Sample C questions

November 2013

Write a C function called `dispatch` with 3 arguments. The first is an enumeration, the second is a pointer to an argument and the third is a pointer to a NULL pointer that will point to the result when the function returns. The return value of the `dispatch` function is 0 in case of success and a non zero value, reporting what the error is, in case of failure.

The `dispatch` function uses its first argument as an index to a constant static table of function descriptors. Each line in the table includes 3 elements: a pointer to a function and the sizes of the structures that are the argument and the result of the function. Each function must be called with two arguments: a pointer to its argument structure and a pointer to its result structure. The fields of this structure will be filled in by the function. Success or failure of the called function is reported through its return value.

For answering, proceed in two steps

1. design the basic functionality of `dispatch`
2. think of everything that could go wrong and, if it happens, report it with an appropriate return code. This includes checking what can be checked in `dispatch` and specifying the minimum checks that the called fonctions must perform and report.

January 2014

Write a C function called « myserial », that, according to the first argument, called todo, that is either 's' or 'd', serialises or deserialises a linked list of structures consisting of a pointer to the next structure of the list (or a null pointer in the last structure), a table of two 32 bit integers, and a string.
The function has 3 arguments: todo, a pointer, called list, to a pointer to the first structure of the list and a pointer, called serial, to a pointer to a memory area (that must have exactly the right size) including the serialised list.
These 4 pointers exist when the function is entered. According to the value of the first argument, the others are input or output. Pointers to output structures or area are zero when the function is entered. The function returns 0 in case of success and -1 in case of failure.

NB: Wikipedia says: serialization is the process of translating data structures or object state into a format that can be stored (for example, in a file or memory buffer, or transmitted across a network connection link) and resurrected later in the same or another computer environment. Serializing is also called sometimes marshalling.

January 2015

Write a C function called « manage_buf ».The function manages a linked list of free buffers. The function has a single argument that is a pointer to a pointer to a struct « mybuf » defined elsewhere. If the argument points to a NULL pointer, the function provides a struct mybuf, else a struct mybuf that is not needed any more is returned to the function. When a free buffer is asked, if free buffers are available in the list, the function provides via the argument the buffer at the head of the list, else it allocates a new buffer. When a buffer is returned, it is inserted at the head of the list. A second list is associated to the list of buffers : this is a list of int counters. Each time the function is called each counter is increased. Buffers associated with counters that reach the constant MAXBUFCALLS are freed.

January 2016

You must write two functions, « mydebug » and « myprint ».

« mydebug » is intended to record the contents of a structure called « structA » at several times during the execution of a program. « structA » consists of an integer, « a », a character string called « b », and a pointer,called « c », to a function.

« mydebug » has two arguments, a pointer to a « structA » and a pointer to a pointer to a « structB » (this pointer is normally null the first time the function is called). The « structB » consists of a pointer to a « structB », a « structA », and the value of the high resolution time, that can be obtained using the function «hrtime_t get hrtime(void) » that returns the time in nanoseconds since an arbitrary time in the past (arbitrary, but the same each time the function is called, of course).
When called, mydebug must obtain the hrtime, then call the function referred to in the « structA » received with hrtime as argument. If the function returns « TRUE », it must write hrtime in a new « structB », copy the « structA » received in argument in the « structB » , and insert this « structB » at the head of a linked list ; the location of a pointer to the current head of that list is received in argument.
If the function returns false, it must do nothing.

« myprint » is intended to be called at the end of the program under test. It receives a pointer to a pointer to a « structB », pointing to a list of « structB ». It must print the integer, the string and the name of the function of each « structA » along with the high resolution time of its recording, in the same order as they were recorded. It must also free each « structB » after printing its contents. The name of the functions can be found in a table of structs consisting of their starting address in memory and a string with their name. That table is a global static variable called func_name_tab.