**실험 UNIX-2 결과보고서**

전공: 컴퓨터공학과 학년: 4 학번: 20212020 이름: 박민준

1. **목 적**

실습 과정에 개발한 fmt에 대하여 결과 보고한다.

1. **문제 풀이 결과**

**2-1. 알고리즘**

fmt를 구현하기 위하여 사용한 알고리즘을 정리하여 기술하시오.

fmt 프로그램은 두 개의 입력 버퍼 L1, L2를 사용해서 파일로부터 텍스트를 한 줄씩 읽는다. 따라서 Count, B\_Flag 등 각 줄의 처리를 위한 변수들을 먼저 초기화한다. 이후 파일로부터 한 줄을 읽어 L1 버퍼에 저장한다. 이 때, 줄의 끝에 있는 불필요한 공백을 제거하는 전처리 작업을 수행하게 된다. 또한, fmt 프로그램은 L1의 내용을 기반으로 현재 줄을 출력할지, 다음 줄인 L2와 합칠지, 또는 새로운 줄로 바꿀지 등을 결정한다. 이 과정에서 R1~R9의 규칙들을 적용하여 판단하게 된다. 결정한 조건에 따라 현재 줄을 출력하고, 필요한 경우 다음 줄을 미리 읽어 처리한다. 파일의 끝에 도달했을 때는 남은 내용을 적절히 처리하고 프로그램을 종료하는데, 마지막 줄이 빈 줄인 경우 등의 특수한 상황도 함께 고려한다. 마지막으로 fmt 프로그램은 주요 기능을 담당하는 ‘Remove\_Blanks\_At\_The\_End’, ‘Get\_Blanks\_Chars’ 등의 함수들을 구현하고, 이들을 조합해 전체 프로그램을 완성하게 된다.

**2-2. 테스트**

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

1. 실습 결과화면을 첨부한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

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

1. ‘Remove\_Blanks\_At\_The\_End’ 함수 : 이 함수는 문자열의 끝에서 불필요한 공백을 제거하는 역할을 한다. 파일에서 한 줄을 읽은 뒤에 줄 끝의 공백이 출력에 영향을 주는 것을 막기 위해 사용되는 함수이다. 만약 문자열의 끝에 줄바꿈 문자(‘\n’)가 있으면 그 위치를 공백 문자(‘ ’)로 대체하고 그 다음 위치에 문자열 종료를 나타내는 문자(‘\0’)를 삽입한다. 이는 출력 시 불필요한 공백을 줄이고 출력 포맷을 형식에 맞게 유지하기 위해 필요한 처리이다.

2. Get\_Blanks\_Chars 함수 : 이 함수는 주어진 시작 위치에서부터 문자열을 분석하여 다음 단어까지의 공백 개수와 단어의 문자 개수를 계산해주는 역할을 한다. Start 위치에서부터 문자열을 순회하면서 공백 문자와 단어 문자를 센 후, 각각의 개수를 참조로 전달된 변수에 저장한다. 즉, 이 함수는 줄바꿈과 공백 삽입의 결정 과정에서 필요한 정보를 제공하는 역할을 한다.

3. Print\_Line 함수 : 이 함수는 fmt 프로그램에서 주어진 텍스트 줄을 처리하고 출력하는 역할을 한다. 이 함수는 텍스트 줄, 현재까지의 글자 수, 플래그를 매개변수를 받는다. 여기서 플래그(B\_Flag) 변수는 이전 줄과 현재 줄 사이에 공백을 삽입할 지의 여부를 결정하는 변수이다. 따라서 이 함수는 주어진 텍스트를 규정된 ‘LIMIT’에 맞춰 적절히 출력하는 역할을 담당한다.

1. 실습시간에 작성한 Makefile의 한줄 한줄의 의미를 설명한다.

cc=gcc -> ‘gcc’ 컴파일러를 사용하기 위해 변수 ‘cc’에 ‘gcc’를 할당한다.

cflags = -W -g

-> 컴파일 옵션으로 ‘-W’와 ‘-g’를 설정하기 위해 ‘cflags’ 변수를 정의한다. 여기서 ‘-W’ 옵션은 모든 경고 메시지를 출력하는 옵션이고, ‘-g’ 옵션은 디버깅 정보를 포함하는 옵션이다.

target = main

-> 최종적으로 생성될 실행 파일의 이름을 ‘main’으로 설정한다.

objects = main.o String\_Manipulation.o Output.o

-> 컴파일 과정에서 생성될 오브젝트 파일들(\*.o)의 목록을 정의한다.

$(target) : $(objects)

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

-> 최종 타겟 파일인 ‘main’ 파일을 만들기 위해서는 ‘objects’ 파일들이 필요하며, ‘gcc’ 컴파일러를 사용해 ‘main’ 파일을 생성한다.

%.o : %.c -> 모든 ‘\*.o’ 파일들은 각각의 ‘\*.c’ 파일들로부터 컴파일된다.

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

-> 각각의 ‘\*.o’ 파일들을 생성한다. ‘$@’는 현재 타겟의 이름을 의미하고, ‘$<’는 현재 타겟보다 더 최근에 갱신된 파일의 이름을 의미한다.

main.o String\_Manipulation.o Output.o : Header.h

-> 위 세 개의 오브젝트 파일들(\*.o)이 ‘Header.h’ 파일에 의존한다.

.PHONY : clean

-> ‘clean’ 규칙을 phony로 선언한다. 이는 해당 규칙이 실제 파일 이름이 아닌 단순 명령어로 동작한다는 것을 나타낸다.

clean :

rm $(target) $(objects)

-> ‘clean’이라는 규칙을 정의하고, 빌드된 파일(최종적으로 생성된 실행 파일)과 오브젝트 파일들(\*.o)을 삭제한다.

1. 규칙 R5를 어떤 알고리즘으로 구현하였는지 상세히 설명한다.

규칙 R5는 입력된 텍스트의 줄이 공백 문자로 시작하는 경우, 해당 줄을 이전 줄과 합치지 않고 새로운 줄로 시작하도록 하는 규칙이다. ‘Print\_Line’ 함수 내에서 R5 규칙을 구현하기 위해 사용된 알고리즘은 다음과 같다.

먼저, 현재 처리 중인 텍스트 줄에서 첫 번째 문자가 공백인지를 확인한다. 만약 첫 번째 문자가 공백인 경우(N\_Blanks>0), 이전 줄과 합치지 않고 새 줄을 시작해야 한다. 이를 위해 ‘Print\_Line’ 함수는 첫 번째 공백이 나타나기 전까지 현재 줄의 내용을 출력하고, 줄바꿈을 수행해 새로운 줄로 처리를 시작한다. 또한, 첫 번째 문자가 공백인 경우(N\_Blanks>0) ‘B\_Flag’ 변수의 값을 0으로 설정해 이전 줄과 합치는 것을 방지하고, 새로운 줄을 시작하게 한다.

마지막으로 해당 함수는 공백으로 시작하는 줄과 그렇지 않은 줄을 처리하기 위해 분기 조건을 사용한다. 만약 공백으로 시작하는 줄이라면 공백 다음의 첫 단어부터 출력한다. 그렇지 않은 경우, ‘B\_Flag’의 현재 값에 따라 이전 줄에 이어서 출력할지 또는 새 줄에서 출력을 시작할지를 결정하게 된다.

1. make의 옵션들에 대하여 정리한다.

1. -f filename : Makefile 이름을 filename으로 사용한다. 해당 옵션을 사용하면 특정 파일이 아닌 다른 Makefile을 사용할 수 있다.

2. -j : 병렬 작업을 실행한다. 여러 개의 작업을 동시에 수행하여 빌드 시간을 단축할 수 있다.

3. -k : 오류가 발생해도 계속해서 다른 타겟을 빌드한다. 일반적으로 Make는 오류가 발생하면 빌드를 중단하지만 해당 옵션을 사용하면 오류가 발생한 타겟을 제외하고 계속 빌드한다.

4. -n : 실제로 빌드하지 않고 빌드 되는 명령을 출력한다. 해당 옵션을 사용하면 어떤 작업이 수행되는지를 미리 확인할 수 있다.

5. -B : 항상 타겟을 빌드한다. 해당 옵션을 사용하면 의존성 여부와 관계없이 항상 빌드한다.

6. -C directory : 지정된 디렉토리로 이동하여 빌드를 수행한다. 해당 옵션은 특정 디렉토리에서 Make를 실행할 때 유용하다.

7. -s : Make의 출력을 최소화한다. 따라서 경고 메시지나 에러 메시지만 출력되고 보통의 메시지나 룰은 출력되지 않는다.

8. –no-print-directory : 디렉토리 변경 메시지를 출력하지 않고 숨긴다.

이 외에도 다양한 make 옵션들이 있지만, 생략하겠다.