Nicholas Castorina

0822759

Project 1 Summary

First in /public/sdk/inc/ntexapi.h, I modified the definition of the SYSTEM\_CSE451\_INFORMATION struct. This struct contained an array of CSE451\_API\_STATUS\_COUNT which contained one index for each API that we were keeping track of the statuses for. Consequently I placed an enumeration named CSE451\_APIS that translated the indices of SYSTEM\_CSE451\_INFORMATION to the APIs. Each instance of CSE451\_API\_STATUS\_COUNT contained an array of STATUS\_COUNT (a struct tuple that contains the status and the count of that status) to represent each of the statuses returned by each of the APIs that we were tracking.

An incStatus function was also placed in this file that increments the status of a particular function and status code, or insert it if it needed to be inserted.

The actual instance of the SYSTEM\_CSE451\_INFORMATION was placed in sysinfo.c, with an extern definition in /base/ntos/inc/ntosdef.h. The instance of SYSTEM\_CSE451\_INFORMATION was placed in sysinfo.c because it keeps track of the statuses that certain APIs return, which is system information. The extern definition in ntosdef is used so every file that modifies the information, modifies the same struct and not different structs. The definitions of the struct are placed in ntexapi.h because this file hold all of the definitions used in the executive routines (which includes sysinfo.c).

I also modified the NtQuerySystemInformation function in the sysinfo.c file, so that it puts all of the data from the sysinfo.c instance of the SYSTEM\_CSE451\_INFORMATION struct into the object that is passed into the function, given that they call the function acquiring about SystemCSE451Information.

I also modified various files in base/ntos/io/iomgr (like write.c, read.c, etc.) so that at each return statement in the API that we are tracking, it catches the return status and increments it. In the case of NtReadFile and NtWriteFile, there is also a macro above the function that adds in the incrementing of the BytesUsed value. I was going to use a try-finally block to catch the status and only increment it in one place, but this doesn’t exist in C.

I also modified test.c, so that it clearly prints the SYSTEM\_CSE451\_INFORMATION after it aquires the data. Instead of keeping track of all of the error counts, warning counts, etc. for each API, after test.c gathers all of the individual statuses, it calculates the error counts, warning counts, etc.

(Note: I also noticed that it only works on the Virtual PC (even when using the provided code only) in debugger (not Enterprise mode).)