REPORT

Part 1:

The syscall function in the sycall.c calls the syscalls and stores the return value of function in eax. Here we are printing the syscall executed and its returned value.

Part 2:

The syscall are designed or implemented systematically which are called from the terminal to certain respective task. Here we have implemented or inserted a syscall name “mydate” into the xv6 code to print the current date and time to the terminal window. To implement or add a syscall to xv6:

Initially, we need to define an available syscall number to “SYS_mydate” (here it is 22) in the “syscall.h” file. Next in “syscall.c” file add “[SYS_mydate] sys_mydate” to “static int (*syscalls []) (void)” and before this declare sys_mydate(void) by adding the line “extern int sys_mydate(void)”. In “defs.h” declare function mydate by adding “int mydate (struct rtcdate *);”. Next in “user.h” file also declare the mydate function. Now in “sysproc.c” add function “int sys_mydate(void){...}”. Next in proc.c define function “int mydate (struct rtcdate *r) {...}”. Here time and date are obtained using “cmostime(...)” function which in called by function “proc.c” and stored into “struct rtcdate”. We need to also create a c file namely “mydate.c” which is implemented when syscall is invoked. Here, we call the functions we created before and print the output to the terminal window. Now finally add the mydate.c file to makefile and modify it.

This is the way how the syscall call flows and they are implemented in xv6. We can trace back the map flow. Now type “mydata” in terminal to get current date and time.