**CSE 3211 Operating Systems Lab**

**Assignment 01**

**Implementation of kprintf and kscanf functions for the kernel**

**Submitted to:**

**Dr. Mosaddek Hossain Kamal**

**Professor**

**Department of Computer Science and Engineering University of Dhaka**

**1: Implementation of kprintf:**

The function prototype:

*void kprintf(uint8\_t \*format,uint8\_t\* outvar);*

The datatypes handled to write the USART to the terminal is given below:

%c : for character

%s : for string

%d : for integer

%x : hexadecimal

%f : for floating point number.

The function is designed to take the *address of the input variable* as a parameter. So, if the input datatype is that of an integer, then the kprintf function will simply take the address that refers to that variable. However, for some of the datatypes the data address needs to be typecasted and for others no particular manipulation is necessary. For integer, floating point and hexadecimal datatypes, the address need to be type casted using uint8 t and then written to the terminal using USART. This operation makes the value retrieving process streamlined and simple. For the character and string data types, no manipulation is necessary as they are in the perfect format for USART writing. The other processes are handled using if else format so that all the case by case operations can be handled. Let us talk about each datatype in detail.

* 1. **The Character Datatype**

This datatype is in the ideal state for USART write and thus no manipulation was done.

* 1. **The String Datatype**

As string comprises of multiple characters, it follows the same format as the character datatype and that’s why it also did not need any manipulation.

* 1. **The Integer Datatype**

The declared outvar pointer pointed to the integer value’s memeory location.We initialize a temp name integer variable where we copy our integer data by calling the my\_memcopy function. After copying the value into to temp variable then conver\_int\_string function called and write to the USART\_WRITE function for print the value.

* 1. **The Hexadecimal Datatype**

Similar to integers, after copying the the value from my\_memcopy function we called convert\_into\_hexString function.

then the string was written to USART

* 1. **The Floating Point Datatype:**

Similar to previous cases the adress was typecasted and then the value was passed to a function called convertFloattoString(). And then the string was passed to USART write funtion for printing. 1 For any other case that might occur, we implemented a function called showError() that will tell us that there has been an error.

1. **Implementation of kscanf**

The function prototype:

*void kscanf(uint8\_t \*format,uint8\_t\* invar)*

The function scans the input that is given by the user. It is simply the preceeding function to the kprintf function. It takes the address in which the value needs to be stored. The address will come as a set of characters. So the input needed to be scanned character by character in order to place the correct value in the correct place.

* 1. **The Character Datatype**

Just like kprintf this datatype does not need a manipulation. The value will simply be read from the address specified by the invar datatype.

* 1. **The String Datatype**

The process for this datatype is just the same as above.

* 1. **The Integer Datatype**

The data and address need to be written in string format and this data is currently in a string format. So the data was converted into integer using convertStringtoInt() function. Then the data was stored.

* 1. **The Hexadecimal Datatype**

In the same way as above, the string was converted to hexadecimal format using the convertStringtoHex() function and then the data was stored.

* 1. **The Floating Point Datatype**
  2. The precedure is the same as above. The function in question is convertStringtoFLoat(). Any other case will be handled by the showError() function and an errror message will be shown.