Ejemplo de Código: Reemplazo de palabras en un archivo

Supongamos que tenemos un archivo de texto llamado entrada.txt con el siguiente contenido:

Hola mundo, bienvenido al mundo de la programación.

Queremos reemplazar la palabra "mundo" por "planeta", de modo que el archivo final quede así:

Hola planeta, bienvenido al planeta de la programación.

Paso 1: Leer el archivo

Primero, almacenamos el contenido del archivo en un buffer:

buffer: "Hola mundo, bienvenido al mundo de la programación."

Paso 2: Buscar la primera ocurrencia de "mundo"

Usamos **strstr(buffer, "mundo")** para encontrar la primera vez que aparece "mundo" en el texto. Esta función devuelve un puntero que apunta al primer carácter de "mundo" dentro del buffer.

Ejemplo de cómo están organizados los datos en memoria:

Dirección de memoria (ejemplo)

buffer[0] = 'H' 1000

buffer[1] = 'o' 1001

buffer[2] = 'l' 1002

buffer[3] = 'a' 1003

buffer[4] = ' ' 1004

buffer[5] = 'm' 1005 <- Aquí empieza "mundo"

buffer[6] = 'u' 1006

buffer[7] = 'n' 1007

buffer[8] = 'd' 1008

buffer[9] = 'o' 1009

buffer[10] = ',' 1010

El puntero resultado ahora contiene:

"mundo, bienvenido al mundo de la programación."

Para obtener la posición dentro del buffer, usamos:

i = resultado - buffer;

Si resultado está en la dirección 1005 y buffer empieza en 1000:

i = 1005 - 1000 = 5

Paso 3: Copiar el contenido antes de "mundo"

Creamos una variable aux para almacenar el texto modificado. Primero, copiamos todo lo que hay antes de "mundo":

strncpy(aux + j, buffer + k, i - k);

j = 0 → indica desde dónde en aux se guardará el contenido.

k = 0 → indica desde dónde en buffer empezamos a copiar.

i - k = 5 - 0 = 5 → la cantidad de caracteres a copiar.

Esto copia "Hola " en aux.

Después, actualizamos j para continuar desde la posición siguiente:

j += i - k;

j = 0 + (5 - 0) = 5

Paso 4: Insertar la nueva palabra

Ahora copiamos "planeta" en aux en la posición j:

strncpy(aux + j, word2, strlen(word2));

word2 = "planeta"

strlen(word2) = 7 (longitud de "planeta")

Ahora aux contiene:

"Hola planeta"

Avanzamos j para seguir construyendo el texto modificado:

j += strlen(word2);

j = 5 + 7 = 12

Paso 5: Seguir con el resto del buffer

Para continuar, actualizamos k para avanzar más allá de la palabra "mundo" en el buffer:

k = i + strlen(word);

Dado que:

word = "mundo" → tiene una longitud de 5

i = 5

Entonces:k = 5 + 5 = 10

Ahora k apunta al texto después de "mundo", es decir:

", bienvenido al mundo de la programación."

Paso 6: Buscar y reemplazar más ocurrencias

Volvemos a usar strstr para buscar otra aparición de "mundo", pero ahora desde la posición k = 10:

resultado = strstr(buffer + k, word);

Esto encuentra la segunda aparición de "mundo" en la posición 25. Repetimos los mismos pasos:

1. **Copiamos el texto antes de "mundo"** en aux desde buffer + k.
2. **Reemplazamos "mundo" con "planeta"**.
3. **Avanzamos j y k**.
4. **Repetimos hasta que ya no haya más "mundo"**.

Lógica de evitar reemplazar la palabra buscada en palabras compuestas

Texto original: "La camarera limpia la cama"

Palabra buscada: "cama"

Palabra para reemplazar: "colchón"

Buscar la primera ocurrencia de "cama"

Usamos **strstr(buffer, "cama")** para encontrar la primera vez que aparece "cama" en el texto. Esta función devuelve un puntero que apunta al primer carácter de "cama" dentro del buffer.

Ejemplo de cómo están organizados los datos en memoria:

Dirección de memoria (ejemplo)

buffer[0] = 'L' 1000

buffer[1] = 'a' 1001

buffer[2] = ' ' 1002

buffer[3] = 'c' 1003 <- Aquí empieza "mundo"

buffer[4] = 'a' 1004

buffer[5] = 'm' 1005

buffer[6] = 'a' 1006

buffer[7] = 'r' 1007

buffer[8] = 'e' 1008

buffer[9] = 'r' 1009

buffer[10] = 'a' 1010

buffer[10] = ' ' 1011

El puntero resultado ahora contiene:

"limpia la cama."

Antes de seguir veremos si es una coincidencia exacta para realizar el reemplazo uso de la función **eslimite** que indica si antes del inicio y después del final hay un carácter limitador como lo sería espacio, coma, punto, final de cadena , etc.

char antes = (i == 0) ? ' ' : buffer[i - 1]; // Si estamos al principio, simula un espacio

char antes =buffer[3-1]= ‘ ‘;

char despues = buffer[i + strlen(word1)];

char despues =buffer[3+4]=’r‘

if (es_limite(antes) && es_limite(despues)) ==False

Como después del final de la cadena hay un carácter que no es una limitante, te significa que no es una coincidencia exacta y se prosigue a hacer lo siguiente:

j = 0 → indica desde dónde en aux se guardará el contenido.

k = 0 → indica desde dónde en buffer empezamos a copiar.

i - k = 3 - 0 = 3 → la cantidad de caracteres a copiar.

word1 = "cama"

strlen(word1) = 4 (longitud de "cama")

Se copiará todo el contenido hasta llegar a la coincidencia

strncpy(aux + j, buffer + k, (i - k) + strlen(word1));

Ahora aux contiene:

"La cama"

Avanzamos j para seguir construyendo el texto modificado:

j += (i - k) +strlen(word1);

j=0+(7)= 7

Seguir con el resto del buffer

Para continuar, actualizamos k para avanzar más allá de la palabra "cama" en el buffer:

k = i + strlen(word1);

Dado que:

word1 = "cama" → tiene una longitud de 4

i = 3

Entonces:k = 3+4 = 7

Ahora k apunta al texto después de la primera coincidencia de "cama", es decir:

"rera limpia la cama."

Y de aquí se usa la misma lógica del ejemplo uno donde solo se realiza cambios con coincidencias exactas