En C, `free(puntero)` y `puntero = NULL` son acciones distintas y se utilizan en contextos diferentes:

1. **`free(puntero)`:**
    - `free` se utiliza para liberar la memoria asignada dinámicamente previamente mediante `malloc`, `calloc` o `realloc`.
    - Después de llamar a `free(puntero)`, la memoria asignada a través de `puntero` se marca como disponible para su reutilización por el sistema.
    - Sin embargo, el puntero `puntero` sigue apuntando a la dirección de memoria liberada. Si intentas acceder a esa memoria después de liberarla, puedes obtener un comportamiento indefinido, ya que el contenido de esa memoria ya no está garantizado.

```c
int *p = (int *)malloc(sizeof(int));
// ... (utilizar p)
free(p);
// Ahora p apunta a una dirección de memoria liberada, y no se debe acceder a *p
```

2. **`puntero = NULL`:**
    - `puntero = NULL` simplemente asigna al puntero la dirección de memoria nula.
    - Esto no libera la memoria asignada dinámicamente; solo indica que el puntero ya no apunta a ninguna ubicación válida.
    - Es una buena práctica establecer un puntero a `NULL` después de liberar la memoria para evitar que se acceda accidentalmente a la memoria liberada.

```c
int *p = (int *)malloc(sizeof(int));
// ... (utilizar p)
free(p);
p = NULL; // Ahora, p no apunta a ninguna dirección de memoria válida
```

En resumen, `free` se utiliza para liberar la memoria, mientras que `puntero = NULL` se utiliza para indicar que el puntero ya no apunta a ninguna ubicación válida. Usar `puntero = NULL` después de llamar a `free` es una práctica común para evitar errores al intentar acceder a la memoria liberada.