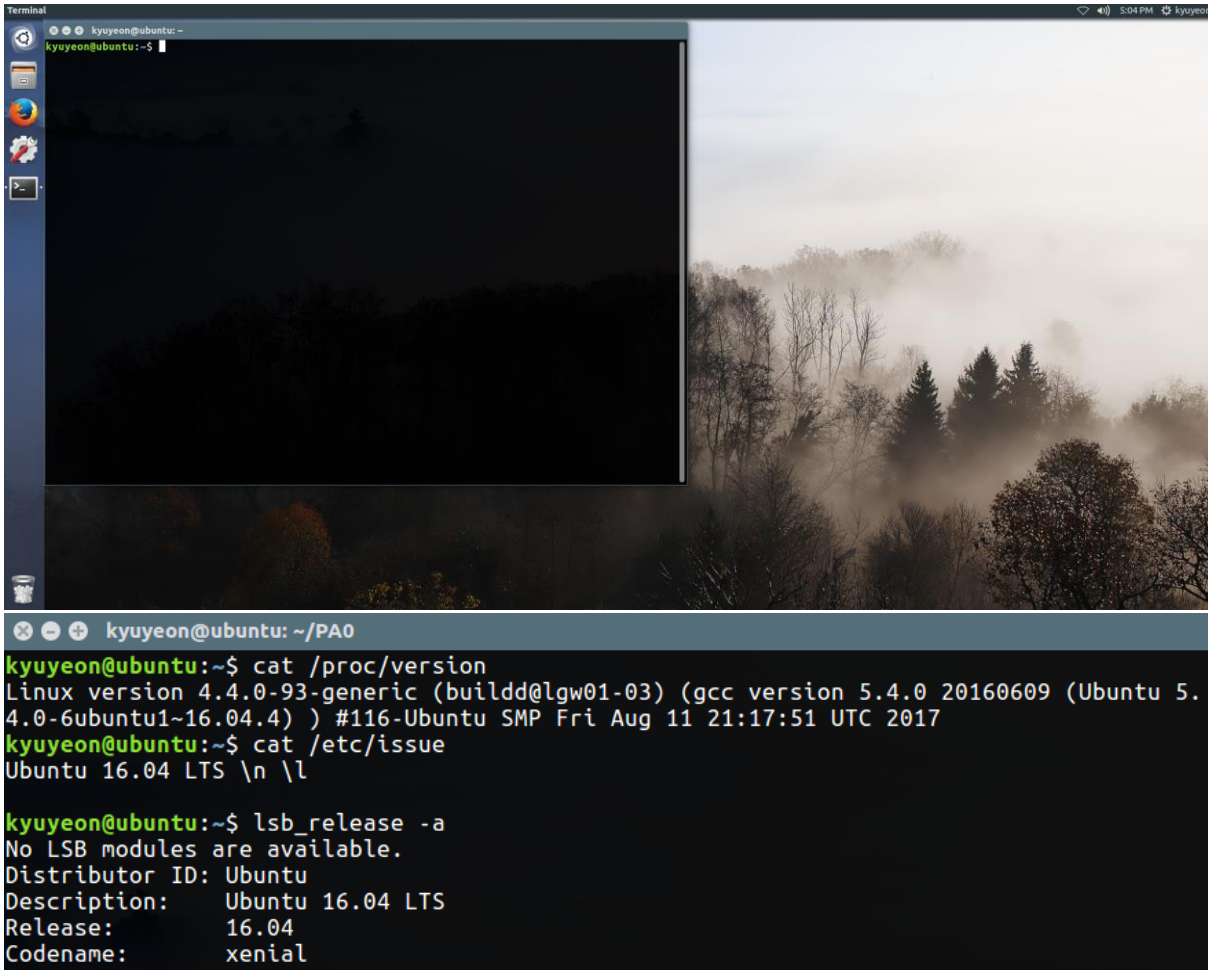
Ubuntu 환경 스크린샷



PA 0 코드 설명

1. atoi2 function

#24 ~ #30: sign이라는 변수를 두어 먼저 이 정수가 음수인지 양수인지 판별하였다. 음수면 sign은 1, 양수면 0이다. 앞에 - 부호가 붙은 경우는 음수로 처리하되, +가 붙거나 아무것도 붙지 않았을 때는 양수로 처리하였다.

#31 ~ #36: 큰 자리의 수부터 차례대로 처리하여 한 수를 읽어 들일 때마다 10씩 return할 값에 곱해주었다. 마지막에 return하기 전에 음수이면 -1을 곱하여 return 한다. 신기하게도 이 방법을 쓰면 INT_MIN의 경우에도 overflow가 발생하는데, 이 경우도 성공적으로 처리된다.

2. atol2 function

atoi2와 return할 변수 타입이 long이라는 점 이외에는 변화가 없다.

3. int2str function

설명하기에 앞서 int2str 함수는 *dest 변수의 메모리를 미리 할당하여 함수에 넣을 필요가 없다. 함수에서 *dest를 위한 메모리까지 할당하여 return해준다.

#70 ~ #86: switch 문으로 num이 INT_MIN인 경우와 0인 경우에 대해서 예외처리를 하였다. INT_MIN인 경우 -1을 곱하면 overflow가 발생하고, 0인 경우에서는 하단의 while문을 단 한 번도 거치지 않기 때문이다. temp는 num을 작은 자리 수부터 순서대로 저장해두는 버퍼이다.

#89 ~ #93: 음수인 경우에 대하여 –를 추가해줌과 동시에 temp를 거꾸로 된 순서로 저장한 dest를 return해준다.

4. strcpy function

null-terminator 문자가 오기 전까지 src를 읽어들여 dst에 저장한다. 이 때 while을 통과하고 나면 조건에 의해 dst의 끝은 null-terminated 되지 않으므로, 끝에 \0을 붙여줘야 한다.

5. strncpy function

count < strlen(src) 일 경우에는 dst가 null-terminated 되지 않는 다는 점에 주의하여 코딩했다. man page에서 정의했듯이 count > strlen(src)일 경우는 뒤에 남는 부분을 모두 \0로 채워 넣었다. 이 외에는 strcpy와 동일하다.

6. strcat function

dst의 \0을 찾을 때까지 index를 늘리고 그 뒤부터 src의 \0 전까지 뒤에 붙인다. 이 또한 while문 조건에 의해 \0을 맨 뒤에 따로 붙여줘야 한다.

7. strncat function

counter 변수에 따른 횟수가 감안된 것을 제외하면 strcat와 동일하다. 그러나

strncpy와는 다르게 count가 src의 길이를 넘어가도 src의 \0에 다다르면 멈춘다.

8. strdup function

우선 함수 내에서 메모리를 할당해야하기 때문에 return할 문자열을 clone이라고 했다. null-terminaotr까지 고려하여 str의 길이보다 1만큼 큰 메모리를 잡은 후, 여기에 위에서 짰던 strcpy 함수를 사용하여 clone에 str을 복사한다. 위의 메모리 할당에 실패하면 NULL을 반환한다.

9. strlen function

\0가 나올 때까지 문자열의 포인터를 하나씩 옮기며 길이를 하나씩 늘려갔다.

10. strcmp function

while 문에 의해 둘 다 \0에 도달하지 않았다면 while 문은 계속 진행된다. lhs안의 문자에서 rhs안에 들어있는 문자를 빼준다. 만약 lhs가 rhs보다 우선이면 dif > 0, 곤가 우선이면 dif < 0을 반환하게 된다. 만약 동일하다면, dif == 0이므로 while 문을 계속 돌게 된다.

둘의 문자열 길이가 다른 경우 마지막에 한 문자열은 \0을 만나는 반면 다른 문자열은 \0이 아니기 때문에 필연적으로 0이 아닌 값이 return된다.

11. strncmp function

횟수를 세어주는 count 변수를 도입해서 일정 횟수만 돌도록 처리해준 점을 제외하면 strcmp와 동일하다.

12. strchr function

str에 대해서 \0가 나오기 전까지 앞에서 차례대로 ch가 있는지 검사한다. 만약 있다면 해당 위치를 return 한다. while 문을 모두 통과했다면 ch가 str 안에 없다는 의미이므로 NULL을 return하게 된다.

13. strrchr function

가장 뒤에 위치하는 ch를 찾아야하기 때문에 우선 p를 NULL로 초기화하고 str의 처음부

터 끝까지 돌게된다. 끝까지 돌 때까지 ch를 찾을 때 p의 위치를 재지정하므로 맨 뒤에 있는 위치를 return하게 된다. 만약 ch가 없다면 p의 값은 NULL에서 변하지 않기 때문에 NULL을 return하게 된다.

14. strpbrk function

str의 문자 하나하나에 대해 accept안의 있는 모든 문자에 대하여 순서대로 검사한다. accept에 포함된 문자 중에서 str와 겹치는 것이 하나라도 있다면 해당 str의 위치가 return 될 것이다. 만약 하나도 없다면 while문을 모두 통과하고 NULL을 return하게 된다.

15. strstr function

p_return은 return해줄 str의 위치, 즉 substr이 등장하는 첫 위치를 저장한다.

p_sub는 str의 매 문자마다 substr의 처음부터 검사하기 위해 둔 변수이다.

#254 ~ #260: p_return과 p_sub의 위치를 초기화하고, 둘 다 \0가 아닐 때까지 while 문을 돌게 된다. 만약 substr을 str에서 찾았다면 p_sub는 substr의 \0에 위치할 것이다. 따라서 #262에서 찾았다는 것을 표시하기 위해 isFound = 1로 변경한다 #266에서 찾은 isFound == 1이라면 바로 p_return을 return하게 된다.

#261 ~ #267: 만약 도중에 둘이 달라 break문을 거치게 된다면 p_sub는 \0에 도달할 수 없다. 만약 이때 안쪽 while 문에서 *p가 \0이었다면, 검사할 남은 str 문자 수가 substr보다 짧다는 의미이므로 더 이상 검사할 필요가 없다. 따라서 #263에서 *p == \0이어도 *p_sub != \0인 경우를 충족하면 NULL을 return하게 된다.

만약 위 과정을 다 통과했다면, 남은 문자에 대해서도 검사할 필요가 있다는 의미이므로 str++해주고 이어서 검사한다.

16. strtok function

사실 p라는 static 버퍼를 둔다는 것 외에는 strtok_r과 차이가 없다. 따라서 정적 버퍼 p를 선언해주고 이 p의 주소를 strtok_r의 매개변수로 넣어주게 되면 strtok 함수가 구현된다.

17. strtok_r function

str은 return할 token의 처음을 가리키는 pointer이다.

*saveptr은 str에 대해서 tokenize를 해줄 iterator역할을 해준다.

**saveptr는 *saveptr을 담는 곳의 주소가 되어준다.

#285 ~ #288: str == NULL이라면 이전 token의 뒤쪽부터 검사를 계속 진행하겠다는 의미이므로, token의 첫 위치인 str을 이전 token의 뒤 쪽 한 칸에 있는 *saveptr로 지정한다. str != NULL이라면 주어진 str에 대해 새로 *saveptr을 지정한다.

#289 ~ #290: 만약 str == NULL 이러가 *str == \0이라면 주어진 문자열이 NULL 포인터이거나 더 이상 생성할 token이 없다는 의미이므로 NULL을 return한다.

#291 ~ #305: *saveptr가 str의 맨 끝에 올 때까지 str의 각 문자에 대하여 delim에 해당하는 문자가 하나라도 있는지 검사를 진행하게 된다. 만약 delim에 해당하는 문자를 찾았다 하더라도 두 가지 경우를 생각해봐야 한다.

한 가지는 일반적인 경우로 str에서 delim을 찾은 위치가 return할 token의 첫 위치하고 다를 경우이다. 이 경우는 return할 token의 길이가 적어도 1이상이라는 의미이므로, *saveptr을 delim을 찾은 위치의 바로 뒤로 옮기고, token을 return해주면 된다.

다른 한 가지는 delim이 str안에서 연속적으로 등장할 경우이다. 즉, str에서 delim을 찾은 위치가 return할 token의 첫 위치하고 같을 경우이다.이 때는 delim을 찾은 위치를 \0로 바꾸면 return할 token의 길이는 0이다. 따라서 이 때는 해당 위치를 \0로 바꾼 후 return할 token의 pointer인 str과 *saveptr을 뒤로 하나씩 밀어준 다음에, 계속 진행한다.

18. memcpy function

일단 n-byte의 메모리에 대하여 1 byte씩 읽어들이기 위해 (char*)로 type-casting을 한 후, dest에 str을 차례대로 붙여넣었다.

19. memset function

일단 n-byte의 메모리에 대하여 1 byte씩 입력하기 위해 dest를 (char*)로 type-casting하고 dest에 ch를 차례대로 입력하였다.

실행결과

다양한 경우를 고려한 main.c 함수를 짜 string_sw.h를 포함하여 실행해본 결과, 모든 경우에 대해 성공적으로 돌아갔음을 알 수 있었다.

2016314827.zip 안에 main.c를 같이 첨부하였다.