컴퓨터학부 / 20192393 / 김현우

**1 . 과제개요**

프로그램 실행 시 내장명령어 ( add, delete, tree, help, exit ) 으로 실행되는 프로그램이다 .

각각의 명령어 마다 역할을 설명하겠다.

**“add”** : 사용자가 입력한 디렉토리를 디몬 프로세스로 모니터링을 하도록 한다.

**“delete”** : 사용자가 입력한 프로세스 번호로 현재 실행중인 디몬 프로세스를 시그널을 보내어 디몬 프로세스를 종료한다.

**“tree”** : 사용자가 입력한 디렉토리를 디렉토리 구조를 tree 형태로 출력한다.

**“help”** : 사용법 출력한다.

디몬 프로세스로 모니터링을 하고 있는 동안 모니터링을 하는 디렉토리에 파일에 대해 생성, 삭제, 수정이 발생 했을 경우 발생한 사건에 따라 로그를 출력해서 확인하는 프로그램이다.

**2 . 구현 기능 ( 함수 프로토 타입 )**

**[ ssu\_monitor.c ]**

int get\_command(char \*command, int argc, char (\*argv)[BUFFER\_SIZE]); // 내장 명령어 구하는 함수

void add(char (\*argv)[BUFFER\_SIZE); // add 실행 함수

int ssu\_daemon\_init(char \*monitor\_list); // daemon process 생성함수

int exist\_directory(char \*dir\_path); // directory 존재 여부 확인하는 함수

int check\_monitoring\_dir(char \*dir\_path); // monitor\_list.txt 에 존재 여부 확인하는 함수

void delete(char (\*argv)[BUFFER\_SIZE]); // delete 실행 함수

int check\_monitoring\_pid(char \*pid); // monitor\_list.txt 에 존재 여부 확인하는 함수

void ssu\_signal\_handler(int signo);

void delete\_monitor\_list(char \*delete\_line); // monitor\_list.txt 에서 해당 줄 삭제하기

void tree(char (\*argv)[BUFFER\_SIZE]); // tree 실행 함수

void draw\_tree(char \*dir\_path, int level); // tree 그리는 함수

void help();

**[ ssu\_daemon.c ]**

void monitoring(char \*monitor\_path, int sleep\_time); // 모니터링 함수

file\_node\* make\_list(char \*path); // 디렉토리 파일 목록 트리화하는 함수

void compare\_list(file\_node \*new\_list, file\_node \*old\_list); // 파일 목록 트리 비교하는 함수

int compare\_file(file\_node \*new\_file, file\_node \*old\_file); // 파일 정보 비교하는 함수

int write\_change\_list(file\_node \*head, int change\_cnt, int status); // 변경 사항 목록 작성하는 함수

void write\_change\_log(int change\_cnt); // 변경사항 파일 기록하는 함수

void init\_list\_status(file\_node \*head, int status); // 모니터링 파일 상태 초기화하는 함수

file\_node\* make\_node(); // 새로운 file\_node 생성 함수

void free\_list(file\_node \*head); // 모니터링 파일 목록 메모리 할당 해제하는 함수

char\* make\_time\_format(struct tm time); // 시간 형식 문자열 생성하는 함수

int ssu\_daemon\_init(char \*monitor\_list); // 데몬프로세스 설정

**3. 상세설계**

**1. [ ssu\_monitor.c ] 흐름도**

스크린샷, 직사각형, 텍스트, 흑백이(가) 표시된 사진

자동 생성된 설명

**2. [ ssu\_daemon.c ] 흐름도**

**스크린샷, 텍스트, 디자인이(가) 표시된 사진

자동 생성된 설명**

**[ 함수 기능별 자세 설명 ]**

**[ ssu\_monitor.c ]**

**1. int get\_command(char \*command, int argc, char (\*argv)[BUFFER\_SIZE]);**

- 사용자가 입력한 값을 통해 “ ”, “\n”, “\0” 을 구분자로 명령어를 구한다.

**2. void add(char(\*argv)[BUFFER\_SIZE);**

- 입력한 경로에 따라 상대경로일 경우 절대경로를 만들어준다.

- 위에서 만든 경로를 가지고 해당 경로가 존재하는 경로인지 **int exist\_directory(char \*dir\_path)** 함수를 이용해서 확인한다.

- 위에서 만든 경로를 가지고 해당 경로가 “monitor\_list.txt” 에 존재하는 경로인지 **int check\_monitoring\_dir(char \*dir\_path)** 함수를 이용해서 확인한다.

- **fork()** 함수를 이용하여 자식 프로세스를 생성해서 **void monitoring(char \*monitor\_path, int sleep\_time)** 함수를 실행한다.

**3. void delete(char (\*argv)[BUFFER\_SIZE);**

- **int check\_monitoring\_pid(char \*dir\_pid)** 함수를 이용해서 “monitor\_list.txt” 에 존재하는 프로세스 번호인지 확인한다.

- **void delete\_monitor\_list(char \*delete\_line)** 함수를 이용해서 “monitor\_list.txt” 에 존재하는 삭제할 줄을 제외하고 “temp.txt” 에 한 줄씩 저장하고 rename 으로 “monitor\_list.txt” 으로 바꾸어 저장한다.

- 해당 프로세스 번호를 SIGUSR! 시그널을 보내 디몬 프로세스를 종료한다.

**4. void tree(char (\*argv)[BUFFER\_SIZE);**

- 입력한 경로에 따라 상대경로일 경우 절대경로를 만들어준다.

- 위에서 만든 경로를 가지고 해당 경로가 존재하는 경로인지 **int exist\_directory(char \*dir\_path)** 함수를 이용해서 확인한다.

- 위에서 만든 경로를 가지고 해당 경로가 “monitor\_list.txt” 에 존재하는 경로인지 **int check\_monitoring\_dir(char \*dir\_path)** 함수를 이용해서 확인한다.

- **void draw\_tree(char \*dir\_path, int level)** 함수를 이용해서 입력받은 해당 디렉토리 내부를 tree 형태로 그린다.

**5. void help();**

- Usage 출력한다.

**[ ssu\_daemon.c ]**

- **int ssu\_daemon\_init(char \*monitor\_list)** 함수를 이용해서 daemon process 설정한다.

- **file\_node\* make\_list(char \*path)** 함수를 이용해서 입력받은 해당 경로에 대해 현재 파일 목록을 저장한다.

- **void compare\_list(file\_node \*new\_list, file\_node \*old\_list)** 함수를 이용해서 현재 파일 목록과 새 파일 목록을 비교할때 파일 목록에서 파일 하나하나씩 비교하기 위해 **int compare\_file(file\_node \*new\_file, file\_node \*old\_file)** 함수를 이용한다.

- **int compare\_file(file\_node \*new\_file, file\_node \*old\_file)** 함수에서 바뀐 파일에 대해 어떤 상태로 바뀌었는지 값을 저장한다.

- **int write\_change\_list(file\_node \*head, int change\_cnt, int status)** 함수로 파일 목록에 존재하는 파일을 확인해서 파일 변경 상태 여부를 저장한다.

- **void write\_change\_log(int change\_cnt)** 함수를 이용해서 저장한 변경 상태를 “log.txt” 에 작성한다.

- **void init\_list\_status(file\_node \*head, int status)** 함수를 이용해서 파일의 변경 상태를 모두 “UNCKECK” 로 바꾸어서 다시 검사하도록 초기화한다.

**4. 실행결과**

**- 프로그램 실행 시**



- 프롬프트 상에서 엔터만 입력 시 프롬프트 재 출력, 지정한 내장명령어 외 기타 명령어 입력 시 예외처리

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**- add 명령어**

- 첫번째 인자로 상대경로와 절대경로 모두 입력가능

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

- 위의 add 명령어 실행 후 “monitor\_list.txt” 상태



- “- t” option 추가해서 실행



- 첫 번째 인자 입력이 없거나 잘못되었을 경우 예외처리

텍스트, 폰트, 화이트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

- 첫 번째 인자로 입력 받은 경로가 존재하지 않거나 디렉토리가 아닐 경우 예외처리

텍스트, 폰트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

- 첫 번째 인자로 입력 받은 경로가 이미 모니터링 중인 디렉토리의 경로와 같거나 포함하거나 포함될 때 예외처리

- 명령어 실행 전 “monitor\_list.txt”

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

- 디렉토리의 경로와 같은 경우



- 디렉토리의 경로가 포함되 경우





- “-t” option 사용이 올바르지 않은 경우 출력 후 프롬프트 재출력



**- delete 명령어**

- 명령어 실행 전 “monitor\_list.txt”

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

- delete 명령어 실행 시



- 명령어 실행 후 “monitor\_list.txt”



- 첫 번째 인자가 “monitor\_list.txt” 에 존재하지 않는 경우 예외처리



-

**- tree 명령어**

- 명령어 실행 전 “monitor\_list.txt”



- 첫번째 인자로 상대경로를 입력한 경우

텍스트, 폰트, 스크린샷, 영수증이(가) 표시된 사진

자동 생성된 설명

- 첫번째 인자로 절대경로를 입력한 경우



- 첫 번째 인자가 “monitor\_list.txt” 에 존재하지 않는 경우 예외처리



**- help 명령어**

- 명령어 실행 시

텍스트, 폰트, 화이트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

**- exit 명령어**

- 명령어 실행 시



**- log.txt 형태**

텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

자동 생성된 설명