

실험 UNIX-2 결과보고서

전공: 컴퓨터공학과

학년: 2

학번: 20191559

이름: 강상원

1. 목 적

실습 과정에 개발한 fmt에 대하여 결과 보고한다.

2. 문제 풀이 결과

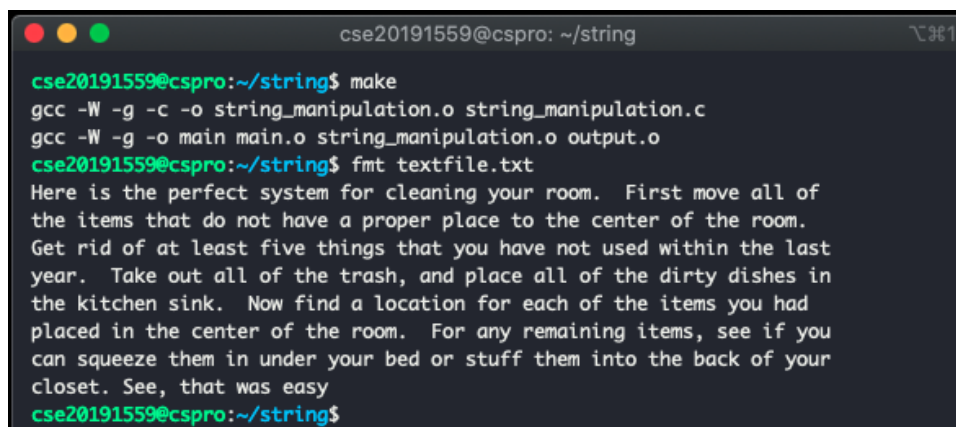
2-1. 알고리즘

fmt를 구현하기 위하여 사용한 알고리즘을 정리하여 기술하시오.

2-2. 테스트

조교가 제시한 테스트 데이터에 대한 출력 결과를 제출하시오. 제출방법은 조교의 지시에 따르시오.

1. 실습 결과화면을 첨부한다.



```
cse20191559@csp: ~/string
cse20191559@csp:~/string$ make
gcc -W -g -c -o string_manipulation.o string_manipulation.c
gcc -W -g -o main main.o string_manipulation.o output.o
cse20191559@csp:~/string$ fmt textfile.txt
Here is the perfect system for cleaning your room. First move all of
the items that do not have a proper place to the center of the room.
Get rid of at least five things that you have not used within the last
year. Take out all of the trash, and place all of the dirty dishes in
the kitchen sink. Now find a location for each of the items you had
placed in the center of the room. For any remaining items, see if you
can squeeze them in under your bed or stuff them into the back of your
closet. See, that was easy
cse20191559@csp:~/string$
```

2. fmt를 구현하기 위해 사용한 함수들과 그 함수들의 목적을 간단히 설명한다.

int main(int argc, char *argv[]) : main 함수, 전체 control 제어

파일 읽기 시도, 실패 시 File open error. 출력.

파일 이름이 없을 경우: Usage: fmt filename | > outfile 출력.

line1, line2의 메모리 할당, 실패 시 에러 문구 출력

while(1) 루프를 통해 함수들 반복 실행.

- B_Line==0 : Print_Line(line1, &Count, &B_Flag),

- else : B_Line = 0.

- count !=0 : B_Flag = 1

- Remove_Blanks_At_The_End(line2) 실행

- line2[0]==' ' 이거나 count != 0인 경우 : 개행 문자 출력, B_flag=Count=0

- else if line2[0]=='\n'

- B_Flag==1 : 개행 문자 출력, B_flag=0, 개행 문자 출력,

- B_line=1, Count=0

- line1, line2 내용 주고받기 (서로 바꾸기)
- line1의 첫 문자가 '\n'이 아닐 때 :
 - i=0, line1[i]가 '\n' 또는 NULL일때까지 i를 늘려가며 반복
 - line[i]=='\n' : '\n' 출력

종료.

void Remove_Blanks_At_The_End(char *line) :

문자열 뒤의 연속된 공백 문자 제거

void Get_Blanks_Chars(char *line, int Start, int *N_Blanks, int *N_Chars) :

문자열의 문자, 공백 수 세는 함수

void Print_Line(char *line, int *Count, int *B_Flag) :

문자열 출력 담당. (미리 설정된 최대 길이에 맞춤)

3. 실습시간에 작성한 Makefile의 한줄 한줄의 의미를 설명한다.

cc=gcc

컴파일러 : gcc 사용

cflags = -W -g

컴파일 옵션

target = main

빌드 대상(실행 파일) 이름

objects = main.o string_manipulation.o output.o

중간 산물 오브젝트 파일 목록 나열

\$(target) : \$(objects)

\$(cc) \$(cflags) -o \$(target) \$(objects)

규칙 적용으로 오브젝트 파일 순차 생성

%.o : %.c

\$(cc) \$(cflags) -c -o \$@ \$<

컴파일

main.o string_manipulation.o output.o : Header.h

헤더 파일 종속성

.PHONY : clean

clean을 명령어로 사용

clean :

rm \$(target) \$(objects)

clean을 입력하면 빌드 결과물, 중간 부산물 (오브젝트 파일) 삭제

4. 규칙 R5를 어떤 알고리즘으로 구현하였는지 상세히 설명한다.

규칙 R5 : 입력줄의 첫 글자가 blank이면 앞줄과 합쳐지지 않게 한다. 만일, 줄의 첫 부분에 여러 개의 blank가 있으면 이 역시 줄을 바꾸어 새 줄에 출력하고 첫 부분의 blank는 첫 번째 blank를 포함해 그 개수만큼 그대로 출력한다.

2번 질문의 main 함수 설명에 상세히 기록되어 있다.

- line2[0]==' '이거나 count != 0인 경우 : 개행 문자 출력, B_flag=Count=0
 - else if line2[0]=='\n'
 - B_Flag==1 : 개행 문자 출력, B_flag=0, 개행 문자 출력,
 - B_line=1, Count=0
- line1, line2 내용 주고받기 (서로 바꾸기)
- line1의 첫 문자가 '\n'이 아닐 때 :
 - i=0, line1[i]가 '\n' 또는 NULL일때까지 i를 늘려가며 반복
 - line[i]=='\n' : '\n' 출력

5. make의 옵션들에 대하여 정리한다.

-C dir	Makefile을 계속 읽지 않고 dir로 이동
-d	Makefile을 수행하며 각종 정보를 출력
-h	옵션에 관한 도움말을 출력
-f file	file에 해당하는 파일을 Makefile로써 취급
-r	내장된 규칙을 없는 것으로 간주
-t	파일의 생성 날짜를 현재 시간으로 갱신
-v	make의 버전 출력
-p	make의 내부적으로 세팅되어 있는 값들을 출력
-k	에러가 나더라도 멈추지 말고 계속 진행

참고 서적

1. Beginning Linux Programming, Richard Stones & Neil Matthew, WROX
2. Learning the bash shell, Cameron Newham & Bill Rosenblatt, O'Reilly
3. Programming with GNU Software, Mike Loukides and Andy Oram, O'Reilly
4. [Debugging with GDB: The GNU Source-Level Debugger](#), Richard Stallman, FSF
5. RedHat Linux 9 Bible, Christopher Negus, John Wiley
6. UNIX in a nutshell A Desktop Quick Reference for SVR4 and Solaris 7 (3rd Edition), Arnold Robbins, O'Reilly
7. UNIX Power Tools Third Edition, [Shelley Powers](#) & [Jerry Peek](#) & [Tim O'Reilly](#) & [Mike Loukides](#), O'Reilly