1 Assignment01 2014314433 이영숙

/* Conversion of string to numeric format */

1. int my_atoi (const char *str);
2. long my_atol (const char *str);

/* Conversion of numeric format to string */

3. char *int2str (char *dest, int num);

/* String manipulation */

4. char *strcpy (char *dest, const char *src);
   -src에 ₩0이 등장하기 전까지 while문으로 처음부터 끝까지 src를 dest에 모두 복사한다.
   -while문에서 src[i]='₩0'상태가 되면 반복을 멈춘다.
   -dest의 마지막에는 ₩0을 넣고 desrt의 포인터를 리턴한다.

5. char *strncpy (char *dest, const char *src, size_t n);
   -src[i]가 '₩0'문자가 아니고 n보다 작은 인덱스에서는 dest에 src를 복사한다.
   -src[i]가 ₩0이거나(src복사완료) 인덱스[i]가 n보다 커지면while 탈출
   -i<n(n바이트보다 적은 바이트만 복사 된 경우, src가 n보다 작아서)일 때에는 n바이트만큼 복사되는 것을 보장하기 위해 나머지 dest[i]에 ₩0 삽입.
   -dest 포인터를 리턴한다.

6. char *strcat (char *dest, const char *src);
   -while문으로 dest의 길이를 파악한다(i).
   -for 문으로 위에서 파악한 dest의 ₩0문자가 저장 돼 있는 위치부터 src를 붙인다.
   -src를 다 붙이고 나면 dest의 맨 마지막에는 ₩0를 삽입해 준다.
   - dest 포인터를 리턴한다.

7. char *strncat (char *dest, const char *src, size_t n); *최대 n바이트 복사(반드시x)
   - dest의 길이 파악(끝 인덱스 파악)을 위해 for 문을 돌면서 ₩0을 만나기 전까지 size 변수를 하나씩 증가 시켜 준다.
   - src의 길이가 n바이트 보다 적고 src가 끝나지 않았을 때 for문을 돌면서 dest에 src를 뒷 붙여준다.
   - src가 끝까지 붙어졌거나, src에서 n바이트만큼 복사가 완료 된 경우에는 for문을 빠져 나온다.
   - src를 다 붙이고 나면 dest의 맨 마지막에는 ₩0를 삽입해 준다.
   - dest 포인터를 리턴한다.

8. char *strdup (const char *str);
   -for문으로 str의 사이즈(마지막 인덱스)를 구한다.
   -위에서 찾은 str의 사이즈만큼 newbie라는 새로운 배열을 동적할당으로 만들어 주고,
   - newbie에 str의 값들을 모두 복사해 준다.
   -newbie 마지막에는 ₩0을 삽입해 준다.

/* String examination */

9. size_t strlen (const char *str);
-str이 ₩0 아닐 때 까지 while로 반복하여 마지막 인덱스 값을 구한다.
-str에서 ₩0가 나오면 while문은 중단되고 이 때 저장된 인덱스 값을
-리턴하면 str의 길이가 구해진다.

10. int strcmp (const char *str1, const char *str2);
-str1크면 1, str2크면 -1, 같은 경우 0 출력
-while문으로 str1과 str2가 둘 다 끝나지 않은 상태에서 둘을 비교하는데, str1이 크면 1을, str2가 크면 -1을 출력하고 둘이 같은 경우에는 인덱스만 증가시키고 반복을 계속한다.
-str1&str2 중에 하나가 ₩0을 만났거나 둘 다 ₩0를 만난 상태에서 반복문은 빠져나오게 된다.
-둘 다 ₩0인 경우에는 같은 문자이기 때문에 0을 리턴하고, str1만 ₩0인 경우는 str2가 더 긴 배열이라는 뜻이므로 -1을 출력하고 그 반대의 경우에는 1을 출력한다.

11. int strncmp (const char *str1, const char *str2, size_t n);
-n바이트의 범위 내에서 while문으로 처음부터 인덱스n-1까지 str1과 str2가 둘 다 ₩0을 만나지 않을 때 비교한다.
- str1이 크면 1을, str2가 크면 -1을 출력하고 둘이 같은 경우에는 인덱스만 증가시키고 반복을 계속한다.
-while을 빠져 나왔을 때, i<n이라면 두 배열 중에 어느 한 쪽이 짧은 상황이다. str1만 ₩0인 경우는 str2가 더 긴 배열이라는 뜻이므로 -1을 출력하고 그 반대의 경우에는 1을 출력한다.
-i>=n인 경우에는 n바이트까지 두 배열이 비교가 끝난 상황인데 둘다 null이 아니므로 두 배열의 n바이트가 같은 경우이다. 0을 리턴한다.

12. char *strchr (const char *str, int c);
-while로 str을 인덱스 0부터 ₩0이 아닐 때까지 c의 유무 체크
-while을 돌다가 c를 만나게 되면 while에서 break로 빠져 나가 c의 위치를 출력하고
-c를 만나지 못한다면 인덱스만 증가시켜서 계속 반복 체크하다가 ₩0를 만나면 while이 끝나고 ₩0의 포인터를 리턴하게 된다.

13. char *strrchr (const char *str, int c);
우선 c가 출현한 마지막 인덱스 값을 저장할 수 있는 'last' 정수형 변수를 선언한다. while문으로 str전체에서 c의 출현을 체크하는데 c가 출현하면 이 위치의 인덱스 값을 last 에 저장하고 다른 문자이면 인덱스 값만 증가시켜서 다음 값을 체크한다. C가 출현하지 않았다면 last 값은 초기 값(-1)이기 때문에 ₩0 문자 주소를 리턴하고, c가 한 번이라도 출현한다면 last 값은 해당 위치의 인덱스 값이 될 것이고 그 포인터를 리턴하면 된다.

14. char *strpbrk (const char *str1, const char *str2);
while문으로 str1과 str2의 길이를 구한다. 이중 for문을 이용해 str1에 str2의 아무 문자 중 하나라도 등장하는지 체크한다. 이 for문에서는 어레이의 인덱스 값을 이용해 반복을 하고 있기 때문에 for문을 진행하다가 str2의 문자가 나타나면 해당 str1[index]의 포인터를 반환한다.

15. char *strstr (const char *haystack, const char *needle);
    1. haystack과 needle의 길이를 구한다.
    2. haystack에서 Needle의 첫 번째 문자가 있는 위치 값들을 모두 저장할 배열(index)를 haystack 길이만큼 동적할당한다.
    3. for문으로 haystack을 돌면서 needle[0]과 같은 값을 가진 haystack의 인덱스를 index 배열에 모두 저장한다. Index 마지막에는 ₩0을 넣어준다.
    4. 이렇게 생성된 index배열의 길이(len)를 구한다.
    5. 이제 index배열에 저장된 haystack의 인덱스들을 다 방문하여 needle값들을 가지고 있는지 확인한다. index배열에 저장된 haystack인덱스부터 시작하여 needle의 길이만큼 haystack과 needle을 비교한다.
    6. Needle과 비교하는 도중 haystack이 ₩0을 만나면 needle이 포함 돼 있지 않은 것이므로 null을 리턴하고, haystack과 needle이 같지 않으면 다음 index 배열 값으로 넘어 가서 위 과정을 반복한다. Haystack과 needle 값이 같은 경우에는 카운트(cnt)값을 증가 시킨다.
    7. 위에서 카운트 값이 니들의 길이보다 같거나 크면, needle이 haystack에 처음부터 끝까지 속해 있다는 것이므로 index 배열의 다음 값으로 넘어갈 필요 없다.
    8. 부분 스트링 needle이 haystack에서 시작되는 포인터를 리턴한다.

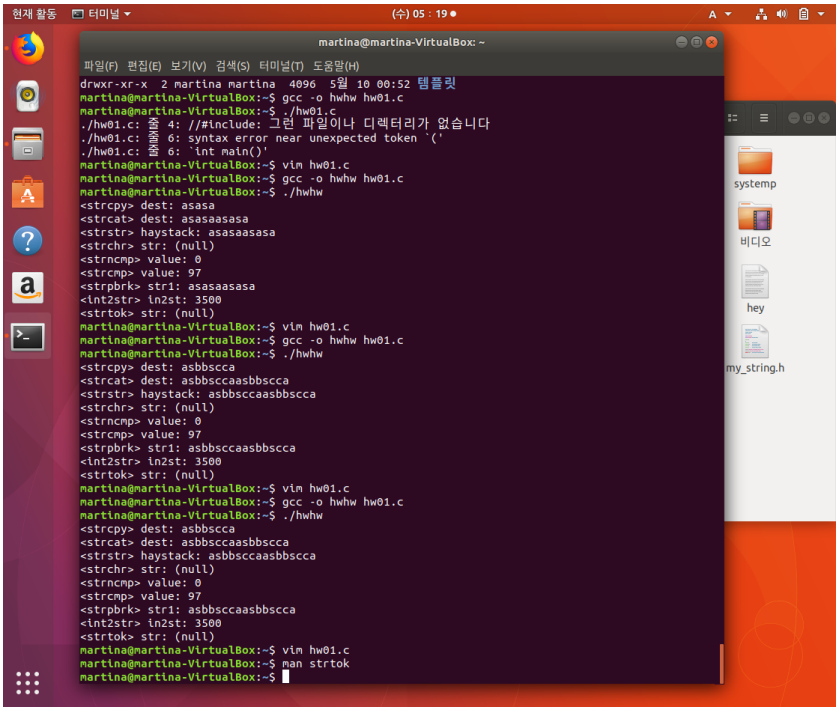
16. char *strtok (char *str, const char *delim);
17.

18. char *strtok_r (char *str, const char *delim, char **save_ptr);

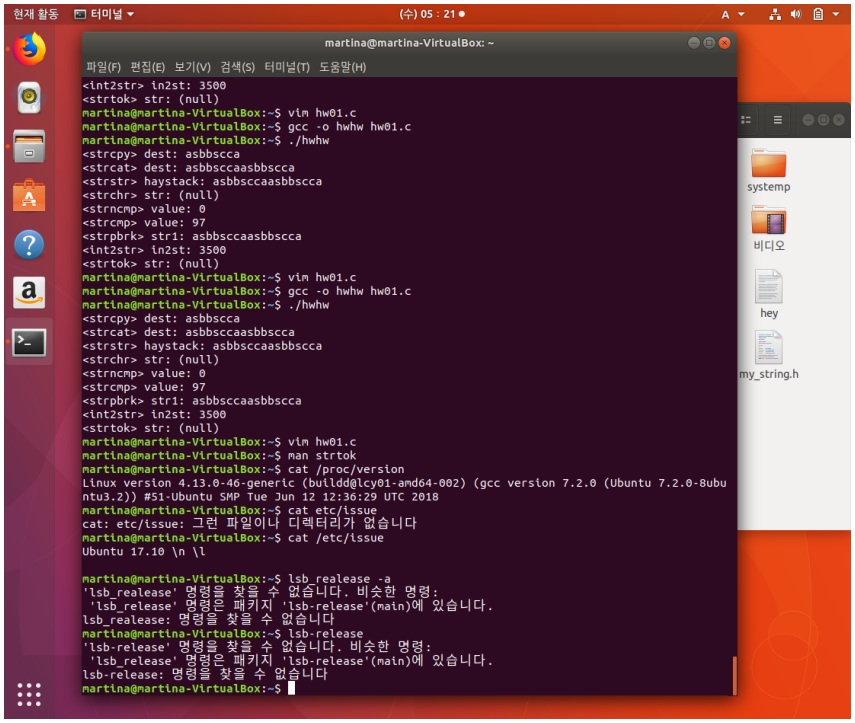
    /* Character array manipulation */
19. void *memcpy (void *dest, const void *src, size_t n);
    dest와 src를 char로 캐스팅 해준다. for문으로 0부터 n만큼 반복하여 두 어레이의 시작 주소부터 시작하여 dest에 src를 n만큼 복사한다.

20. void *memset (void *str, int c, size_t n);
    c를 n바이트만큼 str에 채울 수 있도록 해주려면 일단 void형인 str을 char으로 캐스팅 해 주고(1바이트씩 채워 주는 것이 가능해짐) str의 처음 시작 주소부터 for으로 돌면서 n바이트만큼 채워준다.

4 Assignment01 2014314433 이영숙



리눅스 함수 실행화면(hw01.c라는 연습 파일을 따로 만들어서 사용했음)



Lsb 명령은 실행이 안 됨.