데이터 구조

Binary-Search를 이용한

UNIX 사용자 정보 검색 프로그래밍

컴퓨터 공학

2015211373 권정인

목차

1. 개요 및 작업 환경
2. 전체 코드 및 코드의 요약
3. 컴파일 및 결과
4. 평가 및 개선 방안
5. 개요 및 작업 환경

저는 Windows 10 내에 있는 Ubuntu 16.04 기반 Linux 서브 시스템 내에서, 코드 작성은 VI(VIM)으로, 코드 빌드(컴파일)은 GCC(CLI도구)로 실행 하였습니다.

코드 작성 시작 전, 우선 eClass에서 sorted\_passwd 파일을 Windows의 Internet Explorer 앱으로 제 PC에 다운로드 한 후, Windows에 내장 된 파일 탐색기 앱으로 C:\ 디렉토리로 파일을 옮겼습니다. 그 후, Bash 환경에서 C:\에 있는 sorted\_passwd 파일을 Linux 파일시스템의 제 홈 디렉토리로 복사하였습니다. 그에 대한 명령어는 다음과 같습니다 : *cp /mnt/c/sorted\_passwd ~/sorted\_passwd*

참고로, Linux 서브 시스템 내에서, 실제 C 드라이브의 파티션은 /mnt/c/에 마운트 되며,

Linux 서브 시스템의 홈 디렉토리는 Windows 파일 탐색기에서 *C:\Users\[Windows사용자이름]\AppData\Local\lxss\home\[UNIX 계정명]* 으로 열립니다.

번외로, Linux 서브 시스템의 /home을 제외한 /(루트) 디렉토리는 Windows 파일 탐색기에서 *C:\Users\[Windows사용자 이름]\AppData\Local\lxss\rootfs* 에 있습니다

만일, 파일 작성 코드 작성 및 실행을 제 개인 PC의 Linux 서브 시스템이 아닌 학과 유닉스 서버에서 작성하고 실행한다면, ranwords 파일을 홈 디렉토리로 복사한 후, scp 명령어를 이용해, 학과 서버로 복사해야 합니다. 그에 대한 명령어는 아래와 같습니다 : *cp -P 11800 sorted\_passwd [학과서버 ID]@dalma.dongguk.ac.kr:~/sorted\_passwd*

코드 파일 경로 : *~/hw4.c*

1. 전체 코드 및 코드의 요약

#include "stdio.h"

#include "string.h"

int fileinput(FILE \*fp, char buffer[])

{

int split = 0; //split값 0으로 초기화

printf("Reading User information");

while(fgets(buffer, 100, fp))

{

if(buffer != " ")

{

if(++split % 50 ==0) printf("."); //split을 1씩 증가시키고, 50씩마다(split에 50을 나누어서 0이 될때) ‘.’을 표시합니다

}

}

printf("%d User table estalished.\n", split);

return split; //파일에서 한 줄을 읽을떄마다 split이 증가하므로, split은 총 사용자 수가 됩니다.

}

int compare(char id[], char buffer[])

{

int i, result;

for(i = 0; buffer[i] != '\0'; i++)

{

if(id[i] != buffer[i])

{

result = 0; //id가 buffer와 다르면 result는 0이 됩니다.

}

else

{

if(strstr(buffer, id))

{

int id\_number = split(id);

int buffer\_number = split(buffer);

//id\_number와 buffer\_number은 각각 split(id),(buffer)를 같이 합니다.

if(id\_number = buffer\_number)

break;

}

result = 1;

}

return result;

}

}

int split(char c[])

{

int i, counter = 0;

for(i=0; c[i] != '\0' && c[i] != '\n' && c[i] != ':'; i++) counter++;

return counter;

}

void search(FILE \*fp, char buffer[], int all\_count\_number, char id[]) {

int compare\_counter = 0;

int search\_result = 0;

//compare\_counter와 search\_result을 각각 0으로 초기화

printf("User : %s", id);

while(fgets(buffer, 100, fp))

{

if(buffer != " ")

{

++compare\_counter;

search\_result = compare(id, buffer);

if(search\_result)

{

int i;

printf("\t INFO : ");

for(i = 0; i<100; i++)

if(buffer[i] != '\n')

printf("%c", buffer[i]);

//사용자의 자세한 정보를 1글자씩 100번, 총 100글자를 출력합니다.

else break;

break;

}

else if (compare\_counter == all\_count\_number) {

printf("Failed to find");

break;

}

}

}

printf(" Count : %d\n", compare\_counter);

}

int main()

{

char buffer[128], id[128];

int count = 0, total\_id;

FILE \*fp = fopen("sorted\_passwd", "rt");

system("clear"); //가독성 향상을 위해 유닉스 명령어인 clear에 접근합니다.

printf("Form of User table\n");

if(fp==NULL)

{

printf("File sorted\_passwd not found. Halting..\n");

return 1;

} //파일이 없다면 오류 메시지와 함께 프로그램을 종료합니다

total\_id = fileinput(fp, buffer);

printf("Find user information with Binary search\n");

printf("========================================================\n");

while(1) {

rewind(fp);

printf("Enter User ID : ");

gets(id); //사용자로부터 id를 입력받습니다.

if(id[0] == '\0')

{

printf("Halting..\n");

break; //사용자로부터 아무것도 입력받지 않았을경우(아무것도 입력받지 않고 Enter를 눌렀을때 while을 벗어납니다.

} else search(fp, buffer, total\_id, id); //사용자로부터 값을 입력받았을경우 search함수를 실행합니다.

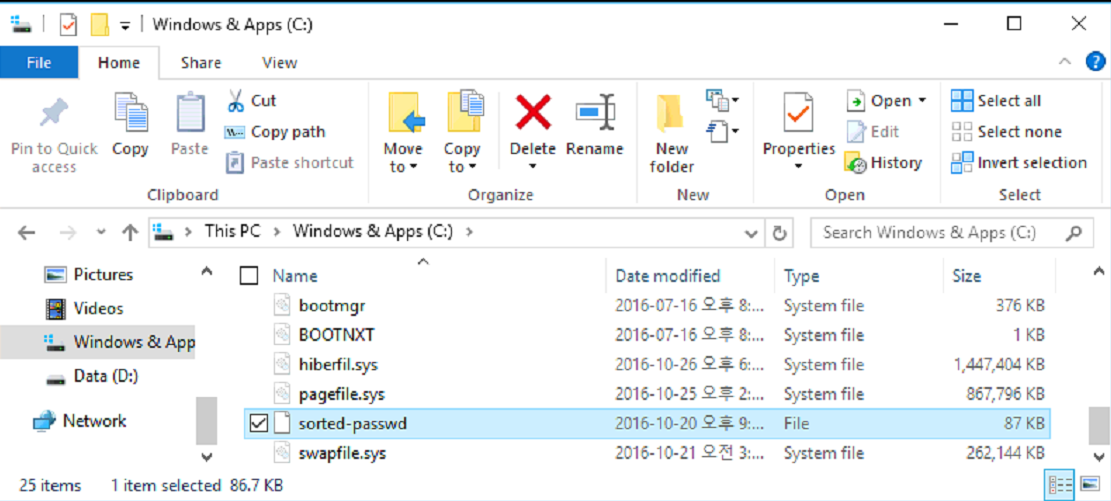
}

fclose(fp); //fp(파일)을 닫습니다.

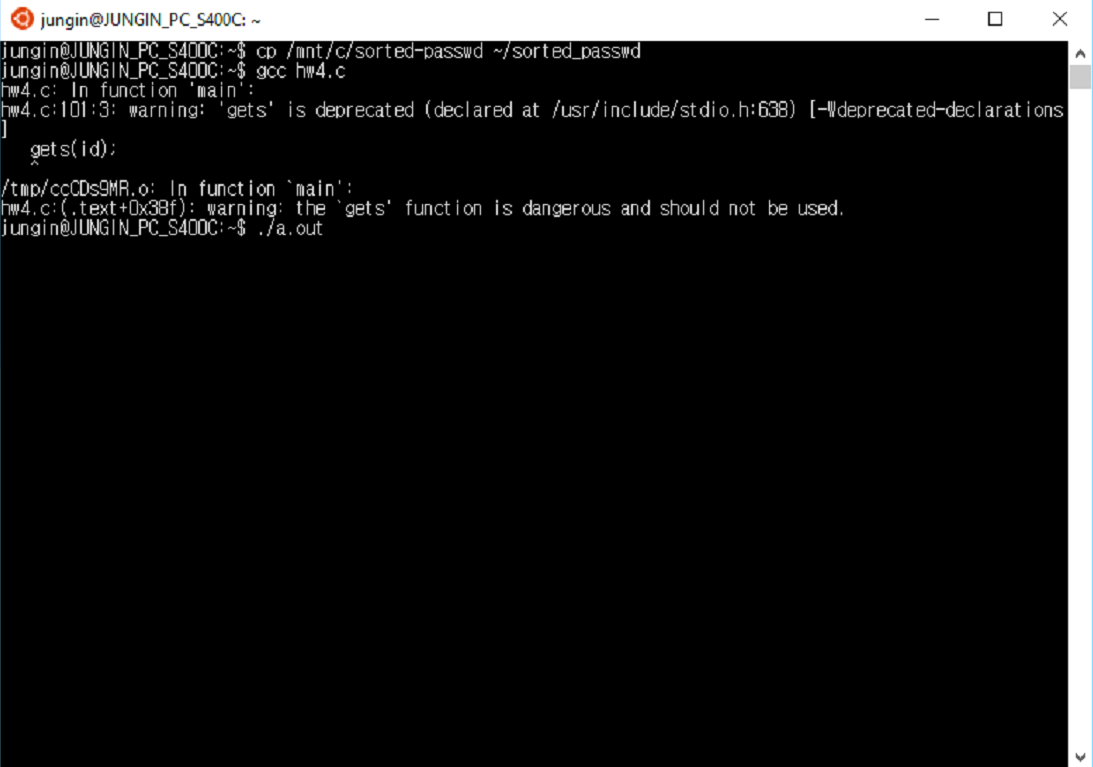
return 0; //0을 리턴하여 프로그램 종료

}

1. 컴파일 및 결과



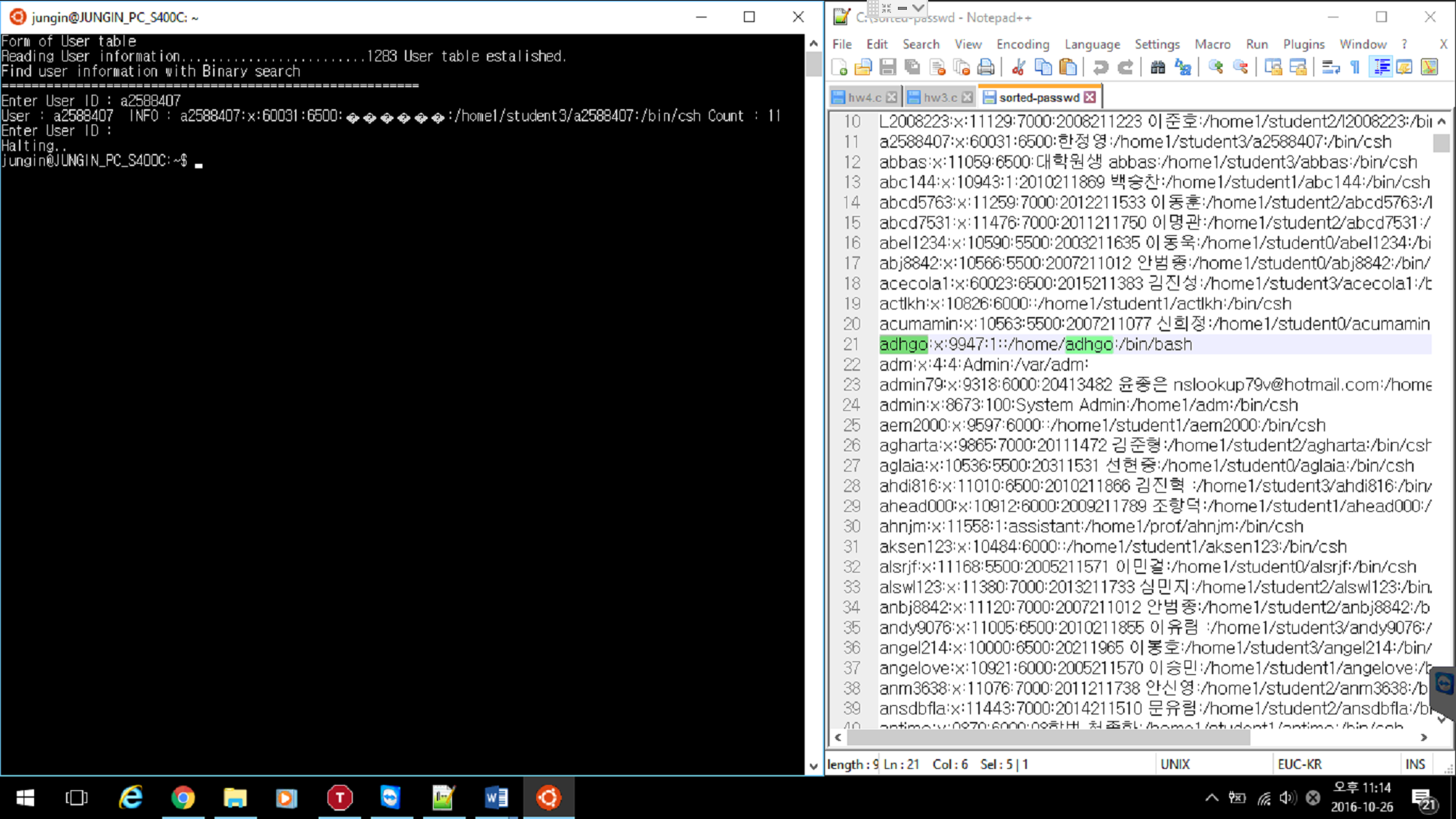
Windows 내의 파일 탐색기를 이용해 접근성이 용이한 C:\로 sorted\_passwd파일을 이동합니다.



Windows 10 내의 Linux 서브시스템에 접근하여 스크린샷 속 명령을 실행합니다.

위에서부터, C:\에 있는 sorted-passwd 파일을 홈 디렉토리로 복사, GCC를 이용하여 코드 빌드(컴파일), 컴파일하여 생성된 a.out 실행

코드를 컴파일 하는 도중 경고 문구가 발생하였지만, 코드에서 사용한 gets의 약한 보안성에 대한 경고일 뿐, 빌드하고 실행하는데는 크게 지장이 없으므로 무시하였습니다.



프로그램 실행 화면입니다. 우선, System(“clear”)를 사용하여, 화면이 초기화된 후, 실행되는 것을 확인할 수 있습니다. 또한 오른쪽의 텍스트 편집기(notepad++) 앱에서 확인되는 바와 같이, 총 사용자 수가 출력됨을 확인할 수 있습니다. 또한 ID를 입력하였을때(오른쪽 텍스트편집기 화면에서 초록색으로 음영되어있는 ID) 정보가 정상적으로 출력됨을 확인할 수 있었습니다. (중간에 한글 출력부분은 정상적으로 출력되지 않습니다. Windows 10 내의 Bash에 대한 고유적인 문제이므로, 코드상의 문제는 아닙니다.) 또한, 아무것도 입력하지 않았을때, 프로그램이 정상적으로 종료됨을 확인하였습니다.

1. 평가 및 개선 방안
2. 코드를 작성할 때, 매우 복잡한 코드 알고리즘을 구현하였습니다. 이번 코드 작성 과정에서, 약 20번이 넘는 에러 발생 빌드를 겪었습니다.
3. 처음 코드 빌드에 성공하고, 실행을 하였을떄, 제가 원하는대로 결과가 나오지 않았습니다. 인터넷에서 검색을 하였지만, 제가 원하는 정보는 나오지않아 공책에 연필로 코드 작성을 여러번 한 결과, 올바른 알고리즘 구현에 성공하였습니다.
4. 이번 과제는 2, 3 번 과제를 적당히 섞으면 성공할 수 있을것 같았지만, 그렇지 않았습니다. 이번 과제는 제 예상보다 훨씬 난인도가 높았습니다.
5. 이제, VI(VIM)에서 C 언어 코드를 작성할 때, 자동으로 들여쓰기가 활성화됩니다. 코드 작성 시 가독성이 향상됨을 확인할 수 있었습니다.
6. 아직도 제가 사용하는 시스템에서는 Bash에서 한글 출력이 제대로 되지 않아, 영문으로 작성되었습니다. 학과 서버에 접속하여 작업을 하려하면, 한글 입력이 제대로 되지 않음을 알 수 있었습니다.
7. 저번 과제와 같이, main 함수에 전체 코드를 작성하기보단, 새로운 함수를 여러개 만들어서 main 함수에서 실행되게 하였습니다. 그 과정에서 함수와 main에서 상호작용중에 변수가 꼬여보이는 상황이 발생하였습니다. 다음에는, 약간 비효율적이라도, 코드를 최대한 main함수에서 작성해보고 싶습니다.