24.- ft_substr.-

Not directly based on any BSD man page, but similar to `strndup(3)`. Associated library “libft.h”.

Synopsis:

```
char *ft_substr(char const *s, unsigned int start, size_t len);
```

Purpose:

- Extracts a substring of a specified length (`len`) from a given string (`s`), starting at a given index (`start`).
- Allocates memory for the substring and returns a pointer to it.

Parameters:

- `s`: The original string to extract from.
- `start`: The index within `s` where the substring begins.
- `len`: The maximum length of the substring to extract.

Return Value:

- Returns a pointer to the newly allocated substring, or `NULL` if memory allocation fails or invalid parameters are provided.

Description:

- Handles edge cases like empty strings, invalid start positions, or lengths that extend beyond the end of the string.
- Allocates memory for the substring using `ft_calloc` to ensure it's zero-initialized.
- Copies the desired portion of the original string to the substring using `ft_strlcpy`.

Code:

```
#include "libft.h"

char *ft_substr(char const *s, unsigned int start, size_t len)
{
    char *substr;
    size_t i;

    // Handle edge cases
    if (!s)                 return (NULL);  // Invalid string
    if (len == 0 || ft_strlen(s) == 0 || ft_strlen(s) <= start)
                             return (ft_strdup(""));  // Empty substring

    // Determine substring length considering string boundaries
    i = 0;
    if (ft_strlen(s) - start > len)
        i = len + 1;  // Length limited by 'len'
    else
        i = ft_strlen(s) - start + 1;  // Length up to string's end

    // Allocate memory for the substring
    substr = ft_calloc(sizeof(char), i);
    if (!substr)            return (NULL);  // Allocation failure
```

```
    // Copy the substring from the original string
    ft_strlcpy(substr, s + start, i);

    return (substr); // Return pointer to the substring
}
```

Code Explanation:

1. **Handle Edge Cases:** Checks for invalid inputs and returns empty substring or `NULL` accordingly.
2. **Determine Length:** Calculates the actual length of the substring based on `len` and string boundaries.
3. **Allocate Memory:** Uses `ft_calloc` to allocate memory for the substring with zero-initialization.
4. **Copy Substring:** Uses `ft_strlcpy` to safely copy the desired portion of `s` to `substr`.
5. **Return Substring:** Returns the pointer `substr` to the newly created substring.

Comments for the `main` Function:

```
int main(void)
{
    char s[] = "Hello, World!";
    char *substring;

    substring = ft_substr(s, 7, 6);  // Extract "World"

    if (substring) {
        printf("Substring: %s\n", substring);
        free(substring);  // Free allocated memory
    } else {
        printf("Error allocating memory for substring\n");
    }

    return (0);
}
```

Key Points:

- **Error Handling:** Handles invalid inputs and memory allocation failures gracefully.
- **Zero-Initialization:** Uses `ft_calloc` to initialize the substring with zeros for safety.
- **Safe String Copying:** Employs `ft_strlcpy` to prevent buffer overflows.
- **Memory Management:** Requires manual `free` after use to avoid memory leaks.