컴퓨터학부 20201841 박세연

1. 과제개요

ssu\_sycn 프로그램은 사용자가 지정한 경로에 대해 해당 경로 및 하위 경로들을 지속적으로 모니터링하고, 파일 변경 사항을 실시간으로 감지하여 백업을 수행하거나 변경 사항을 로그로 기록하는 데몬 프로세스 기반의 프로그램이다. 주요 명령어로는 특정 파일이나 디렉터리를 모니터링 리스트에 추가하는 add, 모니터링 리스트에서 제거하는 remove, 현재 모니터링 중인 디렉터리 리스트를 조회하는 기능인 list, 프로그램의 사용법을 안내하는 도움말 기능인 help, 프로그램을 종료하는 기능인 exit이 있다.

add는 백그라운드에서 모니터링하며 변경 내역을 추적하고 백업할 경로를 입력 받아, 해당 파일 또는 디렉터리에 대한 데몬 프로세스를 추가하는 명령어이다. 입력 받은 경로와 그 하위 경로에서 발생하는 모든 파일 변화는 실시간으로 감지되어 백업되거나 로그로 기록된다.

remove는 PID를 인자로 입력 받아 데몬 프로세스를 제거하여 모니터링을 중지하고 백업 파일을 삭제하는 명령어이다. 더 이상 이 경로에 대한 변화는 감지되지 않으며, 백업 또는 로그 기록도 중지된다.

list는 현재 모니터링 중인 데몬 프로세스 혹은 데몬 프로세스가 관리중인 경로에 대해 트리 형태로 출력하는 명령어이다. 인자를 입력하지 않으면, 현재 실행중인 데몬 프로세스들의 목록을 모두 출력한다. 인자로 PID를 입력 받은 경우에는 해당 데몬 프로세스가 관리 중인 경로에 대해 트리 형태로 출력한다.

help는 위 프로그램에서 사용할 수 있는 모든 명령어에 대한 설명을 출력해주는 명령어이다. help 뒤에 명령어 인자를 넘겨주지 않으면 전체 명령어에 대한 usage를 볼 수 있으며, 인자로 특정 명령어를 입력한다면 해당 명령어의 usage만 단독으로 출력한다.

exit 명령어는 실행 중인 ssu\_sync 프로그램을 종료한다. 사용자는 exit을 입력하기 전까지 계속해서 프로그램을 사용할 수 있고 exit 명령어를 통해 프로그램을 종료한다.

2. 구현 기능

**<ssu\_header.h>**

// 실행가능한 명령어를 담고있는 문자열 배열

char \*commanddata[10] = {

"add",

"remove",

"list",

"help",

"exit"};

// 입력받은 명령어와 인자를 저장할 구조체

typedef struct command\_parameter

{

char \*command; // 명령어

char \*filename;

char \*tmpname;

int commandopt;

char \*argv[10];

} command\_parameter;

// 파일 노드 구조체

typedef struct fileNode

{

char originPath[PATHMAX]; // 파일의 원본 경로

char backupPath[PATHMAX]; // 파일의 백업 경로

char \*fileName; // 파일의 이름 추출

time\_t modTime; // 변경 시간

struct fileNode \*prevFile; // 이전 파일노드를 가리키는 포인터

struct fileNode \*nextFile; // 다음 파일노드를 가리키는 포인터

} fileNode;

// 디렉터리 리스트 구조체

typedef struct dirList

{

struct dirNode \*head; // 디렉터리 노드 리스트의 시작 지점을 가리키는 포인터

struct dirNode \*tail; // 디렉터리 노드 리스트의 끝 지점을 가리키는 포인터

} dirList;

// 디렉터리 노드 구조체

typedef struct dirNode

{

char dirPath[PATHMAX]; // 디렉터리 경로

struct fileNode \*headFile; // 디렉터리에 속한 파일들의 시작 파일을 가리키는 포인터

struct fileNode \*tailFile; // 디렉터리에 속한 파일들의 끝 파일을 가리키는 포인터

struct dirList \*subDirs; // 하위 디렉터리 리스트를 가리키는 포인터

struct dirNode \*prevDir; // 이전 디렉터리 리스트를 가리키는 포인터

struct dirNode \*nextDir; // 다음 디렉터리 리스트를 가리키는 포인터

} dirNode;

typedef struct pathList\_

{

struct pathList\_ \*next;

struct pathList\_ \*prev;

char path[NAMEMAX];

} pathList;

void help(); // help 명령어 실행 함수

int md5(char \*target\_path, char \*hash\_result); // md5 해시 값 계산 함수

int ConvertHash(char \*target\_path, char \*hash\_result); // 해시 계산 함수

char \*GetFileName(char file\_path[]); // 파일명만 추출하는 함수

char \*getDate(); // 현재 날짜와 시간을 문자열로 반환하는 함수

char \*convertDateTime(char \*date); // 날짜를 일정 형식으로 반환해주는 함수

char \*QuoteCheck(char \*\*str, char del); // 따옴표 안에 있는 문자열을 확인하고 해당 문자열의 끝을 찾아서 반환

char \*Tokenize(char \*str, char \*del); // 문자열을 구분자로 토큰화하여 토큰을 반환하는 함수

char \*\*GetSubstring(char \*str, int \*cnt, char \*del); // 문자열을 구분자로 나누어서 부분 문자열을 배열에 저장

int ConvertPath(char \*origin, char \*resolved); // 전체 경로로 변환해주는 함수

void printToLogFile(char \*logfilePath, char \*date, char \*path, int flag); // 파일의 상태를 로그 파일에 출력

**<ssu\_sync.c>**

void printFileEvents(const char \*logFilePath, const char \*targetFilePath); // List -> 경로가 일반 파일일 때 출

int checkPidInFile(char \*pid, char \*filepath); // pid가 존재하는지 확인하고 (존재한다면) filepath에 경로를 넘겨줌

int ListCommand(command\_parameter \*parameter); // List 명령어

void removeStringFromFile(const char \*monitoringListLogFilePath, const char \*string\_to\_remove);

// 삭제하려는 pid 로그 내용만 지우는 함수 (그 문자만 지우고 다른 문자들은 다시 복사)

int emptyDirectory(const char \*dirname); // 디렉터리 비우는 함수

int RemoveCommand(command\_parameter \*parameter); // Remove 명령어

int isDaemonExist(char \*path); // // 해당 데몬 프로세스가 이미 존재하는지 확인하는 함수

void backupFile(char \*path, char \*date, char \*basedir, char \*logfilePath, int flag);

// 백업 파일을 생성하는 함수 (로그파일경로도 함께 넘겨줌)

bool isFileModified(const char \*filePath); // 파일이 수정되었는지 여부를 확인하는 함수

dirNode \*initDirNode(char \*dirPath); // 디렉터리 초기화 함수

void insertFileNode(char \*logPath, dirNode \*toDir, char \*backupDirPath, char \*currentFilePath); // 파일 삽입 함수

void insertDirNode(char \*logPath, dirNode \*toDir, char \*backupDirPath, char \*currentDirPath);

// 디렉터리 삽입 함수

fileNode \*findBackupFile(char \*originalPath, dirNode \*currentDir);

// 현재 디렉터리(currentDir) 내에서 원본 파일 경로(originalPath)에 해당하는 백업 파일 노드를 검색하는 함수

void traverseDirectoryNode(char \*logPath, dirNode \*currentBackupDir, char \*backupDirPath, char \*currentDirPath, int flag1); // 디렉터리 순회 함수

void validateRemovedFiles(char \*logPath, dirNode \*currentDir);

// 현재 디렉터리(currentDir) 내에서 삭제된 파일을 확인하는 함수

void monitoringFile(char \*filePath, int flag1, int period);

// 지정된 파일 또는 디렉터리를 변경 사항을 모니터링하고, 백업 수행

int AddCommand(command\_parameter \*parameter); // Add 명령어

void CommandFun(char \*\*arglist); // 입력된 명령어에 따라 명령 함수 실행

void CommandExec(command\_parameter parameter); // 명령어 실행 함수

void HelpExec(char \*command); // help 실행 함수

void ParameterInit(command\_parameter \*parameter); // 파라미터 관련 구조체 초기화 함수

int ParameterProcessing(int argcnt, char \*\*arglist, int command, command\_parameter \*parameter);

// 명령줄로 받은 명령어들을 정리하고 알맞게 저장하는 함수

int Prompt(); // 사용자에게 명령어를 계속 입력받기 위해 호출하는 함수

void Init(); // 초기화 함수

**<ssu\_help.c>**

// help 명령어를 실행하는 함수로 command 인자로 출력하려는 명령어 받음

// 사용자가 인자를 생략할 경우 임의로 “all” 문자열을 command로 받아 모든 명령어에 대한 usage 출력

void help(char \*command);

void helpAdd(); // add 명령어에 대한 usage를 출력하는 함수

void helpRemove(); // remove 명령어에 대한 usage를 출력하는 함수

void helpList(); // list 명령어에 대한 usage를 출력하는 함수

void helpHelp(); // help 명령어에 대한 usage를 출력하는 함수

void helpExit(); // exit 명령어에 대한 usage를 출력하는 함수

3. 상세 설계

**<메인 흐름도>**

도표, 스케치, 그림, 종이접기이(가) 표시된 사진

자동 생성된 설명

**<add 명령어 – AddCommand() 흐름도>**

텍스트, 도표, 영수증, 평면도이(가) 표시된 사진

자동 생성된 설명

**<remove 명령어 – RemoveCommand() 흐름도>**

**텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

**<list 명령어 – ListCommand() 흐름도>**

도표, 라인, 텍스트, 스케치이(가) 표시된 사진

자동 생성된 설명

4. 실행결과

**(1) ssu\_sync**

텍스트, 소프트웨어, 컴퓨터, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

- ssu\_sync 프로그램 실행 시 백업 디렉터리 생성, 백업 디렉터리 하위에 monitor\_list.log 파일 생성

- 엔터 입력 시 프롬프트 재 출력

- 지정한 명령어 이외의 명령어 입력 시 help 명령어의 결과 출력

**(2) 내장 명령어 1. add**

텍스트, 소프트웨어, 웹 페이지, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

- 일반 파일 경로를 인자로 넘겨주어 add 한 예시

- 생성된 PID를 이름으로 하는 디렉터리 하위에 백업 파일이 잘 생성되고, 해당 PID를 이름으로 한 로그 파일에 변경 기록이 출력

- vi로 파일 내용을 변경하면 로그 파일에 modify가 출력되고, 변경된 파일이 백업

- rm로 파일을 삭제하면 로그파일에 remove가 출력

텍스트, 소프트웨어, 웹 페이지, 웹사이트이(가) 표시된 사진

자동 생성된 설명

- 디렉터리 경로를 인자로 넘겨주고 옵션은 d를 사용한 예시

- d 옵션으로 인해 디렉터리 내의 파일들만 백업 및 추적

- 로그 파일을 살펴보면 파일들만 create

- vi로 해당 디렉터리 내의 파일을 수정하면 로그 파일에 modify가 출력되고, 새로운 백업 파일이 생성

- rm로 해당 디렉터리 내의 파일을 삭제하면 로그 파일에 remove가 출력

- touch로 해당 디렉터리 내에 새로운 파일을 생성하면 로그 파일에 create로 새로운 파일이 생성됨을 출력하고 마찬가지로 백업 파일이 생성

텍스트, 소프트웨어, 스크린샷, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

- 디렉터리 경로를 인자로 넘겨주고 옵션은 r을 사용한 예시

- r 옵션으로 인해 디렉터리 내의 파일들과 서브 디렉터리의 파일들까지 모두 백업 및 추적

- 로그 파일을 살펴보면 서브 디렉터리들의 파일들까지 모두 create 되었음을 확인

- vi로 서브 디렉터리의 파일을 수정하면 로그 파일에 modify가 출력되고 백업 파일이 생성

- rm으로 서브 디렉터리의 파일을 삭제하면 로그 파일에 remove가 출력

- touch로 서브 디렉터리 내에 파일을 생성하면 로그 파일에 create가 출력되고 백업 파일이 생성

스크린샷, 텍스트, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

- t옵션을 사용하여 add 명령어를 실행한 예시

- 예시에서는 t옵션과 함께 30을 인자로 주었다. 즉, 30초 마다 모니터링하라는 명령

- 처음 create 된 시각이 05:53:04 이고, (실제로 vi로 인한 변경은 그 이전에 일어났음) 그 다음 modify 됨을 인지하고 백업된 시각이 05:53:34 임을 로그 파일에서 확인

텍스트, 소프트웨어, 스크린샷, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

- 예외 처리와 관련한 예제

- 경로를 입력하지 않을 경우, add 명령어에 대한 에러 처리 및 Usage 출력 후 프롬프트가 재 출력

- 인자로 입력 받은 경로가 사용자 홈 디렉터리 내의 경로가 아닌 경우 에러 처리 후 프롬프트 재 출력

- 인자로 디렉터리 경로를 받았는데 -d나 -r 옵션을 사용하지 않은 경우 에러 처리 후 프롬프트 재 출력

- 인자로 파일 경로를 받았는데 -d나 -r 옵션이 사용된 경우 에러 처리 후 프롬프트 재 출력

- 인자로 입력 받은 경로에 대해 데몬 프로세스가 이미 존재한다면 에러 처리 후 프롬프트 재 출력

- -t 옵션을 사용할 때 인자로 <PERIOD>를 입력받지 못했다면 Usage 출력 후 프롬프트 재 출력

**(3) 내장 명령어 2. remove**

텍스트, 컴퓨터, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명

- PID를 입력 받아 해당 데몬 프로세스를 제거하고 모니터링을 중지, 백업 파일을 삭제하는 remove

- 현재 존재하는 PID를 확인하기 위해 monitor\_list 로그 파일의 내용을 출력하였고, 그 중 하나인 3030 삭제

- 데몬 프로세스가 삭제되었음을 출력하였고, monitor\_list 로그 파일을 다시 출력해보면 삭제한 PID에 해당하는 줄은 삭제

- 또한 PID를 이름으로 가지고 있는 로그 파일과 백업 디렉터리도 삭제

- 존재하지 않는 PID 번호를 인자로 넘겨 준 경우, 에러 처리 후 프롬프트를 재 출력

**(4) 내장 명령어 3. list**

**텍스트, 소프트웨어, 스크린샷, 컴퓨터이(가) 표시된 사진

자동 생성된 설명**

**-** list 명령어를 사용할 때 인자를 입력하지 않으면 현재 실행 중인 데몬 프로세스들의 목록을 출력

- list 명령어를 사용할 때 인자로 데몬 프로세스의 PID를 넘겨주면 해당 데몬 프로세스 로그를 기준으로 변경 상태와 시간을 트리 형태로 출력

- **해당 데몬 프로세스의 PID가 관리하고 있는 경로가 일반 파일인 경우**, 변경 상태와 수행 시간이 출력

- **해당 데몬 프로세스의 PID가 관리하고 있는 경로가 디렉터리인 경우**, 구현 X

- 인자로 넘겨준 PID가 존재하지 않는 데몬 프로세스 PID인 경우 에러 처리 후 프롬프트 재 출력

**(5) 내장 명령어 4. help**

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

- help 명령어 실행 시 내장 명령어에 대한 Usage 출력

- 인자 생략 시, 모든 명령어에 대한 Usage 출력

- 인자 지정 시, 해당 명령어에 대한 Usage 출력

- 잘못된 내장 명령어 입력 시 에러 처리 후 프롬프트 재출력

**(6) 내장 명령어 5. exit**

**텍스트, 스크린샷, 소프트웨어, 멀티미디어이(가) 표시된 사진

자동 생성된 설명**

- exit 명령어 입력 시 실행 중인 ssu\_sync 프로그램 종료