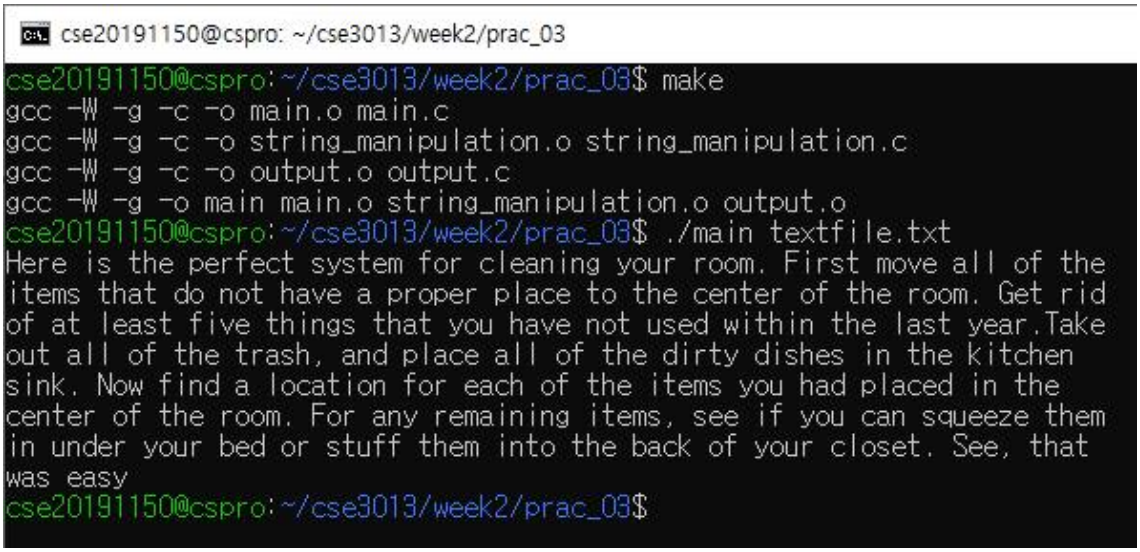
2주차 결과보고서

전공: 신문방송학과 학년: 3학년 학번: 20191150 이름: 전현길

1. 실습 결과화면



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

int main(int argc, char *argv[])

1. 파일 입력: 입력 실패 시 오류 처리
2. 인자가 2개가 아닐 시: 오류 처리, 사용법 출력
3. line1, line2에 메모리를 할당: 실패 시 오류 처리
4. line1의 첫 한 줄 입력: 빈 파일이면 main 함수 종료
5. Remove_Blanks_At_The_END(line1): 파일 전처리, 문장 끝의 공백 삭제

while문 (loop)

실질적 동작: line1과 line2를 교환해 가면서 한 줄씩(또는 256byte 만큼씩) 입력받고, Print_Line() 함수로 출력함. 만약 이번 줄이 개행 문자만 있는 빈 줄이면 개행 문자만 출력하고 다시 입력받고, 한 줄의 시작이 빈 줄로 시작하지 않도록 함.

1. B_Line == 0이면: Print_Line(line1, &count, &B_Flag), 아닐 시 B_Line = 0

2. Count != 0이면: B_Flag = 1

3. line2 입력: line1의 다음 줄을 입력, **비어 있다면 while문 종료**

4. Remove_Blanks_At_The_End(line2): line2 전처리, 문장 끝 공백 삭제

5. line2가 공백으로 시작 && Count != 0: 문자열 개행, B_Flag = 0, Count = 0

   → **line2가 공백으로 시작할 경우 개행 처리**

6. line2가 개행으로 시작: 문자열 개행, B_Line = 1, Count = 0

   6-1. B_Flag가 1이면: 한 번 더 개행, B_Flag = 0 초기화

   → **받은 line2가 개행 문자만 있는 빈 줄일 때, 개행하고 다시 입력받음**

   → B_Line = 1이므로 다음 반복 때 **while문의 1에서 출력이 이뤄지지 않음**

7. line1과 line2의 포인터 교환: line2를 다음 반복문 때 출력

마지막 줄 처리

마지막 줄이 빈 줄이 아니라면, 마지막 문자가 개행 문자일 경우 출력

void Remove_Blanks_At_The_End(char *line)

문자열의 끝에서 공백을 제거하는 함수

개행 문자, 널 문자가 있는 위치에서 역순으로 이동해서 처음 문자가 등장하는 위치의 바로 뒤에 '\n\0' 또는 '\0'을 입력

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

공백 문자와 문자의 개수를 세는 함수

시작 위치에서 문자열이 개행/널 문자로 끝날 때까지 아래와 같은 작업을 수행

1. 공백 문자일 경우: **blank_flag = 1이면 종료**, 아닐 시 공백 문자의 개수를 셈
2. 문자일 경우(≠ 공백/개행/널 문자): 글자 수를 셈, blank_flag = 1 초기화

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

문자열을 길이(LIMIT)에 맞춰 출력하는 함수

B_Flag: 이번 문자열을 이전 줄에서 이어 출력할 수 있는지 알려주는 변수
Count: 현재까지 출력된 문자의 수
Start: 다음 문자열의 출력이 시작되는 위치

1. Get_Blanks_Chars(): 공백 문자 및 문자의 개수를 셈
2. B_Flag == 1:
   2-1. 한 칸을 띄고 현재 문자열을 이전 줄에 이어서 출력
   2-2. Start(다음 문자열의 시작 위치), Count 변수 갱신, 다음 문자열의 수를 셈
   2-0. N_Blanks가 0이 아닐 때 에러 처리
3. B_Flag != 1:
   3-1. (*Count + N_Blanks + N_Chars) <= LIMIT // 문자 수가 제한을 만족하면
      3-1-1. 문자열 출력, Start, Count 갱신, 다음 문자열의 수를 셈
   3-2. else // 문자열이 제한보다 크다면
      3-2-1. Count == 0일 시 출력 후 개행, 다음 문자열의 수를 셈
      3-2-2. Count != 0일 시 공백을 무시하고 개행 후 출력, 다음 문자열의 수를 셈

3. 실습시간에 작성한 Makefile 한 줄 한 줄의 의미를 설명한다.

```
main: main.o cow.o turtle.o dog.o
        gcc -o main main.o cow.o turtle.o dog.o
# 타겟 파일(결과로 생성되는 파일) : 의존성 목록(타겟 파일이 의존하는 파일)
# main.o, cow.o, turtle.o, dog.o 파일을 이용해 main 실행 파일을 생성

cow.o : cow.c
      gcc -c -o cow.o cow.c

turtle.o : turtle.c
      gcc -c -o turtle.o turtle.c
```

```
dog.o : dog.c
	gcc -c -o dog.o dog.c

main.o : main.c
	gcc -c -o main.o main.c
# 위 경우 직접 작성했지만 아래처럼 작성할 수도 있음
# %.o : %.c
#	gcc -c -o $@ $<

.PHONY : clean	# PHONY target을 설정, 실행 파일명이 아닌 명령임을 밝힘
clean :
	rm main *.o
# make clean을 입력하면 main과 *.o(확장자가 .o로 끝나는 모든 파일) 삭제
# 따라서 위 makefile의 경우 디렉토리를 분리시켜야 함
# 정석적으로는 objects = main.o, cow.o, turtle.o, dog.o 등으로 매크로를 사용한 뒤
# rm main $(objects) 형태로 작성하는 게 좋음
```

4. 규칙 R5를 어떤 알고리즘으로 구현하였는지 상세히 설명한다.

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.

R5 규칙은 위와 같다. 먼저 입력줄의 첫 글자가 공백일 경우 앞 줄과 합쳐지지 않는 규칙은 main 함수에서 처리된다. 줄의 첫 부분에 여러 개의 공백이 있든, 한 개의 공백이 있든 간에 main() 함수의 아래와 같은 조건 때문에 개행이 이뤄진다.

```
if (line2[0] == ' ' && Count != 0) {
  putchar('\n');
  B_Flag = 0;
  Count = 0;
}
```

줄의 첫 부분의 blank가 개수만큼 출력되는 것은 Print_Line의 다음 코드 등과 관련된다. 아래 코드는 공백 문자의 개수와 무관하게 문자열을 출력한다. 반면에, 이번 줄에 이미 출력된 문자열이 존재하는 경우(*Count != 0인 경우) 반복문이 Start가 아

닌 Start + N_Blanks에서 시작하기 때문에 공백 문자의 개수는 무시되고 한 칸만 띄워 출력한다.

```
else if ((*Count + N_Blanks + N_Chars) <= LIMIT) {
  for (i = Start; i < Start + N_Blanks + N_Chars; i++) putchar(line[i]);

  Start = Start + N_Blanks + N_Chars;
  *Count = *Count + N_Blanks + N_Chars;
  Get_Blanks_Chars(line, Start, &N_Blanks, &N_Chars);
}

else {
  if (*Count == 0) {
    for (i = Start; i < Start + N_Blanks + N_Chars; i++) putchar(line[i]);

    Start = Start + N_Blanks + N_Chars;
    putchar('\n');
    Get_Blanks_Chars(line, Start, &N_Blanks, &N_Chars);
    Start = Start + N_Blanks;
  }
```

5. make의 옵션에 대하여 정리한다.

플래그	기능
-C <디렉토리>	makefile을 읽기를 중지, directory로 이동
-debug, -d	makefile 실행하는 처리 과정을 모두 출력
-help, -h	옵션에 관한 도움말을 출력
-f <파일명>	<파일명>에 해당되는 파일 makefile로 읽음
-r	내장된
-b	호환성 무시
-B	무조건적으로 모든 타겟 파일을 make
-e	환경 변수를 makefile에 덮어씌움
-p	make의 내부 데이터베이스를 출력
-r	모든 내장 암묵적 규칙(implicit rule)을 비활성화
-R	모든 내장 변수 세팅을 비활성화