<u>PRINTF</u>:

```
#include <stdarg.h>
#include <stdio.h>
#include <unistd.h>

void	ft_putchar(char c, int *count)
{
	write(1, &c, 1);
	*count = *count + 1;
}

void	ft_putstr(char *str, int *count)
{
	int	i = 0;
	if (!str)
		ft_putstr("(null)", count);
	else
	{
		while (str[i])
		{
			ft_putchar(str[i], count);
			i++;
		}
	}
}

void	ft_putdigit(long long int nbr, int base, int *count)
{
	char	*hex = "0123456789abcdef";

	if (nbr < 0)
	{
		nbr = -nbr;
		*count += 1;
		write(1, "-", 1);
	}
	if (nbr >= base)
		ft_putdigit((nbr / base), base, count);
	*count += 1;
	write(1, &hex[nbr % base], 1);
}

void	ft_select_type (va_list args, const char format, int *count)
{
	if (format == 'c')
		ft_putchar(va_arg(args, int), count);
	if (format == 's')
		ft_putstr(va_arg(args, char *), count);
	if (format == 'd')
		ft_putdigit((long long int)va_arg(args, int), 10, count);
	if (format == 'x')
```

```
		ft_putdigit((long long int)va_arg(args, unsigned int), 16, count);
}

int ft_printf(const char *format, ...)
{
	va_list args;
	int		count = 0;
	int		i = 0;

	va_start(args, format);
	while (format[i])
	{
		if (format[i] == '%')
		{
			i++;
			ft_select_type(args, format[i], &count);
		}
		else
			ft_putchar(format[i], &count);
		i++;
	}
	va_end(args);
	return (count);
}
```

GNL con lectura a 1

```c
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>

#ifndef BUFFER_SIZE
# define BUFFER_SIZE 42
#endif

char *get_next_line(int fd)
{
    char a[999999] = {0};
    char *new_s;
    int i = 0;
    int bytes_read;

    if (fd < 0 || BUFFER_SIZE <= 0)
        return (NULL);

    while ((bytes_read = read(fd, &a[i], 1)) == 1)
    {
        if (a[i] == '\n')
        {
            i++;
            break;
        }
        i++;
    }

    if (bytes_read == -1 || (bytes_read == 0 && i == 0))
        return (NULL);

    new_s = malloc(i + 1);
    if (!new_s)
        return (NULL);

    for (int j = 0; j < i; j++)
        new_s[j] = a[j];
    new_s[i] = '\0';

    return (new_s);
}
```

GNL completo

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>

#ifndef BUFFER_SIZE
# define BUFFER_SIZE 42
#endif

char *my_strchr(const char *s, int c)
{
    while (*s)
    {
        if (*s == (char)c)
            return (char *)s;
        s++;
    }
    if (c == '\0')
        return (char *)s;
    return (NULL);
}

char *my_strdup(const char *s)
{
    char *dup;
    size_t len = 0;

    while (s[len])
        len++;
    dup = malloc(len + 1);
    if (!dup)
        return (NULL);
    for (size_t i = 0; i < len; i++)
        dup[i] = s[i];
    dup[len] = '\0';
    return (dup);
}

static char *my_strjoin(char *s1, char *s2)
{
    char *joined;
    size_t len1 = 0, len2 = 0;

    while (s1 && s1[len1])
        len1++;
    while (s2[len2])
        len2++;
    joined = malloc(len1 + len2 + 1);
    if (!joined)
```

```
        return (NULL);
    for (size_t i = 0; i < len1; i++)
        joined[i] = s1[i];
    for (size_t i = 0; i < len2; i++)
        joined[len1 + i] = s2[i];
    joined[len1 + len2] = '\0';
    free(s1);
    return (joined);
}

static char *extract_line(char **buffer)
{
    char *line;
    char *new_buffer;
    size_t i = 0;

    while ((*buffer)[i] && (*buffer)[i] != '\n')
        i++;
    if ((*buffer)[i] == '\n')
        i++;
    line = malloc(i + 1);
    if (!line)
        return (NULL);
    for (size_t j = 0; j < i; j++)
        line[j] = (*buffer)[j];
    line[i] = '\0';
    new_buffer = my_strdup(*buffer + i);
    free(*buffer);
    *buffer = new_buffer;
    return (line);
}

char *get_next_line(int fd)
{
    static char *buffer;
    char temp_buffer[BUFFER_SIZE + 1];
    int bytes_read;

    if (fd < 0 || BUFFER_SIZE <= 0)
        return (NULL);

    while ((bytes_read = read(fd, temp_buffer, BUFFER_SIZE)) > 0)
    {
        temp_buffer[bytes_read] = '\0';
        buffer = my_strjoin(buffer, temp_buffer);
        if (!buffer)
            return (NULL);
        if (buffer && (my_strchr(buffer, '\n')))
            break;
    }

    if (bytes_read == -1 || (bytes_read == 0 && (!buffer || !*buffer)))
    {
        free(buffer);
        buffer = NULL;
        return (NULL);
    }

    return (extract_line(&buffer));
}

int main (void)
{
            char     *line;
            int                   i = 0;
            ssize_t fd1;
            char     file1[1024] = "./testGNL/test3.txt";
            printf("BUFFER_SIZE: %d\n----------------------\n", BUFFER_SIZE);
            fd1 = open(file1, O_RDONLY);
            if (fd1== -1)
            {
                        printf("Fail!\n");
                        return (0);
            }
            while (i <= 12)
            {
                        line = get_next_line(fd1);
                        printf("[fd1]: %s\n", line);
                        free(line);
                        i++;
            }
            close(fd1);
            return (0);
}
```