

Makefile

ssu_backup.c

3-2. 소스코드별 함수 개요

3-2-1. command_add.c

명령어 add를 입력하면 실행되는 파일

3-2-1-1. int main(int argc, char *argv[])

입력된 명령어의 인자를 파싱하고 기본적인 환경 설정을 진행한 후, command_add_execute 함수를 호출한다.

3-2-1-2. int command_add_execute(char *input_path, int is_recursive)

<FILENAME>과 -d 옵션의 여부를 매개변수로 받아 입력된 파일 경로, 인자에 대한 유효성을 검사한다. ($HOME을 벗어나는지, 접근 권한이 있는지 등) 문제가 없으면, command_add_backup 함수를 실행하여 백업을 진행한다.

3-2-1-3. void command_add_backup(const char *path)

백업 경로인 path를 매개변수로 받아 실제로 백업을 진행하는 함수이다.

3-2-2. command_help.c

명령어 help 또는 존재하지 않는 명령어를 입력하면 실행되는 파일

3-2-2-1. int main(int argc, char *argv[])

print_help_main 함수를 호출하여 도움말만 출력하고 종료한다.

3-2-3. command_recover.c

명령어 recover를 입력하면 실행되는 파일

3-2-3-1. int main(int argc, char *argv[])

입력된 명령어의 인자를 파싱하고 기본적인 환경 설정을 진행한 후, command_recover_execute 함수를

호출한다.

3-2-3-2. int command_recover_execute(const char *src_path, const char *dest_path, int is_recursive)

<FILENAME>과 복원 경로(dest_path), -d 옵션 여부를 매개변수로 받아 입력된 파일 경로, 인자에 대한 유효성을 검사한다. ($HOME을 벗어나는지, 접근 권한이 있는지 등) 문제가 없으면, command_recover_backup 함수를 실행하여 복원을 진행한다.

3-2-3-3. void command_recover_process_file_list(const char *src_path, const char *dest_path, int is_recursive)

백업 디렉토리 내 DATE만 다르고 실제 파일의 이름이 같은 파일을 하나로 묶어 복원해야 할 파일의 목록을 구한다. 이런 식으로 목록을 구한 뒤, 각각의 파일에 대하여 command_recover_single_file 함수를 실행하여 실제 복원 작업을 진행한다.

3-2-3-4. int command_recover_single_file(const char *backup_path, const char *dest_path)

backup_path에 있는 파일을 dest_path로 복사한 후, backup_path에 있는 파일은 삭제한다. backup_path에 DATE만 다르고 여러 개의 파일이 존재한다면 그중 하나를 선택하도록 한다.

3-2-4. command_remove.c

명령어 remove를 입력하면 실행되는 파일

3-2-4-1. int main(int argc, char *argv[])

입력된 명령어의 인자를 파싱하고 기본적인 환경 설정을 진행한 후, command_remove_execute 함수를 호출한다.

3-2-4-2. int command_remove_execute(char *input_path, int is_recursive, int is_clear)

<FILENAME>과 -d, -c 옵션의 여부를 매개변수로 받아 입력된 파일 경로, 인자에 대한 유효성을 검사한다. ($HOME을 벗어나는지, 접근 권한이 있는지, 옵션이 중복으로 주어지지는 않았는지 등) 문제가 없으면, <FILENAME>을 백업 디렉토리에 대한 경로로 변환한 후, command_remove_backup 함수를 실행하여 파일 삭제를 진행한다.

3-2-4-3. void command_remove_backup(const char *path, int is_recursive, int is_clear)

백업 경로인 path를 매개변수로 받아 파일과 디렉터리(is_clear가 true일 때만)를 삭제한다.

3-2-5. command_init.c

help를 제외한 모든 명령어 프로그램은 해당 프로그램의 init 함수를 호출함

3-2-5-1. int init(int argc, char *argv[])

새로 실행되는 프로세스의 인자를 파싱한다. 구체적으로 init_filesystem 함수를 호출하고, 어떤 해쉬 함수를 사용할지 정한다.

3-2-6. linked_list.{c, h}

파일 리스트 관리를 위한 연결 리스트(linked list) 자료구조가 구현되어 있다.

3-2-6-1. struct Node

구조체 Node는 다음 두 필드 char **path**[PATH_MAX], struct Node ***next**를 순서대로 갖는다.

3-2-6-2. struct Node *create_node(const char *path)

구조체 필드 path에 매개변수로 들어온 path 값을 넣어 새로운 노드를 생성한다. 생성되는 노드는 동적 할당된다.

3-2-6-3. int get_size_linked_list(struct Node *node)

매개변수로 들어온 node를 root로 가정하고 해당 연결 리스트의 길이가 얼마인지 계산한다.

3-2-6-4. struct Node *get_last_node(struct Node *node)

해당 연결 리스트의 가장 마지막 노드를 리턴한다.

3-2-6-5. void clear_node(struct Node *root)

연결 리스트를 삭제하며, 각 노드들에 할당된 메모리도 해제한다.

3-2-7. debug.{c, h}

디버깅을 위한 함수와 매크로가 정의되어 있다. 다음 함수는 모두 DEBUG가 1로 정의되어 있을 때만 작동한다.

3-2-7-1. debug_print

printf와 같은 동작을 한다. 단, 출력문에 [DEBUG]가 앞에 붙어 출력된다. 매크로로 정의된 가변 인자 함수이다.

3-2-7-2. void debug_hash(const unsigned char *str, int length)

해시를 16진수 포맷으로 출력하는 함수이다.

3-2-8. file.c

파일과 관련된 함수가 정의되어 있다.

3-2-8-1. struct Node *find_all_files(const char *path, int is_recursive, int is_first)

path 아래에 있는 모든 regular file를 연결 리스트의 형태로 리턴한다. is_recursive가 0이면 재귀적으로 탐색하지 않고, path 바로 아래에 있는 파일만 탐색한다. (이 경우, is_first를 1로 주어야 한다.)

3-2-8-2. struct Node *find_same_name_files(const char *path)

path는 regular file의 경로로 주어진다. 이때, path와 같은 디렉터리에 있는 regular file의 경로가 path_<date> 형식인 파일들을 연결리스트 형태로 리턴한다. 즉, <date>를 제외한 파일의 경로가 같은 파일들을 리턴하는 함수이다.

3-2-8-3. unsigned char *read_file(const char *path, int *length)

파일 path의 내용을 읽어 그 내용을 동적 할당하여 리턴한다. 포인터 변수 length에 해당 내용의 길이를 지정해준다.

3-2-8-4. void write_file(const char *path, const unsigned char *content, int length)

파일 path에 content를 length만큼 작성한다. 이때, 파일을 작성할 경로에 디렉터리가 없으면 재귀적으로 디렉토리를 생성한다.

3-2-8-5. void copy_file(const char *dst, const char *src)

src의 파일을 그대로 dst에 복사한다. 각각 read_file과 write_file 함수를 사용하여 구현한다.

3-2-8-6. off_t get_filesize(const char *path)

lstat 함수를 사용하여 파일의 크기를 계산한다.

3-2-8-7. int remove_backup_directory_safely(const char *path)

주어진 경로를 재귀적으로 삭제한다. 이때, 백업 디렉토리는 삭제하지 않도록 하며, 삭제된 디렉토리의 개수를 리턴한다.

3-2-9. hash.c

해시와 관련된 함수가 정의되어 있다.

3-2-9-1. unsigned char *hash_file(const char *path)

파일을 openssl의 내장 함수(MD5 또는 SHA1)을 사용하여 해싱하고 그 결과는 동적 할당하여 리턴한다.

3-2-9-2. int compare_file(const char *path1, const char *path2)

두 파일 path1과 path2를 hash_file 함수를 사용하여 비교한다. 같으면 1, 다르면 0을 리턴한다.

3-2-10. path.c

파일 경로와 관련된 함수가 정의되어 있다.

3-2-10-1. void init_filesystem(char *executable_path)

프로그램을 실행할 때 사용한 인자(argv[0])를 사용하여 프로그램의 위치를 파악한다. 또한, 홈 디렉터리와 백업 디렉터리의 위치와 생성을 관리한다.

3-2-10-2. int is_directory(const char *path)

lstat을 이용하여 해당 경로가 regular file이면 0, directory면 1, 그렇지 않으면 -1을 리턴한다.

3-2-10-3. void get_absolute_path(const char *path, char *ret)

path를 경로 정규화(normalize path) 과정을 적용하여 절대 경로로 만든 후, 그 결과를 ret에 지정한다.

3-2-10-4. const char *get_relative_path(const char *path, const char *base)

경로 base에 대한 경로 path의 상대 경로를 리턴한다. 예를 들어, path=/home/nlog/123이고, base=/home/nlog이면 리턴 값은 /123이다.

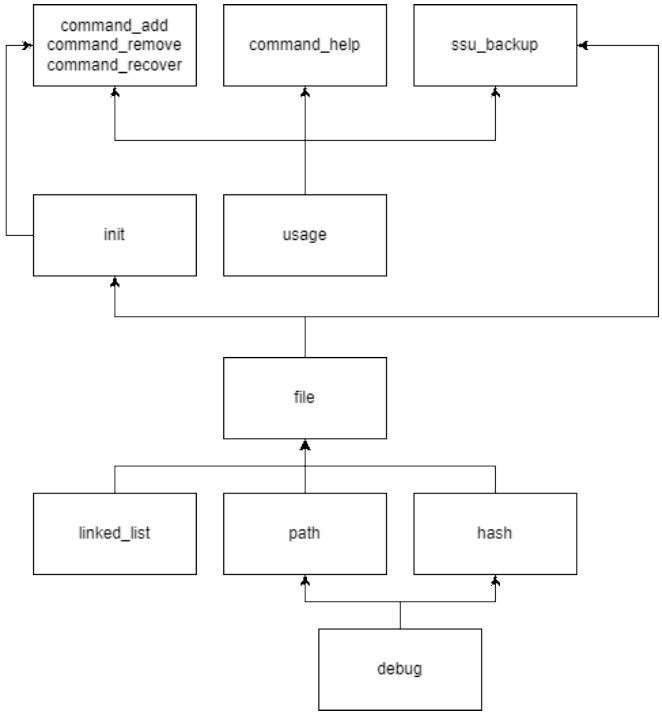
3-2-10-5. const char *get_date_in_path(const char *path)
path에서 date 부분만을 찾아 리턴한다.
3-2-10-6. char *get_parent_directory(const char *path, char *ret)
path의 부모 디렉터리를 ret에 지정한다.
3-2-10-7. int check_path_in_directory(const char *path, const char *directory)
주어진 path가 directory 하위에 있는 파일이면 1을, 그렇지 않으면 0을 리턴한다.
3-2-10-8. void remove_filename_date(const char *path, char *ret)
주어진 path에서 date 부분을 제거한 후, 그 결과를 ret에 지정한다.

3-2-11. usage.c
사용법을 출력하는 함수가 정의되어 있다.
3-2-11-1. void print_help_program()
프로그램 실행 시의 사용법을 출력한다.
3-2-11-2. void print_help_all()
모든 내부 명령어의 사용법을 출력한다.
3-2-11-3. void print_help_add()
add 명령어의 사용법을 출력한다.
3-2-11-4. void print_help_remove()
remove 명령어의 사용법을 출력한다.
3-2-11-3. void print_help_recover()
recover 명령어의 사용법을 출력한다.

3-2-12. ssu_backup.c
프로그램의 메인 파일이다.
3-2-12-1. int main(int argc, char *argv[])
프로그램 인자를 확인하여 어떤 해시 함수를 사용하는지 결정한다. 이후 명령어를 입력받고 그에 맞는 명령어 파일을 fork와 execv 함수로 실행시킨다.

3-3. 종속성(dependency) 그래프

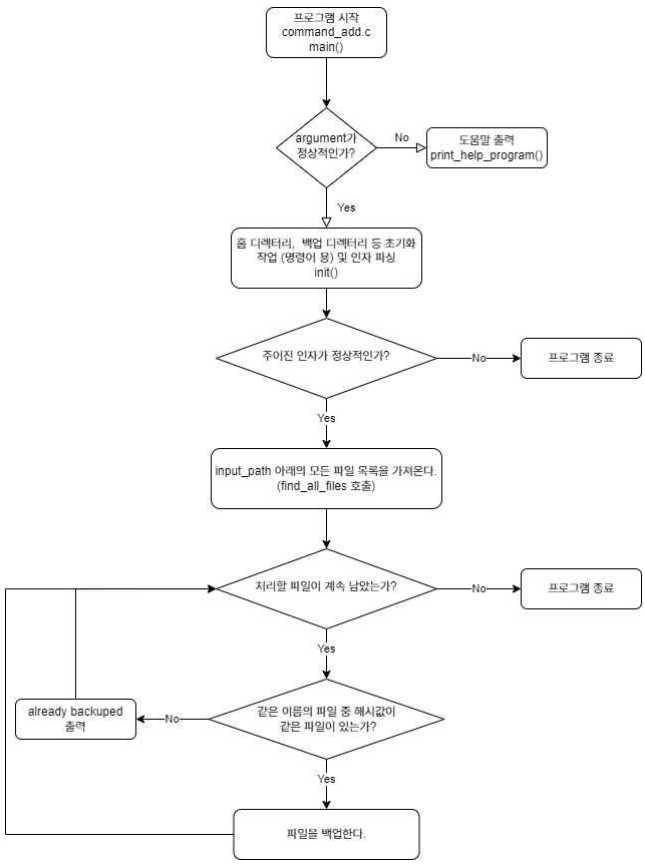
각 모듈(source, header file)이 어떤 모듈에 의존하는지 나타냈다. 작성된 Makefile의 종속성도 아래와 같으므로 Makefile의 종속성에 대한 내용은 생략한다.



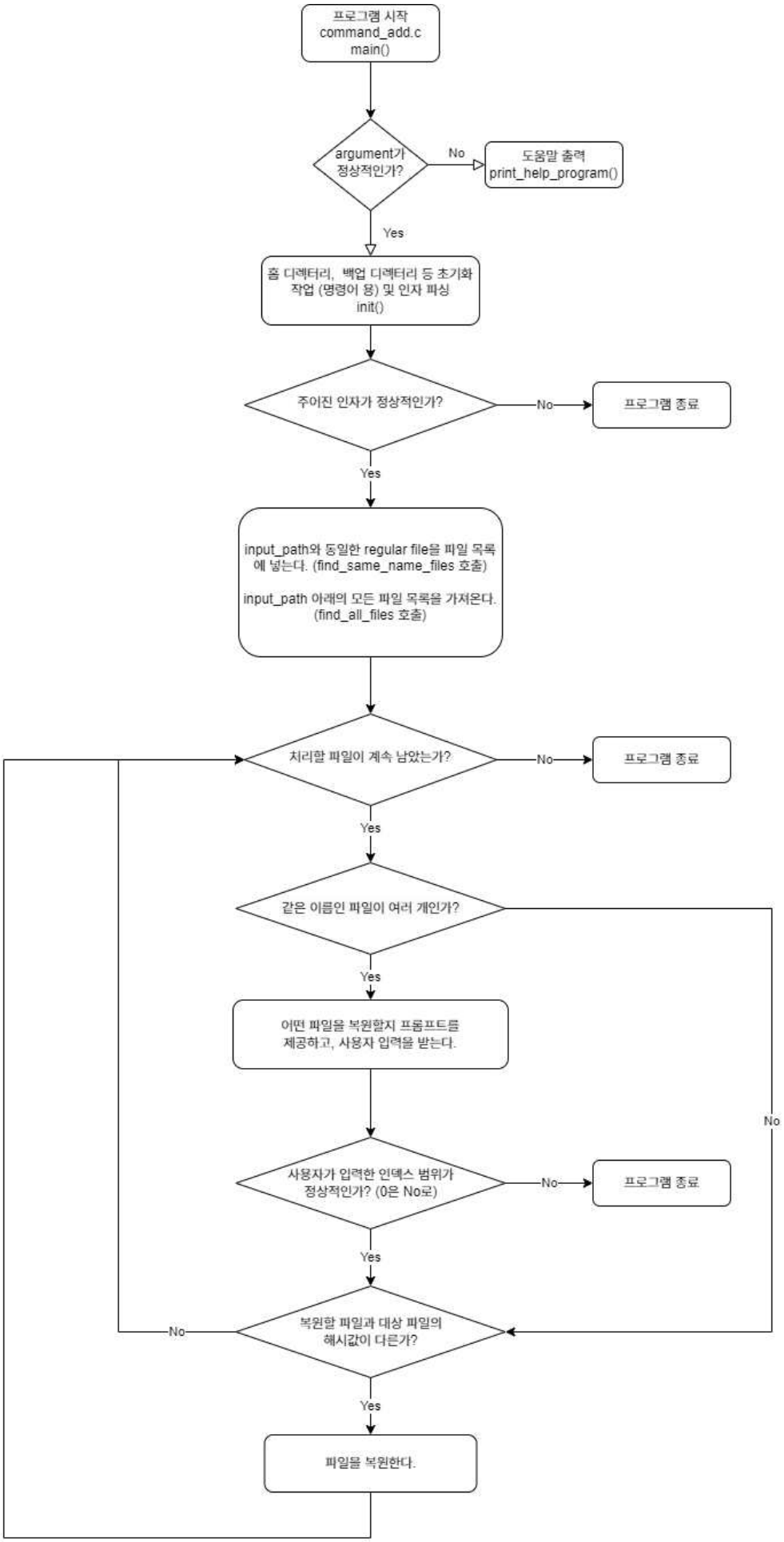
3-4. 순서도

주요 기능(명령어 및 메인 프로그램) 및 주요 함수(file.c, path.c의 일부 함수)의 순서도이다.

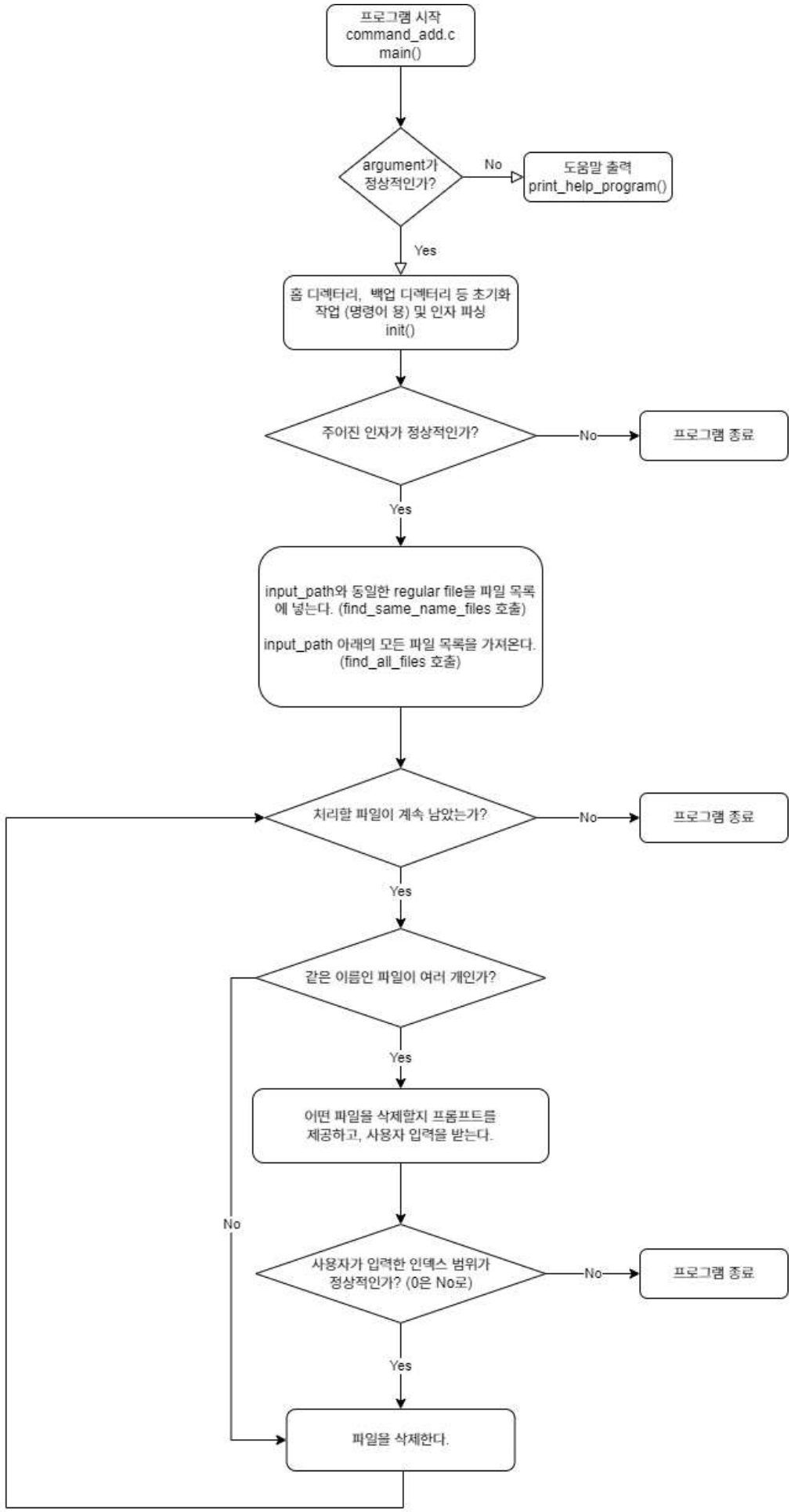
3-4-1. command_add 순서도



3-4-2. command_recover 순서도



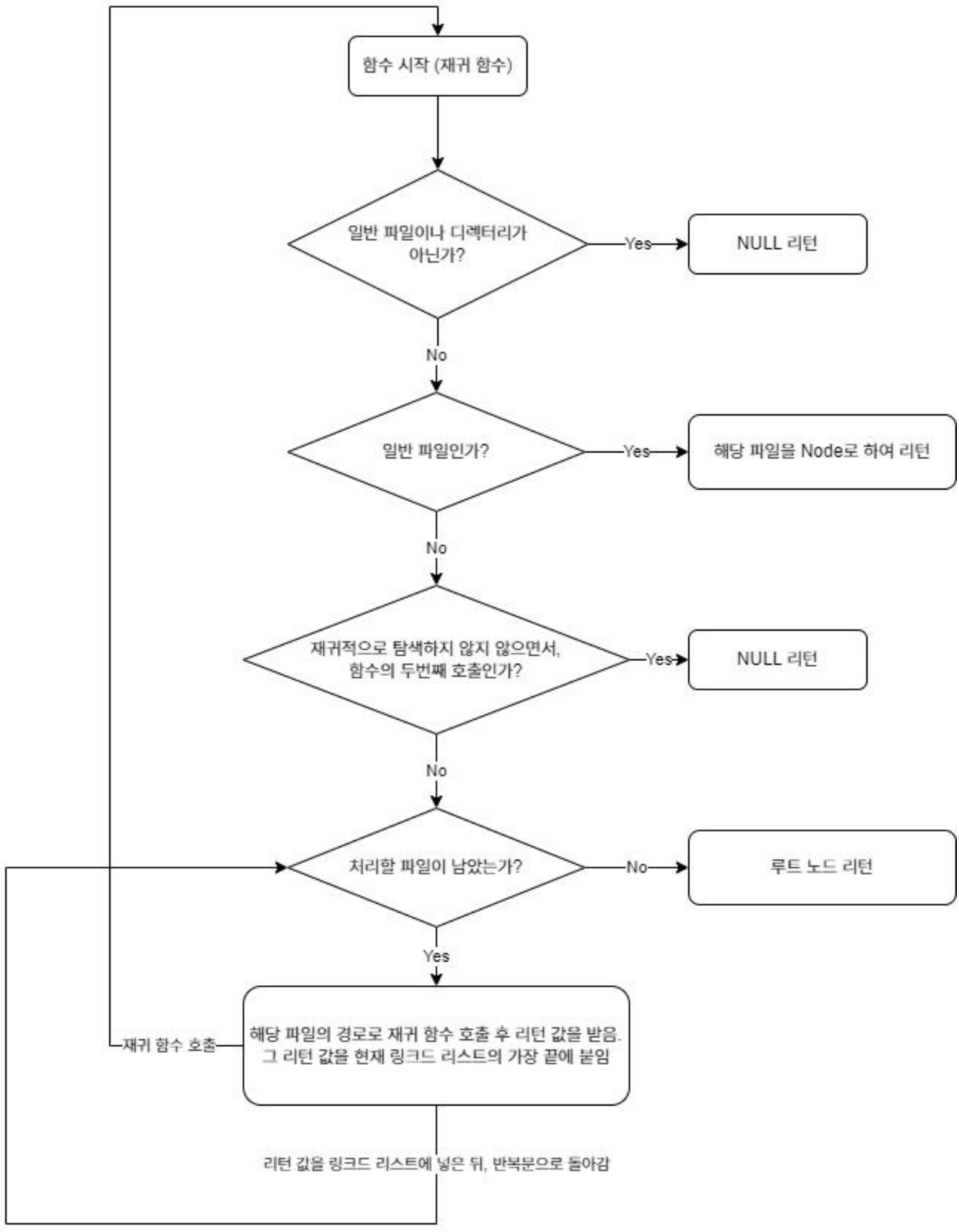
3-4-3. command_remove 순서도



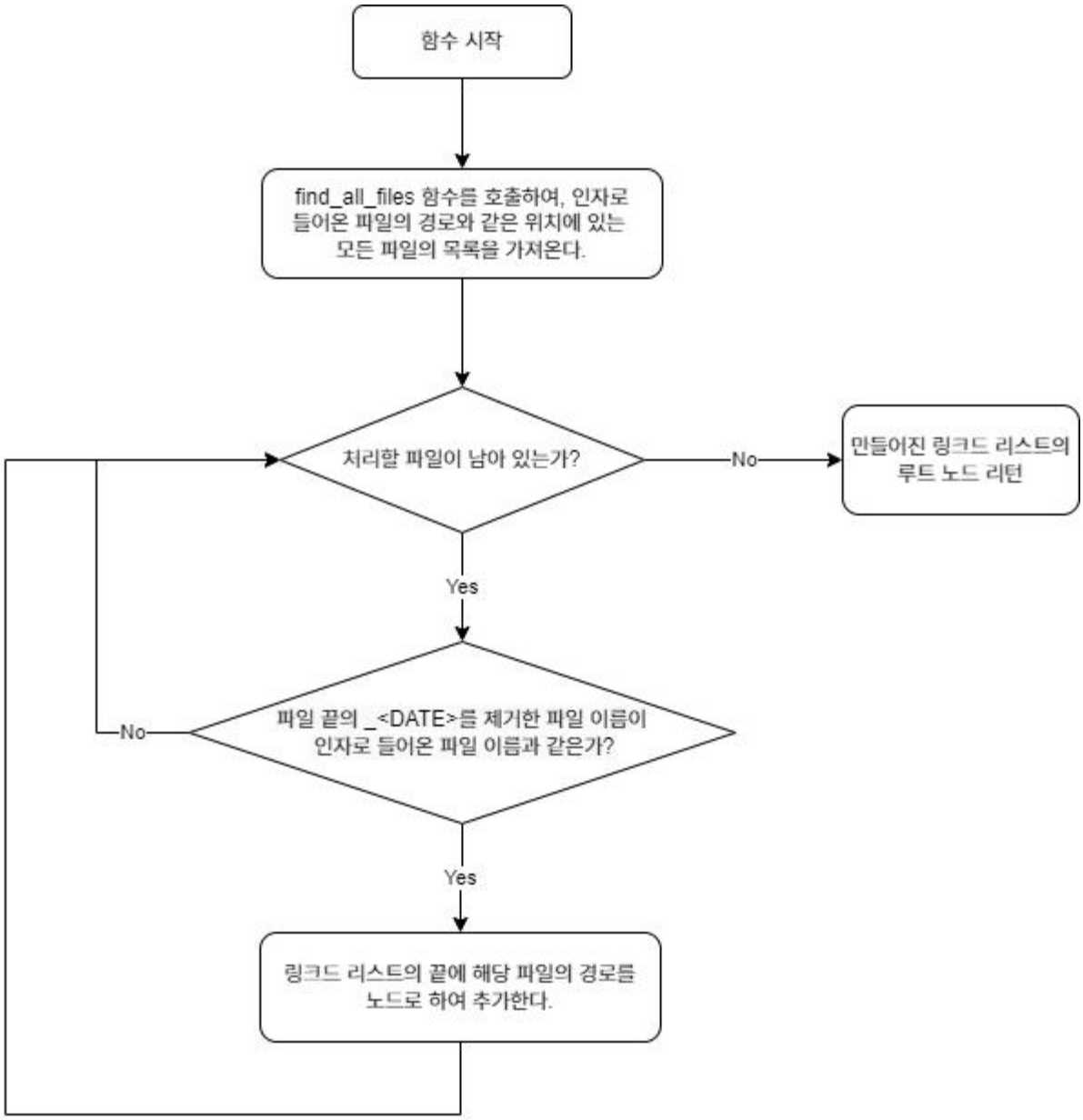
3-4-4. file.c 순서도

file.c에 구현된 주요 함수의 순서도이다. 다음 함수 중 일부는 재귀적으로 정의되어 있거나 재귀 함수를 호출하는 형태이다.

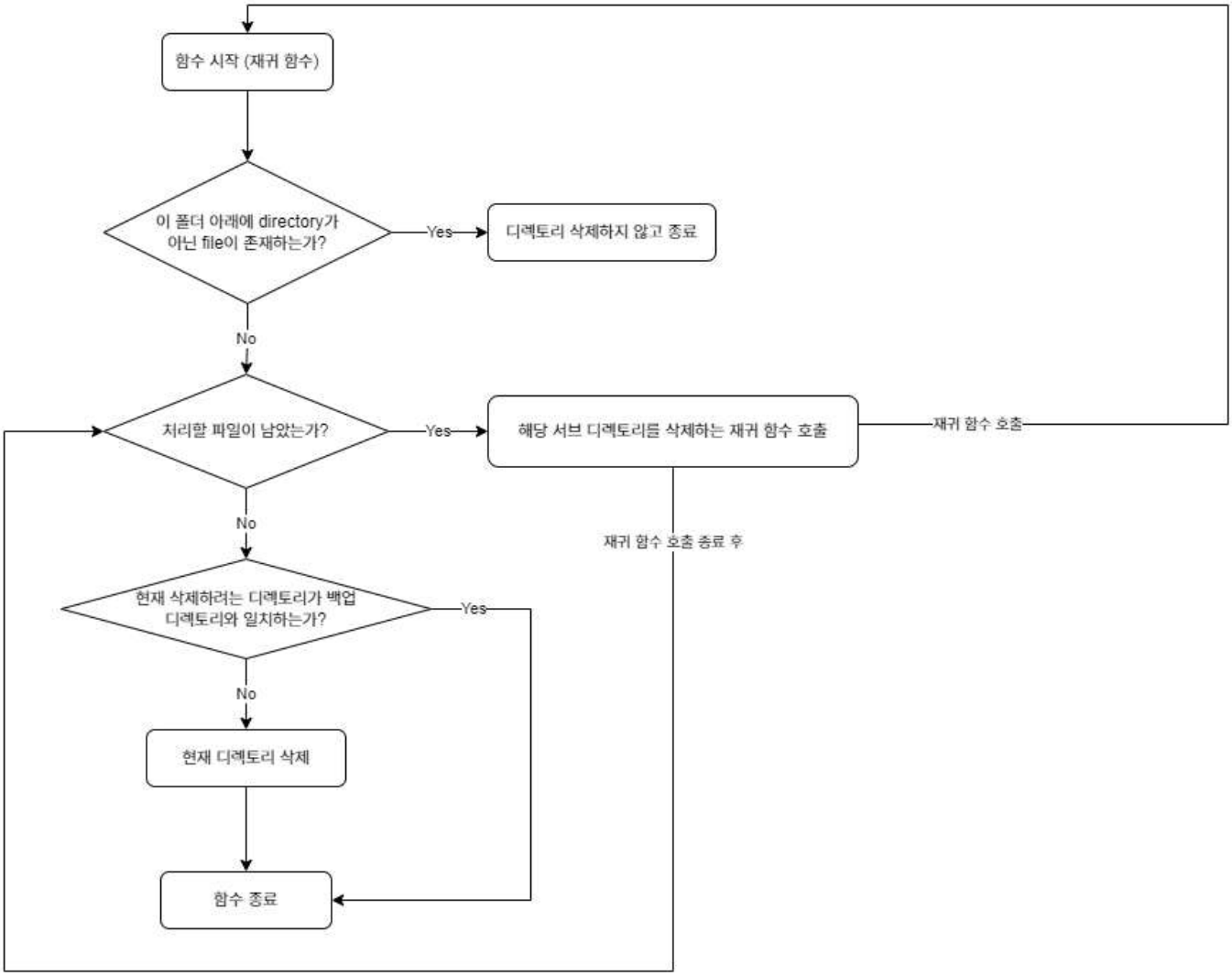
3-4-4-1. find_all_files 함수 순서도



3-4-4-2. find_same_name_files 함수 순서도

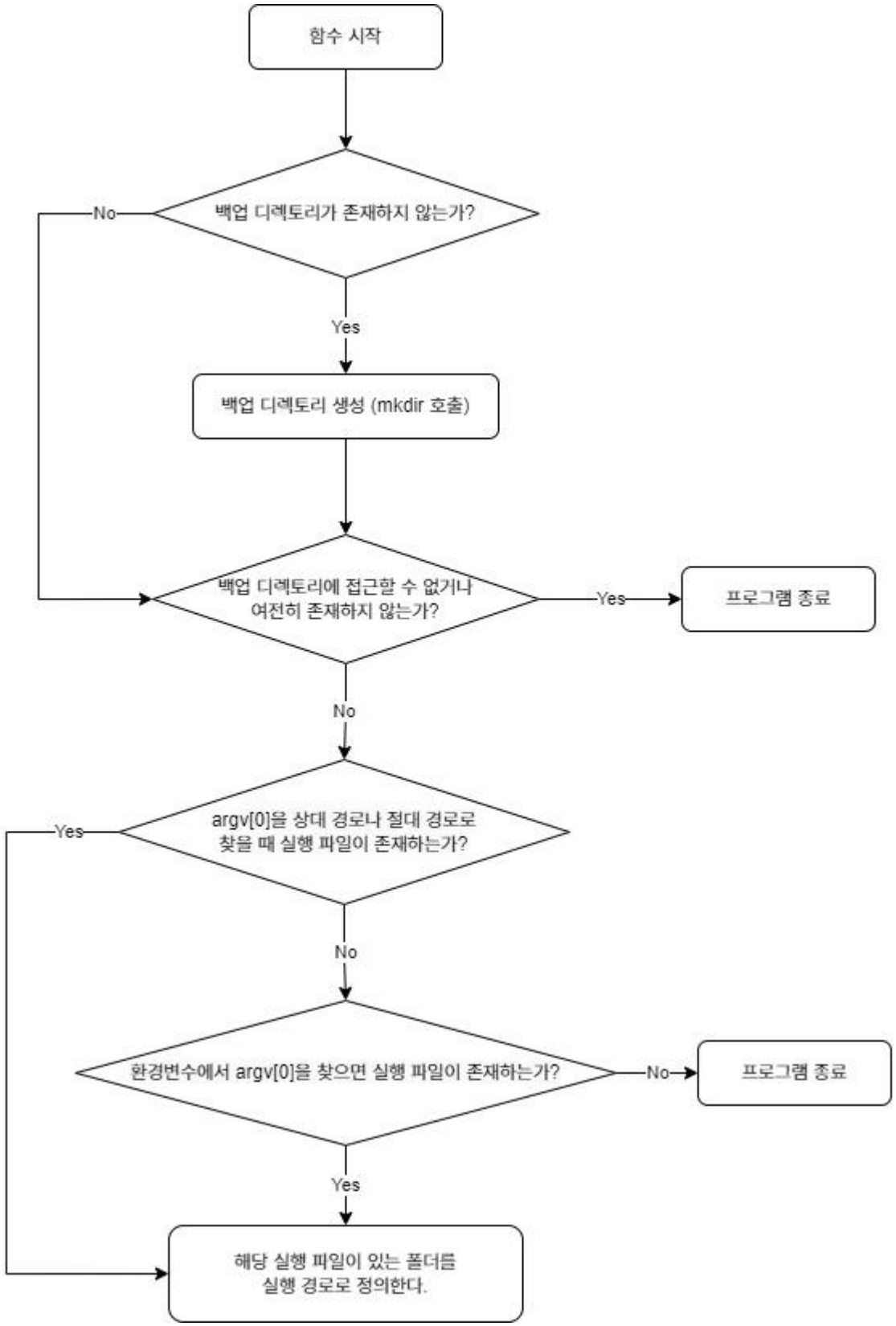


3-4-4-3. remove_backup_directory_safely 함수 순서도

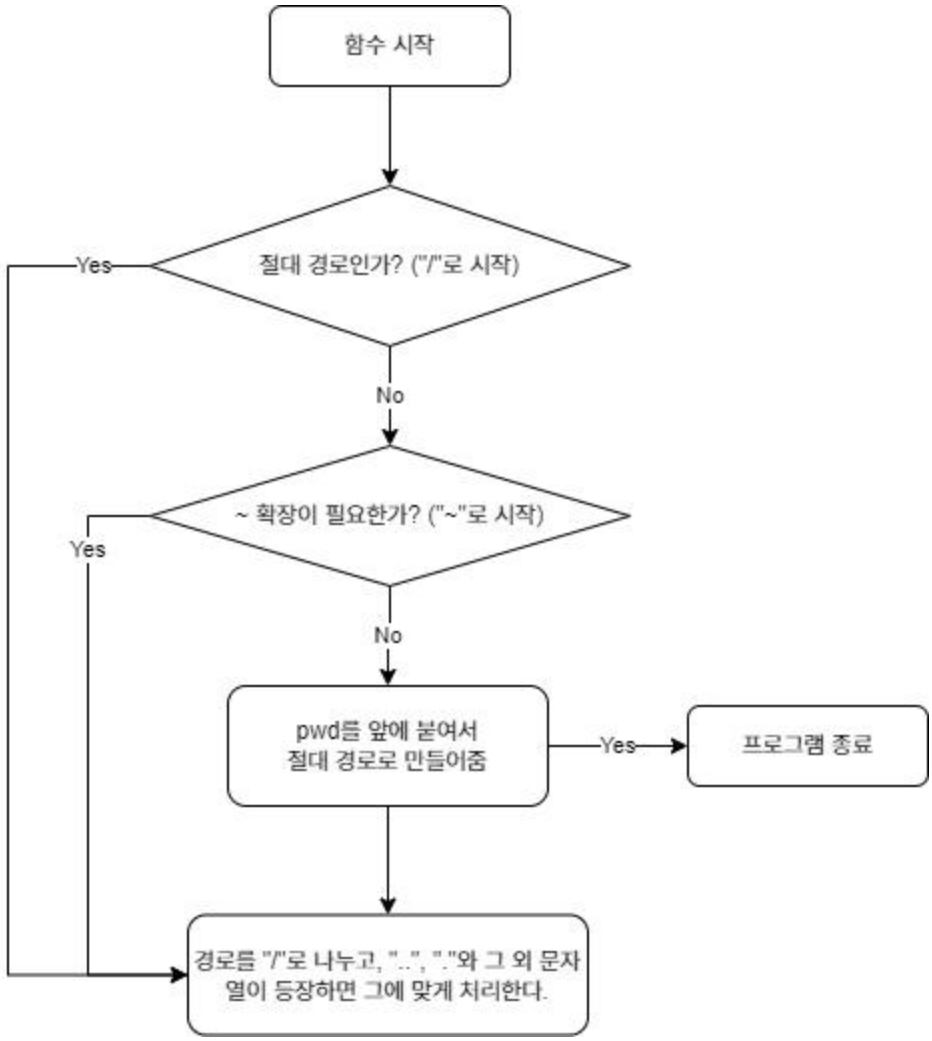


3-4-5. path.c 순서도

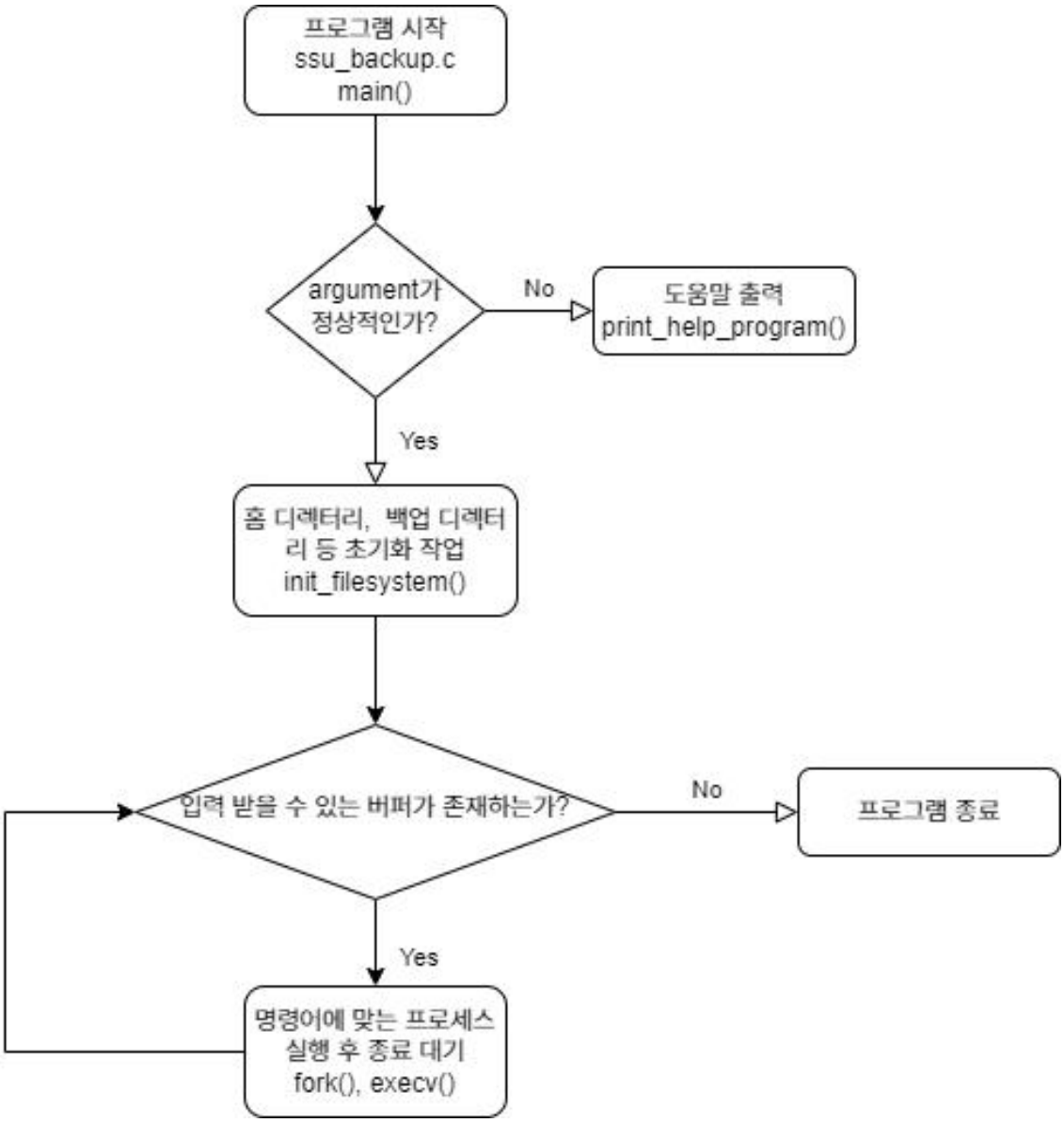
3-4-5-1. init_filesystem 함수 순서도



3-4-5-2. get_absolute_path 함수 순서도



3-4-6. ssu_backup 순서도



4. 실행결과

4-1. 메인 프로그램 (ssu_backup)

4-1-1. 인자 (해시 함수 이름) 작동 확인

```
nlog@nlog-vm-lsp:~/p1$ ./ssu_backup
Usage: ssu_backup <md5 | sha1>
nlog@nlog-vm-lsp:~/p1$ ./ssu_backup md4
Usage: ssu_backup <md5 | sha1>
nlog@nlog-vm-lsp:~/p1$ ./ssu_backup sha0
Usage: ssu_backup <md5 | sha1>
nlog@nlog-vm-lsp:~/p1$ ./ssu_backup sha1
20221494> ^C
nlog@nlog-vm-lsp:~/p1$ ./ssu_backup md5
20221494> ^C
nlog@nlog-vm-lsp:~/p1$
```

4-1-2. 명령어 실행 확인

```
nlog@nlog-vm-lsp:~/p1$ ./ssu_backup md5
20221494> help
Usage:
 > add [FILENAME] [OPTION]
 -d : add directory recursive
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
 > recover [FILENAME] [OPTION]
 -d : recover directory recursive
 -n [NEWNAME] : recover file with new name
 > ls
 > vi
 > vim
 > help
 > exit
20221494> add
Usage:
 > add <FILENAME> [OPTION]
 -d : add directory recursive
20221494> remove
Usage:
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
20221494> recover
Usage:
 > recover [FILENAME] [OPTION]
 -d : recover directory recursive
 -n [NEWNAME] : recover file with new name
20221494> ls
absolute_path.o  command_help     command_remove  data   debug.o  file.o  init.o         Makefile    ssu_backup.c  usage
command_add      command_recover  commands        debug  file     hash.o  linked_list.o  ssu_backup  test          usage.o
20221494> vi
20221494> vim
20221494> exit
20221494>
```

4-1-2. 프로그램 실행 파일 위치와 관계없이 내부 명령어 실행 여부 확인

```
nlog@nlog-vm-lsp:~$ ls p1/ssu_backup
p1/ssu_backup
nlog@nlog-vm-lsp:~$ ls p1/command*
p1/command_add  p1/command_help  p1/command_recover  p1/command_remove

p1/commands:
command_add.c  command_help.c  command_recover.c  command_remove.c  init.c  init.h
nlog@nlog-vm-lsp:~$ ./p1/ssu_backup md5
20221494> add
Usage:
 > add <FILENAME> [OPTION]
 -d : add directory recursive
20221494> recover
Usage:
 > recover [FILENAME] [OPTION]
 -d : recover directory recursive
 -n [NEWNAME] : recover file with new name
20221494> remove
Usage:
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
20221494> help
Usage:
 > add [FILENAME] [OPTION]
 -d : add directory recursive
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
 > recover [FILENAME] [OPTION]
 -d : recover directory recursive
 -n [NEWNAME] : recover file with new name
 > ls
 > vi
 > vim
 > help
 > exit
20221494> ^C
nlog@nlog-vm-lsp:~$
```

ssu_backup의 실행 파일 위치를 찾은 후, 같은 디렉토리 위치의 내부 명령어 실행 파일을 실행한다.

```
nlog@nlog-vm-lsp:~/Downloads$ PATH=$PATH:/home/nlog/p1/
nlog@nlog-vm-lsp:~/Downloads$ ssu_backup md5
20221494> add
Usage:
 > add <FILENAME> [OPTION]
 -d : add directory recursive
20221494> remove
Usage:
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
20221494> exit
nlog@nlog-vm-lsp:~/Downloads$
```

환경변수 PATH에 실행 파일 디렉터리가 등록되어 프로그램이 실행되는 경우에도 처리한다.

4-2. 명령어 add (command_add)

4-2-1. [필수 기능] 단일 파일 및 디렉터리 추가

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> add a
"/home/nlog/backup/test/a_230327211940" backuped
20221494> exit
nlog@nlog-vm-lsp:~/test$ ls ~/backup/
test
nlog@nlog-vm-lsp:~/test$ ls ~/backup/test/
a_230327211940
nlog@nlog-vm-lsp:~/test$
```

단일 파일 추가

```
nlog@nlog-vm-lsp:~/test$ rm ~/backup/ -r
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> add . -d
"/home/nlog/backup/test/a_230327212158" backuped
"/home/nlog/backup/test/c_230327212158" backuped
"/home/nlog/backup/test/b_230327212158" backuped
"/home/nlog/backup/test/d/e_230327212158" backuped
20221494> exit
nlog@nlog-vm-lsp:~/test$ ls ~/backup/test/ ~/backup/test/d/
/home/nlog/backup/test/:
a_230327212158  b_230327212158  c_230327212158  d

/home/nlog/backup/test/d/:
e_230327212158
nlog@nlog-vm-lsp:~/test$ ls . d
.:
a  b  c  d

d:
e
nlog@nlog-vm-lsp:~/test$
```

디렉터리 추가

4-2-2. [필수 기능] 경로가 동일한 파일 백업 (파일의 내용이 다를 때)

```
nlog@nlog-vm-lsp:~/test$ touch a
nlog@nlog-vm-lsp:~/test$ cat a | md5sum
d41d8cd98f00b204e9800998ecf8427e  -
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> add a
"/home/nlog/backup/test/a_230327213826" backuped
20221494> add a
"/home/nlog/backup/test/a_230327213826" is already backuped
20221494> ^C
nlog@nlog-vm-lsp:~/test$ echo abcd > a
nlog@nlog-vm-lsp:~/test$ cat a | md5sum
f5ac8127b3b6b85cdc13f237c6005d80  -
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> add a
"/home/nlog/backup/test/a_230327213845" backuped
20221494> ^C
nlog@nlog-vm-lsp:~/test$ cat ~/backup/test/a_230327213826
nlog@nlog-vm-lsp:~/test$ cat ~/backup/test/a_230327213845
abcd
nlog@nlog-vm-lsp:~/test$
```

4-2-3. [추가 기능] ~ (home directory/tilde) 확장 구현

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> add ~/test/a
"/home/nlog/backup/test/a_230327214531" backuped
20221494>
```

4-2-4. [예외 처리] 첫번째 인자 입력이 없거나 올바르지 않은 경로 입력

```
20221494> add ~/c
cannot access file "~/c" : No such file or directory
Usage:
 > add <FILENAME> [OPTION]
 -d : add directory recursive
20221494> add
Usage:
 > add <FILENAME> [OPTION]
 -d : add directory recursive
20221494> ^C
```

4-2-5. [예외 처리] regular 파일이나 디렉토리가 아닌 파일 입력이나 권한이 없는 경우

```
nlog@nlog-vm-lsp:~/test$ ln -s a b
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> add a
"/home/nlog/backup/test/a_230327214531" is already backuped
20221494> add b
"b" is not regular path and directory
Usage:
 > add <FILENAME> [OPTION]
 -d : add directory recursive
20221494> ^C
```

4-2-5-1. [예외 처리] 디렉토리 백업 시

```
nlog@nlog-vm-lsp:~/test$ ls -al
total 12
drwxrwxr-x  2 nlog nlog 4096  3월 27 21:52 .
drwxr-x--- 22 nlog nlog 4096  3월 27 21:23 ..
-rw-rw-r--  1 nlog nlog    5  3월 27 21:38 a
lrwxrwxrwx  1 nlog nlog    1  3월 27 21:52 b -> a
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> add . -d
"/home/nlog/backup/test/a_230327214531" is already backuped
20221494>
```

디렉토리 백업 시 regular 파일이나 디렉토리가 아닌 파일은 백업하지 않음

4-2-6. [예외 처리] 유저 디렉토리를 벗어나거나 백업 디렉토리 내 파일 백업 시도

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> add ~ -d
"~" can't be backuped
20221494> add ~/backup -d
"~/backup" can't be backuped
20221494> add / -d
"/" can't be backuped
20221494>
```

홈 디렉터리는 “백업 디렉토리 내 파일 및 디렉터리”를 포함하는 경로이므로 예외 처리한다.

4-2-7. [예외 처리] 올바르지 않은 옵션이 주어진 경우

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> add -d
Usage:
 > add <FILENAME> [OPTION]
 -d : add directory recursive
20221494> add . -d -a -b
Usage:
 > add <FILENAME> [OPTION]
 -d : add directory recursive
20221494>
```

4-2-8. [예외 처리] -d 옵션 없이 디렉토리 백업 시도

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> add .
"." is a directory file
```

4-2-9. [예외 처리] 경로가 같고, 파일 내용의 해시값이 같은 파일

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> add a
"/home/nlog/backup/test/a_230327214531" is already backuped
20221494> ^C
nlog@nlog-vm-lsp:~/test$ cat a | md5sum
f5ac8127b3b6b85cdc13f237c6005d80  -
nlog@nlog-vm-lsp:~/test$ cat ~/backup/test/a_230327214531 | md5sum
f5ac8127b3b6b85cdc13f237c6005d80  -
nlog@nlog-vm-lsp:~/test$
```

4-3. 명령어 remove (command_remove)

4-3-1. [필수 기능] 단일 파일 삭제 및 디렉토리 삭제

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> remove a
backup file list of "/home/nlog/test/a"
0. exit
1. 230328090421         3bytes
2. 230328090417         2bytes
Choose file to remove
>> 0
20221494> remove a
backup file list of "/home/nlog/test/a"
0. exit
1. 230328090421         3bytes
2. 230328090417         2bytes
Choose file to remove
>> 1
"/home/nlog/backup/test/a_230328090421" backup file removed
20221494> remove a
"/home/nlog/backup/test/a_230328090417" backup file removed
20221494>
```

단일 파일 삭제 (프롬프트)

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> remove a -a
"/home/nlog/backup/test/a_230327225401" backup file removed
"/home/nlog/backup/test/a_230327225410" backup file removed
20221494>
```

단일 파일 삭제 (-a 옵션으로 한번에 삭제)

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> remove . -a
"/home/nlog/backup/test/a_230327225507" backup file removed
"/home/nlog/backup/test/a_230327225511" backup file removed
20221494>
```

디렉토리 삭제

해당 기능(명령어)은 백업 디렉토리의 파일들에 대하여 종속적으로 구현되어 있다. 따라서, 홈 디렉터리 아래의 파일의 존재 유무와는 관련 없이 독립적으로 동작한다. (ex. remove 명령어의 인자로 받은 경로가 홈 아래에는 없더라도, 백업 디렉토리 아래에만 있으면 삭제 진행함.)

4-3-2. [필수 기능] 백업 디렉토리 내 모든 파일 삭제

```
nlog@nlog-vm-lsp:~/test$ ls ~/backup/ ~/backup/test/
/home/nlog/backup/:
test

/home/nlog/backup/test/:
a_230327225928
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> remove -c
backup directory cleared(1 regular files and 1 subdirectories totally).
20221494>
```

4-3-3. [추가 기능] ~ (home directory/tilde) 확장 구현

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> remove ~/test/a
"/home/nlog/backup/test/a_230327230357" backup file removed
20221494>
```

4-3-4. [예외 처리] 첫번째 인자 입력이 없거나 올바르지 않은 경로 입력

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> remove
Usage:
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
20221494> remove c
Usage:
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
20221494>
```

4-3-5. regular 파일이나 디렉토리가 아닌 파일 입력이나 권한이 없는 경우

```
nlog@nlog-vm-lsp:~/test$ ls -l ~/backup/test/a_230330093145
-rwxr--r-- 1 root root 3  3월 30 09:31 /home/nlog/backup/test/a_230330093145
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> remove a
'/home/nlog/backup/test/a_230330093145' doesn't have read/write permission.
20221494>
```

오직 프로그램만이 백업 디렉토리에 접근한다면, 해당 케이스는 일반적으로 발생하지 않는다. 위 경우는 파일을 백업한 후, 파일의 소유주와 권한을 변경하여 해당 파일에 대한 일반 유저(nlog)의 권한을 제거한 상태이다.

4-3-6. [예외 처리] 유저 디렉토리를 벗어나거나 백업 디렉토리 내 파일 삭제 시도

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> remove ~
"~" can't be removed.
20221494> remove ~/backup
"~/backup" can't be removed.
20221494> remove /
"/" can't be removed.
20221494>
```

4-3-7. [예외 처리] -a 옵션 없이 디렉토리 삭제 시도

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> remove .
"/home/nlog/backup/test" is a directory file, need -a option
20221494>
```

4-3-8. [예외 처리] 올바르지 않은 옵션이 주어진 경우

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> remove -a -c
option 'a' and 'c' duplicated
Usage:
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
20221494> remove . -c
option 'c' and argument 'file path' duplicated
Usage:
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
20221494> remove -c -d -e
Usage:
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
20221494>
```

4-3-9. [예외 처리] 백업 디렉토리가 비어있을 때, -c 옵션 사용

```
nlog@nlog-vm-lsp:~/test$ rm ~/backup/* -r
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> remove -c
no file(s) in the backup
20221494>
```

4-3-10. [기타] 백업 디렉토리 내 같은 경로로 된 파일과 디렉토리가 동시에 존재

```
nlog@nlog-vm-lsp:~$ ls backup/ backup/test
backup/:
test  test_230328090731

backup/test:
a_230328090716
nlog@nlog-vm-lsp:~$ ssu_backup md5
20221494> remove test
"/home/nlog/backup/test_230328090731" backup file removed
20221494> remove test
"/home/nlog/backup/test" is a directory file, need -a option
20221494> remove test -a
"/home/nlog/backup/test/a_230328090716" backup file removed
20221494>
```

백업 디렉토리 내 test 디렉토리와 파일이 동시에 존재한다.
-a 옵션이 주어지지 않는다면, 파일을 먼저 삭제한다. (파일이 여러 개라면 당연히 프롬프트를 제공한다.) 이후, -a 옵션이 주어지지는 않았으나 이미 파일을 삭제했기 때문에 폴더 삭제 시도와 관련된 별도의 에러 처리는 하지 않는다. (이 명령어는 '파일을 삭제하려는 의도'로 처리한 것이다.) 그러나, 폴더만 남아 있는 상태에서는 당연히 -a 옵션이 필요하다.

4-3-11. [기타] 빈 폴더를 삭제하려는 경우

```
nlog@nlog-vm-lsp:~$ ls backup/ backup/test/
backup/:
test

backup/test/:
nlog@nlog-vm-lsp:~$ ssu_backup md5
20221494> remove test -a
Usage:
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
20221494>
```

삭제할 일반(regular) 파일이 없으므로 '올바르지 않은(존재하지 않는) 경로'로 보고 사용법 출력한다.

4-4. 명령어 recover (command_recover)

4-4-1. [필수 기능] 단일 파일 복구 및 디렉토리 복구

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> recover a
backup file list of "/home/nlog/test/a"
0. exit
1. 230328131513         2bytes
2. 230328131521         3bytes
Choose file to recover
>> 0
20221494> recover a
backup file list of "/home/nlog/test/a"
0. exit
1. 230328131513         2bytes
2. 230328131521         3bytes
Choose file to recover
>> 1
"/home/nlog/backup/test/a_230328131513" backup recover to "/home/nlog/test/a"
20221494> recover a
"/home/nlog/backup/test/a_230328131521" backup recover to "/home/nlog/test/a"
20221494>
```

단일 파일 복구 (프롬프트)

```
nlog@nlog-vm-lsp:~/test$ ls ~/backup/test/
a_230328130422  a_230328130448  b_230328130354  c_230328130355
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> recover . -d
"/home/nlog/backup/test/c_230328130355" backup recover to "/home/nlog/test/c"
backup file list of "/home/nlog/test/a"
0. exit
1. 230328130422         2bytes
2. 230328130448         5bytes
Choose file to recover
>> 1
"/home/nlog/backup/test/a_230328130422" backup recover to "/home/nlog/test/a"
"/home/nlog/backup/test/b_230328130354" backup recover to "/home/nlog/test/b"
20221494> exit
nlog@nlog-vm-lsp:~/test$ ls ~/backup/test/
a_230328130448
nlog@nlog-vm-lsp:~/test$
```

디렉터리 복구 (여러 개의 파일 존재 시 프롬프트 제공)

4-4-2. [필수 기능] 새로운 경로로 복구 진행

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> recover a -n new_a
"/home/nlog/backup/test/a_230328130448" backup recover to "/home/nlog/test/new_a"
20221494>
```

단일 파일

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> recover . -d -n ./test_in_test
"/home/nlog/backup/test/new_a_230328160535" backup recover to "/home/nlog/test/test_in_test/new_a"
"/home/nlog/backup/test/b_230328160535" backup recover to "/home/nlog/test/test_in_test/b"
"/home/nlog/backup/test/a_230328160535" backup recover to "/home/nlog/test/test_in_test/a"
"/home/nlog/backup/test/c_230328160535" backup recover to "/home/nlog/test/test_in_test/c"
20221494>
```

디렉토리

4-4-3. [추가 기능] ~ (home directory/tilde) 확장 구현

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> recover ~/test/a
"/home/nlog/backup/test/a_230328160907" backup recover to "/home/nlog/test/a"
20221494> recover ~/test/b -n ~/test_b
"/home/nlog/backup/test/b_230328160907" backup recover to "/home/nlog/test_b"
20221494> exit
nlog@nlog-vm-lsp:~/test$ ls a ~/test_b
a  /home/nlog/test_b
nlog@nlog-vm-lsp:~/test$
```

4-4-4. [예외 처리] 첫번째 인자 입력이 없거나 올바르지 않은 경로 입력

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> recover
Usage:
 > recover [FILENAME] [OPTION]
 -d : recover directory recursive
 -n [NEWNAME] : recover file with new name
20221494> recover a
Usage:
 > recover [FILENAME] [OPTION]
 -d : recover directory recursive
 -n [NEWNAME] : recover file with new name
20221494>
```

4-4-5. regular 파일이나 디렉토리가 아닌 파일 입력이나 권한이 없는 경우

```
nlog@nlog-vm-lsp:~/test$ ls -l ~/backup/test/a_230330093145
-rwxr--r-- 1 root root 3  3월 30 09:31 /home/nlog/backup/test/a_230330093145
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> recover a
/home/nlog/backup/test/a_230330093145 doesn't have read/write permission.
20221494>
```

오직 프로그램만이 백업 디렉토리에 접근한다면, 해당 케이스는 일반적으로 발생하지 않는다. 위 경우는 파일을 백업한 후, 파일의 소유주와 권한을 변경하여 해당 파일에 대한 일반 유저(nlog)의 권한을 제거한 상태이다.

4-4-6. [예외 처리] 입력된 경로가 유저 디렉토리를 벗어나거나 백업 디렉토리 내 파일 복구 시도

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> recover
Usage:
 > recover [FILENAME] [OPTION]
 -d : recover directory recursive
 -n [NEWNAME] : recover file with new name
20221494> recover ~
"~" can't be recovered
20221494> recover /
"/" can't be recovered
20221494> recover ~/backup
"~/backup" can't be recovered
20221494> recover . -n /asdf
"/asdf" can't be recovered
20221494>
```

4-4-7. [예외 처리] -d 옵션 없이 디렉토리 복구 시도

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> recover .
"/home/nlog/backup/test" is a directory file, need -d option
Usage:
 > recover [FILENAME] [OPTION]
 -d : recover directory recursive
 -n [NEWNAME] : recover file with new name
20221494>
```

4-4-8. [예외 처리] 올바르지 않은 옵션이 주어진 경우

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> recover -a -d -c
Usage:
 > recover [FILENAME] [OPTION]
 -d : recover directory recursive
 -n [NEWNAME] : recover file with new name
20221494>
```

4-4-9. [예외 처리] 복원할 파일과 그 대상 파일의 해시값이 같은 경우

```
nlog@nlog-vm-lsp:~/test$ cat a | md5sum
c46e1ffcd01eca4d4f462f05144c0f50  -
nlog@nlog-vm-lsp:~/test$ cat ~/backup/test/a_230328162201 | md5sum
c46e1ffcd01eca4d4f462f05144c0f50  -
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> recover a
"/home/nlog/test/a" and "/home/nlog/backup/test/a_230328162201" are same files.
20221494>
```

4-4-10. [예외 처리] 백업 대상 경로에 대하여 재귀적으로 폴더 생성

```
nlog@nlog-vm-lsp:~$ ls test/
ls: cannot access 'test/': No such file or directory
nlog@nlog-vm-lsp:~$ ssu_backup md5
20221494> recover test/a
"/home/nlog/backup/test/a_230328162201" backup recover to "/home/nlog/test/a"
20221494> exit
nlog@nlog-vm-lsp:~$ ls test/
a
nlog@nlog-vm-lsp:~$
```

4-5. 명령어 ls, vi(m)

4-5-1. [필수 기능] 외부 명령어 실행 확인

```
nlog@nlog-vm-lsp:~$ ssu_backup md5
20221494> ls
backup  Desktop  Documents  Downloads  Music  p1  Pictures  Public  snap  Templates  test  Videos
20221494> vi
20221494>
```

4-5-2. [추가 기능] 시스템 함수 인자 입력

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> ls -al
total 12
drwxrwxr-x  2 nlog nlog 4096  3월 28 16:28 .
drwxr-x--- 22 nlog nlog 4096  3월 28 16:28 ..
-rw-rw-r--  1 nlog nlog    3  3월 28 16:28 a
20221494>
```

4-5-3. [추가 기능] 설치되지 않은 프로그램 실행

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> vim
[ERROR] cannot find "vim" executable path.
20221494>
```

vim이 시스템이 설치되어 있지 않으면 에러 처리한다.

4-6. 명령어 help (command_help)

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> help
Usage:
 > add [FILENAME] [OPTION]
 -d : add directory recursive
 > remove [FILENAME] [OPTION]
 -a : remove all file(recursive)
 -c : clear backup directory
 > recover [FILENAME] [OPTION]
 -d : recover directory recursive
 -n [NEWNAME] : recover file with new name
 > ls
 > vi
 > vim
 > help
 > exit
20221494>
```

4-7. 명령어 exit

```
nlog@nlog-vm-lsp:~/test$ ssu_backup md5
20221494> exit
nlog@nlog-vm-lsp:~/test$
```

4-8. Makefile

4-8-1. build

```
nlog@nlog-vm-lsp:~/p1$ make
file/hash.c: In function ‘hash_file’:
file/hash.c:22:9: warning: ‘MD5’ is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
   22 |         MD5(content, length, ret);
      |         ^~~
In file included from file/hash.c:7:
/usr/include/openssl/md5.h:52:38: note: declared here
   52 | OSSL_DEPRECATEDIN_3_0 unsigned char *MD5(const unsigned char *d, size_t n,
      |                                      ^~~
file/path.c: In function ‘init_filesystem’:
file/path.c:21:29: warning: ‘%s’ directive writing 6 bytes into a region of size between 0 and 4095 [-Wformat-overflow=]
   21 |     sprintf(BACKUP_DIR, "%s/%s", USER_DIR, "backup");
      |                             ^~              ~~~~~~~~
file/path.c:21:5: note: ‘sprintf’ output between 8 and 4103 bytes into a destination of size 4096
   21 |     sprintf(BACKUP_DIR, "%s/%s", USER_DIR, "backup");
      |     ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ssu_backup.c: In function ‘main’:
ssu_backup.c:75:34: warning: ‘%s’ directive writing up to 8191 bytes into a region of size 4095 [-Wformat-overflow=]
   75 |                 sprintf(tmp, "%s/%s", dir, cmd);
      |                                  ^~   ~~~
ssu_backup.c:75:17: note: ‘sprintf’ output 2 or more bytes (assuming 8193) into a destination of size 4096
   75 |                 sprintf(tmp, "%s/%s", dir, cmd);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Build complete
nlog@nlog-vm-lsp:~/p1$
```

4-8-2. clean

```
nlog@nlog-vm-lsp:~/p1$ make clean
rm -f ssu_backup command_add command_remove command_recover command_help *.o
nlog@nlog-vm-lsp:~/p1$
```