|  |
| --- |
| Write a function named `ft\_printf` that will mimic the real printf but //it will manage only the following conversions: s,d and x.  # include <stdarg.h>  # include <unistd.h>  # include <stdio.h>  int f\_strlen (const char \*str)  {  int i;  i = 0;  while (str[i])  i++;  return (i);  }  int ft\_print\_str (char \*str)  {  if (str == NULL)  return (write (1, "(null)", 6));  return (write (1, str, f\_strlen(str)));  }  int ft\_print\_dec(int nbr)  {  int j;  char ch;  j = 0;  if (nbr == -2147483648)  return (write (1, "-2147383648", 11));  if (nbr < 0)  {  j += write (1, "-", 1);  nbr = -nbr;  }  if (nbr > 9)  j += ft\_print\_dec(nbr / 10);  ch = "0123456789"[nbr % 10];  j += write (1, &ch, 1);  return (j);  }  int ft\_print\_hexa(unsigned int hexa)  {  int j;  char nb;  j = 0;  if (hexa > 15)  j += ft\_print\_hexa(hexa / 16);  nb = "0123456789ABCDEF"[hexa % 16];  j += write (1, &nb, 1);  return (j);  }  int ft\_printf(const char \*line\_for\_print, ...)  {  int j;  va\_list ap;  va\_start(ap, line\_for\_print);  j = 0;  while (\*line\_for\_print)  {  if (\*line\_for\_print == '%')  {  line\_for\_print++;  if (\*line\_for\_print == 's')  j += ft\_print\_str(va\_arg(ap, char \*));  if (\*line\_for\_print == 'd')  j += ft\_print\_dec (va\_arg(ap, int));  if (\*line\_for\_print == 'x')  j += ft\_print\_hexa (va\_arg(ap, unsigned int));  line\_for\_print++;    }  else  {  j += write (1, line\_for\_print, 1);  line\_for\_print++;  }  }  va\_end (ap);  return (j);  }  int main()  {  int N1, N2;  {  printf ("--------------------------------string s -------------------------------------");  char line[] = "\n printing %s succeeded \n";  char s[] = "one line";  N1 = printf (line, s);  N2 = ft\_printf (line, s);  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  {  char line[] = "\n printing %s succeeded \n";  char s[] = "";  N1 = printf (line, s);  N2 = ft\_printf (line, s);  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  {  char line[] = "\n printing %s succeeded \n";  char \*s;  s = NULL;  N1 = printf (line, s);  N2 = ft\_printf (line, s);  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  {  printf ("--------------------------------hexadecimal x -------------------------------------");  char line[] = "\n printing one hexadecimal %x succeeded \n";  int x = -1;  N1 = printf (line, x);  N2 = ft\_printf (line, x);  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  {  char line[] = "\n printing one hexadecimal %x succeeded \n";  int x = 0;  N1 = printf (line, x);  N2 = ft\_printf (line, x);  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  {  char line[] = "\n printing one hexadecimal %x succeeded \n";  int x = 0xA;  N1 = printf (line, x);  N2 = ft\_printf (line, x);  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  {  char line[] = "\n printing one hexadecimal %x succeeded \n";  int x = 0x31A;  N1 = printf (line, x);  N2 = ft\_printf (line, x);  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  {  char line[] = "\n printing one hexadecimal %x succeeded \n";  int x = 675;  N1 = printf (line, x);  N2 = ft\_printf (line, x);  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  {  printf ("--------------------------------decimal d -------------------------------------");  char line[] = "\n printing one decimal %d succeeded \n";  int x = -1000000000;  N1 = printf (line, x);  N2 = ft\_printf (line, x);  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  {  char line[] = "\n printing one decimal %d succeeded \n";  int x = 145;  N1 = printf (line, x);  N2 = ft\_printf (line, x);  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  {  printf ("--------------------------------mix 2 ---------------");  N1 = printf ("\n%s%d%s%%\n", NULL, 2, "test");  N2 = printf ("\n%s%d%s%%\n", NULL, 2, "test");  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  {  printf ("--------------------------------mix 3 --------------------");  N1 = printf ("\n%s%s%%\n", NULL, "test");  N2 = printf ("\n%s%s%%\n", NULL, "test");  printf ("\n\t number of characters printf = %i, ft\_printf = %i\n", N1, N2);  }  }  #ifndef FT\_PRINTF\_H  # define FT\_PRINTF\_H  int ft\_printf(const char \*line\_for\_print, ...);  int ft\_print\_char(char ch);  int ft\_print\_line(char \*str);  int ft\_print\_pointer(void \*pointer);  int ft\_print\_int\_dec(int nbr);  int ft\_print\_u\_dec(unsigned int nbr);  int ft\_print\_hexa(unsigned int nbr, char a);  #endif  NAME = libftprintf.a  CFLAGS = -Wall -Werror -Wextra  CC = gcc  SOURCES = srcs/  INCLUDE = include/  OBJ = $(SOURCES)ft\_printf.o \  $(SOURCES)ft\_print\_char.o \  $(SOURCES)ft\_print\_line.o \  $(SOURCES)ft\_print\_pointer.o\  $(SOURCES)ft\_print\_int\_dec.o\  $(SOURCES)ft\_print\_u\_dec.o\  $(SOURCES)ft\_print\_hexa.o\  OBJ\_BONUS = srcs\_bonus/ft\_printf\_bonus.o  all: $(NAME)  bonus: $(OBJ\_BONUS) $(NANE)  $(NAME): $(OBJ)  ar rcs $(NAME) $(OBJ)  %.o: %.c  $(CC) -c $(CFLAGS) -I$(INCLUDE) $< -o $@  clean:  rm -f $(OBJ) $(OBJ\_BONUS)  fclean: clean  rm -f $(NAME)  re: fclean all  .PHONY: all bonus clean fclean re |