

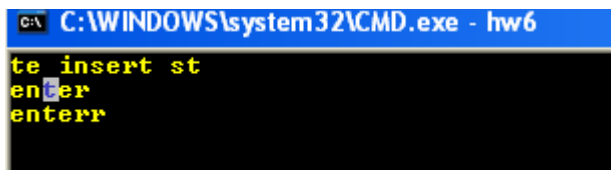
System Design HW6

~ Simple Editor~

2013210111 남세현

VMWare (Virtual Machine) 위 Windows XP 32bit 환경에서 masm5.1 사용

1. 실행화면



2. 목표

간단한 에디터를 만들어야 합니다. 총 256 바이트 입력이 가능해야하고, 수정-삽입 모드 전환과 화살표 키로 커서 이동, 엔터키로 개행 등 기본적인 기능들이 포함되어야 합니다.

3. 구현 방법

저는 ROW, COL 방식이 아니라, 1차원적으로 BUFFER 위의 OFFSET 혹은 INDEX를 의미하는 CURSUR를 만들었습니다. 나중에 이 1차원커서를 2차원 (ROW, COL)로 변환시켜 필요한 계산을 하도록 하였습니다.

$$CURSUR = \text{BUFFER 위의 1차원적인 인덱스} \rightarrow \text{BUFFER}[CURSUR]$$

그러므로, 아래 구현 방법에 나오는 "CURSUR"는 ROW, COL의 커서가 아니라 BUFFER 위의 OFFSET으로 이해하시면 좋습니다. 1차원 커서인지 2차원 커서인지 헷갈릴 수 있는 부분에는 제가 정확하게 "1차원 커서" 등으로 표현을 하도록 하겠습니다. 딱히 추가 표현이 없으면 대부분 1차원 커서를 말하는 경우입니다.

A. HW5에서 구현한 코드를 기반으로 키보드 입력을 받습니다.

B. 현재 눌린 키가 아스키코드면

i. EDIT(수정) 모드일 때는

1. 현재 커서 뒤가 문장의 끝일 경우 INSERT(삽입) 모드처럼 처리

2. 현재 커서 뒤가 문장의 끝이 아닌 경우, 커서 위의 문자를 입력된 문자로 바꿈

ii. INSERT(삽입) 모드일 때는

1. (현재 커서 위치 + 1)부터 한칸씩 뒤로 밀고, 현재 커서 위치에 글자를 넣음.
커서 1 증가시킴.

C. 현재 눌린 키가 특수키면

i. BACKSPACE의 경우

1. 현재 커서 위치가 맨 처음(좌측 상단)이 아닌 경우, (현재 커서 위치+1)부터 하나씩 앞으로 땡기고 커서 위치를 앞으로 한칸 이동.

ii. DELETE의 경우

1. (현재 커서 위치+1)부터 하나씩 앞으로 땡김.

iii. 좌측 화살표 키의 경우

1. 현재 커서 위치가 맨 처음(좌측 상단)이 아닌 경우 하나 앞으로 땡김.

iv. 우측 화살표 키의 경우

1. 현재 커서 다음 위치가 버퍼의 끝이 아닌 경우 하나 뒤로 이동.

v. 아랫 화살표 키의 경우

1. 현재 1차원 커서를 2차원 커서로 변환.
2. 현재 2차원 커서의 위치를 CURRENT ROW, CURRENT COL이라고 할 때, 현재 1차원 커서보다 뒤에 있는 문자들 중
(CURRENT ROW == ROW)
|| (CURRENT ROW + 1 == ROW && CURRENT COL >= COL) 을 만족하는 가장 큰 index의 문자를 찾습니다.

3. 현재 1차원 커서의 위치를 그 문자의 INDEX로 설정합니다.

vi. 윗 화살표 키의 경우

1. 현재 1차원 커서를 2차원 커서로 변환
2. 버퍼의 처음부분 글자부터 2차원 커서 위치를 계산하는데,
(CURRENT ROW - 1 == ROW && CURRENT COL >= COL) 을 만족하는 가장 큰 INDEX의 문자를 찾습니다.
3. 현재 1차원 커서의 위치를 그 문자의 INDEX로 설정합니다.

4. 단, CURRENT ROW 가 0인 경우, 0,0으로 셋팅합니다.

vii. INSERT 키가 눌렸을 경우

1. ISEDIT이라는 값을 토글해줍니다.

2. ISEDIT이라는 값을 통해 우리는 현재 삽입모드인지 수정모드인지 알 수 있습니다. (EDIT : 0, INSERT : 1)

viii. ESC가 눌렸을 경우

1. 메인 루프를 나와서 마무리 준비를 합니다.('E' 항목참고)

D. 키 입력 처리가 끝나면 화면에 출력을 해줍니다. 그리고 다시 올라가 'A' 항목으로 점프합니다.

i. 화면 출력은 BUFFER에 1차원적으로 저장되어 있는 정보를 그대로 출력해줍니다.

ii. ENTER의 경우 CARRIAGE RETURN값이 버퍼에 저장되어 있는데, 방금 CARRIAGE RETURN을 모니터에 출력한 경우, LINE FEED도 출력해 줌으로써 개행이 되도록 합니다.

iii. 글자를 다 출력한 후 커서의 위치를 일일이 계산하여 구한 후 커서의 위치를 셋팅해줍니다.

E. ESC가 눌러서 마무리를 해야 할 경우

i. 처음에 화면 클리어 등 초기화할 때 미리 구해놔던 ATTRIBUTE값을 통해 RESET을 해줍니다.

1. 커서 크기, 화면 COLOR 등

ii. 스크롤을 한 줄만 올리고, 24:00 위치에 커서를 가져갑니다.

iii. 그 위치에서부터 BUFFER에 있는 값들을 재출력해줍니다.

iv. 그리고 21:4C Function으로 종료.

4. 현재 생길 수 있는 문제점

정확하게는 256 바이트의 글자가 아니라 255 바이트의 글자가 입력이 가능할 것입니다. 그 부분이 문제가 될 수 있다는 점을 명시합니다.

그 외에 80글자 이상 입력하면 자동으로 개행이 된다면, 이 숙제에서 필요로 하는 요소들은 전부 구현이 되었습니다.

5. 만들면서 힘들었던 점

저는 원래 C++로 프로그램을 개발을 하는 프로그래머인데, 이번에 어셈블리 언어로 간단한 에디터를 만들면서 힘든 점이 많았습니다.

자주 사용하는 부분은 함수로 만들어서 사용하면 좋을 것 같은데, 어셈블리에서는 함수를 만들어도 주의하지 않으면 잘못된 레지스터 값을 바꾸는 일도 생기곤 합니다. 또한 메모장에서 코딩을 하다 보니 코드 관리가 무척 힘들었습니다.

저도 모르게 이 숙제의 코드를 고급언어 사용할 때의 습관대로 짰습니다. 그렇게 해도 되는 건 모르겠는데, 그래도 오히려 이 숙제를 하면서 왜 고급언어가 나왔는지, 그 고급언어들의 실제 내부는 어떻게 구현되어 있는 것인지 감을 얻을 수 있었습니다.

누군가는 “이런 어셈블리 코딩, 요즘 누가하냐, 필요없는거 아니냐”라고 할 지 몰라도, 저에게는 컴퓨터 시스템이란게 어떻게 이루어져있고 우리가 사용하는 고급 언어들이 어떻게 이루어져 있는지 알아볼 수 있는 좋은 시간이었습니다.

6. 추가로 구현했으면 좋겠는 점

- A. 현재 삽입 모드인지 수정 모드인지 표시
- B. 현재 커서의 위치
- C. 내용을 파일 등으로 저장
- D. 색깔 있는 폰트 설정 가능하게(TXT파일로는 저장이 안되겠지만, 이 에디터로만 읽을 수 있는 파일 포맷을 만들어서.. 마치 .doc이나 리치텍스트포맷 처럼)

정보 표현하는 부분은 매 루프마다 커서 위치를 최하단 좌측, 우측 등에 옮겨서 출력해주면 쉽게 만들 수 있을 것 같습니다.

7. 코드

코드는 외부 파일인 HW6.ASM을 확인해주시시오.